XIII CONFERENCIA LATINOAMERICANA DE INFORMATICA

XI CONGRESO COLOMBIANO DE CALCULO ELECTRONICO E INVESTIGACION OPERACIONAL

ORGANIZADO POR:

ACCIO
ASOCIACION COLOMBIANA DE INFORMATICA

UNIVERSIDAD DE LOS ANDES. BOGOTA

AUSPICIADO POR:

CLEI
CENTRO LATINOAMERICANO DE ESTUDIOS EN INFORMATICA

Noviembre 9 al 13 de 1987
Bogotá, Colombia

EDGAR ENRIQUE RIVAS ORTIZ
OSCAR LOPEZ BELTRAN

A.O
25/87
#

XIII CONFERENCIA LATINOAMERICANA DE INFORMATICA

XI CONGRESO COLOMBIANO DE CALCULO ELECTRONICO E INVESTIGACION OPERACIONAL

Bogotá, 9 al 13 de Noviembre de 1987

ORGANIZADO POR:

CLEI
Centro Latinoamericano de Estudios en Informática

ACCIO
Asociación Colombiana de Informática

UNIANDES
Universidad de los Andes - Bogotá

AUSPICIADO POR:

UNESCO
Oficina Regional de Ciencia y Tecnología
para América Latina y el Caribe

CON LA COLABORACION DE:

- Universidad Nacional de Colombia
- Universidad Piloto de Colombia
- Universidad Autónoma de Manizales

Los organizadores de la "XIII COFERENCIA LATINOAMERICANA DE INFORMATICA" y del "XI CONGRESO COLOMBIANO DE CALCULO ELECTRONICO E INVESTIGACION OPERACIONAL" agradecen muy especialmente la colaboración y apoyo que han brindado las siguientes compañías e instituciones:

El Tiempo
El Heraldo
Avianca
ICM de Colombia Ltda.
Teorema Ltda.
Lógica Ltda.
Procálculo S.A.
Gémini Computadores
UNISYS de Colombia
CopiTécnica Ltda.
Mac Graw Hill

El Espectador
El Siglo
Aviatur S.A.
Coldatos S.A.
Xerox de Colombia S.A.
Colinversiones S.A.
J. Glottmann S.A.
Ecopetrol
SuperMicros y Sistemas S.A.
Diners Club de Colombia S.A.
Colpuertos

La República
El Colombiano
Texins de Colombia S.A.
Educar Informática
Apple Computer Inc.
NCR de Colombia S.A.
MicroMundo
IBM de Colombia S.A.
Legis Ltda.
Ceina S.C.
Telecom

Agencia de Cargas y Turismo de Carga y Aduanas
Compañia Colombiana de Computadores Colcomp S.A.
Empresa Nacional de Computadores y Sistemas S.A.
Sistemas de Tecnología Avanzada S. T. A.
Secretaría de Informática de la Presidencia de la República
Data Processing Supplies Ltda.
Financiera Eléctrica Nacional S.A.
Centro Distrital de Sistematización y Servicios Técnicos SISE
Instituto Nacional de Investigaciones Geológicas y Mineras IngeoMinas
Fondo Colombiano de Investigaciones Científicas COLCIENCIAS

EXORDIO

El espíritu de los tiempos y el estado del arte de la tecnología obligan a considerar a la informática como un factor fundamental en el desenvolvimiento político, económico, social y cultural de cualquier nación.

El impacto económico de la informática, habida cuenta de la amplitud del ámbito que cubre y de la profundidad del mismo, es de tales proporciones que las consecuencias de ignorar el fenómeno, el tratarlo parcialmente como una tecnología más a utilizar, o el no dedicarle con la debida oportunidad el esfuerzo a encauzarlo en bien del progreso nacional, puede provocar un retraso quizás irremediable.

En este orden de ideas resulta inquietante la cuestión sobre el papel que pueden jugar nuestras débiles economías latinoamericanas en el contexto del fenómeno informático mundial, en el cual, en gran medida, sólo nos ha correspondido ser receptores pasivos de tecnologías provenientes de los países desarrollados que concentran en proporción aplastante el poder en este determinante cambio.

Es entonces cuando cobra sentido impulsar un proceso integracionista latinoamericano en materia de informática, por cuanto es apenas obvio deducir las ventajas que se desprenden de la cooperación e integración entre los países de América Latina, promovidas en el marco de una política global, frente a la alternativa de acometer la empresa aisladamente, con la consiguiente duplicación de esfuerzos, debilidad negociadora y pérdida de la capacidad de alcance de prometedores efectos sinergéticos.

Pero ¿ cómo "hacer el camino" en una región donde los intentos integracionistas se han distinguido, si no por su total fracaso, tampoco por su estruendoso éxito ?.

La respuesta parece encontrarse en el dominio de las aproximaciones sucesivas. Los acuerdos suscritos a nivel de los gobiernos, si bien son necesarios, en absoluto resultan suficientes. La crisis a que se vió abocado recientemente el Pacto Subregional Andino, es un claro ejemplo al respecto. Es indispensable complementariamente propiciar el acercamiento entre los estamentos docentes, de investigación y empresariales de los países que integran la región. Reforzar las relaciones que presenta el MODELO SABATINO, compartir el "savoir-faire" alcanzado, propender por la apropiación de tecnología para la región, establecer convenios en los campos de la educación y la investigación, y consorcios en el de la industria - que de por sí estaría aprovechando las ventajas que brinda el mercado ampliado -, y en fin, fomentar el intercambio de inquietudes, son, todos, objetivos deseables, y son precisamente los que han alentado la realización de las sucesivas versiones de la CONFERENCIA LATINOAMERICANA DE INFORMATICA, evento enhorabuena creado por el Centro Latinoamericano de Estudios en Informática CLEI, que ha venido deambulando por la amplia geografía de la gran patria latinoamericana con la plena certeza de la altísima misión que se ha impuesto.

El desafío es grande, pero parece el único camino viable, si es que los países de América Latína, que es nuestra América, desean encontrar el sendero que los conduzca al anhelado desarrollo socio-económico de manera sostenida y con una mayor autonomía, así como lograr una inserción más equitativa en la economía internacional y una digna reafirmación de sus irrenunciables valores culturales en el concierto de las naciones del universo-mundo.

En esta oportunidad, gracias al esfuerzo mancomunado de la ASOCIACION COLOMBIANA DE INFORMATICA-ACCIO y de la UNIVERSIDAD DE LOS ANDES, así como de otras instituciones que se han sumado al empeño, Colombia abre sus puertas generosamente a los hermanos latinoamericanos, y al evocar con su nombre la magnífica osadía del Gran Almirante, es del caso asímismo, proclamar la invitación a atrevernos a llevar a la práctica, al menos en el campo informático, la idea formidable que en el desafortunadamente fallido Congreso Anfictiónico, animó al más grande visionario de América: Simón Bolivar, cuyo legado tenemos el ineludible compromiso de acrecentar.

Jaime Obregón Puyana
Predidente
Asociación Colombiana de Informática ACCIO

Arturo Infante Villarreal
Rector
Universidad de los Andes

Noviembre de 1987

COMITE DE HONOR

XIII CONFERENCIA LATINOAMERICANA DE INFORMATICA
XI CONGRESO COLOMBIANO DE CALCULO ELECTRONICO E INVESTIGACION OPERACIONAL

Dr. Virgilio Barco V.
Presidente de la República

Dr. Fernando Cepeda U.
Ministro de Comunicaciones

Dr. Fuad Char A.
Ministro de Desarrollo Económico

Dr. Pedro Amaya P.
Director del Fondo Colombiano de Investigaciones Científicas COLCIENCIAS

Dr. Alfonso González C.
Director del Departamento Administrativo Nacional de Estadística DANE

Dr. Emilio Aljure N.
Director del Instituto Colombiano para el Fomento de la Educación Superior ICFES

Dr. Ernesto LLeras M.
Presidente de la Asociación de Ingenieros de Sistemas ACIS

Dr. Juan José González
Presidente de la Asociación Colombiana de Usuarios de Computador ACUC

Dr. Alberto Restrepo V.
Presidente de la Asociación de Usuarios de Computador AUC

Dr. Jorge Vidart
Presidente del Centro Latinoamericano de Estudios en Informática CLEI

Dr. Aldo Migliaro O.
Secretario Ejecutivo del Centro Latino-Americano de Estudios en Informática CLEI

PRESIDENCIA COLEGIADA

XIII CONFERENCIA LATINOAMERICANA DE INFORMATICA
XI CONGRESO COLOMBIANO DE CALCULO ELECTRONICO E INVESTIGACION OPERACIONAL

Arturo Infante Villarreal
Rector
Universidad de los Andes

Jaime Obregón Puyana
Presidente Asociación
Colombiana de Informática ACCIO

CENTRO LATINOAMERICANO DE ESTUDIOS EN INFORMATICA

CLEI

CONSEJO DIRECTIVO

Presidente

Jorge Vidart
Escuela Latinoamericana de Informática ESLAI
Argentina

Secretario Ejecutivo

Aldo Migliaro Osorio
Universidad Católica de Valparaíso
Chile

Victor Manuel Toro C. [posesión en Noviembre'87]
Universidad de los Andes
Colombia

Miembros

Newton Braga Rosa
Universidad Federal de Rio Grande do Sul
Brasil

Jaime Obregón Puyana
Asociación Colombiana de Informática ACCIO
Colombia

José Pino Urtubia
Universidad de Chile
Chile

Luis Alberto Meyer
Universidad Católica del Paraguay
Paraguay

Daniel Gascue [tesorero CLEI]
Centro Interdisciplinario de Estudios CIEDUR
Uruguay

Julián Araoz
Universidad Simón Bolivar
Venezuela

COMITE DE PROGRAMA

Presidente

Rodrigo López B.
Universidad de los Andes
Colombia

Miembros

Marcos Borges
Universidad Federal de Rio de Janeiro
Brasil

José Durán
Universidad de Santa María
Chile

Victor Gamarra Herrera
Universidad Nacional de Ingeniería
Perú

Fernando Brum
Interfase Ltda.
Uruguay

Alejandro Teruel
Universidad Simón Bolivar
Venezuela

COMITE EJECUTIVO
XIII CONFERENCIA LATINOAMERICANA DE INFORMATICA
XI CONGRESO COLOMBIANO DE CALCULO ELECTRONICO E INVESTIGACION OPERACIONAL

Presidente

Victor Yockteng M.
Asociación Colombiana de Informática - ACCIO

Miembros

Rodrigo López B.
Universidad de los Andes

Victor Manuel Toro C.
Universidad de los Andes

Ricardo Amaya Umaña
Asociación Colombiana de Informática ACCIO

Pilar Murcia
Universidad de los Andes

Rubby Casallas G.
Universidad de los Andes

Secretaria Ejecutiva

María Isabel Mier F.

Asistente de Contabilidad

Fabio Orozco

=====*=====

COMITE ACADEMICO
XIII CONFERENCIA LATINOAMERICANA DE INFORMATICA
XI CONGRESO COLOMBIANO DE CALCULO ELECTRONICO E INVESTIGACION OPERACIONAL

Presidente

Rodrigo López B.
Lenguajes de Programación - UniAndes

Miembros

José Abásolo
Bases de Datos-UniAndes

Rodrigo Cardoso
Informática Teórica-UniAndes

Carlos Cortés
Teoría de Sistemas-U. Nacional

Alfonso Charún
Telecomunicaciones-U. Nacional

Mauro Flórez
Telecomunicaciones-U.Piloto

Alvaro Galvis P.
Educ. Apoyada por Comp.-UniAndes

Jairo Gómez
Bases de Datos-B.C.H.

José Tiberio Hernández
CAD/CAM-UniAndes

Camilo Rueda
Inteligencia Artificial-U.A.Manizales

Victor Manuel Toro C.
Ingeniería de Software-UniAndes

ASESORES ACADEMICOS
XIII CONFERENCIA LATINOAMERICANA DE INFORMATICA
XI CONGRESO COLOMBIANO DE CALCULO ELECTRONICO E INVESTIGACION OPERACIONAL

Jaime Bohórquez
Programación Lógica-UniAndes

Juan Manuel Damián
Arq. Computadores-UniAndes

María Consuelo Franky
Sistemas Distribuídos-UniAndes

Olga Lucía Giraldo
Estruc. Bases Datos-UniAndes

Roberto Ojeda
Leng. Programación-U.Nacional

Mario José Otero
Paralelismo-UniAndes

Ignacio Pérez V.
Inv. Operacional-UniAndes

Alejandro Quintero
Sist. Operacionales-UniAndes

Leonardo Ramírez
Sistemas Públicos-UniAndes

Francisco Rueda F.
Sist. Operacionales-UniAndes

Jorge Villalobos
Interf. Hombre-Máquina - UniAndes

===*===

ASOCIACION COLOMBIANA DE INFORMATICA
ACCIO

Presidente:	Jaime Obregón Puyana	Ricardo Amaya Umaña
VicePresidente:	Victor R. Yockteng M.	Héctor Múnera Orozco
Secretario:	José Arias Ordóñez	Javier Francisco Garcés
Tesorero:	Alvaro Varela Flórez	Rómulo Niño Neissa
Vocales:	Carlos Cortés Amador	Mauro Flórez Calderón
	Carlos Ardila Arenas	María Esther Ordóñez O.
	Manuel Neira Rojas	Alvaro Zarama Medina
Revisor Fiscal:	Armando García H.	Ana María Pacheco
Directora Ejecutiva:	María Isabel Mier	

===*===

UNIVERSIDAD DE LOS ANDES
UNIANDES

Arturo Infante Villarreal
Rector

Ernesto Gühl Nannetti
ViceRector

Antonio García Rozo
Decano Facultad de Ingeniería

Francisco Rueda Fajardo
ViceDecano Facultad de Ingeniería

Ernesto LLeras Manrique
Director Centro de Investigaciones de Ingeniería

Rodrigo A. López B.
Director Dpto. Sistemas y Computación

INDICE

VOLUMEN I

INFORMATICA TEORICA E INFORMATICA SIMBOLICA

META-CONTROL Y PROGRAMACION LOGICA
ADOLFO KVITCA
ESCUELA SUPERIOR LATINOAMERICANA DE INFORMATICA - ESLAI - ARGENTINA 1

BACKTRACKING INTELIGENTE PARA PROLOG
SILVIA TAKAHASHI
UNIVERSIDAD DE LOS ANDES - COLOMBIA 17

PROGRAMACION LOGICA BASADA EN ASERCIONES: UNA HEREJIA IMPERATIVA.
JAIME BOHORQUEZ V.
UNIVERSIDAD DE LOS ANDES - COLOMBIA 27

UNA EXTENSION DE LISP PARA LA PROGRAMACION ORIENTADA A OBJETOS
GUSTAVO ARANGO
UNIVERSIDAD DE LOS ANDES - COLOMBIA 54

LOGICAS NO MONOTONICAS
RAUL CARNOTA
ESCUELA SUPERIOR LATINOAMERICANA DE INFORMATICA - ESLAI - ARGENTINA 67

IGUALDAD DEFINICIONAL DE EXPRESIONES MATEMATICAS EN LA LOGICA INTUITIVA
JUAN JOSE CABEZAS
UNIVERSIDAD DE LA REPUBLICA - URUGUAY 89

LA IGUALDAD EN TIPOS ABSTRACTOS DE DATOS
RODRIGO CARDOSO R.
UNIVERSIDAD DE LOS ANDES - COLOMBIA 110

PROCEDIMIENTO DE DECISION PARA LA EQUIVALENCIA Y LA CONGRUENCIA OBSERVACIONAL DE TERMINOS REGULARES DE CCS
AMELIA SORIANO MONTES
UNIVERSIDAD CENTRAL DE VENEZUELA - VENEZUELA 118

EDITOR DE AUTOMATAS FINITOS
FELIPE S. QUADROS; ANA M. PRICE
UNIVERSIDADE FEDERALE DO RIO GRANDE DO SUL - BRASIL 138

LOS CONJUNTOS DIFUSOS EN LA MODELIZACION LINGUISTICA
FRANCESCA GRESELIN FERRARI
UNIVERSIDAD CATOLICA DE NUESTRA SENORA DE LA ASUNCION - PARAGUAY 150

EL DICCIONARIO ELECTRONICO DEL ESPAÑOL
CARLOS SUBIRATS RUGGEBERG
UNIVERSIDAD AUTONOMA DE BARCELONA - ESPAÑA 160

ON THE CONNECTIONS BETWEEN RELATIONAL TABLEAUX AND CLAUSE FORM
PEDRO MANUEL SILVEIRA
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO - BRASIL 173

ON A MIN-MAX CONJECTURE FOR REDUCIBLE DIGRAPHS
JAYME LUIZ SZWARCFITER
UNIVERSIDADE FEDERALE DO RIO DE JANEIRO - BRASIL 181

RESOLUÇAO DE EQUAÇOES: UMA METODOLOGIA DE ESTUDO
TIARAJU ASMUZ DIVERIO
INSTITUTO DE INFORMATICA PUC RS - BRASIL 190

UNA VARIANTE DEL SISTEMA DE ROTORES
ERNESTO AZORIN; NELSON BALOIAN
UNIVERSIDAD DE CHILE - CHILE 204

PROPOSICION DE UN LENGUAJE DE NIVEL INTERMEDIO PARA UNSISTEMA DE PRODUCCION DE COMPILADORES
A. CAMPOS; Y. ETEROVIC; G. GARCIA; R. SANTIS
UNIVERSIDAD CATOLICA DE CHILE - CHILE 215

BASES PARA UN LENGUAJE DE DESCRIPCION DE LA SEMANTICA DE LOS LENGUAJES DE PROGRAMACION
A. CAMPOS; Y. ETEROVIC; G. GARCIA; R. SANTIS
UNIVERSIDAD CATOLICA DE CHILE - CHILE 225

DESARROLLO E IMPLEMENTACION DE UN INTERPRETADOR DE LENGUAJE PROLOG PORTABLE Y EXTENDIBLE
LUIS ROBERTO OJEDA; CAMILO PARRA
UNIVERSIDAD NACIONAL DE COLOMBIA - COLOMBIA 236

INGENIERIA DE SOFTWARE Y TECNICAS DE PROGRAMACION

VERIFICAÇAO E VALIDAÇAO DE SOFTWARE NA FASE DE ESPECIFICAÇAO DE REQUISITOS
CLIFTON EDUARDO CLUNIE BEAUFONT; ANA REGINA CALVANCANTI DE ROCHA
UNIVERSIDAD TECNOLOGICA DE PANAMA - PANAMA 237

PARADIGMAS PARA O PROCESSO DE PRODUÇAO DE SOFTWARE E SUAS CONSEQUENCIAS SOBRE OS AMBIENTES DE DESENVOLVIMENTO
SUELI MENDES; ANA REGINA ROCHA
UNIVERSIDADE FEDERALE DO RIO DE JANEIRO - BRASIL 257

DOS APLICACIONES DE LAS TABLAS DE DECISION
LUIS ROBERTO OJEDA CH.
UNIVERSIDAD NACIONAL DE COLOMBIA - COLOMBIA 272

UMA ANALISE REALISTA DA UTILIZAÇAO DO ENFOQUE DA PROTOTIPAGEM RAPIDA NO DESENVOLVIMIENTO DE SISTEMAS DE INFORMAÇAO
MARCOS R. S. BORGES
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO - BRASIL 280

EITIS: UM AMBIENTE DE FERRAMENTAS INTEGRADAS PARA DESENVOLVIMENTO DE SISTEMAS INTERACTIVOS
RUBENS N. MELO
PUC-RJ - BRASIL 290

DISEÑO ESPECIFICACION Y PROTOTIPIFICACION DE SISTEMAS INTERACTIVOS
VICTOR M. TORO C
UNIVERSIDAD DE LOS ANDES - COLOMBIA 302

CARTELERA DINAMICA : DISENO E IMPLEMENTACION
GERMAN RIVERA; IVAN H. REYES
UNIVERSIDAD DE LOS ANDES - COLOMBIA 326

ARQUITECTURA DE UNA INTERFASE USUARIO: UN MODELO GENERAL.
JORGE VILLALOBOS S.
UNIVERSIDAD DE LOS ANDES - COLOMBIA 333

CONSTRUCCION AUTOMATIZADA DE TALLERES DE PROGRAMACION: PEPIM
MIGUEL SANTANA
PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU - PERU 348

MESTRA: UMA METODOLOGIA DE ESCRITA DE DOCUMENTAÇAO INTERNA TRANSPORTAVEL
CAMILO DE LELIS GONDIM MEDEIROS
UNIVERSIDADE FEDERAL DA PARAIBA - BRASIL 363

INSTRUMENTADOR DE PROGRAMAS 'C'
MARCELO S. PIMENTA; VERNER HEIDRICH; ANA M. A. PRICE
UNIVERSIDADE FEDERALE DO RIO GRANDE DO SUL - BRASIL 376

SALTOS NO LOCALES: UNA EXTENSION A TURBO-PASCAL
RODRIGO LOPEZ B.
UNIVERSIDAD DE LOS ANDES - COLOMBIA 388

A PROBLEMATICA DO DESENVOLVIMIENTO DE SOFTWARE NUMERICO
TIARAJU ASMUZ DIVERIO
INSTITUTO DE INFORMATICA PUC RS - BRASIL 398

EL LENGUAJE DECLARATIVO COMO HERRAMIENTA DE IMPLEMENTACION
JUAN GROMPONE; F. BRUM; M. CASAMAYOR; F. SIMINI
INTERFASE LTDA. - URUGUAY 415

PROGRAMAÇAO LOGICA E ENGENHARIA DE SOFTWARE
TERESA CRISTINA DE AGUILAR
UNIVERSIDADE FEDERALE DO RIO DE JANEIRO - BRASIL 427

INGENIERIA DE SOFTWARE Y PROLOG
F. JAVIER DIAZ
ESCUELA SUPERIOR LATINOAMERICANA DE INFORMATICA - ESLAI - ARGENTINA 436

IMPLANTACION DE UN SISTEMA DE CONTROL INTERNO EN UN CENTRO DE PROCESAMIENTO DE DATOS
CARLOS A. PETRELLA
MINISTERIO DE TRASPORTE Y OBRAS PUBLICAS - URUGUAY 447

PROTESTE: UMA FERRAMENTA DE APOIO AO TESTE DE PROGRAMAS
ANA M. PRICE
UNIVERSIDADE FEDERALE DO RIO GRANDE DO SUL - BRASIL 461

LAS REDES DE PETRI COMO UN INSTRUMENTO PARA MODELAR DIAGRAMASDEFLUJOS DE DATOS JERARQUICOS
RICARDO ACEVEDO
UNIVERSIDAD SANTA MARIA - CHILE 463

METODOLOGIAS INTEGRADAS DE SISTEMATIZACION
JORGE ELIECER BARRERA NIÑO
BAVARIA - COLOMBIA 464

COMPUTACION GRAFICA Y DISEÑO ASISTIDO POR COMPUTADOR

MGIT-ANDES: UN MODELADOR 3-D SOBRE MICROCOMPUTADORES BASADO EN OPERADORES DE EULER
JOSE TIBERIO HERNANDEZ P.; JUAN C. QUIROGA CH.
UNIVERSIDAD DE LOS ANDES - COLOMBIA 466

USO DE HEURISTICOS EN COMPILACION DE UN LENGUAJE DE ALTO NIVEL PARA MAQUINADO EXTERNO EN TORNOS DE CONTROL NUMERICO
OSCAR RUIZ S.; RODRIGO LOPEZ B.
UNIVERSIDAD DE LOS ANDES - COLOMBIA 480

FLUX2D Y FLUX3D SOFTWARES DE CAD PARA LA MODELIZACION DE CAMPOS ELECTROMAGNETICOS
ORIETTA SANTANA; GERARD MEUNIER; JEAN LOUIS COULOMB; JEAN CLAUDE SABONNADIERE
ENSIEG - FRANCIA 501

COMPUTACION GRAFICA: METODOS DE APROXIMACION DE CURVAS Y SUPERFICIES
DANIEL FORMICA; CLAUDIO DELRIEUX; DANIEL MAGNI
DEPARTAMENTO DE INGENIERIA ELECTRICA - ARGENTINA 514

INTRODUCCION A LA TOMOGRAFIA AXIAL COMPUTARIZADA
JOSE J. RAMOS RODRIGUEZ
ECUADOR 532

SISTOM: UN SISTEMA PARA VISUALIZAÇAO TRIDIMENSIONAL DE TOMOGRAFIA COMPUTARIZADA
ADRIANO RAMOS GONCALVES; KAREN BASSO; PAULO ROBERTO PARTICHELLI; JACOB SCHARCANSKI; MAGALI T. LONGHI
PONTIFICIA UNIVERSIDADE CATOLICA DO RIO GRANDE DO SUL - BRASIL 545

UTILIZACION COMPARADA DE DIVERSOS ALGORITMOS DE VISUALIZACION DE FACETAS EN C.A.D.
O. SANTANA; J.B. ALBERTINI; B. AUBRY; G. MEUNIER; J.L. COULOMB
LABORATOIRE D'ELECTROTECHNIQUE DE GRENOBLE (UA CNRS 355) - FRANCIA 546

SISTEMA DE ANIMAÇAO AUXILIADO PELO COMPUTADOR
GEOVANNI DE ARAUJO BERTACO; LUIZ FERNANDO MESQUITADESOUZA; MARCELOROSSONI; J. SCHARCANSKI
PONTIFICIA UNIVERSIDADE CATOLICA DO RIO GRANDE DO SUL - BRASIL 548

LOS SISTEMAS DE MANUFACTURACION FLEXIBLE (FMS):UNAIMPLEMENTACION
LIZ JEANNETTE CUBILLOS CANCINO; MARTA LUCIA PARDO MARTINEZ
UNIVERSIDAD NACIONAL DE COLOMBIA - COLOMBIA 549

DEFINICION DE LA BASE DE CONOCIMIENTOS PARA UN SISTEMA DE DIAGNOSTICO DE FALLAS EN CONTROL AUTOMATICO DE PROCESOS
DANIEL SBARBARO H.; DANIEL CAMPOS D.
UNIVERSIDAD DE CONCEPCION - CHILE 551

APLICACIONES NOVEDOSAS

ASPECTOS DE LA INFORMATICA MEDICA: EVOLUCION DESARROLLO Y PERSPECTIVAS EN COLOMBIA
EMILIO SORZANO ESPINOSA
COLOMBIA 552

SISTEMA INTEGRADO DE CONTROL PARA LA UNIDAD DE CUIDADOS INTENSIVOS PEDIATRICA
E. SORZANO E.; C. SANCHEZ FALLA; M. JOSEFA DUQUE;J. ERNESTOCASTRO; F. GONZALEZ FRANCO
COLOMBIA 558

SISTEMA AUTOMATICO DE EVALUACION DIAGNOSTICA EN EDUCACION MEDICA
EMILIO SORZANO ESPINOSA; ALBERTO BERNAL; CARLOS RODRIGUEZ
COLOMBIA 564

ANALISIS MATEMATICO Y CALCULO DE RIESGO EN ENFERMEDADES GENETICAS
EMILIO SORZANO ESPINOSA; ALBERTO MARTINEZ; MARCELA TRIANA; JAIME BERNAL
COLOMBIA 568

MODELO PARA EL CONTROL Y VIGILANCIA EPIDEMIOLOGICA DE LA RABIA
EMILIO SORZANO ESPINOSA; GERMAN PINZON C.; MAURICIO LARA R.
COLOMBIA 572

SISTEMA DE CONTROL DIGITAL PARA EL MUSCULO ESQUELETICO
JORGE ORLANDO CASTRO NOVA; MIGUEL VICENTE PARRADO LOZANO
UNIVERSIDAD NACIONAL DE COLOMBIA - COLOMBIA 577

PROYECTO PILOTO DE INFORMATICA PARA EL MANEJO DE LOS IMPUESTOS EN COLOMBIA
ESPERANZA BEDOYA V.; ALFREDO HERNANDEZ C.
MINHACIENDA - COLOMBIA 597

SISTEMA DE INFORMACION PARA LA ADUANA NACIONAL DE COLOMBIA
ALFREDO HERNANDEZ CIFUENTES; FABIO CAMACHO PARDO
MINHACIENDA - COLOMBIA 610

LA SISTEMATIZACION DE LA SUPERINTENDENCIA BANCARIA COLOMBIANA
ALFREDO HERNANDEZ CIFUENTES; RODRIGO NASSAR MONTOYA
MINHACIENDA - COLOMBIA 629

BASE DE DATOS DE LA DEUDA PUBLICA COLOMBIANA (FENIX)
EDUARDO ASMAR; ALFREDO HERNANDEZ; ALEJANDRO SALAMANCA; ANDRES SUAREZ
MINHACIENDA - COLOMBIA 642

APLICACION DE LOTUS 123 AL MANEJO DE DATOS DE RIEGO
RICARDO SANCHEZ
UNIVERSIDAD DEL VALLE - COLOMBIA 659

SISTEMA UNIVERSITARIO DE INFORMATICA JURIDICA
JESUS SANTOS AMAYA
ICFES - COLOMBIA 668

VOLUMEN II

EDUCACION EN INFORMATICA

AMERICA LATINA: ALTERNATIVA DE LA REVOLUCION INFORMATICA HACIA EL AÑO 2000.
JAIME ACOSTA PUERTA
CRESET - COLOMBIA 670

MODELOS CURRICULARES EN INFORMATICA-COMPUTACION
VICTORIA BAJAR; GUILLERMO LEVINE
INSTITUTO TECNOLOGICO AUTONOMO DE MEXICO - MEXICO 695

INFORMATICA Y COOPERACION: UNA PERSPECTIVA
RICCARDO FERRARI; GERONIMO BELLASAI; BENJAMIN BARAN
UNIVERSIDAD CATOLICA DE NUESTRA SENORA DE LA ASUNCION - PARAGUAY 709

ALFABETIZACION MASIVA EN COMPUTACION EN LA UNIVERSIDAD CATOLICA DE CHILE
PEDRO HEPP K.; LUIS DISSETT
UNIVERSIDAD CATOLICA DE CHILE - CHILE 719

EDUCANDO HACIA LA INGENIERIA DE SOFTWARE: UNA REINTERPRETACION PARA URUGUAY
DANIEL SCHWED
URUGUAY 726

UNA LINEA DE INVESTIGACION EN LOS ESTUDIOS DE INGENIERIA DE SISTEMAS EN LA UNIVERSIDAD NACIONAL DE COLOMBIA.
CARLOS CORTES AMADOR
UNIVERSIDAD NACIONAL DE COLOMBIA - COLOMBIA 738

PLANIFICACION DE INFORMATICA EDUCATIVA A NIVELES UNIVERSITARIOS
CARLOS SCHEEL
UNIVERSIDAD DEL SAGRADO CORAZON - PUERTO RICO 741

INFORMATICA EDUCATIVA

UN AMBIENTE DE DESARROLLO DE SOFTWARE PARA EDUCACION
PEDRO HEPP K.; JAIME NAVON C.
UNIVERSIDAD CATOLICA DE CHILE - CHILE 743

ORIGEN-DESARROLLO E IMPLEMENTACION DE UN SISTEMA COMPUTACIONALDEAPOYOAPROCESOSDE ENSENANZA-APRENDIZAJE
GIOVANNI SENI ZAMBRANO
UNIVERSIDAD DE LOS ANDES - COLOMBIA 754

SISTEMA EDUCATIVO COMPUTACIONAL PARA EL SUMINISTRO DE ANESTESIA
SARA CROITORU EINSTEIN; VIVIAN LUCHAU BOTERO; JORGE MEJIA
UNIVERSIDAD DE LOS ANDES - COLOMBIA 760

CARDIAC: UNA AYUDA DIDACTICA PARA LA ENSENANZA DE QUE ES UNA COMPUTADORA
M. OBREGON S.; S. PATRICIA ALBA S.;J. L. CHAMORRO; MARIA T. OSPINA; G. BALLA; S. VARGAS
COLOMBIA 777

SISTEMA TUTOTIAL CON EJERCITACION Y PRACTICA PARA LA ENSENANZA DE LA SUMA
FERNANDO M. DURAN CAMPOS
UNIVERSIDAD DE LOS ANDES - COLOMBIA 784

SISTEMA DE EDUCACION POR COMPUTADOR SECOM
JOSE ABRAHAM CASTAÑEDA VARGAS; MARIA FERNANDA RAMIREZ DE CASTAÑEDA
COLOMBIA 795

USO DE MINIX PARA LA ENSENANZA DE SISTEMAS OPERATIVOS
MARTIN SANTANA; MIGUEL SANTANA
ESCUELA DE ADMINISTRACION DE NEGOCIOS PARA GRADUADOS. PERU 815

RED DE INTERCAMBIO DE INFORMACION SOBRE INFORMATICA EDUCATIVA
ALVARO H. GALVIS PANQUEVA; OLGA MARIÑO; FRANCISCO RUEDA
UNIVERSIDAD DE LOS ANDES - COLOMBIA 817

SISTEMA MANEJADOR DE BANCOS DE PROBLEMAS
ALVARO H. GALVIS PANQUEVA.; OLGA MARIÑO; FRANCISCO RUEDA F.
UNIVERSIDAD DE LOS ANDES - COLOMBIA 819

SISTEMA DE EVALUACION DEL CONOCIMIENTO DE ESTUDIANTES CON APOYO COMPUTACIONAL
CESAR GUZMAN RENTERIA; CARLOS ALBERTO SANTANA LANCON
INSTITUTO POLITECNICO NACIONAL - MEXICO 821

INVESTIGACION OPERACIONAL

UNA APROXIMACION A LA DINAMICA PERIURBANA
ISAAC DYNER
UNIVERSIDAD NACIONAL DE COLOMBIA - COLOMBIA 822

SISTEMA PARA GENERACION EN MICROCOMPUTADORES DE MATRICES Y REPORTES DE PROGRAMACION LINEAL
B. BRITO; R. CALABRESE; E. CORREA; B. LORENZO
UNIVERSIDAD CENTRAL DE VENEZUELA - VENEZUELA 831

PROGRAMA PARA LA SOLUCION DE MODELOS DE COLAS
MAURICIO ZUÑNIGA CASTAÑEDA; FERNANDO PALACIOS GOMEZ
UNIVERSIDAD DE LOS ANDES - COLOMBIA 850

TIEMPO: SISTEMA PARA ANALISIS GRAFICO DE SERIES DE TIEMPO
F. DA SILVA; E. GRANADOS; B. LORENZO
UNIVERSIDAD CENTRAL DE VENEZUELA - VENEZUELA 871

EVALUACION Y PREDICCION DEL DESEMPENO DE SISTEMAS COMPUTACIONALES
IGNACIO CASAS; FRANCISCO AURTENECHEA
UNIVERSIDAD CATOLICA DE CHILE - CHILE 889

SIMCOL : UNA HERRAMIENTA PARA SIMULAR MODELOS DE REDES DE COLAS
FEDERICO R. ECHAVARRIA R.; FERNANDO PALACIOS GOMEZ
UNIVERSIDAD DE LOS ANDES - COLOMBIA 909

REDES Y SISTEMAS DISTRIBUIDOS

PROTOCOLOS DE CONVERSION PARA EL INTERCAMBIO DE INFORMACION ENTRE BASES DE DATOS BIBLIOGRAFICAS HETEROGENEAS
JOSE ABASOLO PRIETO
UNIVERSIDAD DE LOS ANDES - COLOMBIA 931

TESEO: AMBIENTE DE PROGRAMACION DE SISTEMAS TRANSACCIONALES DISTRIBUIDOS
MARIA CONSUELO FRANKY
UNIVERSIDAD DE LOS ANDES - COLOMBIA 946

COMPILADOR DE EXPRESIONES DEL LENGUAJE RELACIONAL PARA EL SISTEMA TESEO
PERLA LEVY
UNIVERSIDAD DE LOS ANDES - COLOMBIA 972

DETERMINACION Y RESTAURACION AUTOMATICA DE LOS ESTADOS INTERMEDIOS DE UNA APLICACION
MARCO MARCINELLI
UNIVERSIDAD CATOLICA DE NUESTRA SENORA DE LA ASUNCION - PARAGUAY 991

OPTIMIZACION EN TELECOMUNICACIONES
EDUARDO M. LOPEZ SORIANO
ITESM - INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MO - MEXICO 1003

ANALISIS DE LA UTILIZACION MAXIMA DE UN CANAL DE COMUNICACIONES EN UN ENLACE VIA RADIO
BRIA OSCAR NORBERTO; DE GIUSTI MARISA RAQUEL
ARGENTINA 1024

ESPECIFICAÇAO DA CAMADA DE SESSAO PARA A REDE-RIO
LUCI PIRMEZ
UNIVERSIDADE FEDERALE DO RIO DE JANEIRO - BRASIL 1037

SISTEMA DE EMULACION DE TERMINAL Y TRANSFERENCIA DE ARCHIVOS
HUGO SIN TRIANA; MAURICIO ARANGO GOMEZ
UNIVERSIDAD DE LOS ANDES - COLOMBIA 1062

ANALISE E ESPECIFICAÇAO DOS SERVICIOS DE TRANSFERENCIA ACCESO E MANIPULAÇAO DE ARCHIVOS (FTAM)
ESTHER DE CASTRO PACITTI
UNIVERSIDADE FEDERALE DO RIO DE JANEIRO - BRASIL 1078

ANALISIS DEL VIDEOTEX COMO SERVICIO TELEMATICO PARA COLOMBIA
HECTOR ALIRIO ROJAS
UNIVERSIDAD NACIONAL DE COLOMBIA - COLOMBIA 1092

BASES DE DATOS Y ESTRUCTURAS DE DATOS

COMPORTAMIENTO DEL ARBOL-BD EN LAS FASES CRECIENTE - DECRECIENTE Y ESTACIONARIO
O. SANTANA; O. MAYOR; M. DIAZ; G. LOPEZ
UNIVERSIDAD POLITECNICA DE CANARIAS - ESPAÑA 1093

BUSQUEDA DE LOS M-VECINOS MAS PROXIMOS EN EL ARBOL-BD
O. SANTANA; M. DIAZ; O. MAYOR; J. GONZALEZ
UNIVERSIDAD POLITECNICA DE CANARIA - ESPAÑA 1106

AN ALGORITHM FOR DYNAMIC STORAGE ALLOCATION AND ITS APPLICATION TO B+-TREES
RICARDO A. BAEZA-YATES
UNIVERSITY OF WATERLOO - CANADA 1122

ESTRUCTURAS DE DATOS PLEX
JORGE ARGENGDHI; GUILLERMO CICILEO; SONIA RUEDA; MARCELO ZANCONI
CENTRO REGIONAL DE INVESTIGACIONES BASICAS Y APLICADAS - ARGENTINA 1133

ARBOLES QUINTARIOS: ESTUDIO EXPERIMENTAL PARA INTERROGACIONES EXACTAS- PARCIALES- EN RANGO Y EN RANGO PARCIAL
O. SANTANA; O. MAYOR; M. DIAZ; S. REINA
UNIVERSIDAD POLITECNICA DE CANARIAS - ESPAÑA 1148

ESQUEMAS Y ESTRUCTURAS PARA LA BUSQUEDA DE LAS PALABRAS MAS SIMILARES A UNA DADA
O. SANTANA; M. DIAZ; O. MAYOR; J. REYES
UNIVERSIDAD POLITECNICA DE CANARIAS - ESPAÑA 1169

UM SISTEMA PARA AUXILIO AO PROJETO E INTEGRAÇAO DE ESQUEMAS CONCEPTUAIS SEMANTICOS
JANO MOREIRA DE SOUZA
UNIVERSIDADE FEDERALE DO RIO DE JANEIRO - BRASIL 1190

BASES DE DATOS: OPTIMIZACION DEL MODELO INTERNO
RODOLFO SCHMAL S.; YANKO OSSADON N.
UNIVERSIDAD DE TARAPACA - CHILE 1199

MODELO BORROSO DE UN SISTEMA DE RECUPERACION DE INFORMACION
MIGUEL FRANCISCO BUITRAGO B.; WILLIAM JARAMILLO J.
UNIVERSIDAD EAFIT - COLOMBIA 1208

COMPARACION ESTRUCTURAL DE INGRES Y ORACLE EN EL AMBIENTE UNIX
ROY CAHN-SPEYER V.; ALFREDO HERNANDEZ C.; ANDRES SUAREZ R.
MINHACIENDA - COLOMBIA 1221

SISTEMA ESPECIALISTA PARA VALIDAÇAO DE BASES ARUANDA
ANGELA MARIA BOSISIO; MARCIA PACITTI TELES
SERVICIO GENERAL DE PROCESSAMENTO DE DADOS - SERPRO - BRASIL 1242

SOFTWARE Y HARDWARE DE BASE

EL COMPUTADOR PERSONAL COMO ANALIZADOR DE ESTADOS LOGICOS DE BAJO COSTO
GERONIMO BELLASSAI
UNIVERSIDAD CATOLICA DE NUESTRA SENORA DE LA ASUNCION - PARAGUAY 1252

PROJETO E IMPLEMENTAÇAO DO BANCO DE DADOS DE UN SISTEMA PARA PAC DE MICROCIRCUITOS
KARLA PARREIRAS POLANCZYK; NILTON DE OLIVEIRA JUNIOR; ALBERTO HENRIQUE FRADE LAENDER
UNIVERSIDADE FEDERALE DE MINAS GERAIS - BRASIL 1260

O SOFTWARE DO SISTEMA MULTIPROCESSADOR MICRO-BIS
CELSO MACIEL DA COSTA; SIMAO SIRINEO TOSCANI; THADEU BOTTERI CORSO
UNIVERSIDADE FEDERALE DO RIO GRANDE DO SUL - BRASIL 1273

EDUCACION EN INFORMATICA

AMERICA LATINA:
ALTERNATIVA DE LA REVOLUCION INFORMATICA HACIA EL AÑO 2000

Jaime Acosta Puertas
Centro Regional de Estudios del Tercer Mundo - CRESET
Apartado Aéreo 17417
Bogotá - COLOMBIA

Las opiniones expresadas en este artículo son de exclusiva responsabilidad del autor, y por lo tanto no comprometen al CRESET.

RESUMEN

A partir de 1977, Brasil es el único país de AL que optó con decisión, por un desarrollo tecnológico más autónomo. Los demás países del área de mayor o menor industrialización, tienen en la estructura de sus modelos político-económicos, graves deficiencias para impulsar proyectos permanentes de I&D, que nos permitan una menor dependencia tecnológica. Aunque Argentina y Mexico también tienen desarrollos que los orienta a procesos tecnológicos propios.

Ante la evidencia de la Revolución Informática, el conjunto de AL tiene la obligación de pensar su incorporación, en términos que le signifique crear a partir de sí misma, una nueva sociedad que satisfaga las necesidades de la gran mayoría de su población.

El conocimiento profundo de nuestra realidad, permitirá replanteamientos a nuestra sociedad de consumo. Donde el conjunto de las ciencias deberán hacer frente común, ya que es necesario pensar en un nuevo ser MULTIDIMENSIONAL y fundamentado en el trabajo INTERDISCIPLINARIO.

Solo así, América Latina puede hacer la Revolución Informática Alternativa en los años 2.000.

1. AMERICA LATINA. PARADIGMA DE LOS AÑOS 2.000. DEMOCRATIZACION DE LA INFORMATICA Y HUMANIZACION DE LAS NUEVAS TECNOLOGIAS.

Para los países semi-industrializados de la América Latina AL, entre ellos Colombia, la Nueva Revolución Industrial constituye un desafío a creer en su creatividad y un reto para proyectar su conocimiento y prospectar su desarrollo, permitiéndonos espacios específicos y personalizados que nos integre al mundo con identidad propia.

Estos nuevos espacios solo son viables en la medida que cada país y el conjunto de la región, identifique y dinamice estratégicos sectores de desarrollo aún no explotados, e integre a la demanda a los inmensos grupos sociales marginados de los avances científicos y tecnológicos. O sea, más del 80% de nuestra gente.

Para lo cual, la capacidad de innovación endógena, en convenios sub y/o regionales, será necesaria para impulsar proyectos de desarrollo con perspectivas a adquirir una mayor autonomía tecnológica, que apunte a intercambios simétricos, como figura más equilibrada a la asimetría que caracteriza el actual intercambio comercial con los países industrializados PI.
AL podrá hablar de desarrollo industrial, cuando la IT sea un acto cotidiano en su propia industria.

Esto permitirá hacer propuestas tecnológicas alternativas, que nos acredite como aportadores de innovaciones a la Revolución Informática RIN. Como es el caso del convenio científico-tecnológico de Argentina y Brasil, para el desarrollo de programas de Quinta Generacion.

Este es un ejemplo de propuesta de AL al mundo. Y no solo, de una parte de este hacia AL.

En la medida que se logre reivindicar y obtener un espacio en la comunidad científica y tecnológica CCT, se podrá hablar de relaciones interdependientes entre AL y los países desarrollados. Entonces, el pregonado "Redespliegue Tecnológico", será más equitativo, por cuanto seremos participantes creativos y no deslumbrados receptores de utopías y en muchos casos, pasivos compradores de irracionalidades consumistas.

Adquirir un espacio en el desarrollo, implica a nuestros países destinar importantes recursos en I&D, diseñar e implementar políticas tecnológicas a mediano y largo plazo y, señalarle a las transnacionales nuevos esquemas de Transferencia Tecnológica TT, más justas y equilibradas como urge al caso colombiano, en la producción de Bienes Intermedios y de Capital.

Más allá de estos primeros enunciados, la Tercera Revolución Industrial en medio de una prolongada recesión económica debe significar para AL, la oportunidad de gestar particulares modelos

políticos, económicos, sociales y tecnológicos, que articulados, representen nuestra propia visión del mundo hacia una nueva sociedad latinoamericana.

Este sueño obliga al interior de AL una autocrítica al comportamiento de las áreas del conocimiento. Su incidencia en la concepción del Estado, en la capacidad para integrar nuestra heterogénea estructura social, y las razones del rezago tecnológico. Deficiencias que han determinado un complejo, dependiente y tortuoso proceso histórico.

Una adecuada respuesta a estas problemáticas, será necesaria, por cuanto un proceso de mayor industrialización debe trascender la simple creación de nuevas industrias de bienes de capital, reducción de las importaciones de bienes intermedios o, la dependiente modernización de los sistemas de servicios. Esta Tercera Revolución de la Inteligencia, debemos interpretarla como la oportunidad de concebir una nueva sociedad, democratizando la IN y humanizando la incorporación de las NT. Se trata de crear una manera interdependiente de pensar nuestro futuro.

Esto significa relegar el desarrollo unidimensional y economicista, a una visión integral de la ciencia, de las técnicas y de las artes. Los informáticos no pueden ser los nuevos mesías que por sí solos sustituyan a los anteriores (economistas). Esa Nueva Sociedad a partir de la misma AL deberá crear para nosotros y para el mundo un nuevo ser MULTIDIMENSIONAL, que culturalmente nuestro multietnismo lo permite, y que reponda al agotamiento de un esquema individualista, con un ser más humanizado y colectivo.

AL no puede aceptar que aquel tipo de antivalor se inserte aún más y nos desarticule, sin que hayamos tenido la vivencia de nuestro propio desarrollo.

Para esos nuevos hombres y mujeres, detentadores de nuevos métodos del conocimiento, concientizados solo para solucinnes interdisciplinarias, tendrán el compromiso de interpretar nuestra heterogeneidad social, cuya negación ha sido uno de los grandes vacíos de todos los modelos políticos y económicos, que han existido en AL y que solo han permitido a un 10 o 15% de nuestra población, acceder a la "modernización" o al crecimiento material.

AL tendrá en el año 2.000, 850 millones de habitantes (1), quiere decir que siguiendo la actual tendencia de prosperidad social, solo 85 a 120 millones de personas tendrían posibilidades a un mayor bienestar. O sea, menor a la actual población de Brasil o del Japón.

Y esa reducida población no puede ser el mercado interno que jalone proyectos de I&D, para la innovación tecnológica IT, que ha sido la otra omisión de los modelos económicos latinoamericanos.

Ese nuevo ser especializado, pero MULTIDIMENSIONAL, deberá trabajar con la mira puesta a solucionar las necesidades básicas y

de consumo de esos cientos de millones de seres marginados del desarrollo, hoy folklóricos, mañana, la esperanza para que la IN y las NT, se redimensionen dentro de realidades particulares que universalicen sus posibilidades, permitiendo una mayor autonomía tecnológica de AL con la inteligencia de científicos e investigadores propios.

Esta aproximación de conjunto, hará asumir a las ciencias sociales un nuevo papel en la interpretación, análisis y propuestas hacia nuevos procesos nacionales. Especialmente, en el desarrollo tecnológico DT, deberán ayudar a comprender en qué realidad y de qué manera, se deben incorporar los avances de la RIN. Y también a, definir cual tecnología es la más necesaria, partiendo de la base que se deben incorporar los conceptos de la IN y de las NT.

Esta es una manera de poder asimilar la interdependencia académica, como forma que valide la concepción INTERDISCIPLINARIA, hacia un nuevo modelo democrático, que deberá transitar de la representatividad a la participación.

Sin tener esta percepción global e integral del conocimiento, difícilmente lograremos apropiarnos de la RIN.

En síntesis, la forma como entendamos e incorporemos esta Revolución del Conocimiento, hará que una Cuarta esté liderada por nuestros países, ya que solo un nuevo contexto social, que no haya encontrado su oportunidad histórica, es el que puede crear una nueva perspectiva.

Esto implica mirar la universalidad como una serie de especificidades, para que nuestras particularidades las vivenciemos como otra especificidad, con distintos valores y potencialidades, pero con los mismos derechos de ser una estructura en sí y por sí misma, y no tratada como subcultura incorporada a una autodenominada "supercultura", que en muchos aspectos siempre la veremos como extraña.

Es asunto de dignidad, antes que de ideología.

2. LOS RETOS DE LA INFORMATICA ANTE 100 AÑOS DE INDUSTRIALIZACION DEPENDIENTE.

2.1 Pasado y presente de la dependencia tecnológica de AL. Sus efectos para el menor desarrollo endógeno de la IN.

Nadie tiene hoy la verdad revelada, ni en el Norte ni en el Sur. Ambos tenemos atisbos de verdad, hemos hecho análisis, algunos de los cuales son promisorios, pero no podemos aceptar como verdad revelada lo que se piensa en el Norte. Respeto mucho las ideas del Norte, pero no deben ser tomadas por su valor nominal (2).

En el capítulo anterior se indica que las dos grandes deficiencias de los modelos de desarrollo económico en AL, han sido los desestimulos a la innovación tecnológica endógena y, la incomprensión de la heterogénea estructura social de sus naciones.

Esto ha significado un proceso de industrialización dependiente, donde AL todavía es deficitaria en la producción de bienes intermedios y en especial de capital, lo que hace que los procesos de sustitución de bienes de consumo durables, intermedios y de producción, tengan altos índices de componentes importados. Afectando el ahorro interno y fragilizando nuestra posición en el intercambio comercial.

Por otro lado, esa comprensión de nuestra propia complejidad social, interpretada bajo concepciones ideológico-políticas externas, ha incidido para que nuestros países, no hubieran podido afirmar una identidad cultural, y en alguna medida influído para que nuestro papel en la División Internacional del Trabajo DIT, se haya prolongado como simples abastecedores de bienes primarios y de manufacturas tradicionales.

En lo interno, se ha conformado una estructura socio-económica basada en la demanda de un mercado elitista y arribista, lo cual ha limitado las posibilidades de la IT , fomentando la diversificación de manufacturas, intensificando un inconsciente consumismo que equilibre la no masificación de la producción, que beneficie a la gente de todos los ingresos.

En síntesis, ha sido una industrialización sin "Tecnologización", con mercados internos reprimidos que desestimulan el ahorro interno y la acumulación intensiva de capital, que representarían recursos a programas permanentes en I&D.

Estas reflexiones se hacen en la necesidad de CREAR modelos económicos propios, ya que la influencia de las ortodoxias aplicadas o adaptadas a nuestro contexto, no han logrado la estabilidad social en estos países, y que de manera obsecada han insistido en imponer modelos ajenos a la cultura y tradición latinoamericana.

Todo esto como consecuencia de nuestro desempeño en la DIT, que ha determinado submodelos económicos, formando estructuras profesionales submodeladas, para perpetuar el subdesarrollo (3).

Ese fué su erróneo desafío entre los años 50 y 70, que pretendía acelerar el crecimiento = DEPENDENCIA, y no el DESARROLLO = AUTODETERMINACION, para acercarla a los patrones de vida del mundo desarrollado y no a los propios.

Esta presentación general de los efectos causados por los modelos externos a nuestro continente, es para ambientar la discusión de que no existen modelos universales para contextos específicos, ni tampoco tiene como objeto hacer un ajuste de cuentas a un economicismo que en última instancia ha afectado a todas las sociedades del mundo.

Porque ya empieza a ser claro en los economistas del continente, que el desarrollo del análisis del subsistema económico debe ser parte del desarrollo de los otros subsistemas del complejo social. La labor del economista tenderá necesariamente a ampliar su alcance, a coordinarse con el trabajo de otros especialistas y culminar en la planificación social con características de elevada selectividad y de amplitud integral, en las áreas estratégicas que son objeto de las decisiones políticas (4).

Ya no es posible desarrollar modelos económicos en que los factores sociales, históricos, culturales e ideológicos son considerados como referencias exógenas.

Es necesario mostrar estudios económicos recientes con metodología desarrollada en AL, que son alternativas que complementan lo expresado anteriormente. Es el trabajo del grupo de 12 profesionales de distintas disciplinas, nacidos en ocho naciones y dirigido por el Economista chileno Manfred Max-Neef, Premio Noble Alternativo en Economía, quien en su reciente visita a Colombia nos dejó entre otras, las siguientes impresiones:

Los tipos de desarrollo aplicados hasta ahora, no han tenido otros resultados que cerrarnos la puerta para nuestro futuro. Los modelos dominantes son más teología que economía, si fracasa el modelo es porque "la realidad nos hizo trampa" y continúa la vigencia del mismo modelo (5).

Pero lo importante de esta experiencia es que optaron por usar sandalias en vez de mocasines, para caminar y convivir con los problemas, espectativas y realidades de los inmensos grupos económicos de menores ingresos, y producir una teoría que en definitiva aportará al diseño de nuevos modelos económicos, a partir de la realidad latinoamericana, que en de-

finitiva introducirá innovaciones a la Teoría Económica.

Pero, ¿qué tiene que ver esta crítica económica con la IN?

El miedo a la I&D tecnológica, impone una camisa de fuerza a la creatividad de los ingenieros latinoamericanos, para desarrollar una industria de software, prolongando tediosas y estériles especulaciones esotéricas, de cómo debemos hacer parte de la RIN.

Lo expresado en este capítulo, pretende insistir en la reflexión de que esta Tercera Revolución Industrial, obliga la formación MULTIDIMENSIONAL de todos aquellos que hacen parte de la RIN. Por lo cual, el tecnócrata latinoamericano, debe ser un ente político, actuante, deliberante y conocedor de nuestra realidad, para poder proponer políticas de diseño e implementación IN, ajustados a nuestras posibilidades económicas y a prioridades en ciertos sectores, que nos den un liderazgo, o al menos una respetabilidad en el contexto tecnológico mundial. Todo lo anterior, es en referencia al pasado inmediato y al presente.

Respecto al futuro, la necesidad de crear nuestras propias teorías de desarrollo, permitirán en definitiva que la Revolución Informática, sea asimilada, adaptada y desarrollada en nuestros propios países, donde el factor de la mayor o menor integración científica y tecnológica, nos permitirá producir particulares alternativas tecnológicas, donde el CAD, CAM, CIM, CAE, etc. serán diferentes en nuestros países y sus condiciones para su aplicación más flexibles.

No hay que olvidar, que para AL, esta revolución del conocimiento es una revolución de la creatividad.

No podemos seguir dejando en el campo teórico la discusión de las NT y de su transferencia. ¿Qué se suple el país si existen numerosos expertos en Tecnologías de punta, si nuestras industrias siguen alejadas de esos nuevos conocimientos, sino se alienta la transferencia, sin crear la infraestructura adecuada para la transmisión y adaptación de NT?
En ese sentido, la tecnología, su transferencia y el apoyo que demos a nuestros centros de investigación, será factor decisivo para lograr mejorar nuestra posición frente a los más avanzados tecnológicamente (6).

2.2 Desarrollo endógeno de la Informática. Futuro de AL.

En la comunidad científica y tecnológica latinoamericana, existe un consenso para integrar recursos humanos, económicos e infraestructuras, para ejecutar proyectos puntuales en I&D, que permitan desarrollar sectores productivos que represente a los países que participen, líneas tecnológicas autónomas. Esa puede ser la necesidad y la conciencia de

los investigadores. Pero esa no ha sido la conducta que ha identificado a la gran mayoría de políticos latinoamericanos. Que es también el caso colombiano. En alguna medida la falta de claridad sobre la importancia de la IT, para diversificar y complementar los desarrollos industriales de cada país, han sido motivos para la desintegración de proyectos integracionistas, o para relegarlos a estados críticos. Sin olvidar, que los diferentes procesos políticos vividos por la mayoría de las naciones latinoamericanas, han afectado y deteriorado esfuerzos mancomunados, menguándolos aún más con la implantación del modelo Neoliberal, que empieza a principios de los años 70 y desemboca en la gran crisis económica de la presente década.

Pero la historia quiere cambiar.

AL ha empezado a integrarse a través de proyectos políticos, como es el caso de Contadora y el Grupo de Apoyo, que no solo se reune para tratar los asuntos de paz en el continente, sino para llegar a acuerdos integracionistas político-económicos. Eso abre la esperanza que en los años 90 una nueva ideología aparezca en el continente: La AUTODETERMINACION.

Esto significa para la "Revolución de la Inteligencia" nuevas dimensiones, porque será cada vez un mayor propósito una industrialización soportada en la "Tecnologización", los acuerdos Bi o Multilaterales en proyectos de I&D, permitirán un progresivo proceso de autonomía tecnológica y de desarrollo autosostenido.

Porque en definitiva, las políticas de ajuste a corto plazo impuestas por las transnacionales financieras, no han representado la reducción de la deuda externa, por el contrario, su incremento. Pero si han relegado el bienestar de la mayoría de nuestras poblaciones. Por lo tanto, produciendo núcleos desestabilizadores teniendo que recurrirse a mecanismos coercitivos para el control de las justas demandas sociales.

La actitud de ajustes a corto plazo, no solo corresponde a la filosofía inmediatista del Neoliberalismo, sino a la desconfianza de los mismos PI en una importante recuperación económica a corto plazo, que impedirá reabrir sus mercados a nuestras exportaciones manufactureras y conservar el precio de nuestros productos básicos en niveles cada vez más distantes de los bienes tecnificados.

De esta manera tendremos menos divisas para importar NT, por lo tanto, menos recursos para proyectos permanentes en I&D. Lo cual postergaría las posibilidades de un desarrollo más integral de la Informática en AL.

Esto significa que la recesión será prolongada, el proteccionismo no hará significativas conseciones y el flujo de préstamos cada vez más condicionnado a la inseguridad que sienten los prestamistas.

El superavit comercial en bienes manufacturados se ha reducido de manera alarmante en AL, de 39.400 millones de dólares en 1984, se baja a 18.500 millones en 1986 (7).

Todo esto repercutirá al interior de nuestros países, y en definitiva se abrirá una nueva corriente deliberadora a las ortodoxias de izquierda o de derecha.

De igual manera tampoco podemos esperanzarnos en una Nueva División Internacional del Trabajo NDIT, así sea reestructurada por los mismos PI, sin contar con nuestras legítimas necesidades y aspiraciones, entre otras razones, por la mayor competitividad entre ellos mismos, y por el surgimiento de nuevas naciones industrializadas que cada vez adquieren una mayor autonomía tecnológica (Corea, Taiwan, Brasil, etc).

Esa NDIT, implicaría para AL un espacio de desarrollo CT, que en definitiva no le conviene a ningún interés económico del mundo. Pero a nosotros sí.

Pareciera que el mundo desarrollado estuviera destinado a pagar las consecuencias de su delirio materialista y expansionista. ¿Hasta dónde será factible que puedan hacer un alto y reconsiderar su rumbo? Difícil, ante el auge individualista de la libre acción, sin control en sus proyectos y sin control en sus operaciones.

De ahí, la importancia de poder cristalizar iniciativas como el "Proyecto de Alta Tecnología América Latina años 2.000", propuesto por Brasil y Colombia, a la Argentina, Mexico y Venezuela, para crear multinacionales que integren la región a la economía mundial, haciéndolas competitivas con base en una mayor capacidad tecnológica, identificando oportunidades y desafíos provocados por la nueva configuración de los mercados nacionales, regionales y mundiales, socializar el conocimiento a través de la capacidad científica, tecnológica e industrial y, finalmente estimular la iniciación de proyectos multinacionales latinoamericanos, orientados a la investigación, inversión, producción y comercialización.

Estos son los tipos de proyectos que a mediano y largo plazo, llevarán el problema del desempleo y de la pobreza en general, a niveles insignificantes. De resto, las inmediatistas acciones para hacerles frente, solo son respuestas coyunturales y no estructurales.

Pero el núcleo de nuestra dependencia, está en el atraso CT,

y la brecha que nos separa de los PI es en buena parte la razón de nuestra pobreza y atraso. De no enfrentar con recursos económicos, de no rediseñar los programas académicos que formen investigadores en proyectos interdisciplinarios, y directores de proyectos, con una visión MULTIDIMENSIONAL a partir de nuestras especificidades, nos rezagaremos definitivamente en incorporarnos a esta NRI, y nuestra capacidad creadora se sumirá en la cuenta regresiva de la incompetencia creadora, alienándola hasta una no vislumbrada oportunidad, porque esta NRI, no da espera. Y este es un reto del Estado, de los políticos, de la Universidad y de la industria ante el futuro.

Terminemos este capítulo haciendo algunos interrogantes.

¿Por qué, si la I&D es una prioridad en los PI, en los textos de formación de nuestras universidades, escritos en su mayoría en esos países, no aparece el Departamento de Investigación y Desarrollo de Proyectos en los organigramas para una industria, o para un instituto gubernamental?

¿Por qué el Estado no crea incentivos para la Investigación en las industrias?

¿Por qué si nuestro industrial es enamorado de la copia, no reproduce el reto y el esfuerzo de ser innovador?

Pero mientras estos y otros interrogantes se plantean y resuelven, veamos como la RIN nos ahorra tiempo y recursos para hacer Desarrollo Tecnológico en los centros de investigación en universidades a industrias.

3. LA REVOLUCION INFORMATICA Y DE LAS NUEVAS TECNOLOGIAS. ECONOMIAS Y RESPUESTAS A CORTO PLAZO EN PROCESOS DE I&D PARA PAISES CARENTES DE POLITICAS DE DT.

El argumento de una universalización a través de la IN y de las NT, tal como se conciben en los países desarrollados, no es más que un sofisma doctrinario del Centro para el control de la Periferia.

Pero al mismo tiempo, el afán de nuevos mercados y la mayor competencia entre los países que lideran la producción CT, nos permiten conocer de forma más pronta y menos reservada los avances en estas áreas.

Y es ahí donde AL tiene la oportunidad de apropiarse y desarrollar nuevas alternativas tecnológicas, que en su racionalización, debe incorporar los estados anteriores de producción, semiartesanal y mecanizado.

Ya que en nuestros países, ninguna de esas dos etapas, ha podido en su totalidad, ser trascendida por otra. La segunda que obedece a la anterior revolución industrial, nuestros países tampoco la han asimilado, o sea adaptado y desarrollado, por cuanto hemos carecido de políticas para el DT, y porque las formas de negociar la TT tampoco no han permitido una autonomía tecnológica.

Esta NRI, es la oportunidad de integrar los anteriores estados y de permitir su evolución, solo posible por el diseño y aplicación de PDT en proyectos puntuales de I&D. Y en este proceso, los recursos informáticos serán el soporte, permitiendo su desarrollo endógeno. O sea una IN latinoamericana.

De esta manera se logrará en alguna forma integrar las escalas de producción, prospectar nuevas estructuras operacionales y definir unas características de nuestra pequeña y mediana empresa, ya que los parámetros que definen la mini, pequeña, mediana y gran industria en los PI, no es la misma que nuestros mercados internos y que nuestro papel en la DIT nos permite.

Esto nos va demostrando, de cómo en nuestros países este nuevo paradigma científico-tecnológico, nos facilitaría ajustas nuestra infraestructura industrial, a proyectos de DT que se ajusten a las perspectivas de los mercados internos y externos, y por otro lado,sentarnos en la realidad para entender que las NT y la RIN, no pueden todavía eliminar en AL las economías de escala.

Pero mientras se diseñan y ponen en marcha PDT, el conocimiento, la creatividad y solución de nuestras necesidades, no pueden dar espera a que se tomen complejas decisiones en esta materia. Y AL quede cada vez más rezagada ante una demorada concientización, de que solo la I&D nos puede sacar de la pobreza, del atraso y de la dependencia.

3.1 Propuesta de modelo informático para la I&D en industrias de AL.

3.1.1 Los recursos informáticos y entre ellos el CAD/CAM, permiten desde ya a las industrias de nuestros países, y a los centros de investigación estatales y privados, adelantar un proceso de industrialización tecnologizada, más articulada, así la actual carencia de PDT, hayan convertido los escasos procesos de I&D, en instancias desarticuladas y por lo tanto muchas veces desaprovechadas.

En otras palabras, la IN nos permite agregar, clasificar y prospectar lo hasta ahora desarrollado. Esto lo deben hacer el Estado y el sector privado latinoamericano en proyectos locales bi y/o multilaterales.

Porque primero hay que saber que se tiene, para saber que se quiere.

Entre las prioridades para el sector industrial, está en hacer un inventario de aquellas experiencias en desarrollos tecnológicos "clandestinos" en las distintas escalas de producción, ya que en estas experiencias estaría la fuente de desarrollar una Revolución Tecnológica Alternativa en AL.

3.1.2 La concepción tradicional de los laboratorios para el DT , podrán reestructurarse, respecto a su concepción original. O sea, deberán ser infraestructuras MULTISECTORIALES con financiación mixta, para que los desarrollos individuales de cada industria se puedan articular entre distintos sectores y, que los sistemas de CAD sean el laboratorio donde cada empresa pueda desarrollar sus diseños con distintos niveles de respuesta, de acuerdo a la complejidad de la solución, o al grado de desagregación de un producto o sistema.

De esta manera, se eliminaría en un principio el laboratorio en cada pequeña y mediana industria PMI.

Esto quiere decir, micromodelos de Desarrollo Tecnológico Centro-Periféricos en cada país, o en programas sub o regionales.

En esta etapa, será necesario fomentar y proteger una industria de software y racionalizar el tipo de hardware que se importa o ensambla.

Pero la base para que esta etapa sea un núcleo dinamizador de conocimientos y desarrollos endógenos, tanto en el diseño y producción de bienes durables, intermedios y de capital, como en la producción del software,

debe, en primer lugar, soportarse en que es necesario tener la claridad de que la oferta y demanda de productos se debe hacer para mercados locales, que en su mayoría no necesitan del tipo de bienes de consumo durables que ahora inundan nuestro comercio.

Y que los bienes de capital para nuestro mismo mercado y para el de otros países especialmente del TM, no deben ser iguales a los que nos vienen de afuera. Solo pensando en todas estas instancias, AL podrá en su conjunto ser una nueva y diferente potencia industrial en los años 2.000.

3.1.3 El empleo de la IN en I&D, producirá resultados a corto plazo, y de acuerdo a como dobleguemos y desarrollemos nuestra industria de software en laboratorios multisectoriales, hará que también los resultados se logren a menores precios.

3.1.4 Cuando se habla de varios laboratorios multisectoriales, se está trascendiendo el tradicional esquema de desarrollo industrial por procesos SUCESIVOS y sectoriales, para pasar a programas de desarrollo SIMULTANEOS e intersectoriales, y esto porque la Revolución Informática lo permite, entre otras razones el CAD/CAM/CAE, facilitaría una mayor homogeneización e integración vertical y horizontal de los sistemas productivos y, una mayor selectividad de componentes normalizados.

Buscando e identificando nuestra realidad, definiendo y proponiendo nosotros mismos nuestra ubicación en el orden global, ser viable entender, diseñar y realizar proyectos proespectivos. La mira al futuro no puede ser a través de la mente y del lente de otros. Si así lo entendemos en nuestros procesos personales, ¿por qué también no se reivindica para la solución de nuestras aspiraciones nacionales y continentales?

3.1.5 El tener claridad de cómo estos países incorporarían la RIN y la NT, permitirá definir y articular el desarrollo industrial endógeno.

El análisis de sus incidencias en la generación de empleo, bajo el esquema de la multiplicación de productos de innecesario uso, tal como acontece y se agudiza en los PI de Occidente, como consecuencia de la versatilidad y flexibilidad de la IN, no puede aceptarse en estos países, como una verdad a reproducir. Esto quiere decir que la RIN, podría también entrañar graves riesgos a la estructura psíquica individual y colectiva. La dependencia es una limitante para que estos países puedan evaluar previamente, el impacto socio-económico de las nuevas innovaciones, Pero desde una posi-

ción positiva, esa misma versatilidad de la IN, nos permite diseñar bienes que respondan a las particularidades y prioritarias necesidades no solo de consumo final, sino en todas las etapas del desarrollo industrial de nuestros países.

Al adquirir una clara conciencia de lo que significa la "modernización" a partir de nosotros mismos, y no de lo que otros quieren que seamos, y que nunca podremos ser, creará las condiciones para replantar el consumo en las sociedades de AL.

En todo caso, sea cual fueren las políticas de DT, como por las que opten los distintos países de AL, será necesario pensar que sin un sólido engranaje del dispositivo institucional y orgánico, será difícil lograr un proceso tecnológico más acelerado.

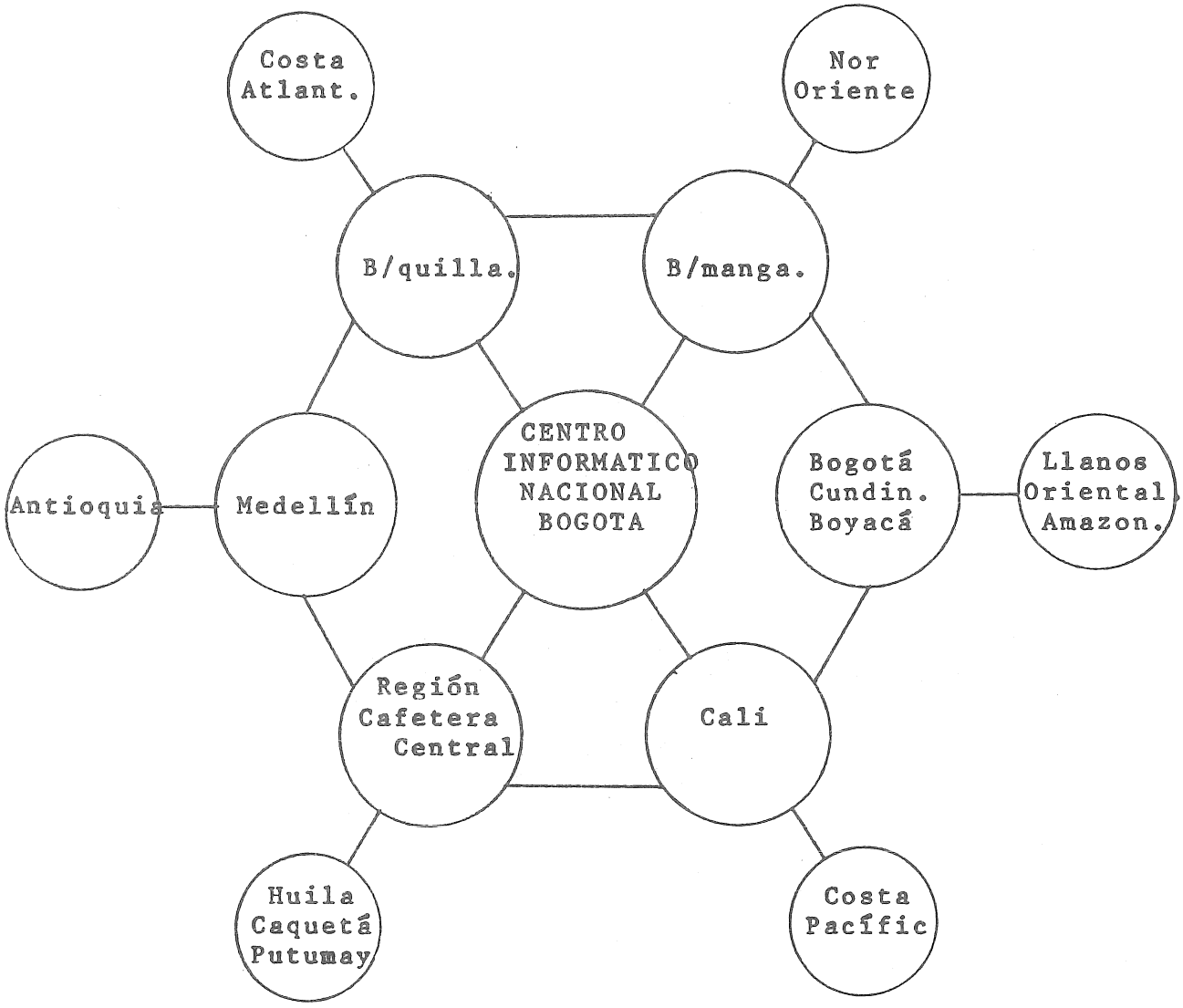
Esto quiere decir que la IT es proporcional al avance institucional orgánico. Donde habrá que reflexionar las razones por las cuales no hay una integración entre la gran empresa (en su mayoría en poder de las transnacionales) y la pequeña y mediana industria, lo cual también dificulta, que la microempresa expansiva no pueda ser dinamizada por la pequeña empresa. Este desligado proceso de industrialización, no ha sido común a los PI en sus etapas de industrialización. Por lo tanto, será un obstáculo al DT en AL. Limitando las posibilidades de la informática en su incidencia a una mayor integración de las escalas y al desarrollo del modelo propuesto en este capítulo.

MODELO MACRO NACIONAL

Segunda Fase

Finalizada la primera etapa podrían consolidar varios centros informáticos - Periféricos conectados a un centro Informático central, que posteriormente se integrará a Redes Bi o Multilaterales con países de AL.

Al consolidarse las redes nacionnales, estas entrarán en un principio a trabajar sobre aquellos sectores priorizados como estratégicos y viables de iniciar inmediatamente su desarrollo, que deberán responder a programas de cooperación científica y tecnológica entre diferentes países de AL.

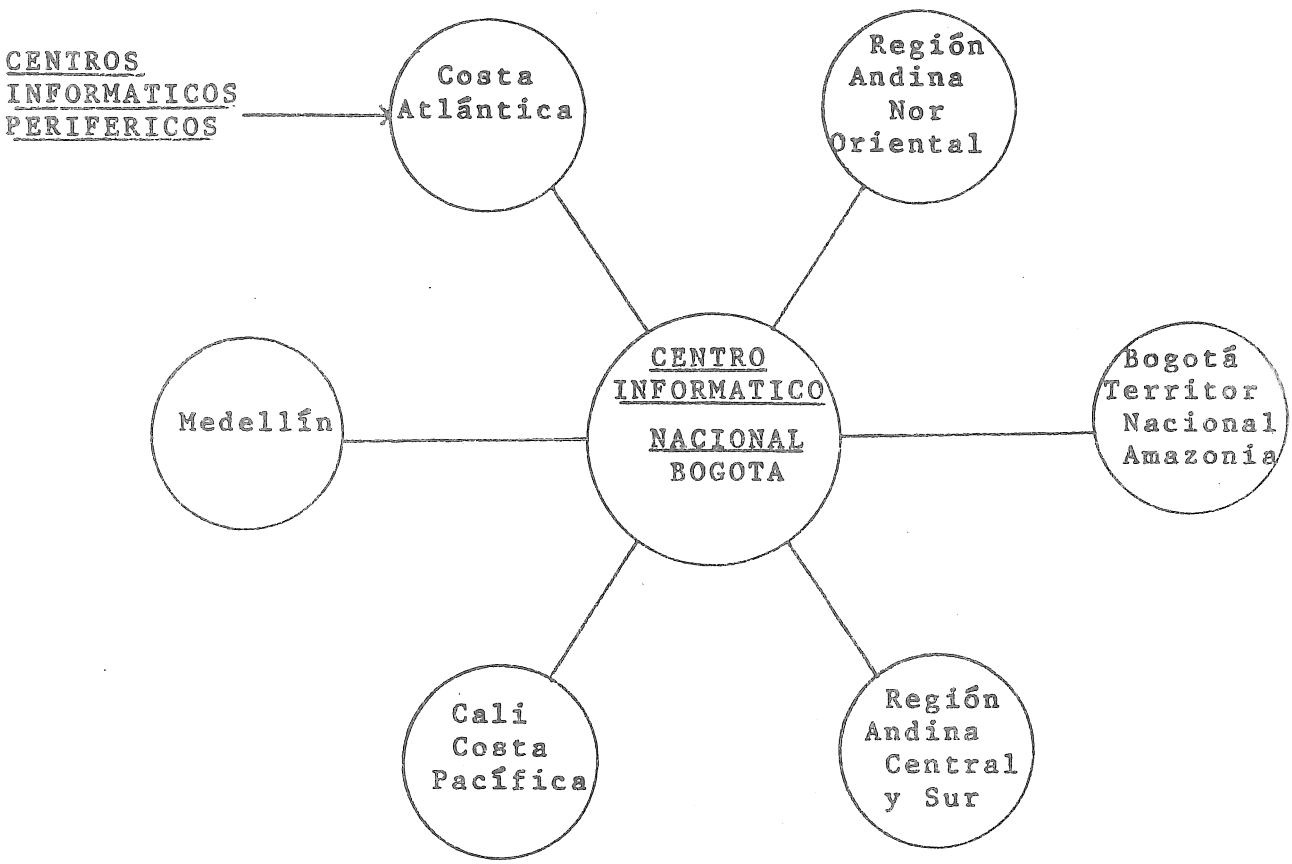


ESQUEMA DE UN SISTEMA INFORMATICO NACIONAL PARA EL DT.

(Ejemplo, Colombia)

MODELO MACRO PARA BASE DE DATOS (1.) Y DIAGNOSTICO (2.)

Primera Fase



Se recomienda hacer experiencias pilotos en tres zonas, con dintinto nivel de desarrollo socio-económico, diferente ubicación geográfica y distintas perspectivas de desarrollo económico:

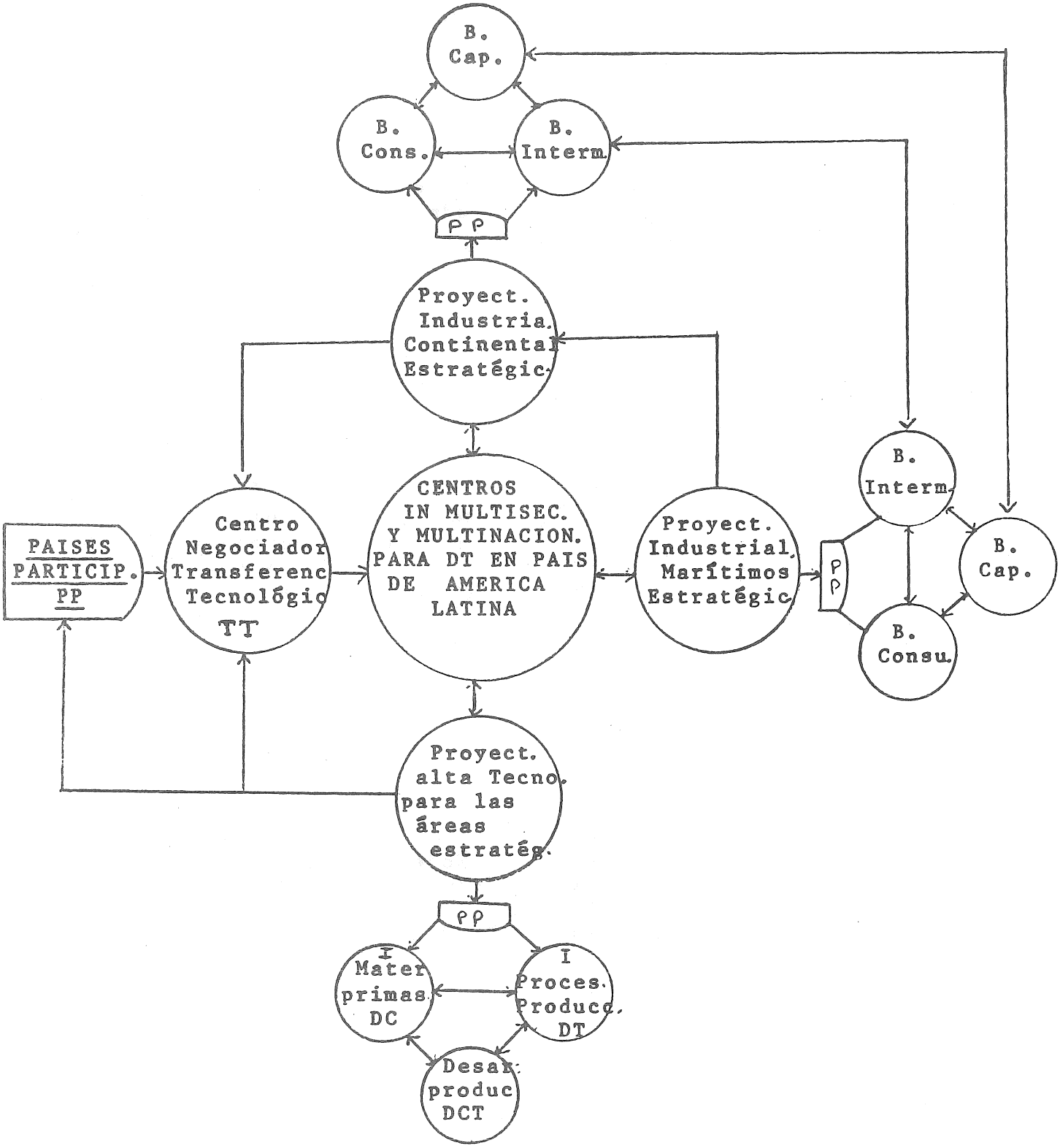
Costa Pacífica = Occidente
Medellín o Cali = Región Andina
Territorios Nacionales = Oriente

Esta experiencia sería únicamente para la primera y segunda etapa (base de datos y diagnóstico).

Realizadas las experiencias pilotos se deben complementar el sistema donde ya se desarrollarían las cinco etapas que conforman cada red nacional (1. a 5.). Para desarrollar el punto 3. ya deberá existir un primer acuerdo de las áreas de DT prioritarias en los países que intervienen en el programa.

ESQUEMA MODELO INFORMATICO PARA EL DT MULTINACIONAL

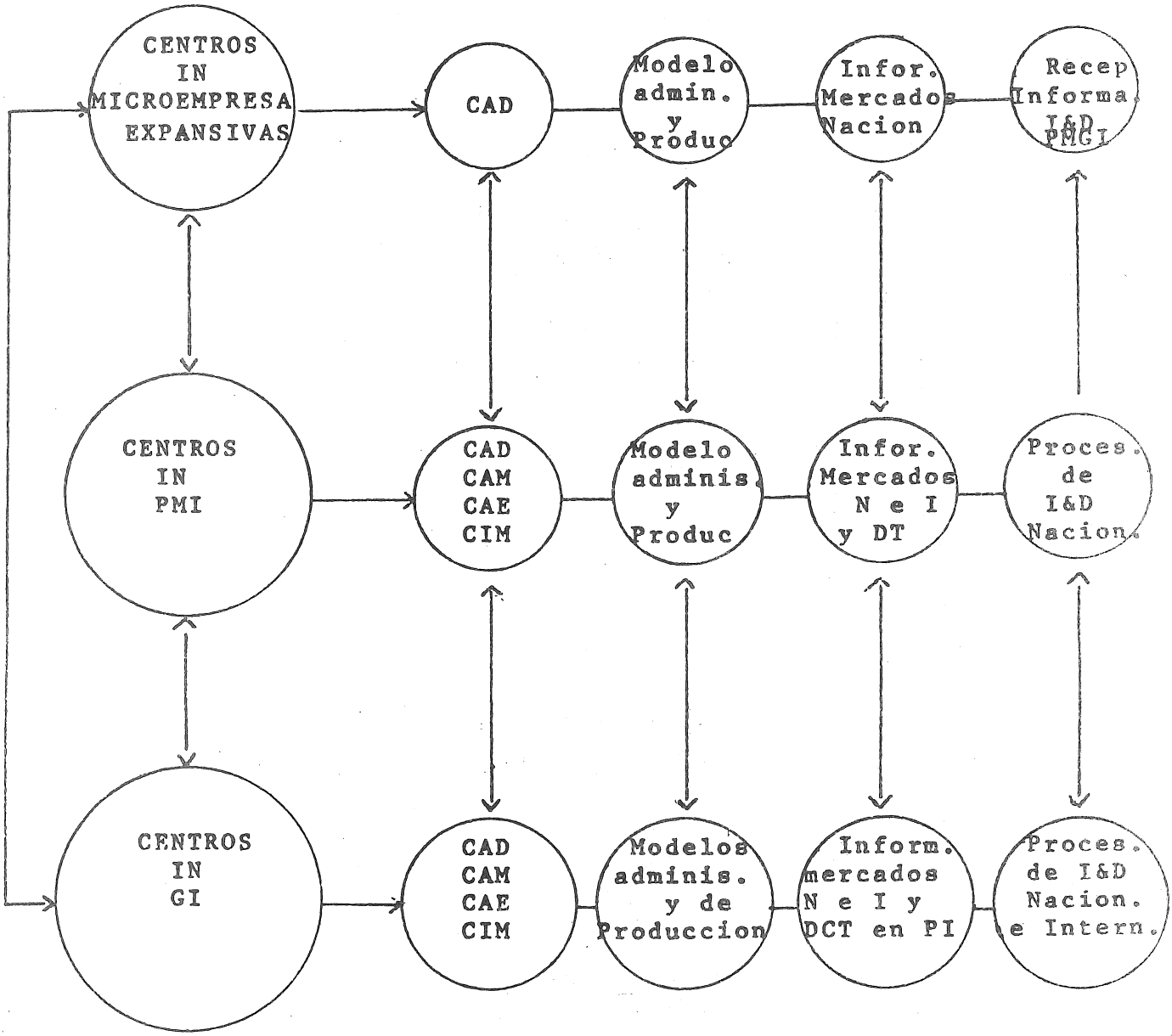
Tercera Fase.



MODELO MACRO DE CENTROS INFORMATICOS LOCALES PARA LA I&D EN LAS INDUSTRIAS DE AL.

Primera Fase - Sub y sectorial

Segunda Fase - Secto y multisectorial



3.4 Reflexiones al enfoque interdisciplinario para el DT en AL.

A lo largo de este documento se ha insistido en que el esquema reduccionista, unidimensional y de soluciones estructurales a través de disciplinas aisladas, obedece a una concepción académica que aceleradamente debemos ir superando.

Por esta razón, cualquier programa de políticas de Desarrollo Tecnológico PDT, debe incorporar no solo economistas, sino también, cientistas sociales de distintas áreas (antropólogos, historiadores, sociólogos, etc.), para la autocrítica y la búsqueda de nuevas posiciones para entender y orientar esquemas que integren nuestra heterogénea estructura social, como soporte fundamental del DT particular que debe enfrentar AL a largo plazo.

De igual manera, si los modelos de desarrollo económico han subestimado la IT a través de procesos propios de I&D, o sea obviado la capacidad creadora de científicos y tecnólogos latinoamericanos, los ingenieros no pueden estar al margen de cualquier proyecto de Prospectiva Tecnológica PT, ni en el diseño de PDT. Esto significa, que desde un principio los ingenieros deben participar en cualquier proyecto de esta índole. ¿Si no, cómo y quién puede orientar desde la realidad y posibilidades tecnológicas inmediatas,nacionales e internacionales, un proceso de DT más autónomo?. ¿En qué sectores podemos ejecutar desarrollos tecnológicos menos dependientes? ¿Quién señalaría las necesidades de infraestructura para procesos de IT escalonada en la industria?

Por lo tanto, si estos proyectos omiten al tecnócrata, corren riesgo de ser nuevos discursos retóricos de gran impacto teórico pero de escasa y casi nula incidencia práctica en lo económico, en lo político y en lo social.

Esto significa para los ingenieros, despertar a una nueva realidad y romper su marginamiento que en parte ellos han consentido.

En nuestros países se olvidan que los ingenieros han hecho las grandes revoluciones tecnológicas que han transformado el pensamiento económico y social de todas las sociedades del mundo y mucho más ahora con la actual RIN y de NT, que si en alguna medida no se desarrollan endógenamente, nos domestica y aliena cada vez más.

4. LOS MODELOS DE DESARROLLO INDUSTRIAL PARA EL TERCER MUNDO TM NO PUEDEN SER COMUNES A TODAS SUS REGIONES.

Existe la tendencia a homogeneizar los problemas y las soluciones para el TM, bajo la misma racionalidad de los indicadores económicos convencionales, que poco dicen del real bienestar de la mayoría de la gente de las nacionaes del TM, y en concreto de AL.

En esa medida, se olvida que el menor o mayor grado de desarrollo obedece a distintos procesos histórico-culturales y a distintos factores políticos, económicos y tecnológicos, que en un momento dado han permitido, o restringido el mayor o menor desarrollo. Y ese es el caso cuando se considera que latinoamérica debe reproducir el modelo de los países asiáticos. Y en definitiva, solo el esquema macro-general de Centros (Japón) - Periferia (Corea, Taiwan, Tailandia, etc.), se podría retomar.

A partir de ahí, un análisis más objetivo, menos dependiente, nos hará concluir que es imposible que nuestros países puedan reproducir el conjunto de esa experiencia.

Simplemente, el desarrollo asiático ha sido diferente al operado en los PI de Occidente. Sin desconocer, que el DCT de estos últimos ha sido el eje de su transformación industrial en cuanto a la técnica. Pero referente a la cultura (aquella verdad que subyace en cada ser del planeta, pero que se ha querido negar y/o deformar en AL), los países del Asia con su milenaria historia, supieran insertarla en su desarrollo industrial. Recuérdese los avances que han efectuado en el campo organizacional y operacional, para una mayor productividad de sus industrias. Y sus ingeniosas estrategias de comercio internacional, obedecen a la fragilidad de sus pobres y pequeños espacios. Esa limitación espacial, también es un factor que moldea la cultura de un pueblo. Y es posible que ahí esté su fuerza o su punto débil.

Ese mismo análisis, que diferencia Asia de Occidente, nos debe servir para entender sin antagonismos, que no podemos ser la Periferia de un Centro (Estados Unidos y Europa), que poco o nada se parece a nuestro complejo mundo cultural.

Y de igual manera, frente a los países asiáticos, donde la diferencia en la riqueza de los suelos y en el tamaño de los territorios, determinan diferenciados desarrollos.

Además, la coyuntura política y económica mundial después de la segunda guerra mundial y el tratamiento en la DIT a los países asiáticos, fué para ellos completamente diferente a la dada a AL.

Pero si en algo queremos imitar a cualquier modelo, sería en

que todos los PI apoyan a ingenieros y científicos para que desarrollen productos y aplicaciones en el campo de la alta tecnología, como lo vienen haciendo en AL, Argentina y Brasil.

Para concluir y tratando de mirar en el nuevo siglo, no es un despropósito pensar en que AL también debe desafiar el reto del diseño del hardware. Miremos por qué:

4.1 La actual recesión en la economía de los países de Occidente, ha empezado a afectar las exportaciones de los países asiáticos. Si estos no le hubieran dado importancia a la ICT, para desarrollar programas de I&D, por ejemplo, en la microelectrónica, estarían de pronto sujetos a la fragilidad de las economías latinoamericanas por haberse aventurado a una industrialización siempre dependiente. Corea y Taiwan, han destinado cientos de millones de dólares al diseño de Chips. Si no lo hubieran hecho, ¿qué podría suceder le a su industria microelectrónica, si solo se hubieran conformado con niveles de seudo TT?

4.2. Los costos de centros de CAD y CAM, que permiten un real desarrollo de productos, o sea, análisis estructural, ergonómico, simulaciones de uso, etc. cuestan en el mercado internacional entre 200 y 300 mil dólares para el CAD, y de 80 a 120 mil dólares para el CAM. Estos costos consideran el uso de microcomputadores y algún tipo de software (8).

Esto quiere decir, que el CAD para microcomputadores que se ofrece en la mayoría de países latinoamericanos, no permite un verdadero proceso de diseño. Esto es un reto a que primero hagamos una producción intensiva del software, al mismo tiempo planear las características del hardware requerido. Porque el CAD de aparente bajo precio que se vende ahora en Colombia, solo sirve como una versátil herramienta de dibujo y archivo de información, y no como un eficiente instrumento para la Innovación Tecnológica. Tampoco podemos esperar qué alcances tienen los cada vez más poderosos microcomputadores, para ver si con ellos podemos hacer I&D.

Como conclusión, no es un despropósito pensar en el desarrollo de desafíos tecnológicos cada vez más complejos. Simplemente miremos al Brasil.

3.2 MODELOS DE SISTEMAS INFORMATICOS NACIONALES PARA EL DESARROLLO TECNOLOGICO INTEGRAL DE AL.

CENTRO INFORMATICO INDUSTRIAL MULTISECTORIAL. Etapas de Desarrollo.

1. BASE DE DATOS MULTISECTORIAL. SOPORTE DEL DIAGNOSTICO (*)

Agrícola	(Sistemas tradicionales, etapa de mecanización, proyectos agroindustriales). Inventario de R. Naturales.
Industrial	(Producción de bienes de consumo durables, intermedios y de capital). Inventario R. Naturales.
Marítimo	(Sistemas tradicionales, sistemas industriales). Inventario Recursos Naturales.
Transporte	(Terrestre, marítimo, aéreo).

2. DIAGNOSTICO DEL DESARROLLO TECNOLOGICO SECTORIAL, IDENTIFICACION DE SECTORES ESTRATEGICOS PARA UN DESARROLLO INDUSTRIAL MULTISECTORIAL SIMULTANEO PARA LA PRODUCCION DE NUEVOS BIENES DE CONSUMO DURABLE, INTERMEDIOS, DE CAPITAL Y DESARROLLO DE UN SOFTWARE ENDOGENO PARA MERCADOS NACIONALES E INTERNACIONALES. PROYECTO DE EVALUACION INTERDISCIPLINARIA (Planificadores, economistas, ingenieros, arquitectos, diseñadores industriales, administradores de empresas, científicos sociales).

3. PROGRAMAS DE PROSPECTIVA TECNOLOGICA MULTISECTORIAL E INTERDISCIPLINARIO. Para el desarrollo articulado y simultáneo de sectores estratégicos identificados en 2.

4. EVALUACION SUB Y/O REGIONAL PARA IMPULSAR PROGRAMAS PUNTUALES DE DT.

5. CREACION DE UN SISTEMA DE INFORMACION MULTISECTORIAL DEL PROGRAMA DE PROSPECTIVA TECNOLOGICA (3.), PARA DESARROLLOS TECNOLOGICOS SIMULTANEOS. RED NACIONAL.

6. CONEXION A UN SISTEMA DE INFORMACION SUB Y/O REGIONALES.

7. PARTICIPACION EN LA ESTRUCTURA, FUNCIONAMIENTO Y FINANCIACION DE ESTOS CENTROS. Estado 70%, capital privado latinoamericano 30%.

8. FINANCIACION ESTUDIOS PRELIMINARES. Corresponden a los gobiernos de AL. de 1. a 6.

9. EJECUCION ESTUDIOS. Centros de Investigación privados e institutos gubernamentales.

(*) Consideración para la base de datos: esta se debe desagregar en: a) Desarrollos tecnológicos locales, incorporando la información correspondiente a tecnología "clandestina". b) Los procesos por TT de las transnacionales. Esto permitirá evaluar y prospectar la capacidad creadora endógena, hacia los sectores de desarrollo estratégico. Por otro lado, trazar nuevos tipos de conducta, para negociar con las transnacionales las formas de transferirnos su tecnología.

3.3 MICROMODELOS DE CENTROS INFORMATICOS PARA I&D EN LAS DISTINTAS ESCALAS DE PRODUCCION Y ARTICULAR UN DESARROLLO INDUSTRIAL.

Este esquema funciona a través de múltiples centros de CAD/CAM/CAE, como primer paso para el desarrollo de infraestructura o laboratorios de DT más complejos. Los cuales serán viables cuando países como Colombia y la gran mayoría de AL, diseñen e implementen serias políticas de DCT a mediano y largo plazo.

CENTROS INFORMATICOS PARA MICROEMPRESAS EXPANSIVAS (Financiación Mixta) (*).	Procesamiento y análisis de su estado financiero, ventas, etc. Sistemas de información para el mercado de sus productos. Sistemas de CAD para el diseño de productos e inducción al diseño industrial. Recepción información de la PMI, comercio, sobre normas de control de calidad, nuevas tecnologías que se producen o importan para su escala, etc.
CENTROS INFORMATICOS PARA LA I&D EN LA PEQUEÑA Y MEDIANA INDUSTRIA PMI. (Financiación Mixta) (*).	Recepción información de las microempresas expansivas. Recepción información de los desarrollos tecnológicos en otros países, especialmente en los industrializados recientemente. Análisis modelos de administración (evaluaciones interdisciplinarias) para observar el impacto de las NT y los modelos de gestión administrativa.
Sistemas de CAD/CAM	- Servicio para el diseño de productos de acuerdo a las necesidades de cada empresario. - Investigación y desarrollo de nuevos bienes de consumo, bienes de capital o componentes para estos, etc.
Procesos de IT	- Desarrollo de software para PMI-CAD/CAM/CAE. - Desarrollo de sistemas flexibles de CIM.

CENTROS INFORMATICOS DE LA GRAN INDUSTRIA (Financiación Mixta) (*)	- Información de la PMI y de las ME. - Información de los desarrollos tecnológicos en otros países del mercado mundial. - Análisis modelos de administración. - Procesos de I&D en el diseño de sistemas de CAD/CAM/CAE. orientados exclusivamente al diseño de bienes intermedios de mayor complejidad, de bienes de capital más tecnificados y a esquemas de CIM más avanzados. - Sistemas de información para dinamizar el desarrollo de las escalas menores. Apuntando a las perspectivas de sus necesidades futuras. De esta manera, podrán orientar el desarrollo de las mismas.

(*) Respecto a su financiación: para todos los casos, la participación del Estado es necesaria; y sus aportes diferenciados (%) para las tres escalas. Aclarando, que la financiación de los centros informáticos para la microempresa expansiva debe ser en su mayoría estatal y de la gran empresa. O inclusive, financiados únicamente por cualquiera de las dos partes.

C I T A S

(1) "El Mundo se volvió estrecho". EL TIEMPO, Lecturas Dominicales. Bogotá, Colombia, Junio 7 de 1987.

(2) Palabras del Doctor RAUL PREBISCH en su discurso del XXI período de Seciones de la CEPAL. Mexico, Abril 24 de 1986.

(3) Un mayor análisis de esta problemática hace el autor de este documento, en un escrito sobre "Los Modelos Económicos de América Latina en el siglo XX. ¿Dependencia o Desarrollo Tecnológico? Bogotá, Septiembre de 1987.

(4) La Planificación en economías Mixtas de Mercado y los Paradigmas del Desarrollo: Problemas y Alternativas. Trabajo del Doctor RENE VILLAREAL de Mexico. Revista de la CEPAL #31, Pag. 57, Abril de 1987.

(5) Utopía y Poder, Espacios y Tiempos de Participación. Doctor MANFRED MAX NEEF, Premio Nobel Alternativo de Economía . EL ESPECTADOR, Magazín Dominical #230. Agosto 23 de 1987. Pag. 17-20.

(6) "La Calidad y la Tecnología no son casos Fortuitos". Doctor FABIO ECHEVERRY CORREA. Presidente de la Asociación Nacional de Industriales de Colombia - ANDI. Declaraciones para el Diario LA REPUBLICA. Julio 8 de 1987.

(7) CEPAL muestra preocupación por situación mundial. Estudio presentado en su XIX Asamblea. Nueva York, Agosto de 1987.

(8) KARL H. PLAETZER. Asesor en Planificación Industrial para países en Desarrollo. Revista ALTA GERENCIA, #22, Bogotá - Colombia, Mayo de 1987.

MODELOS CURRICULARES EN INFORMATICA-COMPUTACION

Victoria Bajar. Instituto Tecnológico Autónomo de México.
Guillermo Levine. Universidad Autónoma Metropolitana (U. Iztapalapa)

Coordinadores del Comité de Modelos Curriculares de la Asociación Nacional de Instituciones de Educación en Informática - ANIEI MEXICO

Resumen

Este trabajo presenta el resultado del COMITE DE "MODELOS CURRICULARES, NIVEL LICENCIATURA , INFORMATICA-COMPUTACION" de la ANIEI (Asociación Nacional de Instituciones de Educación en Informática), fruto de la labor de un grupo de representantes de numerosas instituciones educativas.

La motivación de la elaboración de Modelos Curriculares radica en la imperiosa y urgente necesidad de establecer cómo debemos formar y qué deben saber nuestros egresados en Informática y Computación. El desarrollo nacional en las disciplinas mencionadas exige de la comunidad educativa un esfuerzo para definir, normar e implantar buenos curricula de estudio con rigor, conocimiento, creatividad e ingenio.

Se presenta así un "catálogo exhaustivo" de áreas, subáreas y subsubáreas de conocimiento en Informática y Computación, que "se cruzan" en forma porcentual con cuatro perfiles tipo: Lic. en Informática, Lic. en Sistemas y Computación e Ing. en Comunicaciones y Sistemas Digitales.

1-.Motivación y Antecedentes

Cuando se considera la imperiosa necesidad que existe de aportar elementos propios para el desarrollo nacional de Informática y Computación, surgen multitud de inquietudes, ideas y proyectos, que corresponden a los dominios que se plantean a continuación:

- Conformación de criterios y líneas para el logro de una buena administración informática en general y de recursos de cómputo, en particular;
- Manejo de parámetros realistas a la vez que consistentes para adquisición, selección y configuración de los equipos de cómputo;
- Orientación del potencial humano en el área de programación hacia obras sólidas de ingeniería de software;
- Desarrollo de sistemas, cada vez más variados, más versátiles, para aplicaciones de disciplinas diversas con el fin de convertir en realidad "las promesas" de la Computación y la Informática;
- Aportes y realizaciones en múltiples y diferentes campos del control industrial;
- Implantación de facilidades de teleproceso, de fundamental relevancia en una sociedad que desea comunicarse, para avanzar y evolucionar.

Para alcanzar metas como las consideradas es de importancia medular formar cuadros de profesionales sólidamente preparados, ya que son las vías del estudio y la investigación las que permitirán llegar al nivel de desarrollo independiente que nuestro país requiere. La inquietud de la buena formación profesional existe desde hace tiempo en la comunidad educativa nacional, constatándose múltiples esfuerzos en las instituciones de educación para satisfacerla.

Sin embargo, esos esfuerzos no son siempre convergentes, resentimos una falta de definición y cohesión, nos aqueja una no muy clara escala de prioridades, falta aún consolidar con todo rigor un plan general de desarrollo tecnológico en Computación e Informática y nuestros planes de estudio en estos campos adolecen en muchos casos de cierta despersión, de algunas carencias o de ambiguedades que debilitan la formación de profesionales.

Como ya se dijo, la inquietud relativa a estos últimos aspectos, no es nueva; se remonta a noviembre de 1981 el primer intento significcativo de realizar un aporte formal, en el seno del Primer Foro sobre Formación de Recursos Humanos en Informática, convocadopor la Secretaría de Programación y Presupuesto y la Secretaría de Educación Pública y celebrado en Monterrey, N.L. Surgió así, como resultado del Comité integrado en dicha reunión, el REPORTE DE "DEFINICION DE LA INFORMATICA" (1). Un año más tarde, fruto del Segundo Foro sobre Formación de Recursos Humanos en Informática, convocado por las mismas secretarías (Mexicali, B.C., abril de 1982), la idea inicial cobró más forma, llegandose así al REPORTE DE "DEFINICION DE LOS CAMPOS DE ACCION DE LA INFORMATICA" (2).

Constituída la ANIEI (Asociación Nacional de Instituiciones de Educación en Informática) en Guadalajara el 8 de octubre de 1982, fue su Segunda Asamblea General celebrada en Mérida en junio de 1983, el foro en el que se hizo hincapié en que la ausencia total a nivel nacional, de la definición de un núcleo básico de conocimientos y funciones que deter-

mine con precisión qué debe saber y qué debe acer un profesional de la Computación o de la Informática era una deficiencia que debía supoerarse con urgencia,en los niveles de enseñanza que son del dominio de la ANIEI: superior, medio superior y postgrado. Para abordar el primero, nivel superior, se formó el COMITE DE "MODELOS CURRICULARES, NIVEL LICENCIATURA, INFORNATICA-COMPUTACION",que trabajó desde etonces hasta la fecha, integrado por veintitrés profesores de doce escuelas.

El Comité produjo reportes parciales y resultados que fueron discutidos en múltiples reuniones de trabajo y puestos a consideración y aprobados en las Asambleas Generales de la ANIEI de cd. Victoria, Tamps.(noviembre de 1984) de Guadalajara, Jali.(noviembre de 1985) y de Toluca, Edo. de Mex.(octubre de 1986), dando así forma a los MODELOS CURRICULARES(3)(4) que aquí se presentan.

2- Modelos Curriculares

El trabajo consta fundamentalmente de tres partes principales:

- La definición de cuatro perfiles "tipo" de profesionales en Informática y Computación.
- La formulación de un catálogo de áreas de conocimiento en estos campos del saber, y
- El cruce de áreas y perfiles, bajo la forma de una ponderación porcentual de los temas de estudio, con el fin de definir los conocimientos necesarios en cada perfil.

La metodología seguida en la elaboración de los modelos consistió en:

- Determinación de un marco conceptual sólido, dado fundamentalmente por la delimitación de las áreas de concimiento y de los campos de acción del quehacer informático, en su expresión más amplia, sin excluir ninguna actividad relevante; pero sin caer en un relativismo de adición injustificado;

- Ceneración de información: recopilación, organización y síntesis de áreas de concimiento, incluyendo temas, campos de aplicación, aspectos de investigación y desarrollo, casos de interés, bibliografía, etc.;

- Clasificación y análisis de la información previamente producida;

- Intercambio, discusión de ideas y conciliación de criterios en términos de la estructura conceptual que sirve de marco, para llegar a formulaciones suficientemente consistentes y de concenso.

La complejidad intrínseca de la tarea, aunada a la dispersión geográfica introdujo una serie de atrasos y de reorganizaciones de los grupos de trabajo, aunque se conservó siempre el mismo núcleo y se reafirmó, a lo largo del tiempo, delineándose cada vez con mayor nitidez, la esencia del proyecto.

2.1- Perfiles Profesionales

Corresponden a cuatro dominios de desarrollo profesional en Informática y Computación, que se identifican por los siguientes perfiles profesionales:

- LIC. EN INFORMATICA
- LIC. EN SISTEMAS COMPUTACIONALES
- ING. EN COMPUTACION
- ING. EN COMUNICACIONES Y SISTEMAS DIGITALES,

cuyas definiciones emanadas de la IV Asamblea Anual General de la ANIEI [5] y complementadas por explicaciones suplementarias, son:

LIC. EN INFORMATICA: profesional que en forma interdisciplinaria analiza y sistematiza la información, con el fin de alcanzar los objetivos organizacionales. Debe tener los conocimientos necesarios sobre las organizaciones: estructura, operación, necesidades de información, alcances y objetivos, que le permitan evaluar, seleccionar e implantar sistemas computacionales, así como organizar servicios y administrar recursos informáticos.

LIC. EN SISTEMAS COMPUTACIONALES: profesional capaz de diseñar, desarrollar e implantar sistemas para administrar información útil en la toma de decisiones usando equipo computacional, diseñando métodos y procedimientos que contribuyan a optimizar los recursos de la empresa. Debe conocer metodologías y facilidades para el desarrollo general de software y de sistemas en particular, considerando entre otros: sistemas de información, aspectos de intercomunicación de equipos, interacción con sistemas operativos, teleprocesamiento.

ING. EN COMPUTACION: profesional capaz de diseñar y desarrollar sistemas complejos de software base, de especificar arquitecturas de hardware y de desarrollar -si corresponde- software de aplicación, generando tecnología nacional. Fundamentalmente, debe ser capaz de concebir y diseñar nuevos lenguajes de programación, de construir traductores, de diseñar e implantar sistemas operativos, de diseñar y construir manejadores de bases de datos; de definir, diseñar y elaborar paquetes específicos; y de diseár y desarrollar las metodologías y facilidades necesarias.

ING. EN COMUNICACIONES Y SISTEMAS DIGITALES: profesional capaz de diseñar, construir, instalar, operar y mantener sistemas digitales e interfaces para aplicaciones en computación, teleinformática y control industrial, generando tecnología nacional. Debe ser capaz de concebir, diseñar y construir arquitecturas de hardware, tanto para satisfacer definiciones de funcionalidad como para crear dispositivos electrónicos de fines específicos, debe ser capaz de concebir, diseñar y construir sistemas de transmisión y comunicación de infoprmación.

--*--

Mientras los dos primeros perfiles definen profesionales que se orientan a los usuarios directamente, los dos últimos corresponden a profesionales estrechamente vinculados a los equipos, Y, mientras unos se llaman licenciados, los otros ingenieros. El por qué de estas diferencias de

títulos puede ser objeto de largas y profundas discusiones, que no corresponde abordar aquí. Conviene simplemente seguir la tradición de nombres usuales, por un lado, y recordar las definiciones que da el diccionario de la Real Academia Española [6], por el otro.

LICENCIATURA: grado de licenciado; estudios necesarios para obtener este grado.

LICENCIADO: dícese de la persona que se precia de entendida; persona que ha obtenido en una facultad el grado que le habilita para ejercerlo.

FACULTAD: aptitud, potencia física o moral; poder, derecho para hacer alguna cosa; en las universidades, cuerpo de doctores o maestros de una ciencia; licencia o permiso,...

INGENIERIA: arte de aplicar los conocimientos científicos a al invención, perfeccionamiento o utilización de la tónica industrial en todas sus determinaciones.

INGENIERO: el que discurre con ingenio las trazas y modos de conseguir o ejecutar una cosa; el que profesa la ingeniería.

2.2 Areas de concimiento

Son cinco:

- ENTORNO SOCIAL
- HARDWARE
- MATEMATICAS
- SOFTWARE DE BASE
- SOFTWARE DE APLICACIONES

Cada una de estas áreas está dividida en subáreas, y éstas a su vez en subsuáreas; cada área (subárea o subsuárea) tiene una función específica en el quehacer profesional, lograda a través de sus objetivos explícitos. Cada subárea (o subsuárea) consta además de un grupo de temas de estudio, que podrían ser considerados como materias, parte de materias o grupos de materias según el plan de estudio al que pertenezcan.

El catálogo de áreas de conocimiento es exhaustivo, está fuertemente estructurado y se presenta complementado con un conjunto grande de referencias bibliográficas.

A continuación se muestran en forma esquemática las áreas de conocimiento:

1- **ENTORNO SOCIAL**: conocimientos, normas, experiencias y motivaciones que derivan en la buena integración de las unidades de informática de la sociedad.

1.1- Administración Informática:
Introducción a la administración informática
Administración del desarrollo de proyectos informáticos
Administración de unidades de informática

Aspectos psicológicos incidentes en el desarrollo de software
Auditoría informática

1.2- Aspectos financieros:
Introducción a la contabilidad y finanzas
Costos en informática
Estudios financieros para la adquisición de equipos de cómputo
Costos del desarrollo de software

1.3- Recursos humanos:
Enfoque sistemático de la administración de recursos humanos
Administración de personal

1.4- Política y legislación en informática:
Los derechos humanos frente a la actividad informática
La tecnología como un bien económico
Sistemas de información jurídica inherentes a la informática
Implicaciones legales en las unidades de informática
Políticas

1.5- Relación hombre-máquina:
Interfaz con el usuario
Ambiente
Sistemas computarizados en la oficina y en las organizaciones

1.6- Sociología de la informática:
Impacto de la informática
Informatización de la sociedad

2- **HARDWARE**: estudio de la teoría, técnicas y metodologías el funcionamiento de las computadoras; así como el diseño, construcción y prueba de subsistemas digitales.

2.1- Diseño lógico:
Sistemas de numeración
Algebra booleana
Componentes y tecnología
Métodos de análisis y diseño

2.2- Arquitectura de computadoras:
Historia de los equipos de cómputo
Estructura y subsistemas de una computadora tradicional
Microprocesadores
Microprogramación y programación en lenguaje ensamblador
Arquitecturas especiales

2.3- Teleprocesamiento, redes de computadoras y procesamiento distríbuido:
Teoría de la información
Medios de comunicación
Transmisión de datos y técnicas de codificación
Subsistemas de comunicación digital
Topología de redes
Estructura lógica del funcionamiento de redes

2.4- Electrónica:
Principios de los circuitos electricos y magnéticos
Principios de dispositivos electrónicos
Análisis y caracterización de circuitos integrados
Métodos de conversión A/D y D/A

2.5- Control digital:
Elementos de la teoría de los sistemas y el control
Muestreo, captura de información y servomecanismos
Aplicaciones

2.6- Instalaciones:
Instalaciones para equipo de cómputo

3- **MATEMATICAS**: Base formal que permite el desarrollo de habilidades de abstracción, de alto valor formativo, así como fuente de conocimientos específicos.

3.1- Matemáticas básicas:

3.1.1- Cálculo:
Cálculo diferencial e integral en una variable
Cálculo diferencial e integral en varias variables
Ecuaciones diferenciales

3.1.2- Algebra:
Algebra lineal
Complejos y polinomios
Estructuras algebraicas

3.1.3- Probabilidad y estadística:
Elementos de probabilidad y estadística

3.1.4- Lógica y conjuntos:
Elementos de lógica y conjuntos

3.2- Matemáticas aplicada:

3.2.1- Probabilidad y Estadística:
Probabilidad y estadística

3.2.2- Cálculo número:
Métodos números

3.2.3- Simulación:
Simulación

3.2.4- Investigación de operaciones:
Programación lineal
Análisis de redes
Programación dinámica
Programación entera
Programación no lineal

3.3- Teoría de la Computación:

3.3.1- Matemáticas discretas:
Conjuntos, relaciones y análisis combinatorio
Lógica matemática

3.3.2- Lenguajes y automatas:
Gramáticas formales
Automatas
Computabilidad

3.3.3- Teoría de grafos:
Introducción a la teoría de grafos

4- **SOFTWARE DE BASE**: teoría, técnicas y metodologías para la elaboración de algoritmos computacionales; para la concepción, análisis y diseño de sistemas computacionales; y para la comprensión y construcción de las componentes básicas del software de un equipo de cómputo:

4.1- Algoritmica:
Historia de la computación
Introducción a la algoritmica
Programación básica
Estructuras de información
Comparación de lenguajes de programación
Diseño de lenguajes de programación
Lenguajes especializados
Teoría de algoritmos
Algoritmos de clasificación y búsqueda

4.2- Ingeniería de Software:
Análisis de sistemas
Control de proyectos de programación
Diseño estructurado
Estándares y documentación
Técnicas especiales para manejo de archivos
Control de calidad de productos de software

4.3- Programación de sistemas:
Cargadores
Ensambladores
Macroprocesadores
Compiladores
Tópicos especiales en compilación
Intérpretes
Sistemas Operativos
Utilitarios

5- **SOFTWARE DE APLICACIONES**: herramientas teóricas y técnicas destinadas a aplicaciones de relevancia en el presente y en el futuro próximo.

5.1- Sistemas de Información:

5.1.1- Cuestiones generales:
Enfoques de desarrollo de sistemas

Planeación y programación de proyectos
Análisis de sistemas de información
Implantación de sistemas de Información

5.1.2- Bases de datos:
Diseño de bases de datos
Organización lógica y física de bases de datos
Uso de bases de datos

5.2- Graficación y diseño:

5.2.1- Graficación:
Principios de Hardware
Elementos Matematicos
Componentes de un paquete gráfico simple
Algoritmos de recorte
Entrada gráfica
Estructuración de la imagen
Estándares en computación gráfica
Gráficas en tres dimensiones

5.2.2- Diseño:
Fundamentos de CAD

5.3- Inteligencia Artificial:

5.3.1- Aprendizaje, deducción y solución de problemas:
Programación automática
Deducción
Aprendizaje
Procesamiento de lenguaje natural
Resolución de problemas
Representación del conocimiento
Programas
Lenguajes

5.3.2- Robótica:
Introducción a la robótica
Tecnología de robots
Control y programación de robots

5.3.3- Sistemas expertos:
Sistemas expertos

5.3.4- Visión:
Procesamiento de imágenes por computadora.

--*--

2.3- Cruce de áreas y perfiles

La matriz que se presenta en la tabla 1 expresa la ponderación porcentual -%- de cada área (subárea y subsuárea, si corresponde) y la ponderación respecto de un total de 40 materias -m40- en cada uno de los perfiles. Nótese que en ella no hay ceros, porque los temas previstos

Tabla 1- Cruce de áreas y perfiles

AREAS, SUBAREAS, SUBSUBAREAS (*) / PERFILES			LICENCIADO EN INFORMATICA						LIC. EN SISTEMAS COMPUTACIONALES						INGENIERO EN COMPUTACION						ING. EN COMUNICACIONES Y SIST. DIGITALES					
			%			m40			%			m40			%			m40			%			m40		
			a.b.c	a.b	a	a.b.c	a.b	a	a.b.c	a.b	a	a.b.c	a.b	a	a.b.c	a.b	a	a.b.c	a.b	a	a.b.c	a.b	a	a.b.c	a.b	a
1	1.1		-	9.0	33.5	-	3.6	13.4	-	4.5	27.5	-	1.8	11.0	-	0.25	5.0	-	0.1	2.0	-	1.0	3.75	-	0.4	1.5
	1.2		-	6.0		-	2.4		-	11.0		-	4.4		-	0.5		-	0.2		-	1.0		-	0.4	
	1.3		-	5.0		-	2.0		-	4.5		-	1.8		-	0.25		-	0.1		-	0.75		-	0.3	
	1.4		-	4.8		-	1.92		-	2.0		-	0.8		-	0.5		-	0.2		-	0.5		-	0.2	
	1.5		-	5.4		-	2.16		-	4.0		-	1.6		-	2.5		-	1.0		-	0.25		-	0.1	
	1.6		-	3.3		-	1.32		-	1.5		-	0.6		-	1.0		-	0.4		-	0.25		-	0.1	
2	2.1		-	1.25	8.75	-	0.5	3.5	-	1.0	9.5	-	0.4	3.8	-	5.0	22.5	-	2.0	9.0	-	5.0	46.5	-	2.0	18.6
	2.2		-	2.5		-	1.0		-	2.5		-	1.0		-	5.0		-	2.0		-	6.0		-	2.4	
	2.3		-	2.5		-	1.0		-	3.75		-	1.5		-	4.5		-	1.8		-	12.5		-	5.0	
	2.4		-	0.5		-	0.2		-	1.0		-	0.4		-	5.0		-	2.0		-	10.0		-	4.0	
	2.5		-	0.5		-	0.2		-	0.25		-	0.1		-	2.5		-	1.0		-	11.0		-	4.4	
	2.6		-	1.5		-	0.6		-	1.0		-	0.4		-	0.5		-	0.2		-	2.0		-	0.8	
3	3.1	3.1.1	5.0	12.5	22.5	2.0	5.0	9.0	5.0	13.0	19.75	2.0	5.2	7.9	5.0	13.33	27.5	2.0	5.33	11.0	7.5	17.0	24.5	3.0	6.8	9.8
		3.1.2	2.5			1.0			5.0			2.0			5.0			2.0			7.5			3.0		
		3.1.3	2.5			1.0			2.5			1.0			1.66			0.66			1.0			0.4		
		3.1.4	2.5			1.0			0.5			0.2			1.66			0.66			1.0			0.4		
	3.2	3.2.1	2.25	6.0		0.9	2.4		0.75	3.75		0.3	1.5		1.66	8.33		0.66	3.33		1.0	3.0		0.4	1.2	
		3.2.2	0.75			0.3			1.0			0.4			2.5			1.0			1.0			0.4		
		3.2.3	1.5			0.6			1.0			0.4			1.66			0.66			0.5			0.2		
		3.2.4	1.5			0.6			1.0			0.4			2.5			1.0			0.5			0.2		
	3.3	3.3.1	1.5	4.0		0.6	1.6		1.0	3.0		0.4	1.2		1.66	5.83		0.66	2.33		2.5	4.5		1.0	1.8	
		3.3.2	1.25			0.5			1.0			0.4			2.5			1.0			1.0			0.4		
		3.3.3	1.25			0.5			1.0			0.4			1.66			0.66			1.0			0.4		
4	4.1		-	4.25	11.25	-	1.7	4.5	-	8.75	17.25	-	3.5	6.9	-	12.5	30.0	-	5.0	12.0	-	5.0	8.5	-	2.0	3.4
	4.2		-	4.0		-	1.6		-	6.0		-	2.4		-	7.5		-	3.0		-	2.5		-	1.0		
	4.3		-	3.0		-	1.2		-	2.5		-	1.0		-	10.0		-	4.0		-	1.0		-	0.4		
5	5.1	5.1.1	7.5	17.5	24.0	3.0	7.0	9.6	12.0	22.0	26.0	4.8	8.8	10.4	2.5	5.0	15.0	1.0	2.0	6.0	2.25	4.75	16.75	0.9	1.9	6.7
		5.1.2	10.0			4.0			10.0			4.0			2.5			1.0			2.5			1.0		
	5.2	5.2.1	0.5	2.5		0.2	1.0		0.5	2.0		0.2	0.8		2.0	2.5		0.8	1.0		0.5	1.0		0.2	0.4	
		5.2.2	2.0			0.8			1.5			0.6			0.5			0.2			0.5			0.2		
	5.3	5.3.1	2.0	4.0		0.8	1.6		0.5	3.0		0.2	0.8		2.5	7.5		1.0	3.0		1.0	11.0		0.4	4.4	
		5.3.2	0.5			0.2			0.5			0.2			1.25			0.5			0.5			0.2		
		5.3.3	1.0			0.4			0.5			0.2			2.5			1.0			1.0			0.4		
		5.3.4	0.5			0.2			0.5			0.2			1.25			0.5			0.5			0.2		

(*) denotadas por a. a.b. y a.b.c respectivamente

MODELOS CURRICULARES. INFORMATICA-COMPUTACION

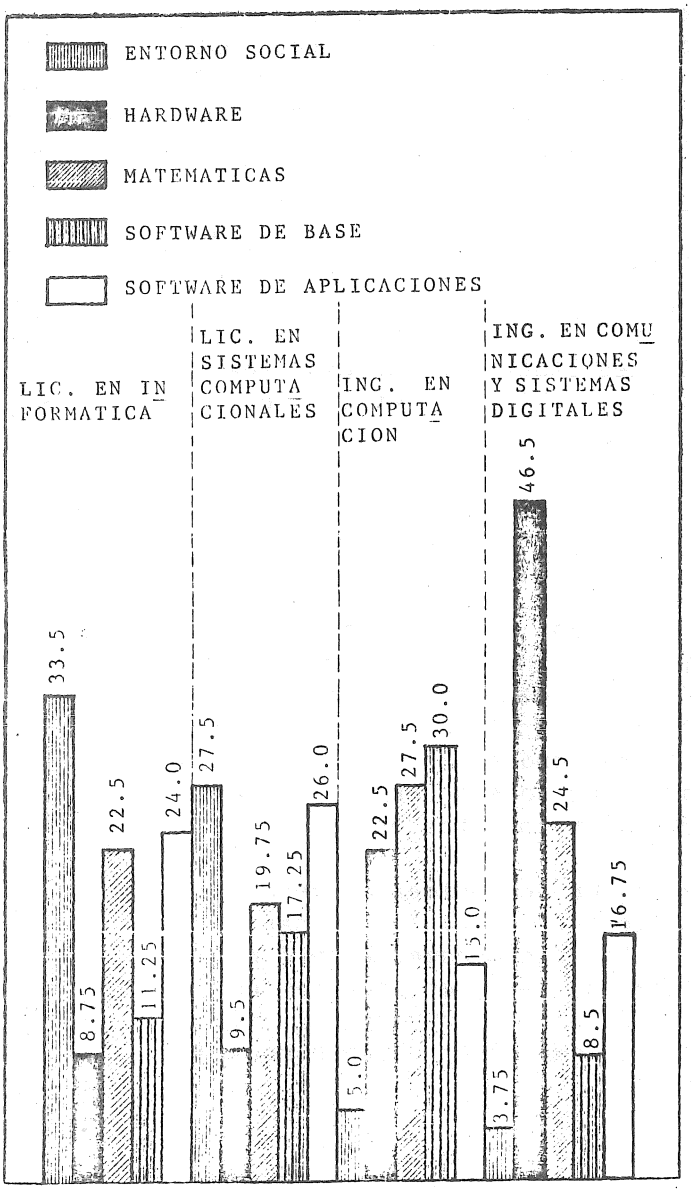


Fig 1- Distribución de áreas según perfiles

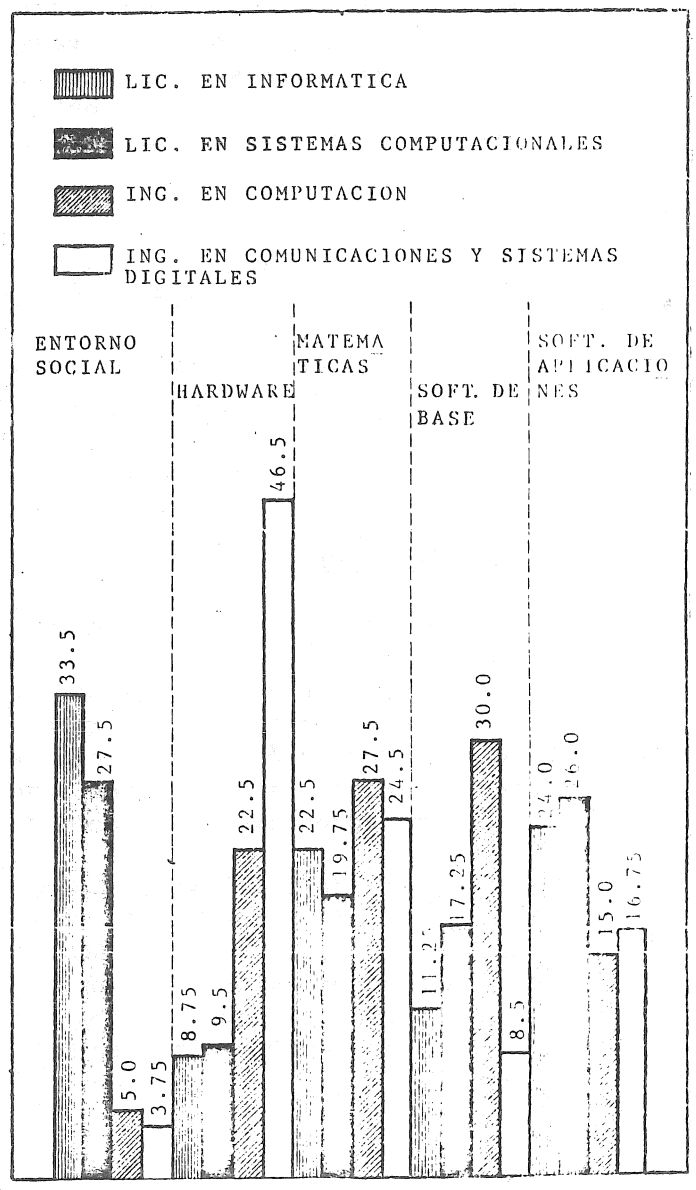


Fig. 2- Areas en cada perfil

deben abordarse en todos los perfiles, aunque la profundidad y el enfoque pueden variar fuertemente de un enfoque a otro: lo que para uno puede ser un conocimiento técnico, profundo, para otro puede ser simplemente a nivel de lo que significan los conceptos; lo que un caso puede involucrar desarrollo para saber hacer, en otro puede interpretarse como saber usar; etc. La ponderación sobre un total de 40 materias se da simplemente para tener otra vara de medida, sin querer decir por esto que los temas corresponden exactamente a materias: mientras en un perfil un grupo de temas puede dar origen a una o mas materias, en otro puede corresponder solo a algunas sesiones dentro de un curso.

Los histogramas que se muestran en la figura 1 y en la figura 2 ilustran las ponderaciones gráficamente, a nivel global. Las cifras globales dan una idea de la medida; pero la diferenciación de perfiles resulta del análisis e interpretación de las cifras parciales.

El cruce de áreas y perfiles es también la resultante del trabajo realizado en la cuarta asamblea anual general de la ANIEI, previamente mencionado [5].

3- Conclusiones

Con la elaboración de éste trabajo, que reune las colaboraciones de un grupo grande de integrantes de la comunidad académica y que cuenta con un concenso de la misma, se pretende disponer de un marco conceptual sólido que permita la configuración de nuevos planes de estudio, así como la adecuación de los actuales, si corresponden; a la vez que se desea eliminar la dispersión y ambiguedades que aquejan en el presente al sistema educativo en el área de informática y computación.

La versión definitiva del trabajo aparecerá en los primeros meses de 1987 y sin duda no será estática; periodicamente deberá revisarse y actualizarse para reflejar los cambios que tan dinámicamente se dan en los campos abordados. Sin embargo, sabemos que existen invariantes y temas ya clásicos que en ningún curriculum de estudio pueden faltar, y que si eso ocurre aún, es una falta grave que debe subsanarse de inmediato.

Referencias

[1] REPORTE "DEFINICION DE LA INFORMATICA"
Coordinador: Guillermo Levine (UAM)
Autores: Guillermo Levine (UAM)
Manuel Alvaréz (Univ. Anáhuac),
Luis Landois (Coleg. de Postgraduados-Chapingo)
Antonio Sanchéz (Univ. de las Américas)
1981.

[2] REPORTE "DEFINICION DE LOS CAMPOS DE ACCION DE LA INFORMATICA"
Coordinadores: Guillermo Levine (UAM)
Manuel Alvaréz (Univ. Anahuac)
Autores: Guillermo Levine (UAM)
Manuel Alvaréz (Univ. Anahuac)
Alfredo Chavéz (Cecyt "Juan de Dios Bátiz)
José A. Echenique (Fac. de Cont. y Admon - UNAM)
Fernando Galindo (UPIICSA-IPN)
Luis Landois (Coleg. de Postgraduados-Chapingo)
Miguel A. Moreno (Univ. Panamericana)
Rique A. Muñoz (Cecyt "Juan de Dios Bátiz-IPN)
Noé Rodríguez (UPIICSA-IPN)
Antonio Sanchéz (Univ. de las Americas)
Raúl Torres (Fac. de Cont. y Admon - UNAM)
Carlos Ulibarri (UIA)
Septiembre de 1982.

[3] "MODELOS CURRICULARES, NIVEL LICENCIATURA, INFORMATICA-COMPUTACION"
(ed.preliminar)
Coordinadores: Victoria Bajar (ITAM),
Guillermo Levine (UAM).
Autores: Victoria Bajar (ITAM)
Guillermo Levine (UAM)
Manuel Alvarez (UNAM)
Hans Fetter (UAM)
Fernando Galindo (UPIICSA-IPN)
Adolfo Guzmán Arenas (CINVESTAV-IPN)
Armando Maldonado (UAM)
Leonel Favila (Dir. Gral de Inst. Tecnolog-SEP)
Rafael Gamboa (ITAM)
Juan M. Ibarra Zannatha (CINVESTAV-IPN)
Luis Landois (Coleg. de Postgrados-Chapingo)
Gerardo León (IIE)
Mario Oviedo (UPIICSA-IPN)
José Pineda (CINVESTAV-IPN)
Ramón Ríos (ITAM)
Isaac Schnadower (UAM)
Américo Vargas (UAM)
Junio de 1986.

[4] "MODELOS CURRICULARES, NIVEL LICENCIATURA, INFORMATICA-COMPUTACION"
Coordinadores: Victoria Bajar (ITAM)
Guillermo Levine (UAM)
Autores: Victoria Bajar (ITAM)
Guillermo Levine (UAM)
Manuel Alvarez (UNAM)
Hans Fetter (UAM)
Fernando Galindo (UPIICSA-IPN)
Adolfo Guzmán Arenas (CINVESTAV-IPN)
Armando Maldonado (UAM)
Raúl Carrillo (Univ. Aut. de Tamaulipas)
Leonel Favila (Dir. Gral. de Inst. Tecnolog-SEP)
José Luis Florez (ITESO)
Rafael Gamboa (ITAM)

Daniel L. García (UPIICSA-IPN)
Juan M. Ibarra Zannatha (CINVESTAV-IPN)
Luis Landois (Coleg. de Postgraduados-Chapingo)
Gerardo León (IIE)
Manuel López (CENAC-IPN)
Mario Oviedo (UPIICSA-IPN)
Norma Pimienta (UPIICSA-IPN)
José Pineda (CINVESTAV-IPN)
Ramón Ríos (ITAM)
Alfonso San Miguel (ITAM)
Isaac Schnadower (UAM)
Américo Vargas (UAM)
Próxima aparición.

[5] IV Asamblea Anual General de la ANIEI Toluca, Edo. de México. 9-11 de Octubre de 1986. (55 participantes de 25 Escuelas)

[6] Real Academia Española
Diccionario de la Lengua Española
Madrid 1970.

Informática y cooperación:

una perspectiva.

Riccardo Ferrari, Gerónimo Bellassai, Benjamín Barán

Laboratorio de Electrónica Digital
Universidad Católica Nuestra Señora de la Asunción
Asunción - Paraguay

Provocada por la *revolución microelectrónica* la integración entre sectores un tiempo distintos produjo un sistema con características autorregenerativas: las *Tecnologías Informáticas* (TI). La capacidad de penetración de las TI en los diferentes sectores productivos, hace siempre más difícil, para quien no está dotado de un adecuado soporte tecnológico, el mantenimiento de las propias partes de mercado. En tal escenario se precisa, para cualquier país, una estrategia que permita defender la propria autonomía y aprovechar las posibilidades creadas por las TI.

A pesar de las barreras que surgen de la competición global de la que las TI son objeto, Latinoamerica puede lograr exito en el desafío tecnologico, costruyendo nuevos modelos de desarrollo, con el aporte de su tradición.

En tal sentido la cooperación internacional puede jugar un rol de particular importancia cuando, como en el proyecto de cooperación no gubernamental descripto, se puedan integrar los aspectos culturales y tecnológicos.

1.0 El Escenario Tecnológico

1.1 Las Tecnologías Informáticas: Un sistema autorregenerativo

El elemento que conjuga Electrónica, Telecomunicaciones, Informática, es el tratamiento de la información. Sin embargo por mucho tiempo la diferencia de los métodos (electromecánicos, analógicos, digitales) ha prevalecido, produciendo nítidos límites de área. Con el paso del tiempo, la siempre mayor aplicación de dispositivos electrónicos ha hecho indefinibles tales límites, hasta que la aparición de los microcomponentes y la consiguiente difusión de las técnicas digitales los ha anulado prácticamente.

En efecto, la posibilidad de tratar cualquier información, ya sea gráfica o sonora, con técnicas digitales, generada por la disponibilidad de microcomponentes con elevada capacidad de memoria y procesamiento, ha provocado el proceso de integración, actualmente en estado avanzado, esquematizado en Fig.1; la expresión adoptada para denominar el complejo resultante es la de *Tecnologías Informáticas* (TI). Esta no solo es la más usada, sino pone en evidencia la característica de lo que va siempre más delineandose como un conjunto de técnicas y máquinas para la recolección, la grabación, el proceso, la transmisión y el uso de las informaciones bajo forma digital. El objetivo, no lejano, es la posibilidad de un sistema integrado en que, por ejemplo, sea posible memorizar en una computadora la voz de salida de un teléfono, procesarla, retransmitirla como comentario de una imágen también digitalizada.

Un sistema en que la salida de cada dispositivo de tratamiento de información pueda ser entrada de otro.

El aspecto más estupefaciente de las TI es sin embargo su evolución a menudo definida en forma exponencial, pero más bien se podría decir geométrica si se considera que en los últimos 20 años el número de componentes por microcircuito es sistemáticamente doblado cada año[1].

El desarrollo de los microcomponentes tiene como éxito inmediato el aumento de potencia de los sistemas de cálculo; si se reflexiona sobre la importancia que dicho aumento de potencia reviste para la automatización de la proyectación, de la producción y de la prueba de nuevos circuitos, y entonces sobre el desarrollo de la microlectrónica misma, nos damos cuenta de una caraterística original de las TI: la capacidad autorregenerativa. Cada evolución en un campo particular repercute en forma general y vuelve a influenciar dicho campo, produciendo un desarrollo "explosivo".

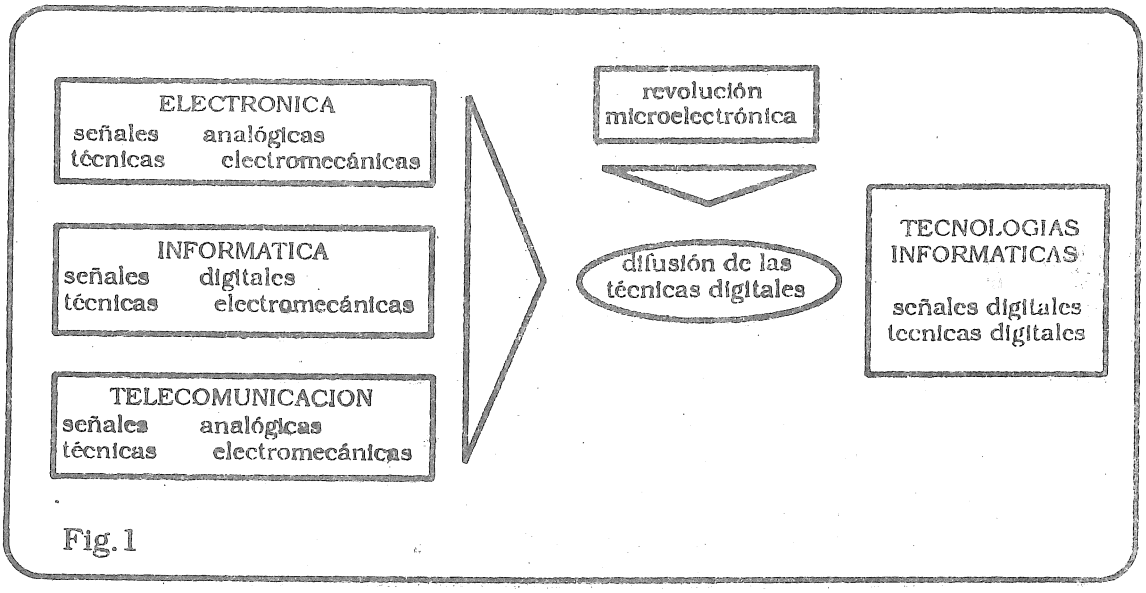


Fig.1

La situación está esquematizada en Fig.2, donde está también indicado el camino de retroalimentación recién indicado.

Descriptos los nuevos límites de sectores, es posible preguntarse dónde nos han conducido y hasta dónde pueden llegar, por lo menos en un futuro inmediato, las TI.

Al ya citado doblamiento anual de la escala de integración se une una drástica reducción de los precios. El costo de un bit de memoria central pasó de los 10 centavos de dólar en 1970 a los actuales 0.05 centavos de dólar, y se prevee que ulteriormente deba reducirse por lo menos 10 veces durante los próximos 5 años.

El costo de un micropocesador de 32 bits, igual a 1000$ en 1980, llegará a los 25$ en 1988.

Todo esto repercute a nivel de sistemas, donde el costo por MIPS se reduce en un 25-30% anual [2].

A través del desarrollo de las técnicas litográficas y la superación de la *barrera del micron* en la producción de los microcircuitos, es posible pensar que alrededor de 1995 sean disponibles componentes 20 veces más veloces que los actuales.

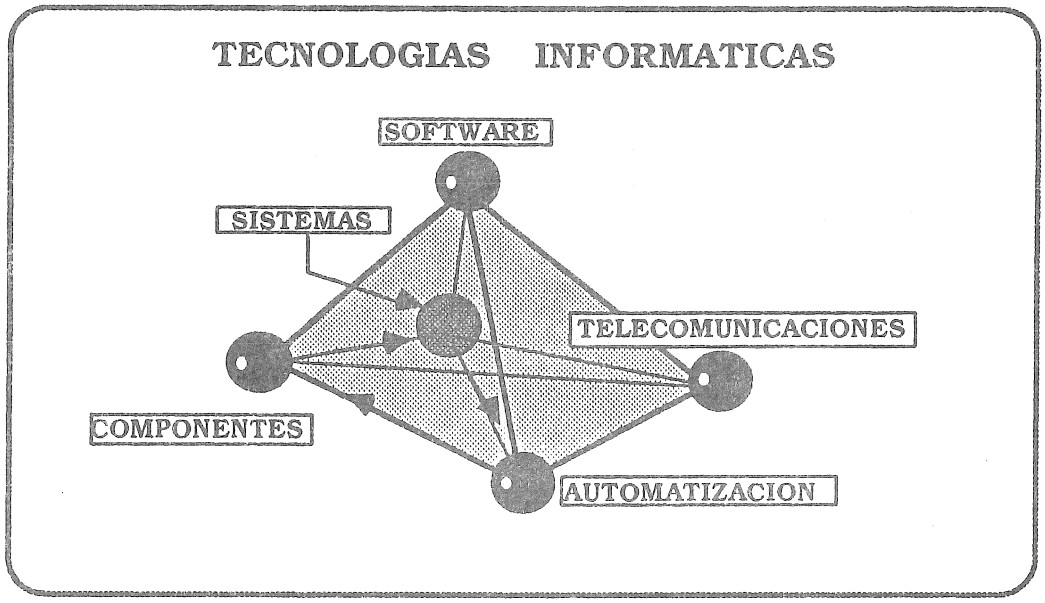
Esto sin contar las posibilidades evolutivas totalmente nuevas, como las basadas sobre *ballistic effect*, actualmente en estado de avanzada experimentación, que podrían llevar a final de los años 90, a circuitos centenares de veces más veloces que los actuales.

En el campo de las memorias de masa, a más de los incrementos de densidad de memorización alcanzables a través de los perfeccionamientos de la tecnología magnética, se presenta el empleo de las tecnologías opticas y laser, con capacidad de grabación superior de dos ordenes de grandeza. Por otra parte el costo siempre menor de las memorias en estado sólido a bajo consumo, prefigura la posibilidad de su empleo para memorización masiva.

En las telecomunicaciones, a las innovaciones que interesan los medios transmisores (fibras opticas, satélites), se unen las de los dispositivos de gobierno de las redes, donde la tecnología digital viene progresivamente a tomar el lugar de la análogica.

La creciente disponibilidad de potencia de cálculo a bajo costo y de medios de comunicación de elevada capacidad de transmisión induce significativos cambios en la arquitectura de los sistemas informáticos.

La tradicional estructura *jerárquica*, en que un potente núcleo central sirve un número más o menos grande de estaciones de trabajo periféricas, dotadas de limitada capacidad de cálculo y de memoria, se la sustituye por una *distribuida*, con niveles diversificados: redes locales (LAN) de personal, integradas con mini, supermini y *mainframe*, conectadas por medio de redes de amplio radio (WAN) y conectadas a las grandes redes intercontinentales.



La estación de trabajo al que tal evolución apunta está basada en microprocesadores de 32 bits con una velocidad de procesamiento de 10 MIPS, memoria central de 1 a 16 MB, periféricos con prestaciones gráficas elevadas y capacidad de reconocimiento vocal.

Esta podría ser huesped de una red global de comunicación, capaz de permitir la trasnmisión de mensajes audio y video digitalizados.

La disponibilidad de sistemas de elevada potencia de cálculo y de bajo costo, permite al sofware de evolucionar a su vez. Los sistemas expertos y la inteligencia artificial son las fronteras de dicha investigación. El tratamiento de las informaciones se trata de sustituir por el tratamiento del conocimiento, la reproducción de mecanismos decisionales, la capacidad de autoprogramación.

La rapidez de los cambios tecnológicos actuales, la amplitud de la difusión de las aplicaciones que derivan de ello no ven precedentes: ningún otro flujo tecnológico (si no, parcialmente, en el caso de la energía eléctrica) nunca tuvo una influencia tan amplia intra e intersectorial. Quizás sea justo definir el evolucionar y el difundirse de las TI como la tercera onda de civilización, un cambio radical en lo generado con el surgir de la industrialización.

1.2 La difusión de las Tecnologías Informáticas

A la característica de sistema autorregenerativo, las TI agregan la de ser extremadamente difusivas en todo el tejido económico y social.

Por cuanto lejos de ser completas, la automatización de la oficina, de la banca, del punto de venta, son ya una realidad difundida, como igualmente están difundidos los ladrillos de la futura automatización de la fábrica: *Computer Aided Engineering* (CAE), *Computer Aided Design* (CAD), *Computer Aided Manufacturing* (CAM), *Robótica*, *Local Area Network* (LAN).

El objetivo perseguido, destinado a ser alcanzado en la última decada del siglo, es la fábrica del futuro donde las funciones clásicas de proyectación, gestión de la producción, elaboración y montaje, movimiento y deposito formarán un único sistema integrado (CIM - *Computer Integrated Manufacturing*) [3].

El aumento de la capacidad productiva y de su calidad -resultante de la introducción de las TI en la actividad industrial, y en cualquier actividad económica- su difusión creciente, son factores que dejan prever como imprescindibles la adquisición de cierto nivel tecnológico de parte de cualquiera que pretenda mantener las propias partes en cualquier mercado.

2.0 El mercado de las Tecnologías Informáticas

2.1 La demanda

Al final de los años 60 en el mundo existían cerca de 120 mil computadoras; al final de los años 70 el número subió a 2.5 millones; hoy se cuentan 30 millones, predominantemente *personal computer*. La difusión de estos últimos ha cambiado radicalmente la forma de la demanda dirigiendola hacia el sector de las comunicaciones. El usuario pide que todos las computadoras puedan dialogar entre sí en forma eficiente. Principal obstáculo para este deseo es la estructura monopolista y estatal del mercado de las comunicaciones, además de la falta de *standards* realmente adoptados por los constructores.

Por otro lado, la enorme difusión de los *personal computer* está produciendo una crisis de madurez del mercado: crecido en tasas del 15-20% anuales por un periodo largo, el mercado estadounidense atraviesa una fase de éxtasis, mientras los síntomas de duda de la demanda tienden a propagarse también a Europa y a Japón.

La demanda tiende a especializarse, se hace más exigente, pide que las numerosas promesas de las TI se hagan realidad.

De hecho, mientras aparecen siempre más cercanas las metas tecnológicas ya delineadas, se evidencia como la realidad actual es incompleta respecto a las necesidades del usuario, que pide ya cuanto es posible ofrecer en un futuro más o menos próximo. A la disminución cuantitativa de la demanda corresponde su crecimiento cualitativo.

2.2 La oferta

El proceso de integración entre Informática y Telecomunicación en curso a nivel tecnológico y largamente requerido por el uso, impulsa a los productores a buscar alianzas que permitan estructurar una oferta adecuada.

El panorama de la producción de TI, ya hace tiempo, aparece marcada por crecientes estrategias de acuerdo y coparticipación.

Sea a nivel geográfico o por línea de productos, el mercado de las TI tiende siempre más a connotarse como global.

Las TI, de siempre oligopolistas, tienden a serlo siempre más. En el área de los componentes, la necesidad de economías de escala, a causa de la bajada de los precios y de la muy alta necesidad de capitales para la investigación y desarrollo, constituye la principal barrera para la entrada. En la gran informática, el nivel muy elevado de investigación y desarrollo para productos competitivos y la necesidad de una compleja y difundida organización comercial representan los obstáculos de mayor relieve para el surgimiento de nuevos productores.

El campo de las telecomunicaciones, estable por más de un siglo, ve cambiar un orden hasta ahora monopolista, gracias a la tendencia de la privatización de las redes, mientras que los grandes productores de informática buscan lograr una penetración a través de acuerdos y adquisiciones de pequeñas sociedades especializadas.

La presencia de 250 empresas productoras de informática en sentido estricto, en el mercado mundial, podría hacer creer en un desarrollo de la pequeña informática. Los análisis más exhaustivos, en cambio, impulsan a creer que se trata de una fase transitoria, destinada a concluirse con un proceso de concentración.

En el sector de los personal computer, las primeras tres marcas ostentan el 60% del mercado mundial; las restantes firmas compiten, entonces, por partes relativamente marginales, en condiciones destinadas a volverse difíciles, bajo el peso de los gastos de investigación y desarrollo continuamente necesarios.

El mercado global impulsa a la concentración y a la redefinición de las estrategias de:

• **Internacionalización:** en el mercado de masa de las TI, la economía de escala es más que nunca necesaria; se precisa producir grandes volumenes de productos con un ciclo de vida de 2 o 3 años; para hacer esto es necesaria una presencia en todos los mercados mundiales. Casi ninguna empresa está hoy en grado de hacer la articulación necesaria, sea por el compromiso económico, sea por las barreras proteccionistas. Se concibe así una nueva estrategia internacional en base a acuerdos e integraciones.

• **Diversificación productiva:** en el ambito de una competición global los productores se dirigen hacia la oferta de una gama diversificada y completa de productos, buscando provecho de los usuarios que no quieran enfrentar el esfuerzo de la gestión de equipos de distintas firmas.

• **Innovación e investigación:** la rapida evolución tecnológica requiere inversiones extraordinarias en este campo. La relación entre inversiones para investigación/desarrollo y facturado está a niveles tres veces superiores al promedio de la industria manufacturera. También en este caso es vital el rol jugado por los acuerdos, especialmente con las universidades.

3.0 Tecnologías Informáticas en Latinoamérica: problemas y perspectivas

Cuando se afronta el tema de la introducción o del desarrollo de las TI en los países no industrializados, y en particular en Latinoamérica, nos encontramos frente a grandes optimismos y grandes pesimismos.

La gran hipótesis es el "salto tecnológico", o sea la posibilidad de un atajo histórico que permita a los países latinoamericanos superar la fase industrial, centralista, necesitada de grandes inversiones, ecológicamente destructiva, aprovechando el último nivel de las TI, sin pasar por las onerosas etapas que las han generado. Juega a favor de esta oportunidad la libertad de acción consiguiente a la ausencia de la necesidad de una reconversión de la estructura industrial preexistente.

Efectivamente, para los países altamente industrializados la automatización de la fábrica es fuente de problemas ocupacionales del todo inexistentes allí donde no esté igualmente desarrollada la producción industrial [4].

Sin embargo, comparando la hipótesis del "salto tecnológico" con el escenario mundial descripto anteriormente, es muy obvio preguntarse cuáles son las probabilidades reales de resultado.

La respuesta a este tipo de preguntas puede ser dada solamente por la historia futura, pero es posible, o más bien necesario, reflexionar sobre los factores de riesgo y sobre los elementos de fuerza. Para hacer esto se precisa detallar en forma ulterior el objetivo, o sea que se quiera lograr.

Si el objetivo es la conquista de una parte, aunque sea mínima, del mercado global de las TI, la solución ya está dada por el análisis de mercado: costos de investigación y desarrollo enormes, economías de escala para obtener producciones rentables y redes de venta y asistencia a nivel mundial hacen difícil la vida de las empresas europeas y estadounidenses e impiden el acceso de nuevos productores.

Como ya se dijo, no hay que hacerse ilusiones sobre la posibilidad de la pequeña informática, por lo menos en lo que se refiere al mercado global. El milagro económico realizado por el Extremo Oriente, que vió países como Korea, Singapur, Taiwan, Malasia, mantener un crecimiento del 10% del PIB, gracias a la producción de TI, ha visto su surgimiento por una convergencia de posibilidades economicas, culturales y politicas bien difícil de recrear.

Si proyectar, producir y vender TI en el mercado mundial es una meta lejana, en el plano del desarrollo de aplicaciones de las mismas están pesando herencias difíciles. Un parque de computadoras dirigido predominantemente a la gestión administrativa [5], a más de haber absorbido notables recursos financieros [6], creó, mantuvo y difundió una mentalidad particular en los especialistas y en los usuarios. Estos últimos continuan viendo el medio informático como algo costoso, complejo, prerrogativa de grandes organizaciones, justificables sólo frente a inmensa cantidad de trabajo. Es decir que la informática es aún concebida como un cuerpo extraño a la estructura organizativa, y además cargado de exigencias tales de alterar ordenes consolidados.

De su parte el especialista se ensimismó en la figura de mediador entre empresas constructoras y públicos más que en el de productor creativo, interprete de las necesidades locales. En tal situación no se excluye que el temor de perder el propio rol, por no saber hacer frente a las nuevas exigencias, impulse a muchos a ser tenaces defensores del *status quo*.

Todo esto viene a influenciar negativamente las grandes posibilidades proveidas por la introducción de microcomputadoras potentes, económicas, flexibles, fácilmente adaptables a cada tipo de aplicación científica, técnica, productiva, administrativa.

¿Si el análisis de los problemas es correcto, qué perspectivas quedan para que las TI no se conviertan en una ocasión perdida?

Como primera cosa es justo preguntarse cuál es la dirección y cuáles son lo primeros pasos posibles, luego es necesario individualizar los recursos.

Aquí no se quieren afrontar los aspectos políticos y económicos, que ya son factores esenciales de logro. Por otra parte economía y política pueden permitir y administrar un desarrollo, pero es bastante imposible que puedan suscitarlo.

• **Dirección:** Las consideraciones a largo plazo deben tener en cuenta al mismo tiempo los aspectos defensivos como los propositivos. Por un lado resulta evidente que las TI juegan un rol de protagonistas en el desarrollo económico, no sólo por su aplicación, sino como producción de alto rendimiento. En el plano defensivo ya es imprescindible, para cualquier país que desee mantener o desarrollar la propia autonomía, apuntar a una capacidad tecnológica nacional. En el caso de que la siempre más indispensable aplicación de las TI a las actividades productivas inundase sin una apropiación tecnológica correspondiente, la consiguiente dependencia resultaría todavía más pesante y de difícil remoción. Si las barreras para la entrada al mercado global de las TI son demasiado elevadas, esto no quiere decir que no sea posible adquirir una posición significativa en sectores estremadamente especializados sea productivos o aplicativos. Aquí puede resultar util citár el caso Italia, país que llegó a conquistar una producción de respeto en el ámbito de la automatización de la fábrica. Si existe una oportunidad es la de no imitar tecnologías ya adquiridas sino tender directamente a productos sustitutivos de las nuevas generaciones. Automatización e Inteligencia Artificial son con seguridad las áreas en que tales productos deberán colocarse.

• **Primeros pasos:** El hecho de que no sea posible competir por hoy en el mercado global no significa que sea igualmente difícil adquirir importancia en el plano del mercado interno. La observación podría parecer obvia desde el momento en que ya varios países de Sud América, Brasil a la cabeza, han embocado de hace tiempo este camino. Pero es necesario subrayar la importancia de desarrollar una vocación tecnológica autoctona. Las aplicaciones verticales, destinadas a las necesidades locales, son un campo que puede revelarse bastante fértil, en términos productivos o por la adquisición de conocimientos tecnológicos. Interfaces, sistemas de control y telerelevamiento, aplicaciones científicas, son ejemplos de producciónes *hardware* y *sofware* que pueden ser afrontadas con provecho, constituyendo entretanto la base para futuros "saltos" en el mercado externo.

• **Recursos:** Es bien difícil desarrollar soluciones tecnológicas de avanzada en ausencia de un adecuado bagaje que, antes que productivo, es técnico, científico, cultural. Es decir: no hay dirección en que moverse y primeros pasos posibles sin los hombres capaces de realizarlos. Esta consideración nos introduce al gran tema de la formación. El gran recurso del subcontinente latinoamericano es la gente y tal recurso puede ser valorizado sólo a través de una adecuada formación. Para que resulte eficaz esta debe al mismo tiempo proveer los conocimientos técnicos y desarrollar la creatividad y la iniciativa. Estas últimas radican en una conciencia de la propia humanidad y de la propia tradición mucho mas que en un aprendizaje mecánico de técnicas. La genialidad no se aprende, sino que se desarrolla, y esto ocurre cuanto más fundado y global es el horizonte humano propio. La tradición cultural y religiosa de América Latina es el gran patrimonio con que sus pueblos pueden afrontar el desafío tecnológico. La opinión de que por la primera vez los países industrializados y los que no lo son se encuentren frente a frente en un viraje histórico y por lo tanto con las mismas armas, muy ilusoria si nos paramos en considerar los aparatos económicos, técnicos y productivos, toma consistencia cuando se incluye la ya evidente necesidad de repensar los modelos y las razones del desarrollo para cualquiera que se presente a la civilización informática. En este sentido el desafío que las naciones latinoamericanas deberán recoger no consiste solo en el desarrollar tecnologías más avanzadas que las actuales, sino en un enfrentamiento global que genere soluciones originales.

4.0 "Laboratorio de Electrónica Digital", una cooperación en el área informática.

De hace varios años la Universidad Católica Nuestra Señora de la Asunción (U.C.) y la Asociación de Voluntarios para el Servicio Internacional (AVSI), organismo no gubernamental italiano, han comenzado una cooperación tendiente a la creación de un laboratorio de electrónica que desempeñe actividades didácticas, investigación y servicio. El programa usualmente es denominado "LED - Laboratorio de Electrónica Digital", por cuanto se trate de

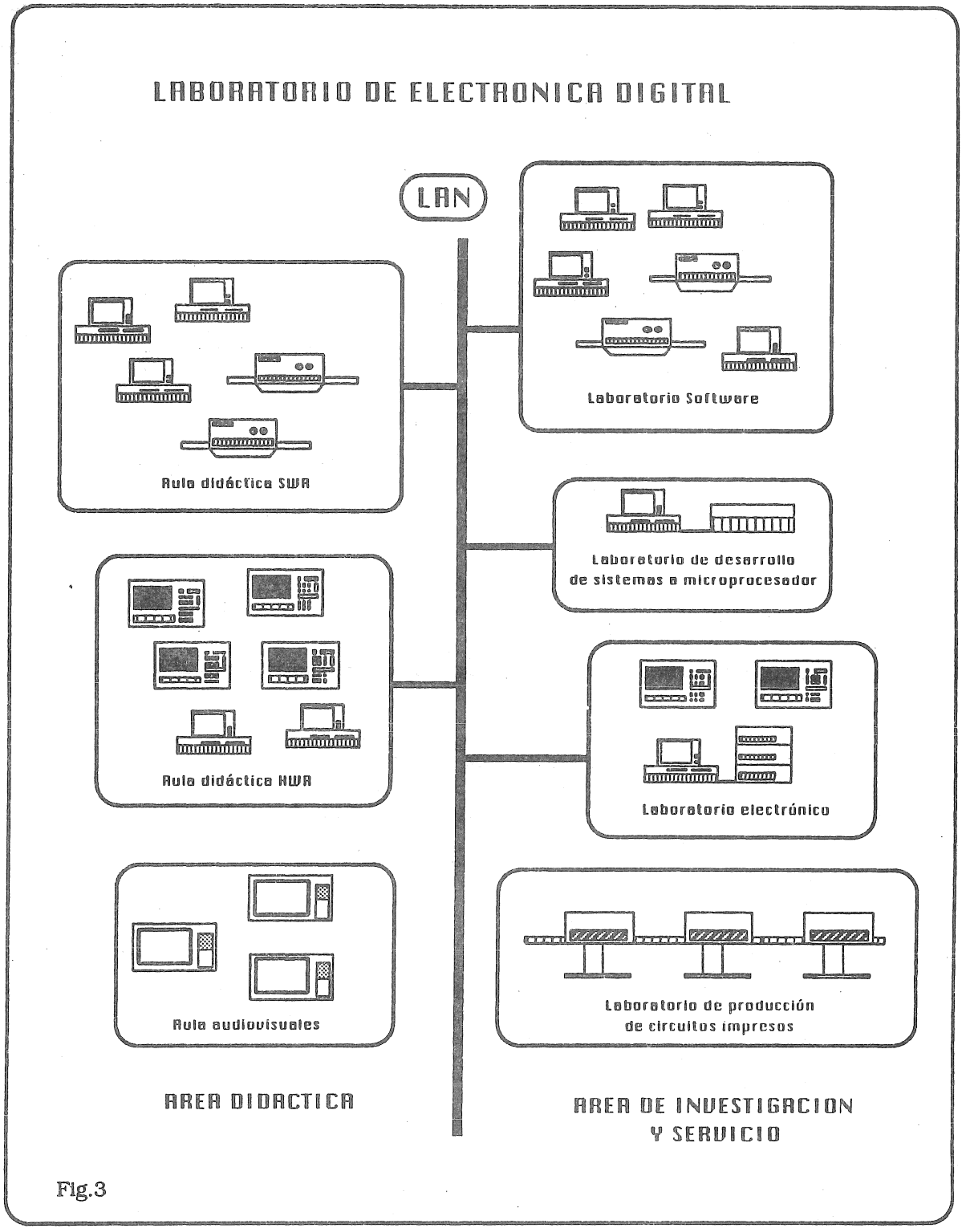


Fig.3

conjunto de laboratorios cuya configuración organizativa se puede ver en la Fig.3. Después de alrededor de tres años de estudio y preparación que han visto el compromiso de personal italiano y local, con la segunda mitad del 86 el proyecto pasó a la fase de realización con la presentación de los curriculum de dos nuevas carreras en Ingeniería Electrónica e Informática, y con el inicio de la puesta a punto de los laboratorios. Esta fase prevee una colaboración, de parte italiana, en términos de recursos humanos y de equipamientos, por un periodo de cinco años. A cargo de la U.C. de Asunción estan las infraestructuras y los gastos para el personal local.

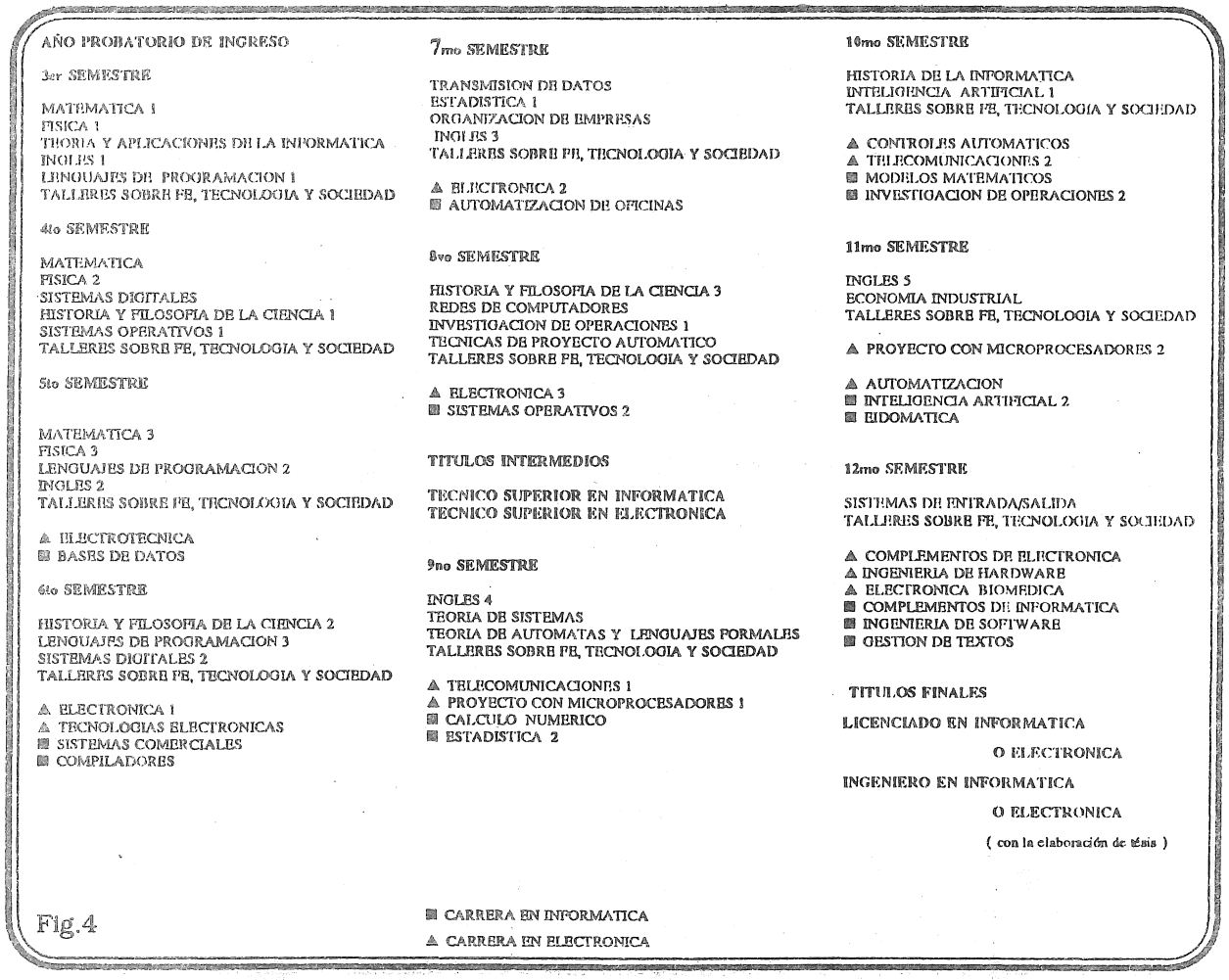


Fig.4

A más de proveer el soporte de laboratorio de las dos carreras, el LED organiza cursos de educación continua para especialistas o para usuarios.

Aquí es interesante subrayar como los aspectos estructurales y metodológicos del proyecto estan orientados a fin de responder a las solicitaciones emergentes del análisis precedente.

El laboratorio está fisicamente organizado en dos sectores:

- Area didáctica

- Area de investigación y servicio.

El área didáctica, dotada con un aula *software*, un aula *hardware* y un laboratorio para audiovisuales, está destinada a ser la base para las ejercitaciones de las carreras de Ingeniería Electrónica e Informática, como también para los cursos de educación continua, dirigidos a especialistas y usuarios, tenidos periodicamente por la U.C..

El área de investigación y servicio formada por:

- Laboratorio electrónico

- Laboratorio "software"

- Laboratorio de desarrollo de sistemas con microprocesadores

- Laboratorio de producción de circuitos impresos.

La misma está destinada a ser utilizada, por los profesores de la U.C. que desarrollen proyectos de investigación en el área de las TI y por los estudiantes de los últimos años, comprometidos en la actividad de investigación para realizar una pasantía adecuada.

El uso de *standards* comúnmente aceptados es un criterio normativo en el desarrollo del proyecto a fin de operar la necesaria independencia de los constructores: por eso el LED se orientó hacia el empleo de computadoras personales con sistema operativo DOS, al sistema UNIX, al uso del lenguaje C.

El contenido de las carreras, cuya lista de materias se ve en la Fig.4, está orientado a la formación de especialistas que desarrollen creatividad e iniciativa a través de la adquisición de una base de conocimiento general de las TI, unida a la profundización de los instrumentos económicos, organizativos y culturales, necesarios en la actividad profesional e empresarial.

La naturaleza del país en que el proyecto opera, caracterizado por un desarrollo bastante limitado sugirió la creación de carreras que no tiendan a una excesiva especialización. Así se crearon dos ramas principales, Ingeniería Electrónica e Ingeniería Informática, que aprovechen un cuerpo común de disciplinas sea de *hardware* o de *software* . Los títulos proveídos son:

- después de cuatro años del curso (incluso el año probatorio de ingreso, año de formación general requerido por la U.C. a cualquiera que se inscriba): *Técnico Superior en Electrónica o Informática* ;

- luego de otros dos años de curso: *Licenciado en Electrónica o Informática* ;

-con el desarrollo de una tesis: *Ingeniero en Electrónica o Informática* .

Los puntos fundamentales del programa formativo pueden ser listados de la siguiente manera:

Sólida formación de base: la preparación de una base matemática y física profundizada es requisito irrenunciable para la obtención de una flexibilidad aplicativa del profesional.

Amplio conocimiento de las TI: si ya en países de alto grado de industrialización la super-especialización se ha demostrado contraproducente, esto es aún más verdadero en el caso de los países que estan presentadose en la introducción de las TI. Por esto la preparación debe resultar lo más amplia posible ya sea bajo el perfil aplicativo o para el de investigación.

Subrayado de las tendencias evolutivas: sea en la didáctica que en la investigación el proyecto prevee el desarrollo de actividades en aquellas áreas que a breve o a largo plazo revestirán una importancia particular:

-Proyectación de sistemas con microprocesadores para control de proceso y recolección de datos, interfaces *hardware* y *software* , énfasis sobre las técnicas de transmisión de datos: tales actividades corresponden a los "primeros pasos" ya mencionados, cuya implementación es improrrogable.

- Introducción en los Sistemas Expertos y en la Inteligencia Artificial; si el costo de las máquinas necesarias para el desarrollo de aplicaciones de inteligencia artificial es hasta ahora prohibitivo, esto no significa que no se pueda comenzar su estudio y su profundización en pequeños sistemas, permitiendo no perder el paso sobre un tema de importancia estrategica.

Adecuada formación cultural: es propia de la Universidad Católica la preocupación de que los años de estudio universitario no se transformen en una simple adquisición de nociones técnicas, en un adiestramiento, sino que sean un tiempo de desarrollo integral del hombre. Una preocupación similar vió la convergencia de la AVSI, en la convicción de que un horizonte cultural amplio, tendiente en buscar el nexo entre cualquier actividad y conocimiento y las razones que hacen humana la vida, sea el tipo de recurso del que hay más necesidad en la actual fase de transformación tecnológica/productiva.

La figura profesional que emerge del proceso descripto es la de un especialista versatil, capaz de dirigirse hacia la investigación en el área de las TI, o hacia actividades aplicativas como el cálculo científico o la proyectación, según la permanencia en el área universitaria o del trabajo elegido.

En todos los casos será siempre más un productor de TI que un usuario evolucionado, o un mediador tecnológico.

Por otro lado los cursos de educación contínua ya sea en un área "hardware" o "software", tienden a formar un usuario exigente, capaz de aprovechar a fondo las posibilidades ofrecidas por las aplicaciones de las TI.

Ya el simple conocimiento de los instrumentos existentes (como: procesadores de texto, bases de datos, hojas electrónicas, instrumentos gráficos) representa una primera importante aproximación a una imágen amigable y útil de las TI. A esta sigue el conocimiento de las posibilidades aplicativas, que llevará a los usuarios a un diálogo siempre más calificado y proficuo con los especialistas, rompiendo la actual inmovilidad.

5.0 Conclusión

El análisis de la situación tecnológica y del mercado, la reflexión sobre la situación latinoamericana, evidencian la necesidad de un notable empeño formativo que apunte, no solo a los aspectos de innovación técnica, sino también a la totalidad de la tradición de Sudamérica.

En esto reside la originalidad más profunda del proyecto LED. El mismo, antes que sobre los aspectos estructurales, se basa sobre el desarrollo de un sujeto homogéneo italo/paraguayo que permita la utilización de las TI como instrumentos para la satisfacción de las necesidades propias, y la génesis de nuevos modelos de integración tecnológico/culturales.

Este sujeto no es otro que un lugar de encuentro entre hombres, que son tales antes que especialistas y técnicos. El verificarse de esto estuvo entre los primeros objetivos del trabajo hasta aquí desarrollado, y representa hoy las más seria garantía de logro.

Bibliografía

[1] Comité de la Presidencia del Consejo de los Ministros para la Ciencia y la Tecnologia, *"Rapporto al Presidente del Consiglio"*, Roma, 1986.

[2] F. De Benedetti, *"Le Chance dell'Europa e le Nuove Tecnologie"* , Informatica Settanta, N. 142, 1986.

[3] B. Lamborghini, C. E. Rossi, *"Problemi e Prospettive della Robotica Industriale in Italia"* , Milano, 1986.

[4] M.J.F. Soupizet, *"D'une informatique de l'offre a une informatique des besoins"* , Agora, N.3, 1985.

[5] V.M. Toro C., M. Castillo H., *"Sobre la formación universitaria en informatica. Intento de sintesis de diversas reflexiones"*, XII Conferencia Latinoamericana de Informatica, Montevideo, 1986.

ALFABETIZACION MASIVA EN COMPUTACION EN LA UNIVERSIDAD CATOLICA DE CHILE

Pedro Hepp K., Ph.D.
Luis Dissett
Escuela de Ingeniería
Universidad Católica de Chile
Casilla 6177 (143) - Santiago

Se presentan las principales experiencias obtenidas luego de casi 3 años de desarrollo del proyecto CRISOL en la Universidad Católica de Chile. Este proyecto ha significado la incorporación masiva de computadores en la universidad, haciendolos accesibles a académicos, alumnos y administrativos de todas las unidades académicas. El énfasis inicial ha sido instalar salas de microcomputadores tanto para uso libre como para cursos estandarizados de alfabetización dictados por el Departamento de Ciencia de la Computación. Actualmente se incentivan los cursos desarrollados y dictados por las propias unidades académicas. El proyecto en su siguiente etapa busca el crecimiento computacional descentralizado, a través de redes de computadores y con apoyo de personal técnico en las mismas unidades académicas.

1. Introducción

Diversas universidades de paises desarrollados han establecido programas de incorporación masiva de la tecnología computacional [1,2,3,4]. Estos proyectos han involucrado gran cantidad de recursos y el apoyo material de empresas como IBM, D.E.C., Apple y otras. Estas experiencias se han desarrollado principalmente en universidades de E.E.U.U., Canadá y Europa.

Este trabajo reporta acerca del proyecto CRISOL, el cual se está llevando a cabo en la Pontificia Universidad Católica de Chile desde comienzos de 1985. Esta Universidad cuenta con aproximadamente 11.000 alumnos, distribuidos en 5 campus en Santiago. También hay diversas sedes en provincias.

La computación en la universidad comenzó a mediados de la década del 60 con un computador IBM 1620. En 1973 se adquirió un Burroughs 3500, en 1977 un DEC-10 y a comienzos de esta década un VAX/780. Luego de casi 3 años del proyecto Crisol, el equipamiento central consiste de un D.E.C VAX/8600, 3 D.E.C. VAX/750 y varios D.E.C. microVAX a los cuales hay conectados aproximadamente 200 terminales. Además, hay alrededor de 400 microcomputadores, principalmente Apple Macintosh, Olivetti M-19 e IBM PC, instalados en dependencias de diversas unidades académicas y en salas denominadas "Salas Crisol", con 15 a 35 microcomputadores cada una, ubicadas en todos los campus de la universidad.

2. Objetivos del proyecto Crisol:

El Proyecto Crisol, aprobado oficialmente por las autoridades de la universidad a comienzos de 1985, define en su documento inicial [5] una política de computación. En sus aspectos más fundamentales, dicha política establece un marco conceptual para el desarrollo de la computación, un esquema para la adquisición y renovación de equipos y un esquema para la administración de los recursos.

En el marco conceptual se establece que "*los computadores tienen el potencial de afectar la productividad y las relaciones humanas en las instituciones, en particular en la universidad. Esto implica difundir y hacer accesible las posibilidades de la herramienta computacional a todos los sectores de la universidad; capacitar masivamente a alumnos, profesores y administrativos, distinguir segmentos de usuarios para quienes la computación representa posibilidades y desafíos semejantes; definir el concepto de servicio de computación como un sistema de apoyo al usuario que consta de máquinas, programas, documentación y asistencia técnica*".

El esquema para la adquisición y renovación de equipo establece la conveniencia de contar con equipos de diversas características, de acuerdo a las necesidades de los segmentos de usuarios; reconoce la importancia de las redes de comunicación; trata el problema de la obsolescencia tecnológica e incentiva la formulación de proyectos conjuntos entre la universidad y las empresas.

El esquema para la administración de los recursos redefine los objetivos del Servicio de Computación, institución encargada de la adquisición y administración de los recursos computacionales centrales, y pone énfasis en definir indicadores medibles de calidad de servicio al usuario a través de planes anuales y trienales evaluables año a año.

Cabe destacar la separación de personas en segmentos de usuarios. Estos segmentos son:

Segmento I: Iliterados en computación, aquellos sin experiencia alguna en computación. Los usuarios de este segmento requieren respuestas a las siguientes preguntas: ¿qué es un computador? y ¿qué posibilidad representan para mi los computadores en el dominio de mi actividad?.

Segmento II: Usuarios de programas envasados y que ya cuentan con alguna experiencia computacional.

Segmento III: Desarrolladores de software y/o hardware, que incluye a quienes requieren de conocimientos más especializados en computación.

La labor se concentró en el segmento I, estableciendo como objetivo académico que para fines de 1988, el 100% de los alumnos que hayan cursado el tercer semestre académico, el 50% de los profesores y el 25% de los administrativos estén capacitados en el uso básico del computador.

3. Planificación, desarrollo y resultados.

El Departamento de Ciencia de la Computación elaboró un plan de trabajo para cumplir los objetivos del proyecto. Luego de las experiencias realizadas a la fecha, se ha visto la necesidad de alterar en forma radical dicho plan.

3.1 Planificación.

El plan de trabajo inicial consitió en enseñar un conjunto de aplicaciones de propósito general, con apuntes de auto instrucción, prácticas y clases estandarizadas para toda la comunidad universitaria. Los objetivos de la estandarización fueron racionalizar la utilización de recursos y simplificar el proceso de clases, evaluaciones y capacitación de ayudantes. Los cursos iniciales fueron dictados por profesores del Departamento de Ciencia de la Computación. Al mismo tiempo se capacitó a un grupo de estudiantes de la carrera de computación para que paulatinamente fuera asumiendo las tareas docentes, bajo supervisión de un profesor. Esto se debe a que el proyecto, en estado de régimen (1988) requeriría de alrededor de 100 cursos anuales.

El curso para el segmento I de usuarios incluye los fundamentos de procesamiento de texto, procesamiento de gráficos y dibujo, planillas de cálculo, bases de datos, más algunos antecedentos tecnológicos sobre el funcionamiento del computador y de redes. Posteriormente hay una evaluación con resultado "reprobado", "aprobado" o "distinguido".

3.2 Desarrollo.

Se realizaron cursos para alumnos, profesores y administrativos en todos los campus de la universidad. Estos cursos fueron de 12 a 15 clases de 1.5 horas cada uno. Además se realizaron diversos cursillos, de 2 a 5 clases cada uno y se realizaron encuentros informales semanales de intercambio de experiencias.

En la medida en que se fueron instalando nuevas Salas Crisol, se fueron requiriendo más ayudantes y más tiempo de profesores para supervisar la labor docente. Debido a la escasez de recursos para afrontar esta mayor demanda, diversas unidades académicas comenzaron a implementar cursos y talleres con sus propios medios, de contenidos específicos orientados a sus necesidades.

3.3 Resultados.

El sistema de cursos mostró que si bien las aplicaciones enseñadas eran de utilidad general, cada alumno deseaba darle énfasis a las materias que le eran más afines a su quehacer. Por ejemplo, era común que los alumnos de arte encontraran muy superficial la cobertura dada a los programas de diseño y demasiado profundo la de las planillas de cálculo. El caso de los ingenieros era exactamente al revés. En cambio, los alumnos de matemática y psicología lamentaban la ausencia de programas estadísticos, los de pedagogía pedían lenguajes como Logo o Basic, etc.

Los cursos estandarizados para alumnos sufrieron, en general, una marcada deserción. En promedio, el 40% de los alumnos inscritos en un curso lo finalizaron. Esto se debe, en parte, al carácter voluntario de los actuales cursos, los que son abandonados debido a la presión académica de los cursos regulares. Otra razón importante es que no parece ser atractivo aprender a utilizar una herramienta fuera de un contexto en que sea relevante aplicarla. Por ejemplo, para un alumno de ingeniería resultan de inmediato aplicables y de gran provecho las planillas de cálculo, sin necesidad de requerir ejemplos que asi lo prueben. Para un alumno de educación, derecho o teología, esto no es necesariamente obvio, y el aprender a utilizar el computador resulta menos atractivo.

En cambio, los cursos iniciados como iniciativas de las propias unidades académicas tuvieron un gran éxito y actualmente la demanda por ellos es

mucho mayor que la capacidad de dictarlos. El éxito de estas experiencias indican que éste es el camino más adecuado, dado el entusiasmo observado en los alumnos y la participación activa de los profesores.

4. Proposiciones para la siguiente etapa

Como resultado de la experiencia acumulada, se propuso apoyar el desarrollo de la computación desde el interior de cada unidad académica e implementar redes de computadores.

4.1. Desarrollo de la computación al interior de cada unidad académica.

Considerando la forma en que las personas aprenden el uso de las aplicaciones, la idea inicial de enseñar computación a toda la comunidad universitaria, en cursos estandard y fuera del contexto de una disciplina, no es adecuada. Nuestra proposición es incorporar la computación como parte de las actividades normales -cursos, talleres, laboratorios- impartidos en cada unidad académica. Esto implica el asignar recursos para contar con ayudantes en las unidades académicas con el único fin de apoyar el proyecto Crisol.

Este personal debiera consistir de un grupo estable de 3 a 5 ayudantes en cada unidad académica, que fomenten, con el apoyo del Servicio de Computación y del Departamento de Ciencia de la Computación, la incorporación de la computación en sus actividades. Pensamos que la orientación de los cursos hace la diferencia en la motivación de los alumnos. Por ejemplo, el uso de programas de diseño es diferente en una escuela de Diseño que en Música o Física.

Se requiere de un coordinador administrativo y un coordinador docente en el Servicio de Computación y en el Departamento de Ciencia de la Computación, respectivamente. El coordinador administrativo es necesario para el trabajo de difusión y recolección de noticias, edición y publicación de boletines, administración de software en salas, mantención de bibliotecas de software, manuales, apuntes, etc. El coordinador docente tiene como función específica la coordinación y apoyo de los grupos de ayudantes y profesores de cada facultad que estan promoviendo el uso de los computadores al interior de sus unidades. Este apoyo se traduce en la

formación de ayudantes, preparación de charlas, talleres y demostraciones, actualización de apuntes y programas prototipos de evaluación, apoyo en la formación de grupos de usuarios, etc.

De la experiencia recogida a la fecha, se concluye que el no haber incluido mayores recursos para apoyar el proyecto con recursos humanos, constituye el impedimento mayor para el logro de las metas establecidas.

4.2 Redes de Computadores

La instalación de redes debiera constituirse en el proyecto central a partir de 1988. Toda adquisición de equipos debiera considerar su incorporación en las redes de la universidad.

Consideramos que objetivos tales como el aumento de productividad y cambio en las relaciones humanos, establecidos en el marco conceptual del proyecto, comenzarán a materializarse con mayor efectividad en un ambiente de redes. Las redes presentan 2 ventajas evidentes:

1) Compartir impresoras y otros equipos periféricos. Actualmente hay una gran falta de capacidad de impresión lo cual desincentiva el trabajo del usuario de sala. Se estima que debiera haber 1 impresora rápida (400 cps) por cada 7 a 10 microcomputadoras y una impresora de calidad de correspondencia por sala Crisol.

2) Accesibilidad al Software. Debido al alto costo del software y a la necesidad de evitar la copia no autorizada de éste, se hace difícil hacer accesible el software en un sistema de máquinas individuales. Una solución es dejar residente y protegido el software en un disco duro accesible a través de la red por todos los usuarios de una sala.

5. Conclusiones

La principal conclusión del proyecto es que la incorporación masiva de computadores en la universidad, en todas las disciplinas y para todos sus miembros, tiene un efecto positivo si es respaldado por la autoridad central, si nace de un proyecto con plazos y metas bien definidas y susceptibles de ser evaluadas y si cuenta con el apoyo entusiasta de los especialistas en computación.

También es importante concluir que no resulta apropiado estandarizar la docencia y material de curso para diseminar adecuadamente esta tecnología: hay que apoyar las iniciativas en cada unidad académica y dejar a los expertos en computación como consultores de los niveles intermedios sin hacerse cargo de la capacitación masiva de todos los miembros de la comunidad universitaria.

Fianlmente, concluimos que no basta con comprar computadores para comenzar un proyecto, es muy importante asignar, dede un principio, recursos para comprar software y contratar personal que administre y desarrolle el proyecto.

6. Referencias.

1. Balcovich, Ed., Lerman, Steven, Pamerlee, R.
"Computing in Higher Education: The Athena Experience"
ACM/IEEE Joint Issue, 1985

2. Zawacky, Edward
"Project Ezra: Focus on Quality Education"
Academic Computing at Cornell Newsletter,
Cornell University, winter 1985.

3. "Faculty Author Development Project"
ACIS/IRIS, Stanford University, Octubre 1986.

4. "Carnegie Mellon University: Reaching for World Leadership
in Educational Computing and Communications."
ACIS, IBM Academic Information Systems, 10/86

5. "Proyecto Crisol. Política para el Desarrollo de la Computación en la
Pontificia Universidad Católica de Chile",
Pontificia Universidad Católica de Chile
Vicerectoría Académica
Santiago, Noviembre de 1984.

Educando hacia la Ingeniería del Software
Una reinterpretación para URUGUAY *

Ing. S/C Daniel Schwed†
Investigador visitante
Department of Information and Computer Science
University of California, Irvine, CA 92717, U.S.A.

Marzo de 1987

Dirección actual:
P.O. Box 6036
Montevideo
URUGUAY

1 Introducción

En este trabajo se analiza la situación pasada y actual de la educación en el área de Ingeniería del Software en el Uruguay[1]. Asimismo se plantean los objetivos futuros hacia los que se pretende dirigir la educación en esta área. Este trabajo resume experiencias del autor como docente del In.Co., y de docentes de la Universidad de California en Irvine.

2 Historia reciente

Las carreras de Ingeniero de Sistemas en Computación y Analista-Programador se encontraban desde el año 1974 sin modificación en su plan de estudios. Este plan de estudios

*Segun Peter Freeman en su presentación *Why Software Engineering is Important to Developing Countries* [Freeman77].

†En uso de licencia del Instituto de Computación, Facultad de Ingeniería, Montevideo, URUGUAY. Este tarbajo ha sido posible gracias a una beca brindada por la Comisión para el Intercambio Educacional entre Uruguay y Estados Unidos (Comisión Fulbright), y el apoyo del Advanced Software Engineering Project, Universidad de California – Irvine.

[1]El único centro de enseñanza de computación de nivel universitario del país, es la Facultad de Ingeniería (F.Ing.) dependiente de la Universidad de la República donde se dictan las carreras de Ingeniero de Sistemas en Computación y Analista-Programador. En la F.Ing. el Instituto de Computación (In.Co.) tiene a su cargo las labores docentes y académicas en el campo de la Informática. El In.Co. es uno de los 8 institutos que componen la Facultad de Ingeniería.

al que llamaremos Plan Viejo, estaba organizado hacia la creación de técnicos de nivel terciario.

La situación no era satisfactoria, como lo muestran los siguientes indicadores:

- La relación alumnos por docente era mayor a 130:1.
- La carga horaria por alumno era de 6.5 minutos docente/semana.
- Los alumnos inscriptos superaban los 4000, o sea mas del 50 % de los estudiantes de la Facultad de Ingeniería.
- Se disponía de solo 28 terminales para todos los alumnos.
- Creciente obsolecencia en el contenido de los cursos.

Como resultado del retorno al estado de derecho en 1985, y con el fin de tratar esta situación, se formó la Comisión Asesora del Area de Computación. Esta comisión fue designada por el Consejo de Facultad de Ingeniería [2] con el fin de llevar a cabo un replanteo de la situación en que se estaba y hacia qué metas se apuntaba.

Los primeros cambios que la Comisión logró introducir fueron:

- Revisión de los contenidos de cada materia.
- Llamados a nuevos cargos docentes.
- Redistribución de los escasos recursos existentes.

Surgió una propuesta docente de reestructura de la currícula, en dos etapas, bautizada Plan Turing 88 [PlanTuring]. Esta propuesta modifica fundamentalmente los contenidos de las materias (Plan de Emergencia) y propone una carrera sustitutiva (la que se definirá a partir del Plan Nuevo), renombrando el título final como "Ingeniero en Computación", y agregando el título intermedio "Analista en Computación", con una orientación más académica. En este momento el Plan de Emergencia se está poniendo en práctica y el Plan Nuevo se esta definiendo.

3 Estado Actual

Las primeras etapas del Plan de Emergencia fueron recientemente aceptadas por el Consejo de la Facultad de Ingeniería y entraron en vigencia en el 87. La puesta a punto de la nueva carrera (Plan Nuevo) será gradual, ya que los alumnos del Plan Viejo podrán terminar su carrera segun éste.

[2] El gobierno de la Universidad está representado por tres órdenes: docentes, estudiantes y egresados. Estos forman parte de los consejos de cada facultad y del consejo central y son elegidos democráticamente. Durante los años 74 al 84 por motivos políticos, el gobierno de la Universidad no se vio representado por los tres órdenes.

3.1 Ingeniería del Software

La idea de introducir el concepto de la Ingeniería del Software en nuestro instituto aparece durante 1985 (como resultado de la reestructura docente y los cambios que se produjeron en los contenidos de las materias indicados anteriormente). Se creó un Area de Ingeniería del Software dependiente del Departamento de Programación del In.Co. y se reestructuró la materia Procesamiento de Datos II (PDII), con esta orientación.

Esta reestructura del curso de PDII se basó en el seminario dictado por el Prof. Lavette C. Teague, [3] sobre "Análisis y diseño estructurados", utilizando bibliografía provista por éste durante su estadía en Montevideo [DeMarco78], [Page-Jones80], [Teague85].

El curso dictado se extendió anexando un taller realizado en grupos de 6 a 8 alumnos, a los cuales se les asignó, según sus intereses, un proyecto real, con usuarios reales (dependencias de la Universidad, Facultades, Ministerios, etc.).

Como resultado de este primer curso se observaron varios puntos:

- Los grupos numerosos de estudiantes (6 a 8) no funcionan bien, debido a problemas de comunicación, falta de definición de roles, incompatibilidad de horarios y diseño por compromiso (diseño por comité). La experiencia sirvió como ejemplo de lo pernicioso que puede ser trabajar en grupos grandes

- Es deseable que los proyectos sean reales. Los proyectos fueron positivos en el sentido de la experiencia ganada, el contacto con la realidad y con los problemas que generan los usuarios, lo que no se les hubiera podido transmitir de otra forma.

- Los usuarios reales tienen poca preocupación por los plazos académicos y se dieron casos en que desatendieron a los estudiantes.

- Los tiempos calculados para los proyectos fueron sub-estimados. El rendimiento de los alumnos fue menor al esperado y los usuarios contribuyeron en esto por lo indicado en el punto anterior.

- La teoría y la práctica deben ser combinadas en forma más gradual. Los cursos no deben tener una parte teórica y una parte práctica separadas.

Es de destacar que la cátedra debió enfrentar numerosos problemas, al disponer solamente de 3 docentes con un total de 33 horas/semana para atender 264 alumnos.

3.2 Formación y Competencia de un Ingeniero del Software

Esta sección revisa los objetivos que deberían alcanzarse a fin de lograr una adecuada formación de Ingenieros del Software. Fairley en [Fairley86], enumera la formación y competencia deseable y/o requerida por un Ingeniero del Software:

[3] California State Polythechnic University, profesor visitante bajo los auspicios del programa Fulbright.

- Análisis de requerimientos
- Especificación funcional
- Diseño del software
- Implementación
- Inspecciónes, revisiones y *walkthroughs*
- Pruebas de las unidades y *debugging*
- Integración y pruebas de aceptación
- Control de calidad
- Preparación de los Manuales del usuario y Guías de mantenimiento
- Mantenimiento del software
- Dirección de proyectos

A fin de lograr una buena educación en el área de Ingeniería del Software, se debe incluir conocimientos de las siguientes cinco disciplinas básicas: ciencias de la computación, administración, habilidades de comunicación, resolución de problemas generales, y diseño [Freeman77].

Es importante una discusión mas detallada de cada una de las disciplinas mencionadas.

1. Ciencias de la computación

   Las ciencias de la computación son la base del conocimiento para un Ingeniero del Software. Un Ingeniero del Software debe conocer:

   - *Matemática discreta*
     Algebra. Teoría de grafos. Lógica.
   - *Estructuras de datos*
     Estructuras recursivas. Algoritmos recursivos. Análisis de algoritmos.
   - *Arquitectura de sistemas*
     Anatomía de los computadores. Compresión de datos, codificación. Arquitecturas de alto nivel. Administración de procesos. Arquitectura del software de alto nivel. Sistemas distribuídos.
   - *Métodos formales*
     Pruebas. Verificación. Abstracción. Especificación. Teoría de Lenguajes Formales. Análisis.

- *Metodología de programación*
  Métodos de verificación. Inspecciones. Pruebas. Codificación. Técnicas, lenguajes, y métodos de diseño.

En resumen, un Ingeniero del Software debe tener una base sólida en ciencias de la computación, pues es en ésta que se basa toda su actividad. El conocimiento amplio de estos temas le permitirá desarollarse a medida que el campo de la Ingeniería del Software avance.

2. Administración

El objetivo principal de la Ingeniería del Software es desarrollar artefactos usables (en tiempo y forma). Los principales temas que deben considerarse son:

- Técnicas de evaluación de costos
- Presupuestación
- Análisis de los requerimientos del usuario
- Técnicas del tipo de PERT, CPM, programación lineal
- Organización teórica y práctica
- Psicología industrial
- Administración de personal

3. Habilidades de comunicación

Un Ingeniero del Software debe participar en la preparación de propuestas, descripción de los requerimientos del usuario, especificación de sistemas, reportes de estado, documentación de sistemas y reportes de mantenimiento.

Las formas de comunicación incluyen la comprensión y preparación de trabajos técnicos y reportes de proyectos. Estas deben ser escritas, orales, formales o informales.

Un Ingeniero del Software debe poder comunicarse tanto en su lengua madre como en cualquier lenguaje técnico que use, su formación debe apuntar a ello.

4. Resolución de problemas generales

Es obvio que a una persona no se le puede enseñar a resolver problemas en general, ya que en determinadas áreas no existe el sustituto de la experiencia. Pero se puede ver que un entrenamiento adecuado puede mejorar sensiblemente la respuesta del individuo.

Las áreas en las que se debe buscar familiaridad y dominio son:

- Formulación de problemas
- Fijación de objetivos
- Estrategias de solución
- Generación de ideas
- Implementación de las soluciones
- Validación de las soluciones

5. **Diseño**

   El diseño es una actividad se que aprende realizándola. No hay sustituto al contacto con el diseño. Obtener experiencia en diseño, es fundamental en la educación de un Ingeniero del Software.

   El énfasis en la teoría es valioso siempre y cuando no desplace otras necesidades básicas de la currícula. El diseño debe ocupar un papel preponderante en la educación de un Ingeniero del Software. Se sostiene que una condición *necesaria* para un Ingeniero del Software, es estar bien entrenado en las técnicas actuales de programación y tener una considerable experiencia en ellas, pero esto no es *suficiente* para hacerlo un buen diseñador [Freeman80].

4 Soluciones a corto plazo

4.1 Prerequisitos.

El plan de estudios vigente contiene una buena raíz para la formación de Ingenieros del Software. Entre los puntos antes mencionados se cubren las áreas de ciencias de la computación y administración en detalle suficiente. Las modificaciones en estas áreas no implicarán cambios sustanciales. Sin embargo en las áreas de comunicación, resolución de problemas, y diseño hay mucho para hacer.

Para ello se deberán procurar cambios en los contenidos de las materias ya existentes, basándose en el concepto de libertad de cátedra (una reestructura de las materias implicaría una modificación tal en la currícula, que debería ser considerada por el Claustro de la F.Ing., lo que tomaría por lo menos seis meses). La Comisión Asesora del Area de Computación ha recomendado a los docentes de las materias relacionadas a los puntos antes mencionados, las modificaciones imprescindibles. La labor de la Comisión será guiar el cambio en forma conceptual y global.

4.2 Modificación del contenido de PDII.

Las modificaciones del programa de PDII ya fueron realizadas. El contenido de la materia es coherente con los objetivos planteados, pero se debe mejorar la forma en que se dicta.

La práctica debe ser revista e integrada a la teoría. Tal vez se deba considerar la inclusión de proyectos de sistemas en forma de competencia o una mayor interrelación entre los grupos de trabajo [Horning77], [Freeman76].

Los plazos de los proyectos deberán ser estrictos, y a pesar de ser deseable no se podrá depender de usuarios externos (ver punto 3.1), hasta no disponer de un plantel docente con mayor carga horaria, que le permita actuar como *buffer* entre los estudiantes y los usuarios.

Se deberá hacer un mayor hincapié en el concepto de diseño, apuntando a ampliar la experiencia de los estudiantes.

5 Soluciones a largo plazo

Las soluciones a largo plazo se basan en las propuestas del Plan Nuevo, (actualmente en estudio) que se pondrá en práctica en forma gradual, dentro de los próximos dos o tres años, luego de ser aprobado por el Claustro.

Es necesario incluir e integrar los conceptos de Ingeniería del Software al Plan Nuevo. Es recomendable preparar a los estudiantes intermedios con un claro concepto de la Ingeniería del Software [Freeman86]. Hay un numero de técnicas y conceptos de la Ingeniería del Software que pueden ser asimilados en las primeras etapas de formación universitaria. La mayor parte de los estudiantes que obtendrán el título intermedio se integrarán al mercado laboral en ese momento (esta idea esta basada en la situación actual que muestra una fuerte tendencia del alumnado a obtener solamente el título de Analista-Programador). Podríamos de esta forma, mejorar significantemente el nivel general de competencia y asimismo proveer una mejor base para la obtención del título final de Ingeniero en Computación.

Deberá permitirse, por medio de cadenas de materias opcionales, que el alumno se decida por la Ingeniería del Software u otra especialidad en los años finales de la carrera.

5.1 Talleres

Es de destacar la separación existente entre los objetivos de una formación en Ingeniería del Software que apunta a la práctica, con la de una orientación teórica pura.

La idea del trabajo en talleres en el Plan Nuevo está orientada hacia lograr un mayor y mejor contacto del alumno con la realidad. Esta idea, buena en su concepción, se ve empañada por el hecho que el ambiente universitario tiene diferencias muy marcadas con el mundo real, como se puede observar en la siguiente tabla [Riddle86] :

Diferencias entre los *Proyectos Educativos* y los del *Mundo Real.*

	Mundo Real	*Educación*
Tamaño	$>$ 10.000 lineas	$<$ 500 lineas
Ciclo de vida	10-15 años	"hasta que corra"
Complejidad	multiples módulos	un módulo
Desarollo	usuario $\neq$ desarrollo	usuario = desarrollo
Mantenimiento	extenso	descartable

La artificialidad de un entorno educativo (tradicional) no aporta un concepto de realidad al alumno. Como destacable se puede mencionar que el alumno no se ve enfrentado a los volúmenes de trabajo que se dan en el mundo real [Shaw76]. El ambiente educativo debe ser contrastado con la realidad, tomádose en cuenta los siguientes problemas:

- Compromisos a corto plazo y part-time de los alumnos
- Muchos estudiantes trabajando en el mismo problema
- La ausencia del usuario final y de continuidad del proyecto
- Falta de una contabilidad efectiva de recursos
- Las tensiones inherentes a un ambiente educativo

Existen distintas alternativas para organizar prácticas que sirven como aproximaciones útiles del mundo real y que proveen experiencias fundamentales en la formación en Ingeniería del Softawre. Las Empresas de Software ("Software Hut") [Horning77], por ejemplo, permiten que varios grupos compitan en un ambiente simulado de venta y soporte de productos desarrollados por ellos. Otros enfoques, [Freeman76], estan basados en el intercambio de productos entre grupos, por ejemplo, un grupo A especifica un sistema, un grupo B lo implementa y luego A lo mantiene.

No debe descartarse totalmente el trabajar con usuarios reales. Esta idea se puede llevar a cabo, disponiendo los docentes de una carga horaria suficiente, con los alumnos más avanzados de la carrera.

5.2 Nuevos cursos

Estos cursos estarán basados en una bibliografía ordenada y orientada hacia un nivel universitario [Fairley85a], [Fairley85b], [Gerhart86], [Pressman82].

En estos cursos se deberán introducir las ideas indicadas en el punto 3.2, dentro de las cuales se pueden destacar:

- El uso de formalismos en Ingeniería del Software.
- El diseño como actividad fundamental.
- La exploración de la idea de ciclo de vida en sus distintos paradigmas.
- La automatización del proceso del software.
- La formalización del conocimiento del proceso del software.

La inclusión de las cadenas de materias opcionales anteriormente mencionadas permitirán al alumno una selección y balanceo según sus intereses y aspiraciones.

5.3 *Bootstraping* de la idea de Ingeniería del Software.

La clave para lograr una buena currícula es establecer un híbrido entre el taller de artes, donde el alumno desarrolla su capacidad creativa, y el equivalente a un hospital de enseñanza, donde los médicos en formación (internos) practican sobre la realidad bajo la supervisión de expertos. Para esto es nesesario organizar laboratorios de desarrollo de software donde se logre la combinación de educación en forma de clases teóricas y el desarrollo de las habilidades prácticas del estudiante.

Estos laboratorios deberán permitir tanto la creación y desarrollo de métodos, herramientas y entornos de diseño y/o programación adaptados a la realidad y necesidades del país, como el desarrollo profesional de los alumnos. Bajo la supervisión de docentes y equipados con técnicas y herramientas actualizadas, los laboratorios permitirán al alumno ganar experiencias irremplazables [Basili86].

Asimismo esto podrá crear una fuente de ingresos por medio de contratos de desarrollo con empresas y organismos nacionales e internacionales. Un modelo de este tipo de cooperación ha sido implementado en California – MICRO program[4]. En el contexto de MICRO, los investigadores universitarios proponen proyectos de investigación aplicada. El costo del proyecto es subvencionado en un 50% por compañías interesadas y la Universidad de California. Debe aclararse que "costo" comprende todos los recursos del proyecto, incluyendo sueldos de investigadores, costo de hardware, software, etc. En nuestro país, dadas las limitaciones presupuestarias de la Universidad, las empresas podrían aportar un 100% de los costos, y el In.Co. como contrapartida contribuiría con los docentes y/o alumnos (que trabajarían en estos proyectos para aprobar sus cursos). Estos ingresos se podrían usar para beneficio del In.Co. en la compra de nuevos equipos, software, herramientas o financiando viajes de educación, seminarios dictados por profesores invitados, etc.

[4]Microelectronics Innovation and Computer Research Opportunities.

6 Resumen

De lo expuesto anteriormente, a fin de lograr una buena educación en el área de la Ingeniería del Software, se debe:

- Incluir e integrar los conceptos de Ingeniería del Software en todos los niveles del Plan Nuevo.
- Realizar talleres que apunten al contacto con el mundo real, con suficiente dedicación docente.
- Crear cadenas de materias opcionales que permitan la especialización coherente del alumno.
- Investigar nuevos paradigmas en el área de Ingeniería del Software.
- Generar herramientas que se adapten a los requerimientos del país.
- Obtener financiamiento a través de contratos de desarrollo con empresas y organismos nacionales e internacionales.

Estos puntos permitirán acercarnos a las metas a las que apunta el Instituto de Computación: la formación de profesionales y la creación de un buen ambiente académico.

[Basili86] Basili, Victor R., *The Experimental Aspects of a Professional Degree in Software Engineering,* Software Engineering Education: The Educational Needs of the Software Community, Springer-Verlag, Junio 1986.

[DeMarco78] DeMarco, Tom, *Structured Analysis and System Specification,* Yourdon Press, 1978.

[Fairley85a] Fairley, Richard, *Software Engineering Concepts,* McGraw-Hill, 1985.

[Fairley85b] Fairley, Richard, *Core Course Documentation, Master's Degree Program in Software Engineering,* Technical Report TR-85-17, Wang Institute of Graduate Studies, Setiembre 1985.

[Fairley86] Fairley, Richard, *Software Engineering Education: An Idealized Scenario,* Software Engineering Education: The Educational Needs of the Software Community, Springer-Verlag, Junio 1986.

[Freeman76] Freeman, Peter, *Realism, Style and Design: Packing it into a constrained course,* ACM SIGCSE Bulletin, vol. 8, num. 1, Febrero 1976.

[Freeman77] Freeman, Peter, *Why Software Engineering is Important to Developing Countries,* Proceedings of the International Conference on Computer Aplications, Bangkok, pp. 1489-1502, Agosto 1977.

[Freeman80] Freeman, Peter, *The Central Role of Design in Software Engineering,* Software Engineering: Research Directions, Academic Press, 1980.

[Freeman86] Freeman, Peter, *Essential Elements of Software Engineering Education Revisited,* Software Engineering Education: The Educational Needs of the Software Community, Springer-Verlag, Junio 1986.

[Gerhart86] Gerhart, Susan L., *Skills versus Knowledge in Software Engineering Education: A Retrospective on the Wang Institute MSE Program,* MCC Technical Report STR-202-86, Junio 1986.

[Horning77] Horning, J.J. and Wortman, D.B., *Software Hut: a Computer Program Engineering Project in the Form of a Game,* IEEE Transactions on Software Engineering, Vol. 3, No. 4, Julio 1977.

[Page-Jones80] Page-Jones, Meilir, *The Practical Guide to Structured Systems Design,* Yourdon Press, 1980.

[PlanTuring] Instituto de Computación, *Plan Turing/88,* Documento interno del In.Co., Agosto 1986.

[Pressman82] Pressman, Roger, *Software Engineering: A Practitioner's Approach,* McGraw-Hill, 1982.

[Riddle86] Riddle, William and Williams, Lloyd , *Technology Selection Education for Software Engineers,* Software Engineering Education: The Educational Needs of the Software Community, Springer-Verlag, Junio 1986.

[Shaw76] Shaw, Mary., *Making Software Engineering Issues Real to Undergraduates,* Software Engineering Education: Needs and Objectives, Springer-Verlag, 1976.

[Teague85] Teague, Lavette and Pidgeon, Christophrer, *Structured Analisis Methods for Computer Information Systems,* Science Research Associates Inc., 1985.

UNA LINEA DE INVESTIGACION EN LOS ESTUDIOS DE INGENIERIA DE SISTEMAS EN LA UNIVERSIDAD NACIONAL DE COLOMBIA: POLITICAS Y ESTRATEGIAS DE INFORMATICA PARA EL DESARROLLO

Resumen. - En esta comunicación se presenta una de las líneas de investigación dentro de los estudios de Ingeniería de Sistemas, en la Universidad Nacional de Colombia.

Para mayores informes sobre esta línea favor enscribir a:

Profesor Carlos Cortés Amador
Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Sistemas
Bogotá D. E. Colombia (Sur América)

Introducción. - Los estudios de Ingeniería de Sistemas por decisión de la respectiva Facultad en cuanto a investigación se refiere, están organizados por Acta 33 de 1985, según unas líneas de investigación que han sido acogidas por el Departamento de Sistemas; dichas líneas son las siguientes:

Linea uno: Desarrollo de teleinformática e ingeniería de software.
Linea dos: Cibernética, máquinas automáticas y computación.
Linea tres: POLITICAS Y ESTRATEGIAS DE INFORMATICA PARA EL DESARROLLO.
Linea cuatro: Investigación interdisciplinaria e Informática.

Para mayor información sobre las dos primeras líneas, el lector de esta comunicación puede escribir al Departamento de Ingeniería de Sistemas de la Facultad de Ingeniería, Ciudad Univer-

sitaria, Bogotá D.E., Colombia. Con relación a la Investigación interdisciplinaria e Informática, un resumen de sus actividades sería el siguiente:

Agrónica: Se está en la fase de su conceptualización y sehan desarrollado algunos trabajos (Profesores Mauro Flórez C., Alfonso Charum D, Carlos Cortés A. y Juan Manuel Andrade).

Informática médica: Los principales investigadores son los siguientes: Lucio Flórez C., Tito Flórez C., Ricardo Martínez R.

Informática interdisciplinaria: Información con el profesor Carlos Cortés A.

Informática Universitaria: Existen muchos trabajos por parte de los profesores Luis G. Astaiza A., Ismael Castañeda F., Alfonso Pérez G., Carlos Cortés A., Alfonso Charum D., José J. Martínez R., Ramón Diaz B., Alberto Jaime S., Jaime Malpica A.

Para mayores informes: Escribir a la dirección antes anotada.

Políticas y estrategias de Informáticas para el desarrollo

Aunque en el total de la actividad investigativa, esta línea es posiblemente la que presenta menor porcentaje, sin embargo sus actividades son bastante destacables.

Los temas más importantes son:

- Políticas y estrategias tecnológicas e industriales (el caso de las tecnologías de la información).
- Cambio tecnológico y su impacto socio-económico y político.
- Transferencia de tecnología, empleo y recursos humanos (el caso de las tecnologías de la información).
- Informática y sociedad.
- Estudios sobre usos alternativos de la informática (informática social) y estudios de prospectiva.
- Innovación (sobre este tema el Instituto de Ensayos e Investigación de la Facultad de Ingeniería ha organizado un programa especial).
- Evaluación y negociación de tecnologías de la información.

Sobre estos temas ha aparecido algunas publicaciones tales como: Informática y Sociedad (Carlos Cortés), Cambio tecnológico (Carlos Cortés), Memorias sobre el Seminario de negociación de software (Alfonso Pérez), Tendencias tecnológicas y políticas de informática (Carlos Cortés). Algunas conceptualizaciones han aparecido en la forma de artículos en diversos periódicos, revistas y anales de congresos. Entre las principales están: Revista Delta (ACCIO), Uno y Cero, Microbyte, Micromundo, Computación iberoamericana, revistas de Colciencias e ICFES y signo y pensamiento .

Entre los proyectos y tesis de grado están las siguientes por áreas:

- Sobre políticas y estrategias: Papel de la planeación industrial en el desarrollo de la informática; evaluación de la planeación en tecnologías de las telecomunicaciones; análisis de políticas de informática (comparación Brasil y Colombia); Integración e Informática; perspectivas de las tecnologías de la información en la microempresa.

- Transferencia de tecnología, empleo y recursos humanos: Análisis de los programas académicos para Ingeniería de Sistemas (Bogotá-años 80-85); análisis y perspectivas de las industrias nacionales de software; Estudio de factibilidad tecnológica sobre producción en electrónica; Enseñanza de la Informática en el bachillerato; Esquema para modelo del manejo de Ciencias y tecnología en las universidades (estudio en el Viejo Caldas); análisis de comunicaciones vía satélite en Colombia; tecnología, organización y trabajo; impacto industrial de las tecnologías de la información, por sectores.

- Informática social: Impacto social y usos alternativos de las telecomunicaciones (estudios de caso de la red pública de datos en Colombia).

- Innovación: Ingeniería del conocimiento, diseño e informática; innovación e informática; innovación, informática y planeación estratégica.

- Evaluación y negociación tecnológica: Estudios sobre ciclos de vida de tecnologías de la información (sector financiero por ejemplo); evaluación y desagregación de tecnologías de la información; negociación de software; evaluación de recursos y planeación (estrategia); factibilidad, estudios y manejo de proyectos; Estudios organizacionales; desarrollo política administrativa; procedimiento; planeación estratégica y estudios de orientación (la mayoría de estos apenas en consideración).

PLANIFICACION DE INFORMATICA EDUCATIVA A NIVELES UNIVERSITARIOS

C. Scheel M.
Director Centro de Cómputos Académicos
Universidad del Sagrado Corazón
Puerto Rico

RESUMEN

Dada la complejidad interdisciplinaria del proceso educativo que se desarrolla dentro de las Instituciones Universitarias, muchas veces es una labor colosal el mantener todos los sectores administrativos, académicos, de investigación-desarrollo, de extensión y de apoyo operativo, satisfechos con el papel hasta ahora desempeñado por la informática como "medio" de transmisión, de enseñanza y de aprendizaje del conocimiento.

El crecimiento de la Informática es vertiginoso, y el área es considerada como estratégica para la mayoría de las Universidades Americanas. Siguiendo el mismo patrón, bajo otras proporciones ocurre algo similar en Latinoamérica.

La "computarización" académica y administrativa de la universidad ha tenido un desarrollo tan rápido que en la mayoría de los casos no se ha podido ni estructurar ni identificar lo que actualmente se tiene, ni mucho menos planificar sobre metas definidas y bien establecidas.

El control de la introducción o evolución de la computarización se ha perdido, o en la mayoría de los casos, se deja en manos de los fabricantes o proveedores de equipos computacionales, quienes ofrecen soluciones empaquetadas utilizando patrones "similares" o típicos tratando de estandarizar el ideal personalizado de la Universidad.

Es por esto que hay que desarrollar una Planificación de Informática Universitaria para cada institución en particular. Hay que adaptar metodologías de planeación estratégica a niveles corporativos empresariales, al ambiente académico, e incorporar la Informática como respuesta factible y establecer procedimientos no-computarizados pero sistematizados, de lo que no puede ser debidamente estructurado para operar "eficientemente" mediante la Informática actual, como facilitador y mejorador del proceso educativo.

La Computación se encuentra en una etapa de análisis, manipulación y síntesis del conocimiento.

Sin embargo, todo este despliege tecnológico, no se ha reflejado en el proceso de enseñanza-aprendizaje. Algunos de los tradicionales sistemas de CAI (Computer Aided Instruction), existentes en el mercado, no desarrollan ni habilidades del pensamiento, ni habilidades para la resolución de problemas o toma de decisiones, y mucho menos se integran de una manera eficaz al curriculum del estudiante.

Prácticamente no ofrecen casi ningún beneficio marginal comparados con un buen libro, o con los medios audio-visuales convencionales.

En este artículo se presenta la problemática anterior, así como una nueva generación de "Sistemas Educativos Basados en la Computación". Se introduce una metodología aplicada a una Institución Universitaria Latinoamericana, en la que se identifica, analiza, diagnostica y recomienda programas, proyectos y actividades encaminados a computarizar efectiva, adaptiva y modularmente las áreas académicas y administrativas de la universidad.

Posteriormente se presenta una evaluación sumativa de la implantación de algunas de las actividades de la metodología de Planificación de Informática Universitaria.

EDUCACION APOYADA

POR COMPUTADOR

UN AMBIENTE DE DESARROLLO DE SOFTWARE PARA EDUCACION

Pedro Hepp. K., Ph.D.
Jaime Navón C., M.Sc.
Escuela de Ingeniería
Universidad Católica de Chile
Casilla 6177 (143) - Santiago

Se propone un conjunto de componentes de software que simplifican la generación de aplicaciones en apoyo de la educación, en especial en el diseño de la interfaz con el usuario. Se muestra la utilidad de estas componentes en la construcción de un laboratorio de estructuras de datos en el cual los alumnos pueden observar el comportamiento de diversas estructuras de datos en situaciones de carga real.

1. INTRODUCCION

La utilización del computador como herramienta de apoyo a la educación ha tenido un significativo auge con la aparición de los microcomputadores, haciendo posible su utilización en apoyo a la docencia en colegios y universidades y a la capacitación en empresas.

El desarrollo de la educación asistida por computador (CAI, Computer Aided Instruction) ha permitido explorar el uso de diversos sistemas puestos a disposición de los profesores para construir sus aplicaciones. Ejemplos son los proyectos Plato de la universidad de Illinois y Ticcit de la corporación Mitre,y más recientemente los lenguajes Smalltalk [Gold 83]y Logo [Pape 80]. Estos sistemas han tenido diversos grados de aceptación entre los profesores quienes tienden a preferir la mayor flexibilidad que les permiten los lenguajes de propósito general, tales como Basic, Pascal y Logo para construir sus propios programas [Niev 86].

Los rápidos avances de la tecnología computacional aumentan dia a dia el potencial de uso de los microcomputadores en educación. Nuevos procesadores de gran velocidad y poder, memorias de varios millones de bytes, pantallas de alta capacidad gráfica que ofrecen la posibilidad de utilizar colores, ventanas, íconos y video interactivo. A esto se suman los avances en psicología cognitiva [Hulm 85] [Reid 85], nuevas ideas en torno a tutores inteligentes [Slee 82], etc.

Sin embargo, esta tecnología hace cada vez más compleja la programación de las aplicaciones, haciendo necesario el conocimiento de técnicas de Ingeniería de Software para obtener programas que sean modificables, robustos y que utilicen adecuadamente los recursos computacionales. Es así como a los profesores con conocimientos básicos de programación les resulta cada vez más difícil aprovechar los avances de la tecnología y deben contentarse con aplicaciones más modestas o bien utilizar programas de generación de aplicaciones CAI, tales como Pilot con los cuales pierden parte de la flexibilidad de un lenguaje de programación.

Puesto que el costo del diseño de la interfaz con el usuario y su implementación representan típicamente más de un 50% del costo total de desarrollar una aplicación [Shoo 83] [Somm 85] y que otros componentes, tales como el manejo de la persistencia representan cerca de un 30% [Atki 83], resulta evidente que éstas son las áreas en que se requiere mayor apoyo.

En este trabajo presentamos una alternativa al profesor con conocimientos básicos de programación. Ofrecemos un ambiente de desarrollo de software en el cual existe una gran facilidad para diseñar las componentes de la interfaz hombre-máquina, automatizando la generación de código a partir del diseño y proporcionandole componentes de software reusables que manejen la persistencia de los datos.

A continuación se presentae el ambiente de software, se muestran algunos ejemplos de diseño

y finalmente se desarrolla una aplicación que implementa un laboratorio de estructuras de datos, destacando la ventajas pedagógicas y las componentes de la interfaz con el usuario.

2. HERRAMIENTAS Y COMPONENTES DE SOFTWARE

Como se ha mencionado, los profesores prefieren la utilización de lenguajes de propósito general debido a la total flexibilidad que les permite, pero esto dificulta la programación de aplicaciones que aprovechen las nuevas tecnologias. Además, se hace cada vez más difícil modificar las aplicaciones para mantención y principalmente ante cambios en el diseño.

Nuestro enfoque se basa en la premisa que es suficiente (y deseable) proporcionar flexibilidad al nivel de diseño de una aplicación, particularmente en el diseño de la interfaz con el usuario, sin ser necesario dejar al profesor acceso a todo el código. Este enfoque permite automatizar la generación de parte del código a partir del diseño, acortando significativamente el costo y tiempo de producir una aplicación. Un enfoque similar es utilizado por el sistema ScriptWriter [Horo 87]

2.1 Herramientas de Software

El software que se utiliza para diseñar la interfaz con el usuario, lo denominamos **herramienta**. Una herramienta es un programa interactivo con el cual un profesor interactúa para especificar su diseño. Una vez especificado, la herramienta genera código para implementar el diseño. En el código generado se incluye en uno o más módulos de software que son posteriormente acoplados a la aplicación.

Algunos ejemplos de herramientas de software útiles para el desarrollo de aplicaciones CAI son las que se utilizan para: la construcción de menús tipo cortina; la definición de ventanas a fin de poder manejar áreas individuales de la pantalla como si fuesen pantallas completas; el diseño de íconos para construir figuar y especificar su desplazamiento en las pantallas; la definición de diálogos para simplificar el proceso de ingreso de opciones y datos por parte del usuario al interactuar con la aplicación , etc.

Este enfoque tiene diversas ventajas. En primer lugar permite la generación rápida de aplicaciones, con un mínimo de funcionalidad, pero con una interfaz completa, a fin de que los usuarios experimenten con ésta y propongan los cambios necesarios. La interfaz es fácilmente rediseñada, generando nuevos módulos con código y el profesor puede ir agregando paulatinamente nuevas funciones. En segundo lugar, la generación de código automático disminuye los errores, ya que éstos son poco a poco eliminados del módulo correspondiente en cada herramienta a medida que son reusadas. En tercer lugar, las herramientas facilitan al profesor la construcción de diversas aplicaciones con cierta estandarización de la interfaz con el usuario, disminuyendo para éstos el costo de aprendizaje de cada nueva aplicación del profesor.

2.2 Componentes de Software

Junto con las herramientas, que generan modulos de software a partir de un diseño, se encuentran las **componentes** de software. Una componente es un módulo de software que cumple una función específica, tiene una clara definición para facilitar su incorporación a una aplicación, pero no es modificable sino por medio de cambios en su implementación. A diferencia de las herramientas, los componentes son "cajas negras", normalmente no modificables.

Típicamente, en cualquier aplicación orientada a CAI, se distinguen 3 grandes categorías de componentes de software :

- componentes de software generales de CAI
- componentes de software del dominio de la aplicación
- componentes específicas de la aplicación

Dado que un objetivo importante es que la aplicación pueda ser desarrollada por las mismas personas que estarán involucradas en su uso, es decir los profesores, es necesario separar conceptualmente estas componentes y en lo posible tratar de reducir al mínimo las del último tipo. Lo ideal es que el profesor encuentre, entre las componentes del primero y segundo tipo todos los elementos para construir un porcentaje significativo de la aplicación.

2.2.1 Componentes de Software Generales de CAI.

Dentro de esta categoría se encuentran todas las componentes básicas del ambiente de apoyo al desarrollo de aplicaciones. Ejemplos típicos de este tipo de componente son aquellas que se requieren en apoyo de los módulos generados por las herramientas como el manejador de mouse y el manejadore de íconos.

Otras componentes son el manejador de gráficos, editor de propósito general, el manejador de la persistencia de los datos, componentes matemáticos, estadísticos y de monitoreo.

El manejador de mouse permite controlar las acciones que el usuario indica con este dispositivo, como por ejemplo, al seleccionar un ítem en la barra de menús. El manejador de íconos permite interactuar con las pequeñas figuras o íconos que han sido incorporados a la aplicación, por ejemplo en lugar de poner un texto que diga que se está trabajando con un árbol binario, es preferible mostrar el ícono correspondiente.

El manejador de gráficos permite mostrar gráficamente los valores especificados en forma tabular e incluye diversos tipos de gráficos, tales como barras, tortas y líneas y formatos de presentación. El editor de propósito general permite manejar todo el ingreso de texto en forma simple y homogénea, e incluye operaciones de cortar, pegar, etc. El manejador de la persistencia de los datos administra almacenamiento y recuperación de información desde ventanas de texto y gráficas. Finalmente, la componente de monitoreo permite almacenar el trabajo efectuado por cada usuario con la aplicación a fin de que el profesor pueda analizarlo.

2.2.2 Componentes de Software del Dominio de la Aplicación.

Son las componentes que están íntimamente ligadas al tipo de aplicación que se está construyendo, pero que podrían ser reutilizadas en otras aplicaciones del mismo tipo. Por ejemplo, si el dominio es el de las estructuras de datos, podrían tenerse componentes como: lenguaje algorítmico, manejador de objetos gráficos , etc. El lenguaje algorítmico permitiría especificar en un lenguaje de muy alto nivel (similar al pseudo-código usado en los textos), un cierto algoritmo. El programa interpretaría este código para ejecutarlo directamente, pudiendo así mostrar en forma paralela este código simplificado, y lo que ocurre con los datos. El manejador de objetos gráficos simplificaría la tarea de manipular los objetos involucrados en una animación.

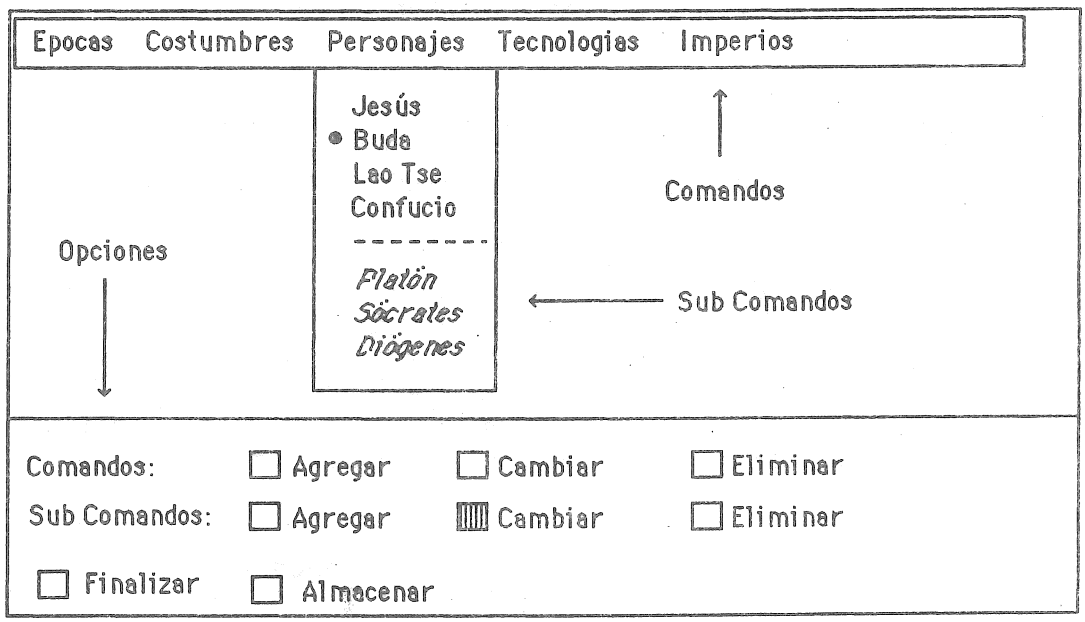
2.2.3 Componentes Específicas de la Aplicación.

En esta categoría entra el resto del código necesario para construir la aplicación. Normalmente incluye software de enlace para unir las distintas componentes que hayan sido incorporadas de los dos primeros grupos, y software específico que resuelve problemas muy particulares y que por lo tanto no puede ser considerado en la segunda categoría. En la medida que la base de software de los dos primeros grupos sea suficientemente completa, el trabajo necesario para hacer la aplicación será reducido.

2.3 Ejemplo de Herramienta: Menús de tipo Cortina

En el caso de diseño de menús de tipo cortina, la herramienta permite especificar la ubicación y contenido de cada uno de los menús. El código generado contiene dos tipos de módulos de software. Un grupo de módulos que maneja los menús activados por una mouse o a través de teclas y otro grupo de módulos que contiene un esqueleto de programa que debe ser completado por el profesor con el código que implementa los comandos de su menú. Parte de éste código es generado automáticamente como es el caso de los comandos para abrir y cerrar archivos, almacenar el contenido de ventanas de texto y gráficas, etc.

La siguiente figura muestra un ejemplo de diseño de menús de una aplicación cualquiera:



El profesor desarrolla su diseño a través de opciones activadas con la mouse o con teclas. Por razones de espacio, en la figura no se muestran otras opciones, como por ejemplo, para indicar si un subcomando está activo en cierto momento, para separar grupos de subcomandos, etc.

3. CASO DE ESTUDIO: UN LABORATORIO DE ESTRUCTURAS DE DATOS.

3.1 Motivación.

La capacidad de estimación aproximada, o de orden de magnitud, de lo bueno o malo que resulta una cierta estructura de datos, es ciertamente difícil de inculcar a los estudiantes en un curso semestral de estructuras de datos [Navo 86] [Navo 87].

Por ejemplo, al tomar un caso tan simple como un árbol de búsqueda binario, difícilmente podrán contestar "a priori" la altura que tendrá un árbol de 10.000 nodos. Se les enseña que la altura, o número de accesos es una función logarítmica del número de nodos (O(log n)). Sin embargo, también se les dice que en el peor caso se llega a una estructura de lista (árboles degenerados). Al calcular el rango de variabilidad para la altura el estudiante obtiene que el número de accesos puede ser desde 14 ($\log_2$ 10000) hasta 10.000 ! La pregunta inmediata es ¿y en un caso práctico cuánto será ?

Una ayuda muy importante en la superación de este tipo de dificultados puede lograrse mediante la generación de un ambiente experimental, donde el alumno pueda observar, y comparar el comportamiento de las estructuras de datos mas conocidas frente a las operaciones básicas. En realidad, casi las mismas razones que hacen aconsejable el uso de laboratorios para aprender Física o Química se mantienen en este caso [Balk 85] [Boot 86].

La idea es efectuar experimentos con volúmenes grandes de datos y poder confrontar así los resultados experimentales con los teóricos aprendidos en las clases. El alumno, despues de la clase teórica sobre árboles binarios se dirige al laboratorio de estructuras (microcomputadores equipados con el software adecuado), y realiza experimentos reales generando árboles con 1.000, 100.000 o 1.000.000 de elementos. Podrá observar entonces, la altura de ellos cuando las llaves son escogidas en forma random o con alguna otra distribución en particular y obtener de este modo una mejor comprensión de cómo se comporta un árbol de búsqueda binario en la realidad.

3.2 Requerimientos Básicos.

Para satisfacer los objetivos anteriores, el software debe satisfacer algunos requisitos fundamentales :

1) Facilidad de Diseño de Laboratorios "Ad Hoc".

El instructor debe poder construir o modificar un ambiente de laboratorio de acuerdo a sus propios objetivos. Lo que se desea es un minilaboratorio a la medida, de acuerdo a los objetivos que se plantee el instructor para la unidad correspondiente.

Por ejemplo, si se están enseñando estructuras de árboles balanceados, el instructor podría "armar" un laboratorio que incluya árboles AVL, árboles 2-3 y árboles binarios simples como referencia para las comparaciones. No sólo las estructuras son piezas intercambiables sino también distribuciones de datos, operaciones y observación de resultados.

2) Interfaz gráfica autoexplicativa.

El alumno debiera poder operar el programa casi sin necesidad de apoyo de documentación

auxiliar (manuales), sólamente basado en lo que observa en pantalla [Böck 86] [Niev 86]. Esto implica una clara preferencia al uso de interfaces gráficas, "pull down menus", ventanas, etc, frente a interfaces de tipo comando en base a texto.

3) Agilidad del ciclo preparación - prueba - observación.

Si queremos que el alumno explore y experimente, debe poder hacerlo con facilidad. Un ejemplo resulta clarificador de este punto. Supongamos que el alumno está investigando el comportamiento de las tablas de hashing para lo cual carga inicialmente la tabla al 80% y luego realiza búsquedas observando el número de accesos requerido. El impulso inmediato es probar que pasaría con factores de carga iniciales diferentes, pero el alumno solo lo hará si eso no le significa especificar nuevamente todos los parámetros para posteriormente esperar 10 minutos por los nuevos resultados. Si por el contrario, el ciclo es ágil, en unos minutos habrá probado como afecta esta variable al desempeño de esta estructura.

4) Volúmenes de Datos grandes.

Una de las motivaciones fundamentales para hacer un experimento de este tipo con el computador, es el poder "poblar" las estructuras con volúmenes de datos acorde a la realidad, es decir, grandes. En contraposición a los ejemplos triviales presentados en clases, i.e., árboles con 4 a 8 nodos, el alumno puede ahora insertar 10.000 o mas nodos en el árbol lo cual es obviamente imposible de hacer sin esta herramienta.

5) Observación gráfica del desempeño.

En lo posible, representar en forma gráfica los resultados de los experimentos en lugar de tabular números (una imagen vale mas que mil palabras). Esto contribuye tambien a agilizar el ciclo, de acuerdo a lo planteado en el tercer requerimiento.

6) Reforzamiento de Conceptos.

Despues de observar el desempeño de la estructura, el alumno debe sacar algunas conclusiones de lo que vió. En este momento debe apoyarse con el conocimiento teórico que hay tras el experimento realizado mediante una precisa exposición del comportamiento esperado.

3.3 Componentes del Software.

3.3.1 Componentes de Software Generales de CAI.

En esta categoría tenemos las componentes que ya han sido mencionadas en el punto anterior:

- manejador de mouse
- manejador de gráficos
- manejador de la persistencia
- editor de propósito general
- constructor de íconos

y los componentes generados por las herramientas para manejar menús, ventanas, diálogos e íconos.

3.3.2 Componentes de Software del Dominio de la Aplicación.

En este caso, el dominio es el de las estructuras de datos, y las componentes necesarias:

- estructuras
- operaciones
- datos
- distribuciones
- observaciones

Las estructuras son las distintas construcciones con las cuales será posible trabajar: árboles, listas, stacks, AVL's, B-trees, etc. Las operaciones se refieren a las distintas cosas que se podrá hacer sobre los datos de las estructura: insertar, buscar, eliminar, rotar, dividir, mezclar, etc. La componente datos permite seleccionar el tipo de dato que se maneja al interior de las estructuras: enteros, reales, strings, dibujos, etc. Las distribuciones, muy relacionadas con los datos, permiten extraer modos de generación de datos en volúmen: uniforme, normal, secuencias, etc. Las observaciones se refieren mas bien al tipo de variable que se observará como resultado de los experimentos: número de accesos, número de comparaciones, número de llamadas de procedimientos, etc.

3.3.3 Componentes Específicas de la Aplicación.

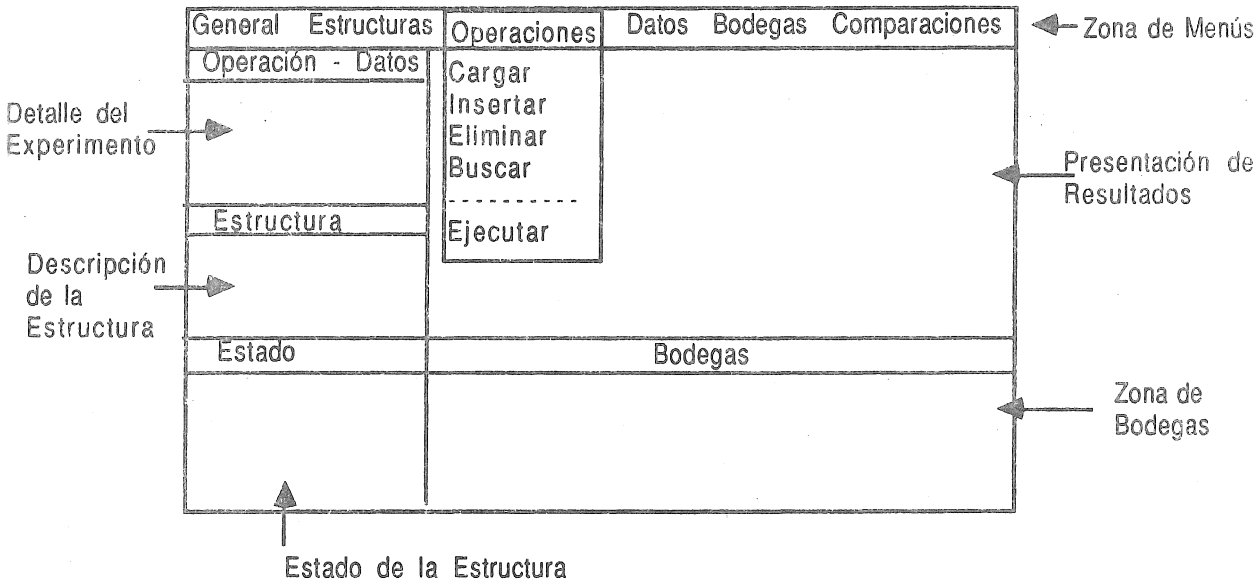
No detallaremos aquí este tipo de componentes por ser muy particular de la aplicación. Corresponde al resto del código que permite construir la aplicación que se describe.

3.4 El Laboratorio.

Supondremos que hemos efectuado ya la etapa de "armado" del laboratorio para lo cual seleccionamos de entre las componentes descritas en el punto anterior lo siguiente :

Estructuras - B-trees y tablas de hashing
Operaciones - Buscar, insertar y eliminar (en ambos casos lo mismo)
Datos - números enteros
Distribuciones - uniforme y secuencial (ordenada)
Observaciones - Número de accesos

Luego de completada la etapa de armado, obtenemos un módulo ejecutable que contiene el ambiente de laboratorio que el instructor ha seleccionado para sus alumnos. Al activarlo vemos una pantalla como lo ilustra la figura.



Existen en esta pantalla 6 ventanas o áreas fundamentales :

1. La zona de los menús.-
   Desde aquí se controlan todas las acciones a seguir. En este caso se definieron 6 menús (utilizando la herramienta descrita anteriormente)

   i) General - Contiene comandos de ayuda (help), comandos para "guardar" el laboratorio en el estado en que se encuentra para uso posterior, comandos para cargar un laboratorio que ha sido guardado, y comandos para terminar el programa.

   ii)Estructuras - Contiene las distintas estructuras con las cuales se puede experimentar. En el estado actual este menú tiene dos items : árboles binarios y tablas de hashing.

   iii) Operaciones - Contiene las operaciones básicas que se efectúan sobre la estructura: insertar, borrar y buscar, ademas de un comando para cargar inicialmente la estructura. Tenemos en este menú tambien el comando para comenzar la simulación.

   iv) Datos - Todo lo relacionado con el dominio y número de datos además de la distribución que siguen los datos que están siendo insertados (uniforme, normal, secuencial).

   v) Bodegas - Permite hacer el manejo comparativo entre varios experimentos relacionados entre sí por el cambio en algún parámetro, o bien entre estructuras distintas con el mismo experimento.

   vi) Comparaciones - Permite controlar la observación de los resultados (presentación es en forma gráfica).

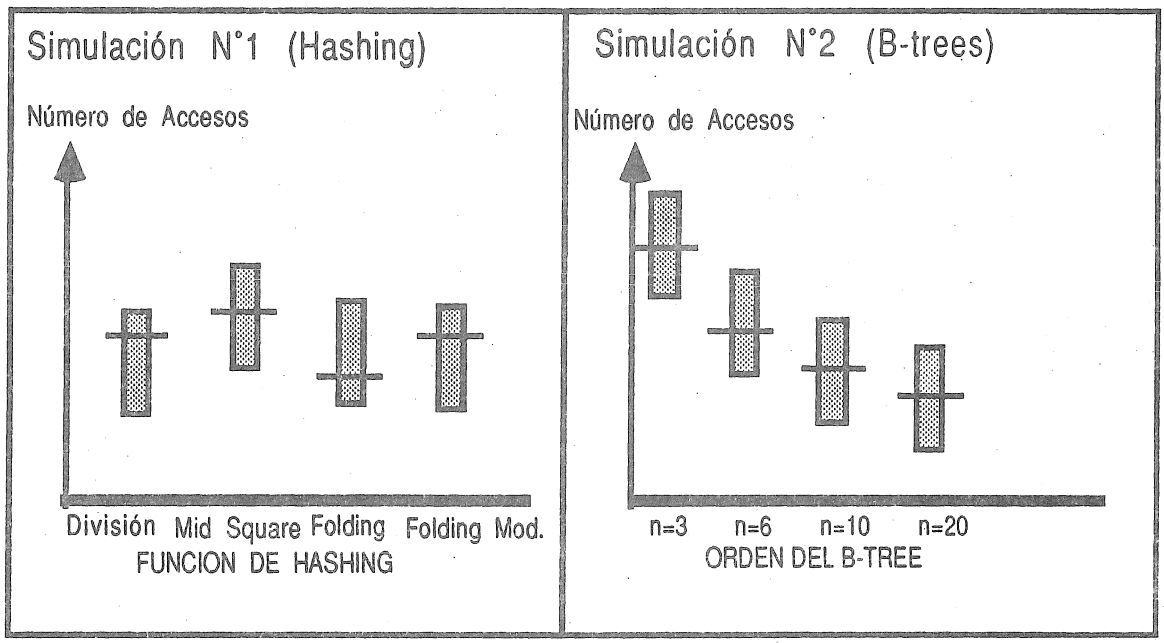
2. La zona del detalle del experimento.
   Aquí aparece el detalle del experimento o simulación que se está efectuando.

3. La zona de descripción de la estructura.
   Aquí aparece el detalle de la estructura que está siendo estudiada. Nombre de la estructura y todos los parámetros que definen la instancia del experimento.

4. La zona del estado de la estructura.
   Aquí se encuentra el detalle del estado de la estructura en un cierto instante. Por ejemplo si se trata de árboles binarios, el número de llaves, la altura, etc.

5. La zona de bodegas.
   Cuando se desea comparar el desempeño de dos o mas estructuras, se puede almacenar el resultado de un experimento o simulación en las llamadas "bodegas" y repetir la simulación cambiando la estructura. El área de bodegas ilustra el estado de las bodegas, es decir los distintos resultados que podríamos analizar comparativamente.

6. La zona de resultados.
   Es el sector donde se observa el resultado de la simulación efectuada. Es posible observar gráficamente el desempeño de las estructuras: cómo varía al aumentar el volumen de datos o frente a cambios en algunos parámetros (ver figura).



4. CONCLUSIONES

El ambiente de desarrollo de software presentado ofrece un conjunto de herramientas de diseño y componentes de software para la construcción rápida de aplicaciones CAI. Este software ha sido probado para armar laboratorios de estructuras de datos que prueban la utilidad de este enfoque. Se espera ampliar el potencial de las herramientas de diseño para aumentar el grado de automatización en la generación de software educacional.

AGRADECIMIENTOS

Este trabajo ha sido realizado gracias al apoyo de la Dirección de Investigación de la Pontificia Universidad Católica de Chile (DIUC, Proyectos 15/1987 y 22/1987), del Consejo Nacional de Ciencia y Tecnología (Proyecto 757) y de IBM de Chile.

5. REFERENCIAS

[Atki 83] ATKINSON, M.P., BAILEY,P.J., CHISHOLM,K.J., COCKSHOTT,W.P., MORRISON, R. (1983) An Approach to Persistent Programming. En **PS-Algol Papers**. University of Edinburgh, Department of Computer Science, PPR-2-83.

[Balk 85] Balkovich E., Lerman S., Parmelee R. (1985) Computing in Higher Education: The Athena Experience". **Communications ACM Vol 28** , Nro 11, Nov , 1214-1224.

[Böck 86] Böcker Heinz-Dieter, Fisher Gerard, Nieper Helga (1986) The Enhancement of Understanding through Visual Representations. **Conference Proceedings on Computer and Human Interaction.** CHI 86 Abril,6 44-50.

[Boot 86] Booth T. et al. (1986) Design Education in Computer Science and Engineering **Computer**, Junio, 20-27.

[Gold 83] GOLDBERG, A., ROBSON, D. (1983) **Smalltalk-80. The Language and its Implementation.** Addison Wesley, Reading.

[Horo 87] HOROWITZ, E. (1987) **ScriptWriter Reference Manual.** Computer Science Department, University of Southern California, Los Angeles, Ca.

[Hulm 85] HULME, C. (1985) Reading: Extracting Information from Printed and Electronically Presented Text. En **Fundamentals of Human-Computer Interaction**, Andrew Monk (Ed.), Academic Press.

[Navo 86] NAVON J. (1986) Aprendizaje de Estructuras de Datos mediante la Experimentación" **II Encuentro Nacional de Educación y Computación**, Viña del Mar, Chile.

[Navo 87] NAVON J. (1987) Basis for a Data Structures Laboratory **11th Educational Computing Conference**, San Francisco, 19-20 Nov iembre.

[Niev 86] NIEVERGELT, J., VENTURA, A., HINTERBERGER, H. (1986) **Interactive Computer Programs for Education. Philosophy, Techniques and Examples**. Addison-Wesley, Reading.

[Pape 80] PAPERT, S. **Mindstorms: Children, Computers and Powerful Ideas.** Basic Books, New York, 1980.

[Reid 85] REID, P. (1985) Work Station Design, Activities and Display Techniques. En **Fundamentals of Human-Computer Interaction.** Andrew Monk (Ed.), Academic Press.

[Shoo 83] SCHOOMAN, M. (1983) **Software Engineering. Design, Reliability and Management**. McGraw-Hill, Nueva York.

[Slee 82] SLEEMAN, D., BROWN, J.S. (Eds.) (1982) **Intelligent Tutoring Systems**. Academic Press, London.

[Somm 85] SOMMERVILLE, I. (1985) **Software Engineering**. 2ª edición, Addison-Wesley, Wokingham.

UNIVERSIDAD DE LOS ANDES

FACULTAD DE INGENIERIA

DEPTO. DE SISTEMAS

TITULO : Origen, desarrollo e implementación de un sistema computacional de apoyo a procesos de enseñanza-aprendizaje

AUTOR : Giovanni Seni Zambrano

DIRECCION : Cra 6 # 8337 apt(501)
BogotáColombia

Este artículo presenta el desarrollo de un proyecto computacional para ilustar que los computadores pueden contribuir significativamente a resolver algunos de los problemas del sector educativo. Asi mismo, intenta mostrar la importancia de orientar la computación educativa a atender necesidades educativas concretas y locales a una institución. Complementariamente soporta la idea de que para maximizar el uso de la computación moderna, con provecho educativo, es necesario una relación e interacción entre los sectores involucrados : Educación y Computación.

MARCO DE REFERENCIA

El proyecto " Comprendemos al Leer" surgió para responder a dos necesidades diferentes : por una parte como compromiso académico del curso E.A.C. de la Universidad de los Andes y por otro lado como respuesta al interés y espectativas de una institución educativa de la ciudad.

Una institución conciente de que el computador ha conquistado un sitio en las escuelas de todo el mundo desarrollado se apresuraba a constituirse en privilegiada protagonista de la revolución informática.

No faltaron los temores dentro de la institución sobre las consecuencias que el proyecto podría tener en una actividad tan tradicional como la del proceso enseñanza-aprendizaje.Sin embargo, no es posible estar ausente de este increible proceso causado por el impacto del computador en la educación.

Con la moderna tecnología se puede mejorar, pero también debilitar, la cultura local cuando se la usa simplemente para transplantar elementos culturales foráneos. El proyecto debía constituirse en respuesta a una necesidad concreta y local a la institución.

EL COMPUTADOR COMO SOPORTE A UNA NECESIDAD EDUCATIVA

Sin duda , son muchas las áreas de educación en las cuales el computador puede convertirse en herramienta eficaz y recurso para el desarrollo de destrezas intelectuales. Se seleccionó el área de español en los cursos intermedios de educación básica primaria.

Como resultado de un estudio sobre identificación de necesidades de apoyo computacional en esta área y población objetivo, se encontro dificultad en los niños para extraer las ideas principales que se transmiten a través de un texto.

Se tenía ahora una población objetivo, un área y un tema para el proyecto.

ENFOQUE Y ESTRUCTURA DEL PROYECTO

El siguiente paso fué recoger las directrices metodológicas que tienen en cuenta la edad mental y cronológica de los niños asi como las diferentes teorías que existen como sustento al diseño de ambientes de aprendizaje.

En ningún momento se pretendía el aprendizaje repetitivo de conceptos, sino la utilización del computador como recurso invaluable que permitiera a los niños desarrollar habilidades y destrezas necesarias para leer comprensivamente.

No es fácil entrar en el corazón de los mecanismos que los niños utilizan para aprender a leer pero sí es claro que no se trata de un proceso de educación vertical, centrado en transmisión de conocimientos.

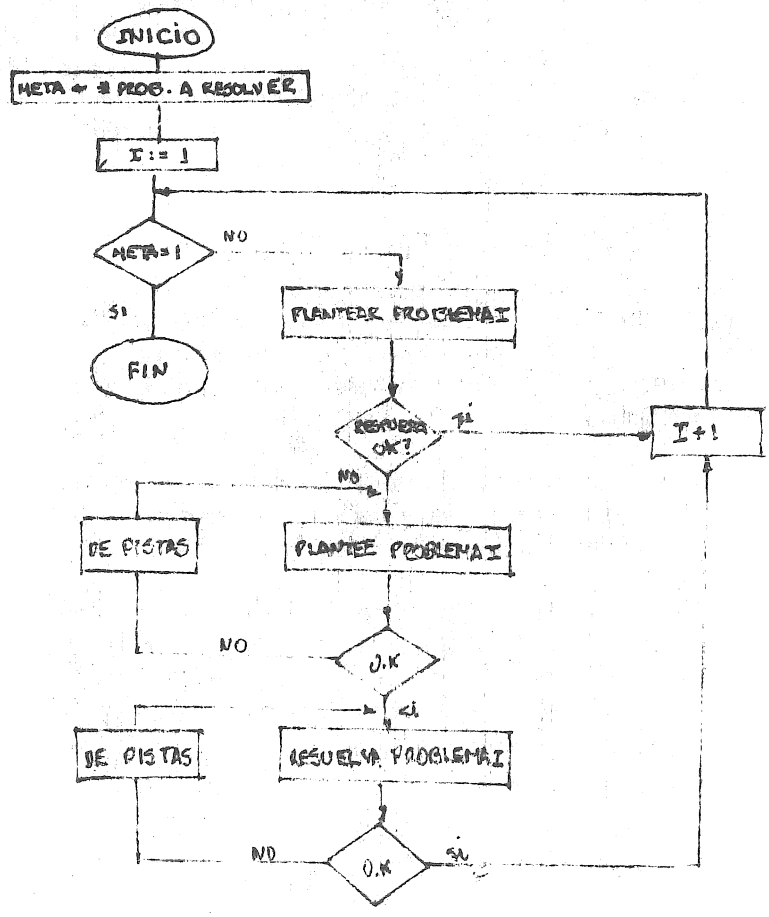
Esta consideración nos hizo concebir un sistema que, antes que transferir e imponer las ideas del educador hacia el alumno, le ayudara a desarrollar sus procesos de pensamiento.

Fue ideado un sistema que le ofreciera al alumno la posibilidad de ejercitar y practicar las destrezas que estaban siendo aprendidas en la clase.

Proporcionar varios niveles de pistas al alumno que le permitan ir descubriendo la solución de los ejercicios fue la siguiente decisión sobre el enfoque que tendría el proyecto.

De esta forma "Comprendemos al leer" puede clasificarse como un sistema de ejercitación y práctica con retro información heurística.

El macro-algoritmo general para este tipo de aplicación puede verse de la siguiente forma :



El sistema "Comprendemos al Leer" no fuerza al alumno a fijarse una meta sino, por el contrario, le permite abandonar su uso en el momento que lo desee. Sin embargo está diseñado de forma tal que una historia sobre el desempeño del alumno es guardada. Así el alumno podrá utilizar el sistema desde el lugar donde lo abandonó la última vez.

El macro-algoritmo presentado puede hacer creer que se dispone de infinitas pistas para proporcionarle al alumno cuando se equivoque. Ciertamente esto es imposible en una aplicación real donde un número máximo de niveles de ayuda debe determinarse.

El sistema "Comprendemos al Leer" proporciona tres niveles de ayuda diferente asociados a cada una de las preguntas.

TEORIAS DE APRENDIZAJE QUE INFLUENCIARON AL PROYECTO

Mencionemos a continuación las ideas, extraidas de las diferentes teorías de aprendizaje, sobre las cuales el diseño del proyecto está apoyado.

Que los ejercicios se constituyan en algo motivante que provoque expectativas en el alumno y sean significantes para el (Gestalt), que el contenido se presente de una buena forma : animada y novedosa (En buen Gestalt), que cada alumno marche a su propio ritmo recibiendo un tratamiento individual (Gestalt) y enfrentando un programa de ejercitación que conduzca al alumno escalando por diversos niveles de dificultad poco a poco (Conductismo : Programación de la Instrucción) y recibiendo reconocimiento cada vez que responda bien (Conductismo) son la mejor justificación al micro-mundo utilizado en el sistema.

LA IMPLEMENTACION DEL PROYECTO

El proyecto "Comprendemos al Leer" tiene como punto de partida, para la implementación, al programa "Voy Parriba". Este programa había sido desarrollado por Juan C. Rodríguez y Moisés Brainsky con un micromundo que encajaba de excelente forma a las necesidades de "Comprendemos al Leer".

En el micro-mundo de "Comprendemos al Leer" se presenta un edificio de ocho pisos cada uno de los cuales corresponde, en su orden, a lecturas de diferentes grados de dificultad.

Todos los cuartos de un piso deben ser recorridos (Contestar satisfactoriamente las preguntas sobre el texto en cuestión) antes de poder pasar al siguiente piso. El objetivo es llegar al último piso del edificio para dar un paseo en helicóptero.

No es fácil definir cuál debe ser la conducta de entrada de los alumnos que van a utilizar el sistema pues de acuerdo con el grado de dificultad del texto analizado, se requerirá el dominio por parte del alumno de diferentes conceptos.

El sistema está provisto de mecanismos que le permiten al personal docente, encargado del sistema, modificar parcial o totalmente todos los textos, preguntas, respuestas y pistas manejados en el micro-mundo.

De esta forma los profesores podrán cambiar dinámicamente las lecturas de acuerdo con los grados de dificultad deseados.

También son usados a lo largo del sistema recursos visuales (colores) y sonoros para mantener atraída la atención del estudiante.

Los efectos sonoros son útiles para propósitos de refuerzo (cuando la pregunta es satisfactoria) o para mantener de manifiesto una respuesta no acertada.

A lo largo del sistema se hace uso de ventanas para presentar el micro-mundo y para comunicarse con el alumno (presentación de preguntas, captura de respuestas, mensajes de error, etc.).

SUBSISTEMAS QUE INTEGRAN AL PROYECTO

Para terminar esta descripción a grandes rasgos del origen y características del proyecto "Comprendemos al Leer" podemos mencionar los diferentes sub-sistemas que lo componen.

Un sub-sistema de ayuda dirigida a los alumnos. No es un tutorial sobre como manejar el sistema sino la presentación de algunas de sus características para familiarizar al lumno con el programa antes de ser utilizado por primera vez.

Un sub-sistema para jugar. Aquí se despliega el micro-mundo, las lecturas y las preguntas que debe enfrentar el alumno.

Recordemos aqui que este software está diseñado para permitirle al usuario controlar el ritmo de avance en el material. Por esta razón, siempre aparecerá un mensaje explicando que comandos (teclas) son permitidas para continuar.

El juego está provisto de tres niveles de información de retorno asociados a cada pregunta. Luego de agotados estos tres niveles, se desplegará el último nivel de retro-información hasta que el alumno escoja la respuesta correcta o tome la opción de abandonar.

Esta restricción de implementación resalta la importancia del personal docente en el éxito del sistema. La información de reforno deberá diseñarse cuidadosamente para una buena utilización del proyecto.

El sub-sistema de listar alumnos permite ver por pantalla el código o nombre y el último piso alcanzado por cada usuario que se encuentre registrado en el sistema.

El sub-sistema borrar alumnos permite retirar la historia de un alumno del sistema.

El sub-sistema de manejo de textos permite consultar y/o modificar la información de retorno asociada a las preguntas; consultar y/o modificar las diversas preguntas y respuestas y manejar las lecturas.

En este punto podemos descubrir que tanto los profesores como los alumnos seran usuarios del sistema, por tanto un mecanismo de claves es provisto para que los estudiantes no tengan acceso a los módulos reservados para el uso del personal docente.

RELEVANCIA DE ESFUERZOS

Finalmente cabe anotar que "Comprendemos al Leer" ha sido escrito en lenguaje Pascal UCSD y diseñado para utilizarse en computador Apple II plus o compatible con 64K de memoria principal y con al menos una unidad de disco.

A pesar de las grandes limitaciones en recursos, que esta máquina y lenguaje representan, el proyecto "Comprendemos al Leer" es un buen testimonio de las cosas que se están haciendo en la Universidad de Los Andes bajo la dirección del Profesor Alvaro Galvis.

Gracias a su asesoría y enseñanzas "Comprendemos al Leer" es una muestra de lo que llamaríamos acertada recepción tecnológica. Acertada porque fue desarrollado de acuerdo a las circunstancias particulares de una institución y porque fue la unión de los esfuerzos de profesionales de la educación y de la informática.

SISTEMA EDUCATIVO COMPUTACIONAL PARA EL SUMINISTRO DE ANESTESIA

Ing. Sara Croitoru Eisenstein
Ing. Vivian Luchau Botero
Doctor Jorge Mejía

DEPARTAMENTO DE INGENIERIA DE SISTEMAS
UNIVERSIDAD DE LOS ANDES
BOGOTA, COLOMBIA

RESUMEN:

El objetivo de este sistema es ofrecer una herramienta educativa que permita, tanto al Médico Anestesiólogo como al estudiante de medicina, practicar sus conocimientos de suministro de anestesia en un paciente hipotético durante la simulación de una cirugía. Así, el usuario puede reconocer los efectos de sus acciones sobre el paciente sin los riesgos que esto supone en la vida real.

El sistema está orientado a dos tipos de usuarios: un "experto", que lo alimenta con el conocimiento médico necesario y un usuario "estudiante", que administra anestesia al paciente durante la simulación de la cirugía. Para el usuario "experto" se desarrollaron herramientas que permiten editar el conocimiento y para el "estudiante", un programa que lleva a cabo la simulación.

Para obtener los efectos de las drogas inhaladas e intravenosas en el paciente se utilizaron dos modelos farmaco-cinéticos, los cuales fueron implementados mediante técnicas de control. Para combinar dichos efectos, se utilizaron técnicas de sistemas expertos que permitieron separar el conocimiento del programa, facilitando así las labores de validación y complementación del conocimiento.

Actualmente, las herramientas computacionales para el funcionamiento del sistema se encuentran desarrolladas; sinembargo, debido a la complejidad y cantidad del conocimiento involucrado, no ha concluido la tarea de obtención y validación de este.

El sistema fué desarrollado en Turbo Pascal y funciona bajo sistema operacional DOS en máquinas IBM PC y compatibles con 512K de memoria principal.

Introducción

El uso de los computadores en medicina ha adquirido cada vez más importancia debido a la progresiva complejidad de la práctica médica y gracias al desarrollo de herramientas, sencillas de manejar, que apoyan al médico en su labor diaria.

Dentro de la medicina los sistemas computacionales han tenido una buena acogida en el campo de la educación. Se destacan especialmente los Sistemas Expertos que guían al estudiante en la labor de diagnóstico, y los Simuladores los cuales permiten enfrentar al estudiante con situaciones clínicas especiales.

Particularmente, en el área de anestesia se han desarrollado sistemas que apoyan en la determinación del plan anestésico y en la administración de anestésicos inhalados. Luego de revisar bibliografía referente a dichos sistemas, decidimos utlizar otro enfoque.

Inicialmente nos planteamos como objetivo desarrollar un sistema capaz de simular algunos de los aspectos del cuerpo humano sometido a una cirugía y bajo el efecto de las drogas que comunmente se administran durante la anestesia (anestésicas o no), con el objeto de permitir a los médicos y/o estudiantes de medicina observar las acciones que estos medicamentos despiertan en el organismo.

De esta manera, al disponer de un paciente simulado, el usuario puede practicar sin riesgos sus conocimientos y aprender a reconocer los efectos de sus acciones sobre el paciente, para diferenciarlos de aquellos derivados de otros factores (como, por ejemplo, enfermedad); adquiriendo así una actitud vigilante en cuanto a las variaciones que pueda inducir sobre sus pacientes. Además, una vez el sistema cuente con una base de conocimiento más completa, el usuario podrá aprender cómo reaccionan ante la anestesia pacientes con alteraciones fisiopatológicas poco comunes, casos a los cuales el médico puede no tener acceso durante su período de entrenamiento formal.

Una vez definidos los objetivos del sistema, decidimos utilizar modelos farmacológicos desarrollados por expertos, validados y publicados. Así, escogimos el modelo de drogas inhaladas de W.W. Mapleson (1976) y un modelo farmacocinético bicompartimental publicado por M. Wood (1985).

Luego, desarrollamos un modelo fisiológico basados en la recopilación de información proveniente de textos, consultas con expertos y artículos.

Dadas las características de la información recopilada, decidimos que lo más conveniente era desarrollar, desde el punto de vista computacional, un sistema compuesto por:

- técnicas de control para el manejo de los modelos farmacocinéticos (el inhalado y el intravenoso)

- un sistema experto que, a partir de los datos generados por los modelos, simulara el efecto de las drogas y procedimientos sobre el paciente.

Con la finalidad de reforzar el carácter educativo del sistema, se diseñó un módulo que permitiese mantener un registro de los cambios producidos durante la simulación, con el fin de permitir al usuario entender las alteraciones producidas por los medicamentos sobre las variables del paciente.

Por ser el sistema modular, el usuario tiene la posibilidad de modificar el conocimiento de tipo médico para actualizarlo y ampliarlo con la nueva información que vaya obteniendo.

Objetivos del Sistema

El objetivo fundamental del sistema MECCA es contribuir en la educación de los médicos para el suministro de anestesia, ofreciendo al usuario otra alternativa para practicar sus conomientos teóricos, con la garantía de eliminar el riesgo que esto implica para un paciente, presentándole un caso hipotético en el cual el computador tiene el papel de paciente y el usuario el de médico anestesiólogo.

El objetivo del sistema no es ensenar al estudiante cómo determinar un plan anestésico, sino permitirle aplicar sus conocimientos planteando él mismo el plan anestésico y observando el efecto que tiene sobre el paciente. Se busca ofrecerle un modelo que le permita realizar una estrategia adecuada para la vigilancia sobre el paciente durante la anestesia, luego de haber elaborado un plan anestésico y observar los efectos de sus acciones sobre el paciente hipotético.

En general, la contribución educativa que ofrecerá el sistema al estudiante es permitirle practicar otro tipo de razonamiento diferente al que él utiliza en sus clases teóricas ; cuando el estudiante adquiere una serie de conocimientos teóricos se encuentra en un ambiente de estudio en el cual se utiliza un razonamiento que va desde las causas a las consecuencias (de la enfermedad a los síntomas), pero en el ambiente de trabajo el razonamiento es al inverso, se reconoce la causa a partir de las consecuencias (se diagnostica la enfermedad a partir de los síntomas). Esto se refuerza gracias a que el sistema permite llevar un registro de los cambios ocurridos a las variables (es decir, de los cambios de estado sucesivos del paciente), para que el usuario no experto analice y verifique el porqué de una reacción del paciente, ofreciéndole una explicación teórica que justifique dicho cambio.

El sistema le ofrece al profesor (médico experto) la posibilidad de complementar sus clases teóricas, con casos en los cuales se puedan destacar los aspectos importantes sin poner en peligro la vida de un ser humano, y permitiéndole manipular las características del paciente.

Delimitación

La anestesia consiste en un bloqueo de la sensibilidad a los estímulos dolorosos.
Hay dos tipos de anestesia : General y Regional.

La anestesia general se define como un estado transitorio de inconsciencia producida por agentes anestésicos, con pérdida de las sensaciones en todo el cuerpo.

La anestesia regional es capaz de bloquear la conducción nerviosa de los impulsos. Esta acción es reversible y hay una recuperación completa de la función fisológica. Pero a diferencia de la anestesia general no produce estado de inconsciencia.

La administración de anestesia general requiere de un control más cuidadoso del estado del paciente debido a su complejidad y riesgo.
La anestesia regional es un procedimiento que requiere habilidad manual, la cual no se puede adquirir mediante el uso de un modelo computacional, sino mediante práctica con pacientes.
Gran parte de las cirugías se realizan con anestesia general, motivo por el cual en este sistema se escogió la técnica de anestesia general y por ende el tipo de cirugías que puedan requerir de ésta.

El control del cuerpo humano durante la anestesia es bastante complejo, por lo cual sería necesario tener en cuenta (medicamente) todo lo que ocurre en el paciente. Sinembargo por simplicidad académica, es necesario hacer algunas restricciones sobre los tipos de drogas a utilizar, los anestésicos, tipos de cirugías y variables a controlar.
De todas maneras la estructura del sistema MECCA está en capacidad de aceptar todo el conocimiento que el usuario experto desee tener en cuenta.

Se tomará como referencia una historia clínica definida por el usuario experto. En este caso se ha probado con un paciente sin ninguna otra complicación adicional al motivo de la intervención, sobre el cual se pueden realizar cirugías previamente definidas al sistema. (Actualmente se ha probado con apendicectomía).

El Sistema permite la administración del anestésico que el usuario seleccione, ya sea solo o asociado a otros anestésicos en cualquier momento del procedimiento.

Se tienen en cuenta variables de estado de tipo clínico y de tipo fisiológico. Las primeras son aquellas que se pueden medir directa y objetivamente sobre el paciente y cuya evaluación se realiza normalmente en una intervención, mientras que las segundas no se toman explícitamente en cuenta durante la intervención debido a su dificultad para ser medidas, pero se calculan mediante un modelo matemático previamente definido por el experto.

Estrategia Educativa Utilizada

El sistema es básicamente un simulador. Por este motivo está definido mediante una serie de estados en los cuales puede estar el paciente en un momento dado.

Es decisión del médico experto determinar las variables tanto clinicas como fisiológicas que le interesa tener en cuenta en un caso, seguramente con el propósito de hacer énfasis sobre estas, dejando de lado las que en dicho caso no tienen tanta importancia académica.

Se trata de simular los cambios que se producen sobre las variables de estado del paciente durante un procedimiento quirúrgico determinado, causados por :

- Efectos de la Cirugía.
- Efectos de cambios que haga el usuario en cuanto a la aplicación de anestésicos y oxígeno.
- Efectos producidos por las drogas que le sean administradas.
- Efectos producidos por los líquidos que le sean aplicados.

En un simulador, como su nombre lo indica, se espera que el comportamiento de aquello que se simula esté reflejado por el funcionamiento del sistema.

El experto (en este caso el médico anestesiólogo), que conoce muy bien la naturaleza del objeto de la simulación (el cuerpo humano durante la anestesia) y las leyes que rigen su funcionamiento prepara el conocimiento básico. Esto incluye la caracterización de las reglas de funcionamiento, y de las variables, los líquidos, drogas y anestésicos que se van a permitir durante la intervención. Pero él no determina la prescripción de una forma única para atacar el problema; tal cosa queda a voluntad del usuario estudiante.

Educativamente se busca que quien interactue con el simulador pueda aprender del sistema, sin las limitaciones o peligros que la vida real pueda tener. El sistema permite al usuario experimentar para aprender.

Cuando el estudiante aprende una serie de conocimientos en sus clases teóricas se encuentra en un ambiente de estudio en el cuál se razona de las causas a las consecuencias, es decir, se estudia un diagnóstico indicando cuales son sus síntomas asociados. Por el contrario en el ambiente de trabajo y en la práctica a partir de los efectos (síntomas) se deben descubrir las causas (diagnósticos).

Los sistemas de modelaje y simulación en medicina son muy importantes desde el punto de vista educativo puesto que permiten al estudiante ejercitarse en el tipo de razonamiento que deberá utilizar en la práctica y al cual no está acostumbrado: deducir un diagnóstico a partir de los síntomas en vez de tomar un diagnóstico y describir sus síntomas asociados.

El aprendizaje se realiza por descubrimiento porque al introducir cambios el usuario observa sus efectos pero no se le evalua. El usuario aprende de su interacción con el sistema.

El sistema será un complemento a las clases, al aprendizaje teórico y a las prácticas en el suministro de anestesia. Le servirá para llevar a cabo una práctica sin riesgo dando prioridad a lo que debe aprender en un caso específico.

Descripción General

El sistema MECCA es un modelo mixto que usa técnicas de modelos de control y técnicas de sistemas expertos.

Se implementaron 2 modelos farmacocinéticos : el inhalado y el intravenoso; basados en modelos matemáticos previamente definidos.

El modelo farmacocinético inhalado genera variaciones en la profundidad anestésica del paciente, la cual a su vez produce efectos directos sobre el estado del paciente, es decir, sobre las variables.

De manera similar el modelo farmacocinético intravenosos se basa en el cálculo de la concentración plasmática de la droga en el organismo. Dicha concentración afecta también el estado del paciente.

Estos dos modelos por sí solos son estáticos, por lo cual se requiere la prescencia de un modelo farmacocinético que refleje los cambios (sobre las variables de estado del paciente) producidos por estos modelos y adicionalmente los cambios de las variables que afectan otras variables. Este modelo farmacodinámico no es un modelo de control sino que está basado en técnicas de sistemas expertos. Es decir, se tiene una base de concimiento expresado mediante reglas de producción de las siguientes maneras :

- SI <causa> ENTONCES <efecto>
- HAGA QUE <efecto>

En la primera para que el efecto se lleve a cabo es necesario que se presente la causa, mientras que en la segunda forma el efecto se produce incondicionalmente.

Puesto que paralelamente a la anestesia se desarrolla la cirugía, ésta también debe ser simulada.

La cirugía esta definida como un conjunto de etapas y transiciones de una etapa a otra, ya sea por el transcurso del tiempo o por la ocurrencia de un evento especial.
Cada etapa tiene asociada una transición por tiempo y posiblemente transiciones por eventos.

Durante cada ciclo de ejecución del programa se chequea la transición a otra etapa; es decir se verifica si puede ocurrir una transición por un evento especial, si este no ocurre se verifica si por tiempo la etapa ha concluido.

Con el fin de generar un evento, la cirugía tiene asociada una probabilidad de que ocurran problemas durante su ejecución (se calcula aleatoriamente). A su vez las transiciones por evento tienen asociada una probabilidad de ocurrencia.

El procedimiento es el siguiente :

Si la etapa actual tiene asociadas transiciones por eventos se genera aleatorimente una probabilidad de problemas para el ciclo si dicha probabilidad es menor que la probabilidad de que se generen problemas durante la cirugía se busca generar un problema en la etapa actual. Por este motivo se suman las probabilidades asociadas a las transiciones por evento de la etapa actual y se genera una probabilidad entre cero (0) y dicha sumatoria; en este momento se comienza a recorrer la lista de transiciones llevando un acumulado de las probabilidades, si dicho acumulado es mayor que la probabilidad generada aleatoriamente se escoge esa transición por evento, de lo contrario se continua con la siguiente transición.

El sistema MECCA consta de cuatro programas que son necesarios para su funcionamiento.

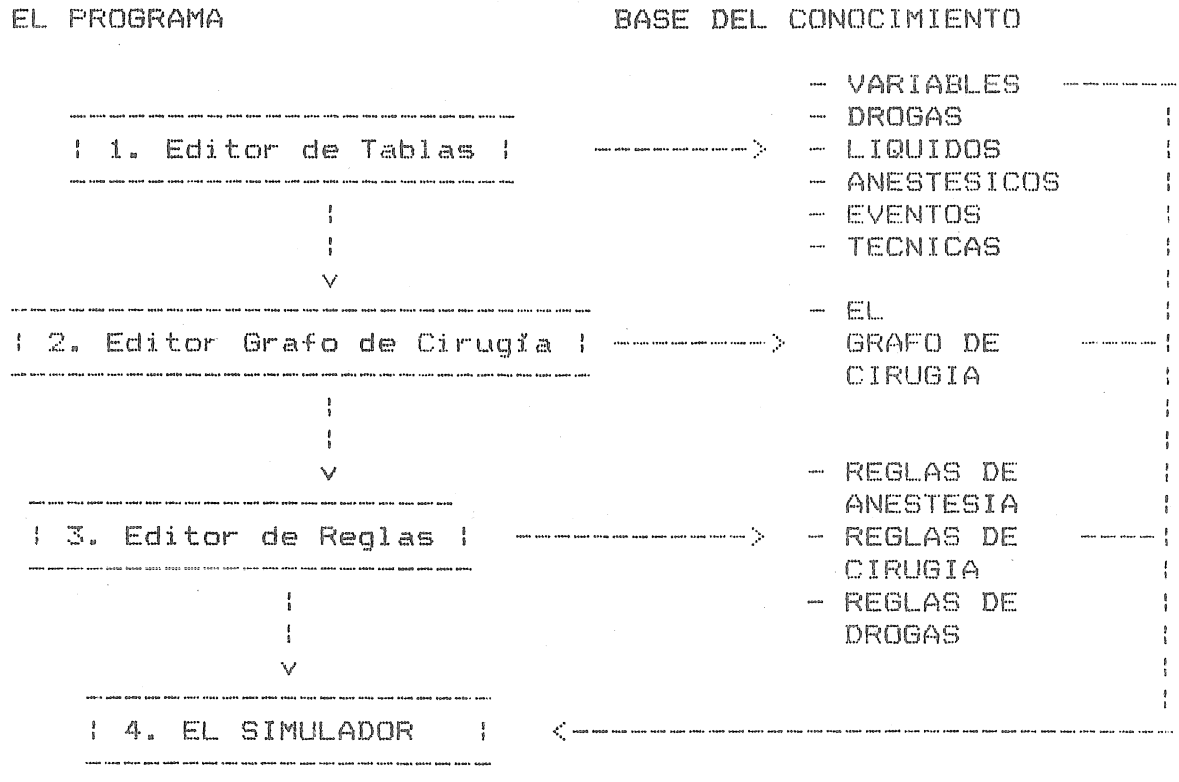
Estos son :

1. El Editor de Tablas.
2. El Editor del Grafo de Cirugía.
3. El Editor de Reglas.
4. El Simulaldor.

Inicialmente el usuario experto debe definir el conocimiento que se va a manejar. Para este fin utiliza los editores; comienza con el editor de tablas, el grafo de cirugía y finalmente edita las reglas que van a manejar al sistema (reglas de anestesia, reglas de cirugía y reglas relacionadas con la aplicación de una droga).

El usuario experto debe definir con la ayuda del ingeniero del conocimiento los Métodos; estos son procedimientos programados que representan una gran dificultas para ser expresados mediante reglas debido a la complejidad de los cálculos que requieren. Se emplean generalmente para el cálculo de las variables fisológicas. A pesar de ser estos procedimientos programados se busca definirlos de una manera clara y concisa que no represente problemas al usuario experto.
Sinembargo se debe avisar al sistema MECCA la existencia de dichos métodos en el editor de Tablas, para que este los pueda manejar .

El siguiente diagrama muestra los programas que componen el sistema y su interacción con la base del conocimiento:



El Editor de Tablas, El editor del Grafo de Cirgugía, El Editor de Reglas son herramientas que permiten al usuario experto definir el conocmiento que se va a tomar en cuenta, generando la Base del Conocimiento. (Ver Diagrama).

El Editor de Tablas

Permite al usuario experto definir los elementos con los cuales va a trabajar el sistema relacionados con el paciente mismo y con el tratamiento que se le puede aplicar.

El Estado del Paciente está definido en un momento dado de la intervención por los valores que toman las variables. Tanto los hechos iniciales acerca del paciente como el conjunto de variables clínicas y fisológicas que lo describen son definidas mediante este editor.

El Editor de Tablas captura la información acerca de :
- Las Variables Clínicas y Fisológicas.
- Los hechos iniciales del paciente.
- Las Drogas.
- Los Liquidos.
- Los Gases (Anestésicos y Oxígeno).
- Los Metodos que van a ser utilizados y las variables que estos afectan.
- Eventos Especiales y Técnicas Elementales.

Para mayor información a este respecto consultar la sección 3.

Una vez que se tienen las variables, se puede definir el paciente hipotético dándole valores iniciales a éstas.

Editor del Grafo de Cirugía

Permite al usuario experto definir una cirugía cualquiera.

La cirugía se define para efectos del sistema como un grafo cuyos estados corresponden a nodos del grafo y cuyas transiciones ya sea por tiempo o por causa de un evento especial ocurrido corresponden a los arcos.

Depende de la definición de los eventos especiales puesto que la transición de un estado a otro del grafo puede ser causada por un evento especial que ocurra durante la cirugía.

Genera un archivo donde está el grafo de cirugía con las características ya descritas.

Editor de Reglas

Mediante el editor de reglas el experto puede definir las leyes que rigen el sistema tanto para anestesia y drogas que se aplican como para la misma cirugía.

La importancia de las reglas radica en que estas contienen todo el conocimiento del experto anestesiólogo y del médico relacionado con la cirugía, para que ocurran los cambios en el estado del paciente, es decir, para que la "Simulación" pueda ser efectuada. Es decir las reglas dan "movilidad" al sistema.

El Simulador

Una vez que el experto ha definido todo el conocimiento que se va a tomar en cuenta, el simulador puede funcionar.

Las facilidades que ofrece el programa al usuario pueden resumirse de la siguiente manera:

Inicialmente se pregunta al usuario cuál cirugía desea practicar.

Puesto que el manejo del tiempo durante la anestesia es de gran importancia, se permite al usuario determinar el intervalo de tiempo en el cual desea monitorear las variables y todo lo que sucede en la cirugía, es decir, cuánto tiempo transcurre para que los cambios sean actualizados. Es importante aclarar que esto no tiene relación alguna con la rapidez con la que los cambios ocurren.
Dicho intervalo permite contraer o dilatar el tiempo, de acuerdo con el deseo del usuario. Es decir el tiempo es "regulable".
La importancia de llevar un control del tiempo, desde el punto de vista del usuario radica en que se deben tomar decisiones constantemente y el no tomar ninguna desición es una de las acciones posibles, puesto que esto afecta el estado del paciente. El tiempo es muy importante durante la intervención.

El sistema permite también al usuario llevar un registro de los cambios ocurridos durante la intervención. Esta facilidad es de gran importancia académica puesto que permite saber y entender al usuario el porqué de una reacción del paciente ante determinada circunstancia, ya que permite consultar la regla que determinó dicha reacción y a su vez devolverse en el tiempo consultando los cambios de los valores y las reglas que los produjeron.

Adicionalmente este registro ha contribuido notoriamente a refinar y verificar la validez del conocimiento introducido, puesto que permite ver al usuario experto las variaciones producidas por las reglas definidas.

La idea general del funcionamiento del sistema es la siguiente:

Una vez que se ha pedido toda la información pertinente se inicia la intervención.

Se presentan al usuario en pantalla las variables que el experto definió y consideró pertinente que se mostraran

constantemente. Se actualizan las variaciones de dichas variables cada vez que se cumple el intervalo de tiempo definido por el usuario al principio de la intervención.

Puesto que el programa está desarrollado con base en lo que ocurre en una sala de cirugía de un hospital colombiano, se ha buscado que la simulación sea aproximada a la realidad, sinembargo se permite medir variables que en este medio no se pueden medir puesto que no existen las herramientas necesarias.

Cuando se inicia la anestesia en la sala de cirugía el paciente es anestesiado con anestésico inhalado mediante la máquina de anestesia (la cual contiene óxido nitroso y oxígeno) y/o con anestésicos intravenosos. También se le aplican otras drogas que contribuyan al bienestar. Una vez que el paciente ha sido anestesiado, puede ser intubado con el fin de suministrarle oxígeno.

Posteriormente, el anestesiólogo puede dar la orden de iniciar la cirugía. Se inicia entonces el mantenimiento. A partir de este momento se siguen aplicando drogas y anestésicos y se comienza a vigilar al paciente mediante la medición de algunas variables de interés y la realización de exámenes que permitan determinar el estado general del paciente y la profundidad anestésica de éste.
Constantemente el anestesiólogo ayuda al paciente a respirar mediante una bomba, es decir le ayuda con la ventilación.
También puede suministrarle líquidos determinando cuantas gotas de líquido se suministran en un minuto.

De acuerdo con esto el sistema MECCA, permite realizar las siguientes opciones :

- aplicación de una droga
- aplicación de un anestésico u oxígeno
- aplicación de un líquido
- ayuda en la ventilación
- técnicas elementales

El usuario puede consultar las variables fisiólogicas y clínicas, los hechos y el Registro de cambios.

El médico anestesiólogo tiene la capacidad de controlar también un poco la cirugía porque es quién da la orden de iniciarla (tal como ocurre en una sala de cirugía), de pararla o reanudarla cuando estime conveniente. El usuario con solo presionar una tecla puede llevar a cabo alguna de las opciones anteriores, el sistema nunca le obligará a tomar una acción, es decir, no se le presenta un menu de opciones sino que el decide tomar una acción. No hacer nada significa también tomar una decisión, puesto que esto puede afectar el estado del paciente.

Después de haberse iniciado la cirugía comienza ésta a desarrollarse paralelamente con la anestesia. Se simulan las transiciones de un estado a otro y la ocurrencia de un evento especial de acuerdo con una probabilidad.

La cirugía y la anestesia no se pueden separar puesto que están intimamente ligadas con el paciente, es decir, lo que ocurre en la cirugía depende en parte del estado del paciente y lo que ocurre en la anestesia (las decisiones de anestesia) afectan naturalmente el estado del paciente. Pero las decisiones de anestesia son responsabilidad del usuario anestesiólogo, mientras que lo que ocurre en la cirugía es controlado por el sistema.

Finalmente, la intervención termina cuando la cirugía ha concluido y el anestesiólogo decide terminar el mantenimiento y los cuidados de la postcirugía.

Conclusiones

En la implementación del sistema educativo de simulación de pacientes durante la anestesia MECCA utilizamos tanto Técnicas de Control como de Sistemas Expertos.

Las técnicas de control se utilizaron en la implementación de los modelos farmacocinéticos, puesto que estos están basados en procedimientos matemáticos previamente definidos y validados. Se utilizan técnicas de sistemas expertos para implementar el modelo fisiológico del paciente y para la evolución de la simulación; es decir, el sistema experto permite el desarrollo de la intervención puesto que genera los cambios en las variables del paciente y el avance de la cirugía.

Los sistemas expertos tienen la ventaja de que el conocimiento está separado del programa; lo cual facilita al usuario experto las labores de validación y complementación del conocimiento. Esta característica es de gran importancia en el conocimiento médico ya que este es incompleto y no se puede asegurar su certeza porque está cambiando constantemente. Modelar el funcionamiento del cuerpo humano no es una tarea sencilla y menos aún cuando éste se considera bajo una situación especial como es la anestesia.

Los sistemas de control permiten desarrollar modelos matemáticos bastante precisos y fáciles de manejar; sinembargo la programación de dichos modelos puede resultar engorrosa para el experto y rígida para permitir modificaciones en el conocimiento. Los sistemas de control requieren de un proceso de validación bastante complejo, puesto que requieren la representación del conocimiento mediante modelos matemáticos que pueden resultar difíciles de manejar. Por otra parte, los sistemas expertos permiten al experto expresar el conocimiento manera muy cercana a la forma como él razona, sin recurrir a modelos matemáticos que pueden resultar inadecuados y difíciles de manejar.

Las técnicas de sistemas de control resultaron muy adecuadas para la implementación de los modelos matemáticos validados como son el modelo de drogas inhaladas de W.W. Mapleson y el modelo farmacocinético bicompartimental publicado por M. Wood. Por otro lado, mediante el uso de las técnicas de sistemas expertos se logró permitir libertad al usuario para expresar el conocimiento de la manera como el experto estime conveniente.

Teniendo en cuenta estas observaciones, concluímos que el uso de sistemas expertos para la construcción de herramientas computacionales en el área de la medicina puede resultar muy conveniente. Esto es especialmente cierto por la característica modular que tienen los sistemas expertos, al

separar los "datos" (conocimiento de tipo médico) del control (programa) se independiza el sistema del tipo de conocimiento que utiliza.

La labor de obtención del conocimiento médico ha resultado muy difícil dada la naturaleza de este. Como indicábamos antes, el funcionamiento del cuerpo humano es muy complejo, esto hace que exista una vasta cantidad de conocimiento acerca de éste. Intentar recopilar una parte significativa del conocimiento existente sobre el comportamiento del cuerpo bajo la acción de diversos medicamentos y de una cirugía es una tarea que puede tomar mucho tiempo.

El diseno e implementación del simulador represento una tarea larga y un tanto compleja por la naturaleza del conocimiento; se pretendio desarrollar una herramienta que simulara los cambios de estado de un paciente por la aplicación de la anestesia durante el desarrollo de una cirugía.

Este programa se desarrolló con fines educativos, lo cual imprime al sistema unas caracteristicas especiales, puesto que ademas de la simulacion se implementaron unas herramientas que facilitan al usuario su proceso de aprendizaje.
Desde el punto de vista educativo, el simulador tiene la ventaja de permitir al estudiante practicar sus conocimientos evitando los peligros que puedan presentarse en la vida real. El estudiante puede experimentar para aprender.

La esencia educativa del sistema **MECCA** no solamente se ve reflejada en el simulador, sino que también mediante el Registro de Cambios permite al estudiante formalizar los conceptos, puesto que da respuesta al porqué de una reacción del paciente ante determinada circunstancia.

Aunque no esté completamente validado el conocimiento, el sistema tiene una característica educativa muy importante al permitir al estudiante observar relaciones de causalidad entre variables; es decir, aunque en su estado actual los valores de las variables del paciente durante la simulación pueden no ser muy reales, se pueden observar claras relaciones entre ellas, elemento muy importante desde el punto de vista educativo.

Dado el objetivo del sistema y la estructura de este, el sistema **MECCA** puede estar sujeto a mejoras desde varios puntos de vista :

- desde el punto de vista técnico
- desde el punto de vista médico
- desde el punto de vista educativo

Desde el punto de vista técnico podría decirse que el sistema

está "completo" puesto que la estructura que está implementada permite dar soporte al tipo de conocimiento que se maneja y que ha sido considerado en el diseno. Sinembargo, podrían efectuarse mejoras que permitan manejar el conocimiento de una manera más eficaz y que ofrezcan al usuario tanto experto como estudiante mayores facilidades para el manejo del sistema.

Para hacer más eficaz el manejo del sistema, se podría pensar en desarrollar un generador automático de tablas que permita manejar las estructuras básicas de una manera más uniforme. Este generador de tablas permitiría al usuario definir cuantos objetos de conocimiento diferentes desee, para así ampliar el tipo de conceptos que se manejan sin tener que entrar en labores de programación.

En cuanto al tipo de reglas que maneja el sistema, podría pensarse en ampliar el tipo de operaciones que soporta para incluir entre sus condiciones el operador lógico OR y permitir funciones como Logaritmo.

Actualmente el sistema, ofrece un solo nivel de simulación en el cual el usuario tiene acceso a todas las variables definidas y el tiempo no transcurre durante las consultas y y acciones del anestesiólogo. Con el fin de hacer el sistema más real puede plantearse al usuario la dos niveles de dificultad: el primero de ellos con las características que actualmente están implementadas y el segundo impediría al usuario estudiante la consulta de variables que en la realidad no pueden ser medidas debido a que no se cuenta con las herramientas necesarias. En este último nivel se buscaría hacer la simulación en tiempo real con el fin de que el médico pueda realizar la intervención en una ambiente más real en el cual el tiempo es de gran importancia, puesto que la demora en la toma de una decisión puede influir en el estado del paciente.

Desde el punto de vista médico hay una gran cantidad de mejoras que se pueden realizar, pero se debe tener en cuenta que el conocimiento involucrado es siempre susceptible de ser modificado y complementado. El conocimiento que se ha considerado hasta ahora no ha culminado aún su proceso de validación, puesto que las reglas no están completas.

Hay varios aspectos del conocimiento que pueden ser mejorados para lograr un mejor funcionamiento del sistema en sus múltiples interaciones, estas mejoras se refieren fundamentalmente al área de control y son :

- El modelo farmacocinético de las drogas no inhaladas, pues actualmente solamente funciona con la administración de drogas por vía intravenosa, suponiendo una absorción completa e instantánea. Ocasionalmente se administran drogas por otras vías en donde la absorción es lenta e incompleta, o por vía

venosa en infusión donde la absorción es completa pero toma mucho más tiempo.

- El manejo de los datos hemodinámicos se hace de una manera relativamente sencilla, y podría desarrollarse un modelo más complejo y por lo tanto más aproximado a la realidad; que tenga en cuenta otras variables hemodinámicas que actualmente no están contempladas en el sistema, como por ejemplo: presión venosa central, resistencias vasculares, contractilidad cardiaca, etc..

Además, sería conveniente que el sistema simule los datos del electrocardiograma (actividad eléctrica del corazón), vigilancia que se realiza normalmente durante la anestesia.

- El modelo de control de la respiración y del PaCO2 también podría mejorarse para que tenga en cuenta cosas tales como producción de CO2, es decir, cambios metabólicos que podrían ocurrir en el paciente.

- Por último se puede incluir información acerca de entidades poco comunes que generan alteraciones fisiopatológicas especiales y que requieren un tratamiento diferente por parte del anestesiólogo, es decir, se podrían plantear casos de pacientes con otras complicaciones adicionales al motivo de la cirugía.

Desde el punto de vista educativo, una ampliación muy importante que se podría pensar en desarrollar sería incluir al sistema un módulo que se encargara de hacer comentarios acerca del desempeno del usuario durante la administración de la anestesia. De esta manera, al recibir "retroinformación" del sistema, el usuario podría eventualmente revisar sus conocimientos en aquellas áreas en que esto sea necesario.

También se puede pensar en utlizar el conocimiento en el cual está basado el sistema para realizar otro tipo de sistemas educativos, como por ejemplo un sistema que apoye al usuario en la elaboración de planes anestésicos.

Teniendo en cuenta todos estos elementos, se puede deducir que el Sistema MECCA no está concluido y que posiblemente siempre pueda verse sujeto a modificaciones debido a la naturaleza de sus objetivos y al tipo de conocimiento que maneja, ya que en medicina no se tiene la verdad absoluta.

Aunque no esté completo, consideramos que el sistema cuenta con una base de conocimiento adecuada y, después de una labor de refinamiento y obtención de más conocimiento se puede pensar en su utilización en algún hospital.

CARDIAC

" Una Herramienta didáctica para la enseñanza de Qué es una computadora?"

Director:
Mateo Obregón Sargent.

Sandra Patricia Alba Saavedra.
Gerardo Esteban Balla Guerrero.
Jorge Luis Chamorro Erazo.
María Teresa Ospina
Santiago Vargas Martín.

Jorge Luis Chamorro E. Calle 56#35-15 of 302
Bogotá - COLOMBIA

Con el fin de brindar nuevas herramientas para aprender el funcionamiento básico de una computadora, se implementó CARDIAC (CARboard Ilustrative Aid to Computation) en una computadora personal, por ser un medio más accequible a cualquier usuario.

CARDIAC está basado en un proyecto desarrollado por la Bell Laboratories, para que a manera de juego didáctico se entiendan los componentes y el funcionamiento básico de una computadora.

CARDIAC hace entendible los conceptos de CPU, memoria, unidad de control, registros y demás elementos de una computadora. Esto se logra a través de una relación directa con la computadora, donde se ofrecen unos códigos básicos de operación que permiten sumar, restar, llamar subrutinas, hacer corrimientos lógicos, operaciones de lectura y escritura.

Por último, clarificar estos conceptos le permitirá al usuario un mejor aprovechamiento de los recursos que una computadora le puede brindar.

Introducción.

El desarrollo de este trabajo surge como un proyecto de investigación realizado por parte de un grupo independiente de estudiantes que, paralelamente a los temas de estudio obligatorios, deseaban profundizar en un tema particular; con el fin de presentar soluciones a las inquietudes que a nivel estudiantil surgen en la mayoría de las personas que cursan la carrera de computación.

CARDIAC deriva su nombre de "CARboard Illustrative Aid to Computation"; diseñado y desarrollado por Bell Telephone Laboratories como una ayuda práctica a manera de juego didactico para entender las computadoras y la programación.

La inexistencia de ayudas de este estilo en computadoras personales -cada vez más populares- fué determinante en la elección del proyecto. Esta versión de CARDIAC fué implementada en una microcomputadora IBM PC utilizando el lenguaje Lisp.

Objetivos.

El objetivo del grupo es complementar la formación academica recibida, trabajando nuevas ideas en computación teórica y aplicada.

El objetivo de CARDIAC es responder a la pregunta que se hacen algunas personas: Que es una computadora?, dado que actualmente nadie puede desentenderse del fenómeno informático que cubre todos los aspectos de la vida cotidiana. Este proyecto brinda las herramientas esenciales para que el usuario pueda posteriormente entender el funcionamiento básico a nivel de programación de cualquiier clase de equipo.

El aspecto más interesante, es la manera didáctica y sencilla como facilita entender el funcionamiento de una computadora, proporcionando así un entendimiento completo de los conceptos que la integran.

Está dirigido a todas aquellas personas que se inician en el area de sistemas, sin que interese su posterior especialización. No se trata en ningún momento de enseñar a escribir código, sino mostrar como este se ejecuta.

El objetivo de este aprendizaje es correr el velo muy generalizado que hace parecer la computadora como una caja negra que ejecuta procesos mágicamente. En este aspecto, el hecho que el proyecto este implementado en una microcomputadora, implica que el usuario tenga un contacto directo con la máquina, brindando la familiaridad y confianza necesarias para asimilar la computadora y por ende clarificar los conceptos que abarca.

En la mayoría de los casos, la enseñanza de los conceptos CPU, memoria , dispositivos de E/S es aislada, y generalmente teórica, sin llegar a una conceptualización total; siendo necesario complementarla a traves del tiempo, con las desventajas que esto implica.

A diferencia de esto, CARDIAC es un juego didáctico en donde la fuente de potencia es la persona misma. El interés que ésta presenta interactuando con CARDIAC de una manera directa y práctica, lo llevan paso a paso a entender los elementos esenciales para la comprension de Qué es una computadora.

Es necesario hacer énfasis en que la condición indispensable para obtener un beneficio, reside en el interés que para la persona representa conocer la manera elemental de programar la maquina.

El hecho que CARDIAC sea una abstracción de lo que es una computadora, permite resaltar un punto que muchas veces pasa inadvertido: las computadoras no son máquinas que resuelven todos los problemas.

A continuación se explicarán los componentes y funcionamiento de CARDIAC, los elementos son: CPU -compuesta por acumulador, decodificador de instrucciones y contador de programa-; memoria y dispositivos de E/S.

La memoria es una parte esencial para el funcionamiento de CARDIAC, en esta se almacenan los datos/instrucciones que forman parte de un programa. Se encuentra dividida en cien celdas enumeradas de la 00 a la 99, cada una de las cuales contiene un dato o una instrucción. Las posiciones 00 y 99 están reservadas por el sistema, para almacenar en la 00 la instrucción 001 que actúa como bootstrap de CARDIAC, (autocargador del programa); y en la 99 las instrucciones cuyo formato es 8XX, que significa salto a una posición de memoria; siendo esta una manera simple de implementar una subrutina.

La unidad de control regula la operación del registro de instrucciones

Sus funciones básicas son:

- Utilizar el contenido del contador del programa (PC), como la dirección desde la cual traerá la siguiente instrucción al registro de instrucciones.

- Incrementar el PC en 1, una vez ha cargado la instruccion.

- Activar el registro de instrucciones para ejecutar la instruccion corriente.

La función del registro de instrucciones es almacenar la instruccion durante el lapso de tiempo en que se está procesando. Una vez terminada esta ejecución, se repite el ciclo de CPU.

El contador del programa guarda la secuencia que debe el programa.

El acumulador es la unidad aritmético-lógica; en este los numeros son adicionados, restados o sujetos a operaciones de corrimiento de digitos, adicionalmente es posible conocer su signo. El acumulador soporta operaciones de suma o resta hasta 999, como máximo valor posible en un operación.

CARDIAC opera a través de códigos compuestos por tres digitos, el primero indica la operación a efectuar y los dos últimos información referente a la celda de memoria. Los códigos de operación son:

0 : Entrada de la cinta a la memoria.
1 : Borre el acumulador y sumele el contenido de la celda XX.
2 : Sume al acumulador el contenido de la celda XX.
3 : Salto condicional. Verifique el signo del contenido del acumulador y dependiendo de la respuesta posicione al PC en XX.
4 : Corrimiento del valor existente en el acumulador.
5 : Movimiento desde una celda de memoria a la salida.
6 : Almacene el contenido del acumulador en la celda XX.
7 : Reste del acumulador el contenido de la celda XX.
8 : Salto incondicional, posicionando el PC en XX.

9 : Pare y posicine el PC en 00.

De esta manera CARDIAC ofrece al usuario elementos suficientes para construir programas de cierta complejidad.

Dentro del enfoque técnico, el diseño de CARDIAC utiliza una clase de programación interesante, basada en una manipulación de objetos (programación centrada en objetos).

La programación centrada en objetos es un nuevo enfoque dado a la programación de computadoras que altera la relación entre codigo de un programa y sus datos. En el enfoque tradicional donde el código es el encargado de manipular los datos, ya sea leer, transformar o escribir, se presenta el problema de la dependencia directa con respecto a la estructura de datos seleccionada; si es alterada el codigo no servirá lo que implica una reescritura de este.

Con la programación centrada en objetos, los datos tienen una estructura en la cual ya no son entes aislados sino que ahora tienen asociadas unas propiedades que indican la clase de dato (tipo), y unos procedimientos que manejan este tipo. Así, el código de un programa se limita a proveer el control de la ejecución, sin trabajar directamente sobre los datos.

Trabajando así es posible brindar una mayor flexibilidad a un programa, de tal manera que si se agregan nuevos tipos, no sera necesario modificar el código, puesto que los nuevos datos traeran su propia información de manejo.

La clase de programación utilizada, establece una similitud con las experiencias reales a nivel de procesos, las cuales se basan en la independencia de los objetos establecidos y en la interrelación de éstos a través de un manejador.

El usuario tiene interacción con CARDIAC a dos niveles: el primero un acceso directo para escribir programas, y el segundo, le ofrece todas las herramientas para modificar y manipular cada uno de los elementos que componen CARDIAC.

Un manejador de entrada está encargado de llamar a la CPU. Dado que los objetos se comunican por medio de mensajes, debe existir un controlador del paso de estos (message-passing), función que realizaa la CPU entre los objetos acumulador, contador de programa, flags, memoria, dispositivos de E/S. Adicionalmente la CPU tiene las subrutinas que manejan el conjunto de códigos de operaciones a ejecutar. De esta manera se logra unificar un punto de entrada del usuario al manejador de mensajes (CPU).

Dentro de CARDIAC exiten también como objetos los errores y el paquete de ayuda, los cuales proveen al usuario de una aclaración específica en una situación determinada.

Se utilizó el lenguaje LISP, entre otras razones porque facilita el manejo de comunicación entre funciones.

SISTEMA TUTORIAL CON EJERCITACION Y PRACTICA PARA LA ENSEÑANZA DE LA SUMA.

Grupo de Informática Educativa, Universidad de los Andes

Autor: Fernando Mauricio Durán Campos.

Apartado Aéreo 4976, Bogotá. Tel: 286-61-85

Este artículo presenta una experiencia en desarrollo de software educativo. El sistema que se describe es una tesis desarrollada en el Grupo de Educación Apoyada con Computador del Departamento de Sistemas de la Universidad de los Andes.

Este sistema constituye una herramienta adicional para el profesor en la enseñanza de la suma en el curso primero de primaria. El sistema facilita la adquisición de la operación por medio de la actividad personal, promoviendo así, la participación directa del alumno en el proceso de aprendizaje, adicionalmente, para el profesor sirve como un medio de control del avance y de la situación académica de todo un curso. El sistema se encuentra dividido en dos partes principales, la aplicación propiamente dicha de enseñanza de la suma, y otra parte conformada por una serie de aplicaciones que sirven para manipular un curso de estudiantes. La primera parte presenta cuatro módulos, conjuntos, adición con unidades, adición con decenas y adición con centenas, los cuales abarcan el programa dado por el Ministerio de Educación Nacional. La segunda parte está constituida por tres aplicaciones, crear curso, listar curso y modificar curso.

INTRODUCCION.

En los últimos años, gracias a la reducción de costos de Hardware y a la difusión y masificación de los sistemas, ha aumentado el interés del sector educativo en el uso de apoyos computacionales en los diferentes campos de la instrucción. Este interés abarca todas las áreas del conocimiento (matemáticas, idiomas, ciencias, etc.), todos los niveles instruccionales (primaria, bachillerato, universidad, institutos técnicos, etc.) y todos los tipos de sistemas de apoyo: herramientas de apoyo logístico y administrativo, medios principales o auxiliares para el proceso de enseñanza-aprendizaje, etc.

En respuesta a este interés y necesidad del país de utilizar los avances tecnológicos en beneficio de la educación, el Grupo de Eduación Apoyada con Computador E.A.C. del Departamento de Sistemas de la Universidad de los Andes está trabajando en tres áreas de aplicación educativa de la informática: desarrollo de herramientas de apoyo a la administración de la instrucción, como bancos de problemas y herramientas estadísticas; software de apoyo al proceso mismo de enseñanza-aprendizaje, es decir sistemas tutoriales, heurísticos, simuladores que refuercen la enseñanza de un area específica para una población específica y alfabetización computacional (v.gr. enseñanza sobre la informática misma).

Dentro de los diferentes tipos de sistemas de apoyo al proceso de enseñanza-aprendizaje se encuentran los sistemas tutoriales y de ejercitacion y práctica, en los cuales el computador conduce al alumno a través de secuencias preestablecidas de actividades para lograr unos objetivos predeterminados.

El sistema que se describe en este artículo es un tutorial con ejercitación y práctica de apoyo a la enseñanza de la SUMA para niños de primero elemental (6-7 años). En este artículo se resumen las etapas básicas del desarrollo del proyecto, según el esquema de ingeniería de Software Educativo planteado por el Grupo E.A.C.

ANALISIS DEL PROBLEMA EDUCATIVO.

DESCRIPCION DE LA SITUACION.

En el proceso de desarrollo de aprendizaje de las Matemáticas los primeros conceptos que conoce un niño son los de número y el conteo de éstos, conceptos que se adquieren mediante un proceso lento pero sin mucha dificultad.

El primer gran obstáculo que realmente se le presenta al niño es la adquisición de las operaciones aritméticas, empezando por la adición. Es en este punto en donde muchos niños empiezan a tomar cierta aversión a las Matemáticas; ésto puede deberse a la mala enseñanza del concepto, que se puede presentar por falta de recursos que permitan mayor claridad en las explicaciones y cierta individualización en la enseñanza.

Según los programas del Ministerio de Educación Nacional, es en el curso primero de primaria en el cual se introduce la operación de la adición, estableciendo el concepto y llevando al niño a la comprensión del mismo mediante manipulación con unidades, decenas y centenas. Para entender estos conceptos, el alumno debe tener un dominio pleno de las nociones de número y conteo. El problema está directamente relacionado con niños que estén cursando el primer año de enseñanza primaria, en que se ve por primera vez la suma, pero se extiende a otros niños de cursos más avanzados que no adquirieron en forma correcta la operación.

Los recursos con que se cuenta actualmente para la enseñanza de la suma son los mismos de hace varios años, es decir, poco se ha avanzado al respecto. Estos recursos son fundamentalmente los textos, los juegos didácticos y los medios de comunicación (radio y televisión).

Los textos y la televisión se basan primordialmente en el uso de imágenes por medio de las cuales se trata de introducir la operación. La televisión y la radio impiden en todo momento la participación del alumno, siendo nulo todo tipo de interacción, lo que puede llevar al niño a perder el interés por aprender. Los juegos, por el contrario, son métodos activos que establecen una participación directa por parte del alumno, pero en algunos casos presentan problemas debido a que producen en el niño una reacción mecánica ante las diferentes instancias de los mismos, sin hacer explícito el conocimiento y habilidades que se están manejando.

ANALISIS DE LA SITUACION PROBLEMATICA.

La enseñanza tradicional de las matemáticas básicas utiliza en muchos casos esquemas rígidos en los que el alumnos es un receptor pasivo al que se bombardea con fórmulas y reglas. El profesor no cuenta con recursos suficientes para generalizar el concepto en varios contextos ni, lo que es peor, para permitir la participación individual de los alumnos. El niño, en este esquema, aprende mecánicamente la regla como una receta de símbolos sin entender su contenido y tiene luego problemas de abstracción y generalización que pueden frenar todo su proceso educativo.
Otro aspecto del cual peca la enseñanza actual es que no permite el intercambio intelectual en grupo, el niño no tiene la oportunidad de exteriorizar su percepción de los concepto aprendidos ni compararla con la de los demás; puede tener ideas erradas que resultan muy difíciles de detectar.

Para lograr la construcción de las operaciones en el niño es necesaria la investigación por parte del mismo alumno, para ésto se le deben crear situaciones psicológicas al niño que lo lleven a adquirir estas operaciones y lo motiven a hacerlo. Siempre se deben tener en cuenta los esquemas anteriores de que dispone el alumno y a partir de ellos desarrollar la nueva operación. Al niño se le debe permitir desarrollar su pensamiento con plena libertad, por lo cual se le debe conducir a construir sus nociones y operaciones mediante la acción personal, acción que está basada en la investigación.

Es importante que los niños asimilen las operaciones como un conjunto de conceptos, es decir que su concepción no se quede simplemente en la comprensión de eventos parciales, sino que forme un todo. Si lo que se quiere es que los alumnos adquieran mediante investigación personal el conjunto de un sistema de operaciones y no las operaciones de ese sistema en forma aislada, hay que orientar la actividad presentando adecuadamente el problema. Una buena medida para lograr el objetivo es reducir el alcance y complejidad del problema de tal forma que sea posible encontrar la solución del problema sencillo y avanzar luego hacia el problema más complejo. De esta manera se van construyendo poco a poco las operaciones, logrando al final la concepción total de las mismas.

Para evitar la adquisición errónea de las operaciones por parte del niño, es importante estudiar en grupo las reacciones falsas que puede originar la solución de cierto problema, para que los alumnos comprendan las razones por las cuales un procedimiento no es correcto y para que puedan captar las diferencias y las relaciones entre la reacción correcta y el error

El sistema computacional consta fundamentalmente de dos partes, una primera parte es en forma tutorial, en la cual se presentan diferentes conceptos al niño, los cuales ya deben haber sido tratados claramente por el profesor, por lo cual su profundidad es tal que permite condensar lo básico del concepto. Esta parte contiene todos los conceptos básicos, necesarios para la comprensión final de la operación de la adición. Una segunda parte, y tal vez la más importante es la de ejercitación y práctica, mediante la cual el niño puede participar directamente en el proceso de aprendizaje y por medio de la cual puede controlar su evolución, deteniéndose en los aspectos que le resulten más complicados de adquirir. En esta parte es indispensable contar con un sistema de motivación que estimule al niño a resolver correctamente los ejercicios y a avanzar a niveles cada vez más complejos. El juego favorece la retroalimentación, haciendo más ameno el proceso de apredizaje e incita al niño a buscar formas de solución a los obstáculos que se le presenten durante la comprensión de la operación. Para que este juego resulte interesante y cumpla su función motivadora, debe plantearse en términos de la población objetivo: niños de primero de primaria.

MACRODISEÑO DE UNA SOLUCION EDUCATIVA APOYADA CON COMPUTADORES.

OBJETIVOS DEL SISTEMA.

El sistema está orientado a niños de primer año de enseñanza primaria y por medio de éste se busca facilitar el aprendizaje de las Matemáticas en este nivel, específicamente en cuanto se refiere a la operación de la adición. Lo que se pretende es facilitar la adquisición de la operación por medio de la actividad personal, promoviendo así, la participación directa del alumno en el proceso de aprendizaje.

Según el programa del Ministerio de Educación Nacional, en el primer año de primaria se debe enseñar la operación de la adición a nivel de números enteros y con un alcance tal que se desarrolle sobre unidades, decenas y centenas.

El contenido del sistema abarca los siguientes puntos:

1.- Hacia la comprensión de la adición, por medio de conjuntos.

a.- Noción de conjunto.
- i) Clasificación de los objetos (conjuntos).
- ii) Relación elemento conjunto (pertenencia).
- iii) Relación entre dos conjuntos (menor o mayor número de elementos).
- iv) Hallar el número de elementos de un conjunto (cardinalidad).
- v) Simbolización del elemento neutro.

b.- Iniciación de la situación aditiva.
- i) Unión de conjuntos.
- ii) Número de elementos resultante en el conjunto unión.
- iii) Identificar la adición como una manera de hallar el número de elementos de la unión de dos conjuntos disyuntos.
- iv) Resolver problemas de situaciones aditivas con los números de 0 a 9.

2.- La adición.
- i) Orden numérico (0 a 9), anterior y siguiente.
- ii) Identificación de la decena, números de 11 a 19 expresados en términos de 10 + x.

ANALISIS DEL PAPEL QUE JUEGA EL COMPUTADOR EN LA IMPLEMENTACION DE LA ALTERNATIVA DE SOLUCION SELECCIONADA.

En síntesis, el problema que se le presenta al educador en la enseñanza de la suma es que esta operación como cualquier otra, requiere de una comprensión total por parte del niño, la cual no puede adquirirse por simple observación de imágenes, sino que requiere de la investigación personal por parte del alumno. Actualmente es difícil que se presente este proceso investigativo, ya que requiere de la individualización de la enseñanza, hecho poco viable en nuestro medio. Adicionalmente, en la enseñanza de una nueva operación matemática se debe evitar la mecanización del proceso de aprendizaje ya que éste puede crear hábito, siendo errónea la adquisición del concepto.

En este trabajo se propone una solución que le dé al niño la oportunidad de adquirir el concepto en forma individual mediante la práctica independiente, la cual requiere de un esfuerzo netamente personal y luego combinar esta práctica con una acción en grupo mediante la cual se puedan corregir los errores de cada alumno. La idea es que el niño pueda ir evolucionando de acuerdo a sus capacidades, al ritmo que él desee y mediante una interacción amena que le permita disfrutar del proceso de aprendizaje, brindándole plena confianza. Además, es de suma importancia para la adquisición de la operación, que el alumno pueda practicar lo suficiente sobre los nuevos conceptos adquiridos, de tal forma que pueda corregir oportunamente los errores que presente.Por último debe garantizar una respuesta inmediata a las inquietudes de los alumnos, mediante un sistema de retroinformación diferencial durante la ejercitación.

El computador resulta el medio más adecuado para el trabajo de ejercitación individual, ya que individualiza la enseñanza y permite al niño avanzar al ritmo deseado; adicionalmente puede generar cualquier cantidad de ejercicios de diversa índole que relacionen los conceptos adquiridos gradualmente por el alumno. Además, el computador puede brindar toda la confianza necesaria al niño de tal forma que el aprendizaje resulte ameno obteniéndose con ello una mayor concentración y por supuesto una mayor aceptación por parte del alumno. También es muy importante la retroalimentación que puede dar el computador, lo cual puede servir para que el niño se de cuenta a tiempo de sus errores de tal forma que los pueda corregir. Para el maestro, el computador resulta muy útil ya que le permite llevar un control más directo sobre la evolución de cada uno de sus alumnos en el proceso de adquisición de una nueva operación.

En el contexto del curso, este sistema sirve como un recurso adicional para el profesor; el maestro es el encargado de introducir los conceptos de la nueva operación y el sistema computacional es el medio que le permite al alumno ejercitarse a su propio ritmo, reforzando, interiorizando y generalizando el concepto.

El computador permite lograr una completa comprensión y ejercitación de la nueva operación, de manera gradual, a un ritmo dado por el alumno, pues él es quien determina en qué momento adquirió un concepto y si puede o no intentar adquirir otro. Por otra parte, el computador le permite al alumno manejar su secuencia de instrucción, dándole la facilidad de volver en cualquier momento a estudiar conceptos vistos anteriormente. De esta manera, el proceso es evolutivo, pero permite en cualquier momento volver sobre esquemas anteriores.

Para el trabajo grupal se proponen actividades de discusión y análisis en la clase, que aunque pueden hacer referencia a la ejercitación individual en el computador,es conveniente desarrollarlo con otros medios en que se facilita más el trabajo intergrupal.

iii) Orden numérico (0 a 99) (decenas, unidades).
iv) La centena.
v) Resolver problemas de situaciones aditivas con los números de 0 a 100.

Es de anotar que el contenido anterior se ciñe a las disposiciones especificadas por el Ministerio de Educación Nacional tras la reestructuración del sistema educativo (Decreto ley 1002 de 1984).

ESTRATEGIAS EDUCATIVAS.

Se usan como estrategias educativas el aprendizaje por investigación personal y la retroalimentación positiva.

Para facilitar la investigación personal, se le da al niño inicialmente una breve explicación de cierto concepto, el cual ha sido previamente tratado en clase y relacionado con éste se dan al alumno ciertos ejemplos que le permitan comprender el concepto y que le permitan familiarizarse con el sistema y el modo de interucción, finalmente se le plantean los ejercicios.

Para hacer estimulante el proceso de aprendizaje y para motivar al niño de tal forma que encuentre en el sistema un medio de diversión, se utiliza un juego, el cual facilita la ejercitación y plantea al niño cierto reto, incitándolo a alcanzar cada día mejores resultados. Este juego por supuesto, facilita la retroalimentación positiva.

El juego consiste en construir el mayor número de figuras posible, las cuales se forman con pedazos que aparecen cuando el niño responde acertadamente a una pregunta. Para ésto se tiene un banco de figuras de robots y personajes de caricaturas, conocidos para los niños de primaria.

ASPECTOS COMPUTACIONALES PARA IMPLEMENTAR LA SOLUCION EDUCATIVA.

REQUERIMIENTOS DEL SISTEMA.

Para su implementación se necesita un microcomputador con ayudas gráficas que permitan presentar las pantallas de una manera amena y con diversos dibujos que llamen la atención al niño, además, éstas deben facilitar la implementación del juego. Adicionalmente, el microcomputador debe permitir la emisión de notas musicales, indipensables para la retroinformación. Por las razones anteriores, el sistema está implementado en un microcomputador "MACINTOSH PLUS".

DESCRIPCION DEL SISTEMA.

El sistema sigue el programa del curso primero de primaria. Para su implementación se divide el sistema en cuatro módulo principales:

Conjuntos,
adiciones involucrando unidades,
adición con decenas,
adición con centenas.

Cada módulo a su vez se divide en dos partes, tutorial y ejercitación. En la parte tutorial se le presentan los conceptos en forma interactiva al estudiante, haciéndole preguntas y explicandole detalladamente cada tema, cuando el niño responda mal. En la parte de ejercitación se le plantean al niño varios ejercicios correspondientes al tema previamente explicado en el tutorial, el número de estos ejercicios está dado por el número de componentes que tenga la figura (del juego), que se esté armando en el momento, es decir, el final de estos ejercicios se consigue al completar una figura.

Los módulos están distribuídos de la siguiente forma.

Conjuntos.

Este módulo presenta al alumno la enseñanza del concepto de CONJUNTO. Para ésto, el proceso se divide en seis etapas:

Clasificación de objetos.
Relación elemento conjunto (pertenencia).
Relación entre dos conjuntos (menor o mayor número de elementos).
Unión de conjuntos.
Número de elementos de un conjunto (cardinalidad).
Número de elementos resultante en el conjunto unión.

Adición involucrando unidades.

Este módulo presenta las primeras sumas al niño, las cuales tienen como respuesta números menores que 10. El proceso de enseñanza con unidades se divide en cuatro etapas:

Identificar la adición como una manera de hallar el número de elementos de la reunión de dos conjuntos disyuntos.
Orden numérico (0 a 9), anterior y siguiente.
Situaciones aditivas con los números de 0 a 9.

Adición con decenas.

Este módulo introduce el concepto de Decenas y enseña al niño a resolver sumas cuyo resultado es menor que 100. La enseñanza de este módulo se divide en cuatro etapas:

Identificación de la decena.
Números de 11 a 19 expresados en terminos de dos sumandos.
Orden numérico (0 a 99) (decenas, unidades).
Situaciones aditivas con los números de 0 a 99.

Adiciones con centenas.

Este módulo introduce el concepto de centenas y explica como resolver sumas que tienen resultado menor que 1000. La enseñanza de este módulo se divide en tres etapas:

Identificación de la centena.
Números de 101 a 199 expresados en terminados de dos sumados.
Situaciones aditivas con los números de 0 a 999.

DISEÑO COMPUTACIONAL.

MODULOS DEL SISTEMA.

El sistema se encuentra divido en dos partes principales; una primera parte la constituye la aplicación propiamente dicha, de enseñanza de la suma. La otra parte está conformada por una serie de aplicaciones que tienen por objeto manipular un curso de estudiantes que tiene acceso a la aplicación de enseñanza de la suma.

La primera parte presenta cuatro módulos fundamentales; conjuntos, adición con unidades, adición con decenas y adición con centenas. Estos módulos se encuentran divididos en dos partes: Tutorial y ejercitación. En la parte tutorial se presenta la enseñanza de un concepto y en la parte de ejercitación se le da la oportunidad al niño de prácticar y evaluar sus conocimientos. La parte de ejercitación se divide a su vez en dos partes: Ejercicios y retroalimentación, la retroalimentación es la respuesta que se da al alumno de acuerdo a lo el conteste a una pregunta planteada, ésta puede ser positiva o negativa dependiendo de los resultados que de el niño a los problemas que se le presentan.

Con relación al manejo del curso, se tienen tres aplicaciones, que son:

Crear curso.

Esta aplicación se encarga de la creación interactiva de un curso de estudiantes. Cada estudiante se identifica por su nombre y sus dos apellidos. El curso además indica el último tema al que ha podido llegar el estudiante y cual fué el último tema que estudió (puede ser un nivel menos avanzado, si el niño decidió retroceder para repasar). Al crear el curso estos campos se inicializan en 0.

Listar curso.

Esta aplicación le permite al profesor mirar como se han desempeñado todos los estudiantes, es decir, saber hasta que tema ha llegado cada uno y cual fué el último tema estudiado por el.

Modificar curso.

Por medio de esta aplicación el profesor puede modificar un curso, es decir, puede adicionar un nuevo estudiante y también puede cambiar el último tema visto por un estudiante existente y el tema más avanzado al que ha llegado un alumno.

IMPLEMENTACION.

El sistema ha sido implementado en un microcomputador MACINTOSH PLUS, el cual posee una capacidad en memoria principal de 1 Megabyte. El sistema operacional utilizado es el FINDER versión 5.3. El ambiente de programación bajo el cual se desarrolló el sistema es TURBO PASCAL versión 1.0.

La filosofía de programación está totalmente determinada por las facilidades que ofrece el MACINTOSH PLUS, entre las cuales se tienen: Manejador de recursos, manejador de eventos, manejador de diálogos, Quickdraw, etc. Adicionalmente, se han tenido en cuenta las ventajas que ofrece TURBO PASCAL, como son las unidades y la segmentación.

Una unidad es un módulo que permite definir constantes, tipos de datos, variables, funciones y procedimientos, para ser usados por diferentes unidades que conforman una aplicación. El cuerpo de una unidad se divide en dos partes: interfase e implementación.

Interfase.

En esta parte se encuentran todas las declaraciones que son accequibles por otras unidades de una aplicación.

Implementación.

En la parte de implementación se encuentra el cuerpo de los procedimientos y funciones declarados en la parte de interfase.

Un segmento es un módulo cuyo código compilado ocupa máximo 32 Kbytes. Los segmentos pueden ser cargados y descargados en forma dinámica, lo cual permite crear sistemas grandes con TURBO PASCAL, ya que el uso de segmentos optimiza el consumo de memoria.

Debido a su gran tamaño, la aplicación (SUMANDO) fué implementado usando varias unidades y a la vez utilizando diferentes segmentos.

Las unidades son las siguientes:

UNGLOBAL.

Esta unidad contiene la declaración de constantes, tipos de datos, variables, procedimientos y funciones que son globales para toda la aplicación, es decir que son usadas por todas las demás unidades.

UNRETRO.

En esta unidad se encuentran los procedimientos encargados de la retroalimentación de los ejercicios de conjuntos y unidades.

UNRET1.

Esta unidad es complementaria a la anterior, ya que también presenta procedimientos encargados de retroalimentación, pero en este caso de los ejercicios de decenas y centenas.

UNEJER.

Aquí se encuentran los procedimientos encargados de ejecutar los ejercicios de conjuntos, unidades y decenas.

UNEJER1.

Esta unidad es complementaria a la anterior y también presenta procedimientos encargados de ejecutar ejercicios, en este caso del módulo de centenas.

UNPRESEN.

Contiene los procedimientos encargados de la presentación del sistema y del

tutorial correspondiente a cada módulo, asi como del control del juego.

Adicionalmente, se cuenta con el módulo principal de la aplicación SUMANDO, el cual contiene la declaración de las unidades usadas y de los diferentes segmentos creados. En este módulo se encuentran las instrucciones encargadas de todo tipo de inicialización asi como de finalización de la aplicación.

En las aplicaciones encargadas del manejo del curso de estudiantes que pueden accesar la aplicación SUMANDO, no se crearon unidades debido a que el tamaño de dichas aplicaciones no lo requeria. Cada una de estas aplicaciones está implementada en un solo módulo principal y no necesitan el uso de segmentación.

Además de la aplicación, el sistema cuenta con manuales de usuario y de mantenimiento con el fin de prestar soporte al sistema desarrollado. En el manual de usuario se describe la manera de usar la aplicación SUMANDO, y se le da la instrucción al profesor para el manejo de las aplicaciones que tienen a su cargo el curso de estudiantes que tiene acceso a la aplicación SUMANDO. En el manual de mantenimiento se describe la manera de mantener la aplicación SUMANDO de acuerdo al desarrollo que se haga de la misma y a los resultados que se obtengan de ella, lo mismo se hace para las aplicaciones encargadas del manejo del curso.

CONCLUSIONES DE IMPLEMENTACION.

El hecho de implementar la aplicación SUMANDO en el computador MACINTOSH PLUS ha sido todo un reto computacional, debido a la novedad de este microcomputador y a la filosofia de programación que posee. Por este motivo se recomienda que en caso de querer utilizar el MACINTOSH PLUS para implementar un sistema, se haga primero un estudio pormenorizado de sus capacidades y de los recursos que posee para facilitar la implementación y para adecuarla a su filosofía.

En estos momentos han salido al mercado nuevos compiladores del lenguaje Pascal, tales como 'MPW. PASCAL' y 'LIGHTSPEED PASCAL', los cuales permiten segmentos ejecutables superiores a 32 Kbytes, lo que resulta en una ventaja importante con respecto a TURBO PASCAL, por este motivo es aconsejable utilizarlos ya que en los otros aspectos son muy similares a TURBO PASCAL.

El hecho de que TURBO PASCAL no permita segmentos mayores de 32 Kbytes fué el mayor problema presentado en la implementación de la aplicación SUMANDO; no bastó con modularizar en diferentes unidades y a su vez en diferentes segmentos la aplicación, los segmentos debieron ser llenados de tal manera que cada uno ocupara prácticamente todos los 32 Kbytes. Inicialmente se organizaron los segmentos de acuerdo con las invocaciones de procedimientos, de manera que el árbol de llamadas de procedimientos fuera equilibrado en todos los casos, obteniendo así un árbol de altura mínima y mayor eficiencia. Sinembargo este método no funcionó y se debió escoger el primero mencionado (llenar al máximo los segmentos sin importar la altura del árbol de procedimientos). Se recomienda estudiar con detenimiento este aspecto antes de implementar una aplicación en el MACINTOS PLUS utilizando el compilador de TURBO PASCAL.

CONCLUSIONES GENERALES.

El sistema desarrollado, ante todo, debe ser tenido en cuenta como una herramienta adicional para la enseñanza de la suma en el curso primero de primaria, no como un

medio único para el aprendizaje de esta operación. Esto debido a que la aplicación está dirigida a estudiantes de primer año de primaria, los cuales tienen un promedio de 7 años, edad en la cual es importante para la formación del niño combinar diferentes aspectos analizados en este artículo, entre los que se destaca la socialización del aprendizaje, la cual no brinda el computador.

De esta forma y para lograr los objetivos planteados, es necesario adecuar un sistema integrado para la enseñanza de la suma, sistema en el cual siempre debe haber una cabeza dirigente, la cual corresponde al profesor. Los otros componentes de sistema son, los alumnos primordialmente y los recursos de enseñanza, entre los que sobresale el computador, en este caso el MACINTOSH PLUS, con la aplicación SUMANDO.

La manera correcta de actuar en el proceso de enseñanza es la siguiente:

El profesor dicta su clase aprovechando los recursos que tiene a mano, tales como libros, tablero, láminas, etc., y una vez considere que un tema ha sido explicado en su totalidad (por ejemplo, clasificación de conjuntos), debe llevar a los niños a practicar en el computador dicho tema, para lo cual se puede disponer de una o dos semanas según la distribución del curso, y teniendo en cuenta que la aplicación SUMANDO, consta de 16 temas. En paralelo a este trabajo se debe hacer un análisis en grupo de los resultados obtenidos al estudiar determinado tema para de esta forma corregir las malas interpretaciones y los errores que presenten los estudiantes

El sistema, por su tamaño y por los temas que abarca, presenta un desarrollo a largo plazo, por lo que en estos momentos es difícil opinar sobre su validez y por este motivo se hace necesario iniciar una etapa de uno o dos años de pruebas, para lo cual es indispensable contar con el apoyo de algún colegio que preste este servicio. Sinembargo, a esta alturas y una vez terminada la aplicación, se puede decir satisfactoriamente, que ésta ha sido desarrollada para abarcar en su totalidad todos los objetivos propuestos, por lo que se espera que su utilidad sea bastante elevada, para bien de la educación colombiana.

Es indiscutible que la falta de color en el computador MACINTOSH PLUS, afecta de cierto modo la acogida que presente el niño hacia el sistema, de todos modos es importante saber que una vez se cuente con este recurso, no será demasiado difícil adecuar la aplicación, ya que ésta, está basada en el uso de gráficas, las cuales pensando en posibles mejoras técnicas, fuerán implementadas como recursos, lo cual permite la adaptación de nuevas gráficas a la aplicación sin necesidad de realizar cambios a las unidades que conforman la misma.

S I S T E M A D E E D U C A C I O N

P O R C O M P U T A D O R "S E C O M"

AUTORES : JOSE ABRAHAM CASTAÑEDA VARGAS

MARIA FERNANDA RAMIREZ DE CASTAÑEDA

DIRECCION : Diagonal 127 A # 9 B-35 - BOGOTA D.E.

El Sistema de Educación por Computador "SECOM", es la forma más revolucionaria, útil y segura de cambiar fundamentalmente los elementos del proceso de Enseñanza-Aprendizaje, dentro del país y posiblemente del exterior, al brindar en forma dinámica y permanente la actualización y mejoramiento de los medios que intervienen en el proceso educativo, generando a los educandos y profesores, nuevas maneras técnica y pedagógicamente equilibradas, dentro de un proceso eminentemente CIBERNETICO.

El Sistema produce sus propios registros de control que permiten el análisis y estudio de los resultados y por ende la modificación de contenidos, ayudas y medios evaluativos, haciendo cada vez, más comprensible estos elementos, dentro de un proceso ininterrumpido, que genera en forma objetiva y determinante, los indicadores que inicien los cambios necesarios para la optimización del proceso, y otros elementos para la transmisión del conocimiento.

Parte de la base que el conocimiento es uno solo, dentro de cada Area, suministrado en dosis, de acuerdo a los niveles de Aprendizaje, en un continuo y recionalizado edificar educo-formativo, consciente e interdisciplinario, que acelera la captación del estudiante y conduce al profesor a su actualización.

SISTEMA DE EDUCACION POR COMPUTADOR "SECOM"

1 - ANTECEDENTES

Desde mediados de 1980, se ha venido trabajando en la generación de un Sistema Educativo, que fundamentado en el Computador, permita la renovación total de la docencia y de los métodos de Aprendizaje utilizados en el país, especialmente aquellos dirigidos a la niñez y la primera infancia, como pilares éstos, de la modernización y proyección de la Educación.

No obstante, también se ha utilizado el Sistema en el campo universitario, y más aún, en el Sistema de Educación a Distancia, como prueba defintiva de la bondad del Sistema, en esta difícil modalidad de Enseñanza.

Los primeros pasos dados en 1978 y 1979, resultaron de la incorporación a la docencia universitaria, de una serie de ayudas y medios didácticos para el desarrollo de las cátedras, que condujeron a la creación de un sistema de apoyo a la Enseñanza, en virtud de los asombrosos resultados obtenidos en las diferentes materias en que se utilizaron. Estos resultados se puedan sintetizar en :

- La captación aumento en un 40%, en su primera versión,
- La posibilidad de repetición,en forma personal o colectiva, aumento el porcentaje anterior,
- La posibilidad de mejoramiento del material surgió en forma inmediata, y
- El Tiempo de las cátedras se redujo en un 30% a un 40%.

Lo anterior orientó este proceso experimental, a la búsqueda de mejores formas de suministrar la docencia, en aquellas materias, que de suyo, son de muy lenta comprensión y los niveles de aprendizaje son muy bajos.

Pero a su vez, esta serie de descubrimientos fué mostrando - también, la obsolecencia del material que habiamos preparado y la imperiosa necesidad de encontrar un método más eficaz, para su renovación y mejoramiento, que permitiera a la vez, la actualización con la misma velocidad de su renovación.

Era preciso encontrar tecnicas que contemplaran la modificación y actualización del material, en forma precisa, al alcance de los profesores, ofreciendo una fácil reproducción.

Aquí es donde surge la posibilidad de utilización del Computador, como medio para la actualización del material. Desde ese momento, su irrupción dentro del campo de la educación, ha modificado sustancialmente los conceptos, convirtiendose en el eje central de toda innovación.

A medida que la Tecnología desarrolla nuevos elementos que pueden incorporarse a estos propositos, este instrumento ha venido aumentando su valor, viniendo a ser el más elemental factor como elemento renovador de la Educación y el generador de los futuros sistemas de Enseñanza.

El Computador ha dado origen a la ejecución de acciones muy específicas, relacionadas con sus inmensas posibilidades, que puestas al servicio de la Enseñanza-Aprendizaje, desembocan en dos grandes Areas, a saber:

- Diseño de Instrucción
- Manejo y Administración de la Instrucción

Estos elementos, especialmente el segundo, representan un universo en donde se genera una nueva Metodología en el proceso innovador y revolucionario de la Informática.

Cada día son mayores las utilizaciones y programas, no solo para el Diseño de Instrucción, sino para el manejo y administración de la propia instrucción. Naturalmente que resulta casi imposible determinar el límite de la una y de la otra y por general similitud, se tiende a ver el uso del equipo, como un medio para el diseño de cursos, sin descubrir su verdadero prodigio, es decir, la utilización del Computador como un medio, eficaz y revolucionario dentro del sistema de enseñanza.

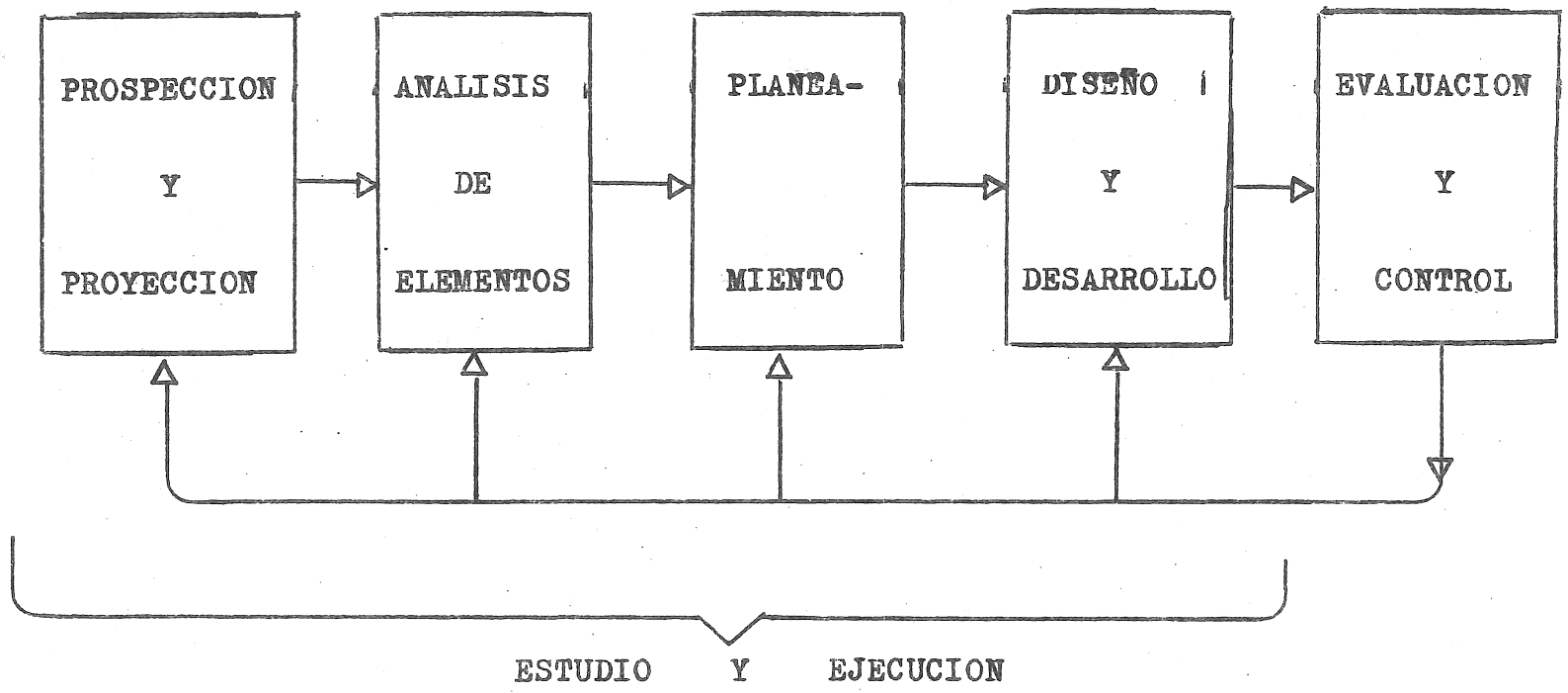
En paises desarrollados el cambio originado por el Computador en la Educación, ha sido tan fuerte, que ha obligado a dictar Leyes, en las cuales, han dado un tiempo perentorio a los profesores para su capacitación en la utilización del Computador, como un medio de Enseñanza.

En Colombia hemos desarrollado diversas versiones, tendientes todas, a la preparación de cursos de enseñanza en diferentes tópicos, pero sin prestar atención a lo realmente fundamental, que consiste en la utilización del Computador como un medio de Enseñanza-Aprendizaje, vale decir, como el generador de un nuevo Sistema de Educación.

Desde 1983, cuando se presentaron las primeras experiencias sobre el particular, la mayor parte de sus autores han venido mejorando el contenido y calidad de los cursos, pero ninguno ha desarrollado, hasta el presente, un sistema que convierta al Computador en un medio básico a la producción del material, le incorpore los aspectos indispensables sobre Evaluación, y menos aún, lo torne como parte de un sistema integrado, vinculado a la docencia, con su propio caracter, produciendo una forma de alimentación correctiva, etc.

Veamos el esquema general, que ha conducido a la creación del Sistema de Enseñanza por Computador.

ESQUEMA GENERAL PARA DESARROLLAR EL SISTEMA



ETAPAS BASICAS PARA LA CREACION Y DESARROLLO DEL SISTEMA

2 - FILOSOFIA DEL SISTEMA

Teniendo en cuenta el nivel de captación, inherente como un elemento propio a la individualidad de cada ser humano, es indispensable concebir y definir la estructura de un sistema, que analice, diagnóstique y plantee también soluciones, cuando la situación lo requiera, en las muchas instancias del proceso de Aprendizaje, con diversas alternativas y posibilidades.

El Sistema debe alimentar, regular y controlar el desarrollo instruccional, atendiendo con la misma prioridad los aspectos de diseño y construcción del material, como de manejo y administración de las formas de Enseñanza, dentro de un contexto pedagogico, de tal manera, que alcance su propia esencia, al interior de un proceso Cibernético.

Se fundamenta este Sistema, en el acopio y utilización de los planteamientos y logros, que a través de la historia, han venido generando, los estudiosos y entendidos de estas materias. De esta manera recoge los resultados que se han obtenido en el quehacer ininterrumpido de encontrar mejores metodos de transmisión-captación de los conocimientos, hasta lograr un procedimiento que de manera gradual, reincorpore sus frutos, convirtiendolos en nutriente crítica, de nuevas y permanentes metas.

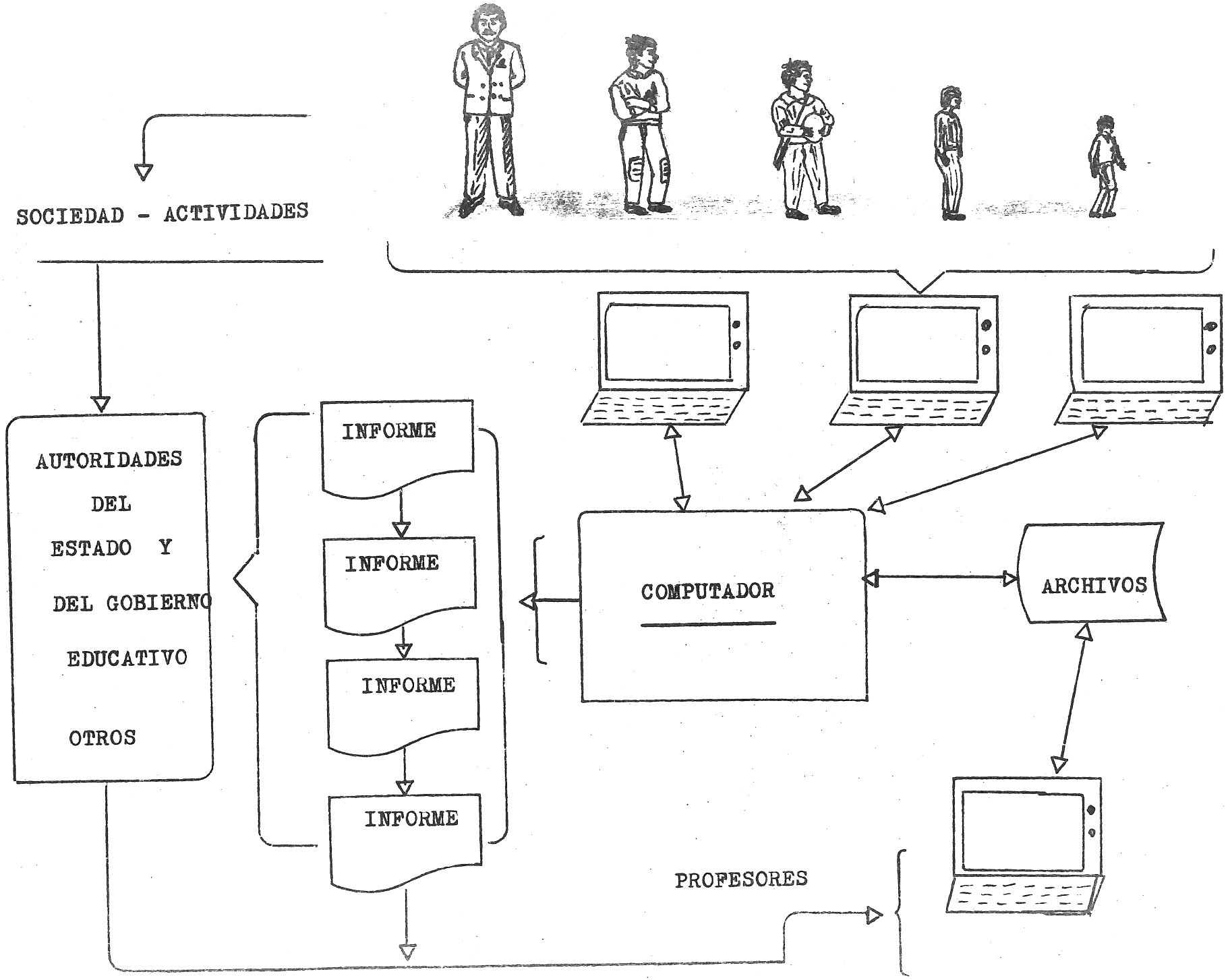
El Sistema conlleva, la generación bajo los mismos preceptos, del proceso Evaluativo-instruccional, en cualesquiera de los niveles del conocimiento y de las etapas en que se emplee, de los mecanismos básicos, que permitan la regulación, el control, la actualización y finalmente la retro-alimentación general del mismo proceso.

Es decir, debe contener y contiene dentro de sus aspectos estructurales y conceptuales, la dinámica de su propia regulación, al incorporar dentro de su propia funcionalidad los elementos críticos, que originen el ajuste y modificación de los contenidos y conceptos.

Esta concepción es posible, por las divisiones del conocimiento y sus naturales fraccionamientos, de los cuales se desprende la conclusión de puntualizar, los contenidos y criterios que deben utilizarse para su modificación.

De igual manera permite a los docentes la crítica de su propio trabajo, al suministrarles los elementos de juicio, que le permitan precisar los puntos debiles del contenido, así como la revisión del aspecto evaluativo, instándolos a su corrección y por ende a su actualización.

Por último, le facilita al estudiante la comprensión, por la ampliación y complementación del material instruccional.


SOCIEDAD - ACTIVIDADES
AUTORIDADES DEL ESTADO Y DEL GOBIERNO EDUCATIVO
OTROS
INFORME
INFORME
INFORME
INFORME
COMPUTADOR
ARCHIVOS
PROFESORES

3 - ESTRUCTURA DEL SISTEMA

Para la funcionalidad del Sistema se ha creado una estructura jerarquizada, equivalente o representativa de los niveles del proceso instructivo y paralelamente se ha creado también, una segunda estructura que permite registrar las diversas instancias de captación y comprensión del tema analizado, en una escala de valores, lo suficientemente detallada que arroja en forma precisa, los resultados de las reacciones de los educandos a los temas tratados por el Sistema.

Esta operabilidad del Sistema, se fundamenta en la estructuración, en un simbolismo similar al anterior, de la escala del conocimiento, en sus areas y contenidos fraccionados de tal forma, que permiten establecer con precisión a que corresponden las reacciones presentadas por el estudiante a los temas, sub-temas, unidades, capitulos, contenidos evaluativos, etc.

Partiendo de un contenido básico, clasificado segun el orden anterior, correspondiente a un tema especifico, se fracciona éste, en unidades menores, que faciliten la creación del material y su incorporación en los pasos siguientes del proceso, creando en cada uno de ellos las diferentes alternativas de complementación y adición, susceptibles todos, de un paso correctivo de revisión, complementación, adición o bien, de un replanteamiento expresivo, pedagógico, de forma o de contenido. En igual forma se procede para el análisis del proceso de Evaluación y el de reacción del educando a los pasos expuestos.

La estructuración se realiza identificando las inter-relaciones que deben existir entre los contenidos, los archivos que estos generan, los procesos necesarios para la operacion y manejo del Sistema, los resultados que se obtienen y los registros de las frecuencias que van arrojando los estudiantes, tanto en el proceso de comprensión, como en el de retorno a traves de las reacciones a los contenidos de Evaluación.

A un nivel macro la estructura se puede esquematizar en un diagrama como el de la pagina precedente, en el cual se identifica lo que aspira obtener de los procesos del Sistema, en términos de informes y pantallas, para responder a las necesidades del entorno y la afectación que éste puede realizar sobre la operación del mismo Sistema.

Además es conveniente y necesario artícular operativamente los flujos de información, que se dan entre las fuentes generadoras de datos, los procesos, los archivos y los usuarios de la información del Sistema.

ESTRUCTURA CODIGO DE MATERIAS

TIPO	RAIZ		FAMILIA		MATERIA
	AREA	SUB-AREA	GRUPO	SUB-GRUPO	- O CLASE - CONTENIDO - CATEDRAS
1/4	1/5	1/9	1/9	1/9	01 / 99
1	2	3	4	5	6 - 7

POSICIONES O CARACTERES

NOTA : PUEDE AGREGARSE UN OCTAVO (8) DIGITO PARA SEÑALAR EL NIVEL DE LA ENSEÑANZA, ASI :

1 = JARDIN	6 = Post-Gradual
2 = PRIMARIA	7 = Doctorado
3 = SECUNDARIA	8 =
4 = TECNICA	9 = Cursos Libres
5 = UNIVERSITARIA	0 = Otros

4 - ESTRUCTURA DEL CODIGO DE MATERIAS

Esta estructuración se fundamenta en cuatro niveles de información, cada uno de ellos totalmente independientes, pero integrados dentro del Sistema, como Indices o indicadores del nivel de detalle de la información de contenido de la respectiva materia, en un proceso de desagregación que llega a los mayores detalles, en función del desarrollo de la Enseñanza, es decir, en funcion del proceso de instrucción, afianzado en nuevos conceptos sobre el manejo de contenidos y su consiguiente actualización.

Estos cuatro niveles de la estructuración del Codigo, permiten el acceso selectivo a los mismos, dentro de una concepción dinámica del manejo de contenidos, reduciendo en esta forma los costos de operación y actualización.

Veamos la estructura del Código de Materias (Codificación de Materias), que se explica en la siguiente forma:

4.1 - TIPO (Topología)

- 4.1.1 - Teoría
- 4.1.2 - Tecnología Formativa - Específica
- 4.1.3 - Tecnología Aplicada - Complementaria
- 4.1.4 - Tecnología Práctica - Laboratorios-Talleres-Seminarios-Proyectos-Etc.

4.2 - RAIZ

Esta a su vez se divide en dos grupos, así:

- 4.2.1 - Area
  - 4.2.1.1 - Matemática
  - 4.2.1.2 - Comunicación
  - 4.2.1.3 - Ciencias Naturales
  - 4.2.1.4 - Ciencias Sociales
  - 4.2.1.5 - Humanidades
- 4.2.2 - Sub-Area
  División de cada una de las Areas. Desagregación y ordenamiento dentro de cada una de las Areas.

4.3 - FAMILIA

Las Areas y Sub-Areas se dividen a su vez en :

- 4.3.1 - Grupos
  Son las agrupaciones o clasificaciones de Sub-Areas.
- 4.3.2 - Sub-Grupos
  Agrupaciones o clasificaciones de los grupos. Desagregaciones de los grupos. Clasificación de los Sub-grupos de contenido.

4.4 - MATERIA (Contenido)

Esta clasificación u ordenamiento, identifica el contenido específico de cada una de las materias, en una sucesión relativa al suministro de conocimientos en orden al desarrollo de los cursos o programas, de acuerdo al nivel de ubicación de los estudios.

Esta relativa sucesión de conocimientos, representa en esencia, la forma como ha sido construida la secuencia de los cursos dentro de los programas o niveles de aprendizaje, es decir, la dosificación y suministro de porciones, en consonancia con los objetivos de las cátedras, materias y cursos.

Estos contenidos y sus repsectivos Syllabus, identifican o corresponden a un numero secuencial dentro de los diseños de las cátedras o materias, en cada familia, Area, grupo, etc. de conocimientos.

Los niveles subsiguientes en donde se trata la desagregación de los elementos de cada materia, corresponde a la estructuración y codificación de contenidos que veremos mas adelante.

ESTRUCTURA DE CONTENIDOS

CODIFICACION DE MATERIAS Y CONTENIDOS		
CODIGO MATERIA	CODIGO CONTENIDO	DESCRIPCION
1/7 (Opcion 1/8) 1 8	1/7 9 15	1/30 16 46

M O D U L O S

CODIFICACION		CONTENIDOS	
CODIGO MATERIA 1 8	MODULO 1/9 9	CONTENIDO	= 1

FASCICULOS

CODIFICACION	CONTENIDOS		
CODIGO MATERIA 1 8	MODULO 9	FASCICULO 01/99 10 11	CONTENIDO = 2

UNIDADES

CODIFICACION	CONTENIDOS			
CODIGO MATERIA 1 8	MODULO 9	FASCICULO 10	UNIDAD 01/99 11 12 13	CONTENIDO = 2

CAPITULOS

CODIGO MATERIA 1 8	MODULO 9	FASCICULO 10	UNIDAD 11 12 13	CAPITULO 14 15	CONT = 2

5 - ESTRUCTURA DEL CODIGO DE CONTENIDOS DE LAS MATERIAS

El Código de Contenidos (Códigos de los elementos de los Módulos) representa la forma como se presenta y artícula la - desagregación de los Contenidos.

Esta codificación obedece a la organización de los contenidos de las materias, en Módulos y Fascículos y la subdivision de éstos, en Unidades y Capítulos.

Veamos como se organiza este fraccionamiento de los Contenidos de cada materia.

5.1 - Módulos

Esta división corresponde al fraccionamiento de una materia en diversos Módulos, de acuerdo al contenido - extension y facilidad de comprensión. Estando numerados de 1 a 9 para su integración.

5.2 - Fascículos

Los Módulos a su vez se dividen en Fascículos, es - decir fracciones en que se desagregan los contenidos del Módulo. Estos estan numerados de 01 a 99.

5.3 - Unidades

Cada uno de los Fascículos, a su vez, esta dividido en Unidades, numeradas de 01 a 99, para una mayor - desagregación del contenido del Fascículo.

5.4 - Capítulos

Las Unidades a su vez, estan divididas en Capítulos que permiten el fraccionamiento de las Unidades en - contenidos de orden menor, los cuales estan numerados de 01 a 99.

Esta estructura y codificación de los Contenidos de cada - una de las Materias, corresponde a las porciones del contenido instructivo de las parcelas o unidades de orden académico en que se ha fraccionado el suministro de conocimientos en cada una de las Areas, Familias, etc.

El funcionamiento y operabilidad de esta estructura se hallan integrados dentro del Sistema, de tal forma que constituyen elementos operadores determinantes, sin los cuales no podrían definirse los aspectos y puntos de mayor relieve dentro del proceso de Enseñanza-Aprendizaje.

Es importante destacar la facilidad que presenta esta estructura para la modificación y actualización del material instruccional a tal punto, que los docentes efectúan los - cambios en los elementos, sin tener que reeditar el contenido.

ESTRUCTURACION DEL SISTEMA DE EVALUACION

CODIFICACION DE MATERIAS	
CODIGO DE MATERIA	DESCRIPCION

CODIFICACION DE CONTENIDOS		
CODIGO DE MATERIA	CODIGO DE CONTENIDO	DESCRIPCION

CODIFICACION DE EVALUACIONES-PREGUNTAS			
CODIGO MATERIA	CODIGO CONTENIDO	CODIGO PREGUNTAS	DESCRIPC.

SISTEMA EDUCATIVO - SUB-SISTEMA DE EVALUACION

El Sistema de Evaluación corresponde a un Sub-Sistema, dentro del Sistema de Educación por Computador, siendo uno de los pilares básicos de la integración y funcionamiento del Sistema.

El Sistema de Educación por Computador, tiene directa relación - con el contenido de las Materias de los diferentes programas académicos, en cada una de las Instituciones de Enseñanza, y a su - vez tiene relacion, con las Areas del conocimiento contenidas en la estructuración del Código de Materias, con la estructuración del Código de Contenidos y con la propia estructuración del Banco de Preguntas.

El Sistema de Educación por Computador, esta fundamentado en cuatro niveles de información, cada uno de ellos totalmente independiente,,pero integrados y funcionalizados por el Sistema, como indicativos o Indices del nivel de la información de Contenidos, en un proceso de desagregacion que llega a los mayores detalles, en concordancia con el desarrollo de la instruccion, es decir, del - proceso de Enseñanza-Aprendizaje.

El Sistema de Evaluación constituye un punto convergente de los - elementos que integran el Sistema Educativo, al arrojar los resultados inherentes al proceso de la Enseñanza, con la correspondiente identificación de las partes que denotan inconsistencias, o - bien frecuencias desusuales, que implican corrección inmediata o ajuste a mediano plazo, segun el caso.

Por este medio, el evaluativo, podemos visualizar las dificultades presentadas por el proceso, los niveles de captación de los educandos, la necesidad de corrección de los instructivos, incluso la medida de aceptación de los temas y contenidos de las preguntas.

Igualmente sirve de canal de alimentación del Sistema, al proporcionar a los profesores, los elementos determinantes de sus ajustes y mejoras al material instruccional. Es también elemento de juicio, dentro del proceso, para indicar los niveles de agrado y comprension de los temas.

Por último el Sistema de Evaluación, permite detectar, una vez - eliminado el mismo proceso como resultado susceptible de correccion o mejoramiento, con todos sus alcances y ponderaciones, la medida que cada extracto estudiantil, denota como característica representativa de su captación y comprensión del tema expuesto y evaluado, sino además orienta de manera definitiva en la puntualización de la calidad y bondad pedagógica del material utilizado como medio instruccional.

ESTRUCTURACION DEL BANCO DE PREGUNTAS

CODIFICACION DE MATERIAS-CONTENIDOS Y BANCO DE PREGUNTAS			
CODIGO MATERIA 1/7 (8)	CODIGO CONTENIDO 1/7	CODIGO PREGUNTA 1/6	DESCRIPCION 1/30

CODIFICACION BANCO PREGUNTAS - NIVEL DE CONDUCTA				
CODIGO MATERIA	CONTENIDO	NIVEL CONDUCTA	DESCRIPCION	= 1
		. . .		

CODIFICACION BANCO PREGUNTAS - CLASE PREGUNTA					
CODIGO MATERIA	CONTENIDO	NIVEL	CLASE	DESCRIPCION	= 2
			. . .		

CODIFICACION BANCO PREGUNTAS - NUMERO PREGUNTA						
CODIGO MATERIA	CONTENIDO	NIVEL	CLASE	NUMERO	DESCRIPCION	= 3
				. . .		

1 - ESTRUCTURA DEL BANCO DE PREGUNTAS

Esta Estructura contiene los elementos y la funcionalización de las Evaluaciones, en cada una de las Materias, de acuerdo a un esquema básico de integración y de la utilización de las diferentes formas que usualmente se acostumbran dentro de esta modalidad.

Esta estructuración obedece a la jerarquización de las Pruebas, en consonancia con una serie de ordenamientos de magnitud y secuencia, de los elementos que las componen.

Entendemos obviamente, que las características y las formas normales utilizadas en este tipo de trabajo, son y serán modificadas, por una parte por el desarrollo sico-pedagógico, y por otra por los resultados obtenidos en la operación del Sistema y las secuencias lógicas derivadas de su funcionamiento.

El objeto de crear un Banco de Preguntas, obedece al propósito de generar procedimientos aleatorios para la selección de las preguntas, en forma tal, que no sea necesario estar incluyendo permanentemente nuevas preguntas. El Banco se crea y revalua en forma sistemática como resultado de la función del Sistema.

Veamos como esta concebida su Estructura :

1.1 - NIVEL DE CONDUCTA

Se han tomado los niveles representativos de las diferentes formas en uso, actualmente, asi :

1.1.1 - Análisis
1.1.2 - Evocación
1.1.3 - Aplicación
1.1.4 - Síntesis
1.1.5 - Razonamiento

1.2 - CLASES DE PREGUNTAS

Clasificación de las preguntas de acuerdo a su naturaleza y características, así :

1.2.01 - Verdadero - Falso
1.2.02 - Verdadero - Falso o Corrección
1.2.03 - Verdadero - Falso - Opinión
1.2.04 - Si - No
1.2.05 - Hecho - Opinión
1.2.06 - Pregunta Directa
1.2.07 - Completar Respuesta
1.2.08 - Correspondencia
1.2.09 - Selección Multiple con respuesta única

1.2.10 - Selección Multiple con respuesta multiple
1.2.11 - Clasificación
1.2.12 - Información Suficiente
1.2.13 - Análisis de Relación
1.2.14 - Ensayo Limitado
1.2.15 - Ensayo Extenso
1.2.16 - Oral
1.2.17 - Ejecución
1.2.18 -

1.3 - NUMERO DE LA PREGUNTA

Esta numeración corresponde al ordenamiento de los temas evaluativos en relación con las características del tipo de pregunta, pudiendo recogerse hasta novecientas noventa y nueve posibilidades, dentro de cada de las Clases enumeradas arriba.

Se ha determinado un volumen de preguntas de esta naturaleza, para atender el crecimiento paulatino de este tipo de modalidad, tambien se ha pensado como un medio de generacion de temas de evaluacion, que cubra la totalidad del tema tratado en el proceso instruccional, sin tener que acudir a la renovación del material con frecuencia.

Esta Estructura permite el crecimiento en forma paulatina de acuerdo a las necesidades y a la utilización de los temas de Evaluación.

Igualmente para aumentar las posibilidades del Banco de Preguntas, se ha pensado la utilización en forma aleatoria para multiplicar las oportunidades y evitar la frecuencia de las mismas preguntas.

La elaboración de los temas evaluativos, nace de la concepción del material instruccional, por esta razón cada una de las partes que componen el Sistema, aun que tienen su propia naturaleza y características, que les permiten trabajar en forma separada, han sido integradas dentro de la concepción y creación general de Sistema.

Estan perfectamente eslabonadas y coordinadas que constituyen la base del Sistema, de tal forma, que se complementan y corresponden, en el funcionamiento y operación total de Sistema, como elementos indispensables y secuenciales del proceso.

CODIGO DE LOCALIZACION

GEOGRAFICO		ESTABLECIMIENTOS		DESCRIPCION DEL
DPTO	MCPIO	TIPO	NUMERO	LUGAR o ESTABLECIMIENTO
1/2	1/3	1/2	1/3	1/30

CODIGO GEOGRAFICO - DPTO

DEPARTAMENTO	DESCRIPCION
. . . .	

= 2

CODIGO GEOGRAFICO - MCPIO

DEPARTAMENTO	MUNICIPIO	DESCRIPCION
	. . .	

= 3

CODIGO ESTABLECIMIENTO- TIPO

CODIGO DPTO	MCPIO	T I P O	DESCRIPCION
		. . .	

= 2

CODIGO ESTABLECIMIENTO - NUMERO

CODIGO DPTO	MCPIO	TIPO	NUMERO	DESCRIPCION
			. . . .	

= 3

2 - ESTRUCTURA DEL CODIGO DE LOCALIZACION

Finalmente, si también queremos información sobre la ubicación de los Centros de Educación, los tipos de Centro y los resultados obtenidos de la operacion del Sistema, se ha incorporado al mismo Sistema, la Estructura de un Código de Localización que indica geográficamente, por tipo y clase de centro educativo, los resultados del trabajo realizado en los diferentes niveles del proceso de educación.

Se ha incorporado a esta Estructura la división geo-política del país y una clasificación de los centros educativos a fin de poder ubicar por sectores o regiones, la respuesta que brinda el Sistema.

Falta solamente dentro de este contexto, la ubicación por grupos de edad y por sexo de los educandos sometidos al trabajo con el Sistema, pero esta información nos la brinda el archivo general de estudiantes en cada unidad o Centro educativo, que ha sido prevista dentro de los elementos del Sistema.

El Código de Localización, en esencia, corresponde a los siguientes elementos :

2.1 - GEOGRAFICO

2.1.1 - Departamento

Utilizando la codificación establecida por el DANE, se puede representar el Departamento, Intendencia o Comisaria del territorio nacional. Este Código consta de dos digitos, es decir, de 01 a 99 posibilidades para codificación.

2.1.2 - Municipio

Para esta codificación también se utiliza el Código del DANE, que indica el Municipio al cual corresponde la información y de esta manera, es posible intercambiar datos con todos los Organismos del Estado, especialmente con los establecimientos del sector educativo. Este Código consta de tres digitos de 001 a 999.

2.2 - ESTABLECIMIENTO EDUCATIVO

Esta codificación se ordena y clasifica de acuerdo a los tipos de establecimientos o centros docentes y a su numero de registro. Se explican así :

2.2.1 - Tipo de Establecimiento

Esta ordenación se refiere al tipo de establecimiento, si es privado o no y dentro de ésta, al tipo de docencia que suministra.

2.2.1.1 - Oficial

2.2.1.1.1 - Jardines
2.2.1.1.2 - Pre-escolar
2.2.1.1.3 - etc,

2.2.1.2 - Privado

2.2.1.2.1 - Jardines
2.2.1.2.2 - etc.

2.2.2 - Número de Registro

El Número de Registro corresponde al número asignado por el Estado a cada uno de los Centros Educativos. Este registro puede provenir de Secretaría de Educación regional o del Ministerio de Educación.

Con esta Estructura de Localización se puede establecer en forma perfectamente definida, las reacciones que presentan los sectores, regiones, centros, modalidades educativas, en fin, es posible puntualizar y establecer comparaciones y frecuencias del desarrollo del Sistema Educativo.

El Sistema de Educación por Computador "SECOM", proporciona la solución a los problemas consuetudinarios del país y brinda igualmente, la posibilidad inmediata de renovar y actualizar las formas educativas, revolucionando de paso el proceso de Enseñanza-Aprendizaje, para los educandos por una parte y proyectando a los docentes a su verdadera labor educacional, actualizandolos y manteniendolos al alcance de las innovaciones dentro del campo.

USO DE MINIX PARA LA ENSEÑANZA DE SISTEMAS OPERATIVOS.

Martín SANTANA[1], Miguel SANTANA[2]

[1]Escuela de Administración de Negocios para Graduados
Apartado 1846, Lima 100, PERU

[2]Pontificia Universidad Católica del Perú
Escuela de Graduados
Apartado 1761, Lima, PERU

RESUMEN

La enseñanza en el área de los Sistemas Operativos siempre se ha visto confrontada a una serie de dificultades, especialmente en nuestros países, donde las Universidades sufren de una escasez notoria de recursos:

a) Disponibilidad de máquinas con las cuales experimentar.

b) Restricciones debidas a las características del sistema de estas máquinas (multiproceso, tiempo compartido, niveles de protección, acceso a los dispositivos hardware, etc).

c) Falta de herramientas de experimentación.

d) Complejidad de la interfase que define las rutinas de base de un sistema (creación de procesos, comunicación/sincronización entre procesos, uso de la memoria virtual, entradas/salidas, etc).

e) Falta de homogeneidad en el diseño de las máquinas y mucho más entre máquinas diferentes.

Estos problemas conducen en muchas ocasiones a la realización de cursos casi exclusivamente teóricos, con pocos ejercicios y aún menos aplicaciones prácticas; es una situación paradójica, pues se trata de un área eminentemente práctica, donde la experiencia juega un rol preponderante.

Algunos de estos problemas han podido ser resueltos gracias a la vasta difusión de los micro-computadores y, en especial, del PC-IBM y compatibles. Efectivamente, el bajo precio de éstos, unido a las condiciones especiales de venta que benefician al sector enseñanza, han permitido equiparse a la mayoría de las Universidades e inclusive a muchos de sus alumnos. Por otro lado, la simplicidad del sistema operativo de estas máquinas (MS-DOS, CP/M u otros) ha jugado en nuestro favor; por ejemplo, la ausencia de mecanismos de protección y el fácil acceso a los

dispositivos de la máquina han permitido diseñar trabajos de mayor envergadura (tratamiento de interrupciones, drivers de E/S) e inclusive proyectos completos (sistema tiempo real, sistema de archivos, etc). Otra consecuencia de la popularidad de estas máquinas es la facilidad con la que uno puede conseguir todo tipo de informaciones sobre ellas: descripción de los vectores de interrupción, interfase de escritura de un driver de E/S, manuales hardware, etc.

La contribución del sistema UNIX también ha sido de mucha importancia, tanto por los conceptos innovadores que introdujo como por las facilidades que brinda para la realización de todo tipo de herramientas. La mayor parte de los especialistas modernos en Sistemas Operativos se han formado a través de UNIX, tanto por haberlo usado intensivamente como por haber construído herramientas para él o haberlo portado hacia otras máquinas. Desde este punto de vista, UNIX constituye un excelente banco de experimentación; sin embargo, su complejidad, aunada a las restricciones de acceso a sus fuentes, dificultan su uso en la parte práctica de la enseñanza de Sistemas Operativos.

Desde hace algunos meses, ha aparecido en el mercado una nueva versión de UNIX, denominada MINIX, con fines estrictamente educativos. Se trata de un nuevo diseño de UNIX, totalmente independiente de AT&T y realizado por el Profesor Andrew Tanenbaum de la Universidad de Vrije (Amsterdam). Una característica importante de MINIX es que ha sido implementado en los PC-IBM, lo que le garantiza un público bastante amplio; por otro lado, el sistema es vendido por un precio derisorio (alrededor de 80 dólares) y bajo una modalidad bastante novedosa: el cliente está autorizado a hacer copias del sistema, inclusive para otras personas. Otra característica interesante de MINIX es que el sistema es entregado con sus fuentes; de esta manera, el usuario puede modificarlo y regenerarlo, sin ningún problema.

Es obvio que un sistema de este tipo representa una excelente oportunidad para los profesores de Sistemas Operativos. Así lo entendió la comunidad informática mundial, que ha demostrado un inmenso interés por MINIX: ediciones agotadas, creación de fórums reservados a MINIX en las principales redes mundiales, desarrollo de una gran cantidad de nuevas herramientas para él, etc. Muchos cursos de Sistemas Operativos han sido desde entonces remodelados en función de MINIX, tanto en Estados Unidos como Europa.

Es nuestra opinión que las Facultades de Informática de América Latina no deben perder esta ocasión y, por el contrario, tratar de aprovechar al máximo este movimiento. Es por esta razón que estamos organizando los nuevos cursos de Sistemas Operativos, tanto en ESAN como en la Universidad Católica del Perú, en base a MINIX. El objetivo de esta comunicación es bastante simple: compartir nuestra experiencia con las personas interesadas y presentarles nuestras inquietudes al respecto.

Red de Intercambio de Información sobre Informática Educativa

Grupo de Informática Educativa, Universidad de Los Andes

Alvaro H. Galvis Panqueva, Olga Mariño, Francisco Rueda

Apartado Aéreo 4976, Bogotá. Tel 286-6185

Propósito

La red pretende favorecer la ubicación, recopilación, organización y difusión de información entre personas y entidades que se interesan por la Informática Educativa. Para ello articula fuentes y usuarios de este bien.

Servicios

A través de la red de intercambio los participantes pueden obtener los siguientes servicios :

1. Publicación sobre **Informática Educativa**, con periodicidad cuatrimestral, a partir del primer semestre de 1988. Cada número se dedicará a un tema específico sobre el que habrá al menos un artículo completo; contará con una sección de novedades documentales recibidas, otra sobre publicaciones, eventos y productos relacionados con el tema. Según evolucione la publicación, contará con otras secciones.

2. Servicio especializado de **acceso a publicaciones periódicas relacionadas.** Las tablas de contenido de las revistas a que la biblioteca del CIFI tiene acceso (por sí misma o por medio de los centros de documentación cooperantes) permitirán a los usuarios pedir copia de artículos de su interés.

3. Servicio especializado de **consulta a bases de datos nacionales o internacionales** relacionadas con el tema, a petición y como servicio pagado por los usuarios.

Participación

Hay dos tipos de usuarios en la red : institucionales y personales. Los interesados en participar pueden pre-inscribirse llenando el desprendible para inscripción adjunto. Las personas que anteriormente han llenado este formulario ya están pre-inscritas y recibirán una comunicación a finales de 1987 sobre los mecanismos operativos y costos de participación en la red.

Es muy posible que la red sea financiada, parcial o totalmente, con fondos de COLCIENCIAS. Por este motivo no se han establecido los aspectos operativos y de costos. El Grupo de Investigación sobre Informática Educativa y el Centro de Documentación del CIFI darán el soporte técnico para el funcionamiento de la red.

INSCRIPCION A LA RED DE INTERCAMBIO DE INFORMACION SOBRE INFORMATICA EDUCATIVA

NOMBRE : _____ MIEMBRO : _ personal _institucional

INSTITUCION : _____ CIUDAD : _____

DIRECCION : _____ TELEFONO : _____

Formación Básica : _____

Actividad profesional : _____

Experiencia en Informática Educativa : _____

Actividades de Interés en Informática Educativa (Marque con X donde corresponda) :

__ Recibir información documental y referencial vía **Revista de Informática Educativa.**

__ Recibir información sobre cursos y eventos, nacionales e internacionales, relacionados con la Informática Educativa.

__ Intercambiar Información y Materiales (impresos o computarizados) en el área.

__ Recibir asesoría para el desarrollo de proyectos en Informática Educativa

__ Prestar asesoría para el desarrollo de proyectos de Informática Educativa

Por favor haga llegar este formulario diligenciado al

Grupo de Informática Educativa, Of. W302, UNIANDES

Apartado Aéreo 4976, Bogotá

SISTEMA MANEJADOR DE BANCOS DE PROBLEMAS

Grupo de Informática Educativa
Universidad de los Andes

Alvaro Galvis Panqueva, Olga Mariño, Francisco Rueda
Apartado Aéreo 4976, Bogotá. Tel: 286-61-85

Con esta comunicación se pretende dar a conocer el Sistema Manejador de Bancos de Problemas , desarrollado por el Departamento de Ingeniería de Sistemas de la Universidad de los Andes. A continuación se presentan los motivos que llevaron a considerar la elaboración de este banco y una descripción funcional del sistema.

Justificación:

El diseño y elaboración de items para pruebas académicas representa una gran carga para el profesor. Esto es especialmente cierto en los cursos con muchos grupos o secciones en los que es necesario asegurar una calidad homologada a través de diferentes, de igual complejidad, para los diferentes grupos o en cursos en los que se desea hacer una evaluación diferencial, generando pruebas diferentes sobre un mismo tema.

La utilización de un banco de problemas que pueda reflejar la estructura de contenidos del curso y en el que se puedan crear, modificar y consultar problemas, para generar con ellos pruebas diferenciales parciales y totales, facilita el trabajo de los profesores y permite ir completando y depurando los items de evaluación de un curso de un semestre a otro, lo cual redunda en una mejor calidad de las evaluaciones y del curso en general.

La anterior necesidad motivó la elaboración de un proyecto por etapas. Una vez realizado el análisis de necesidades se elaboró un diseño global el cual se implementó con seis tesis de grado en el departamento de Ingeniería de Sistemas: dos para el manejador del banco de problemas, una para el editor de texto, una para el de fórmulas, una para el editor gráfico y una sexta tesis para integrar los módulos y unificar la interfaz. Actualmente el proyecto se encuentra en prueba piloto y ajustes.

Banco de problemas sistematizado:

El sistema manejador de bancos de problemas facilita y agiliza la creación de pruebas académicas y mantiene información detallada y actualizada sobre los problemas y ejercicios de un curso. Además de servir para la elaboración de pruebas, el banco permite al profesor suministrar periódicamente ejercicios al estudiante y, le puede facilitar la preparación del curso al proporcionarle ejercicios que pueden ser presentados como ejemplo en la clase.

Tipo de Información

El sistema maneja tres tipos de información: cursos, temas y problemas. Cada curso puede tener hasta 20 temas principales; un tema es una lección o unidad de un curso y para un tema pueden definirse hasta 5 niveles de subtemas. Este esquema permite copiar fielmente la estructura de objetivos y lecciones de un curso.

Los problemas son la unidad básica del banco; para cada tema o subtema se pueden definir tantos problemas como se quiera. Un problema consta de tres partes: la identificación, la hoja de vida y el texto o cuerpo del problema. La identificación consta de un número y un descriptor que resume o describe brevemente el problema. La hoja de vida es un conjunto de atributos que describen características especiales del problema y permiten clasificar y seleccionar problemas (p.ej: complejidad, uso de gráficas, tipo de pregunta, etc.). Por último se tiene el cuerpo mismo del problema: éste incluye el enunciado o planteamiento, la pregunta, la respuesta o un prototipo de ésta y el desarrollo o explicación de la misma. El cuerpo del problema puede tener texto, fórmulas y gráficas.

Descripción funcional del sistema

La primera vez que se utiliza el banco debe indicarse una clave de acceso al banco y el nombre de la institución que lo utiliza. Para cada curso que se incluya en el banco, el profesor debe diseñar la estructura de la hoja de vida de sus problemas (nombre, tipo y rango de cada atributo). Una vez establecidos los parámetros básicos del sistema, el usuario puede trabajar con los módulos de mantenimiento del banco y de procesamiento de la información almacenada para la elaboración de pruebas.

MODULO DE TEMAS

Este módulo permite adicionar un nuevo tema (dando su nombre y ubicación dentro de la estructura); asimismo, modificar un tema, eliminarlo previa confirmación del usuario y revisar interactivamente el temario del curso. Además, permite seleccionar un tema o subtema como tema para procesamiento de información (tema actual).

MODULO DE PROBLEMAS

Con el módulo de mantenimiento de problemas se pueden adicionar, modificar, eliminar y trasladar problemas de un tema a otro. Este modulo trabaja con un problema del tema o subtema actual escogido por el profesor. Adicionar un problema consiste en determinar su hoja de vida, crear el texto del enunciado y, si el usuario lo desea, darle un descriptor. Para la creación del texto del problema se cuenta con un editor mixto: de texto, gráficas y fórmulas. La opción de modificación permite cambiar cualquiera de los componentes del problema y la eliminación borra un problema del banco. Con la facilidad de translación el profesor puede mover un problema de un tema a otro, si considera que los objetivos que evalúa corresponden más a éste.

ELABORACION DE PRUEBAS

El módulo principal del sistema es el de elaboración de pruebas, en el que, a partir de la información almacenada en el banco de preguntas se pueden seleccionar problemas para pruebas y generar las evaluaciones con el formato que establezca el profesor. Con este módulo pueden seleccionarse problemas de uno o más temas, especificando valores o rangos para algunos atributos de la hoja de vida; pueden elaborase exámenes a partir del conjunto de problemas seleccionados o escogiendo uno a uno problemas del tema actual. Los problemas de un examen pueden imprimirse con el encabezado y formato escogidos por el profesor. El sistema permite guardar la información de hasta 8 exámenes de cada curso, los cuales pueden consultarse, modificarse o borrarse.

PROCESOS VARIOS

Este modulo de mantenimiento le permite al profesor cambiar los atributos de la estructura de la hoja de vida; el sistema actualiza automáticamente la hoja de vida para los problemas ya creados (v.gr. desabilita los campos que se eliminan).

Requerimientos de implantación

Para este sistema se requiere un equipo con la siguiente especificación: Sistema operacional MSDOS (equipo compatible IBM); memoria principal mayor que 512 Kb; tarjeta gráfica IBM standard; disco duro, con un mínimo de 10 Megabytes; impresora que permita reproducir gráficas; teclado con funciones programables.

Estado actual del sistema

Las opciones del sistema están plenamente probadas. Se está mejorando el editor gráfico para hacerlo más amigable y transportable (v.gr. ejecutable sobre diferentes tarjetas gráficas). Por otra parte, se está haciendo una prueba de carga y eficiencia del sistema con el banco de problemas del curso de estadística del departamento de Ingeniería Industrial.

Proyecciones

Para versiones posteriores se espera incluir manejadores de diversas impresoras. También se desea agregar un módulo de corrección automática de pruebas de respuesta única para facilitar la labor de corrección de los profesores y para uso interactivo de los estudiantes.

Mayores Informes

Grupo de Informática Educativa, Of. W302, UNIANDES
Apartado Aéreo 4976, Bogotá Tel: 286-61-85

SISTEMA DE EVALUACION DEL CONOCIMIENTO DE ESTUDIANTES CON APOYO COMPUTACIONAL

por M.C. César Guzmán Rentería (México)
A/P Carlos Alberto Santana Lancón (México)

El sistema "Evaluación Mecanizada del Conocimiento de estudiantes por Computadora" (EMCEC) tiene la función de generar exámenes de evaluación de conocimientos por computadora. EMCEC cuenta con el apoyo y auspicio de Unesco y fue elaborado en la Sección de Computación del Departamento de Ingeniería Eléctrica del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV/IPN), Apartado Postal 14740, (07000) México D.F., México.

La finalidad de generar exámenes por computadoras es aprovechar los beneficios que proporciona un ordenador para el manejo y administración de una considerable cantidad de información. Un examen de admisión o evaluación, normalmente se elabora manualmente para ser aplicado a un conjunto de alumnos. Difícilmente, por el costo y esfuerzo que implica, se elaboran exámenes diferentes para el mismo conjunto de alumnos. Actualmente con un computador se tiene la facilidad de diseñar un sistema que permita tomar en forma aleatoria, de una base de datos, las preguntas relacionadas con un tema determinado. De este modo los exámenes que se generan son diferentes uno de otro, sin variar el grado de dificultad o de profundidad. En concreto la finalidad es tener un sistema que aplique exámenes en forma automática y de diferentes áreas del conocimiento, a cualquier conjunto de individuos a evaluar.

Las ventajas de un sistema de este tipo son diversas. Permite que instituciones nacionales (CONACYT, SEP) o internacionales (UNESCO, IBI, etc.), ofrezcan becas con criterio objetivo, pues es fácil aplicar el examen (la computadora lo aplica y lo califica). Resulta mejor comparar las calificaciones de exámenes generados por el sistema, que comparar calificaciones procedentes de diferentes universidades. El examen puede conformarse (a criterio del evaluador), para otorgar diferentes pesos a cada una de las áreas de evaluación. Así los requisitos de ingreso pueden variar de una escuela a otra, para exámenes de admisión.

El Sistema EMCEC se elaboró en un computador personal IBM-PC utilizando el compilador TURBO-PASCAL (versión 3.0), a través del sistema operativo MS-DOS. Para el buen funcionamiento del sistema se requiere : Un computador IBM-PC o compatible, de 256 Kb de memoria; dos manejadores de disco flexible y una impresora de 80 columnas. Las características de funcionamiento y estructura del EMCEC se enuncian a continuación:

- Maneja un banco de datos, el cual se forma por textos de preguntas de examen y sus respectivas respuestas. A su vez en él se pueden efectuar todas las funciones más comunes de mantenimiento de archivos.
- Presenta la facilidad de clasificar los problemas por áreas de conocimiento.
- Es posible diseñar la estructura de un examen, combinando preguntas de diferentes áreas de conocimiento.
- Las técnicas de selección de problemas en forma aleatoria, son tales que difícilmente se pueden crear exámenes iguales.
- Es posible llevar un control de los datos de los estudiantes. Esta información es almacenada en archivos específicos, a los que es fácil consultar o modificar.
- Cuenta con procesos de evaluación, que realizan reportes gráficos (histogramas), y cálculos estadísticos acerca del nivel académico de los alumnos que presentan examen.

El sistema se está aplicando en los programas académicos de la Maestría en Computación en el CINVESTAV/IPN, con la finalidad de optimizar el proceso de exámenes de admisión y también para detectar las fallas y desventajas del mismo.

INVESTIGACION OPERACIONAL

UNA APROXIMACION A LA DINAMICA PERIURBANA

ISAAC DYNER R.
Profesor Asociado

UNIVERSIDAD NACIONAL DE COLOMBIA

FACULTAD DE MINAS

Medellín, Agosto de 1987

RESUMEN

En los últimos lustros las contribuciones de las áreas de Sistemas e Investigación Operacional en la Planificación Urbana han sido cada vez más importantes. No obstante, su apropiación es todavía inadecuada en los países en vías de desarrollo.

En general, el proceso de urbanización de las ciudades latinoamericanas ocasiona rápidas transformaciones en los usos del suelo. Las ciudades se expanden desordenadamente y ocupan áreas rurales aptas para la producción agropecuaria; por tanto, se hace necesario que el crecimiento urbano tenga consideración especial por el ordenamiento territorial.

Con la metodología utilizada en el presente trabajo se permite el estudio de diversos escenarios para el análisis de políticas de usos del suelo.

El modelo desarrollado es de fácil complementación y manejo. Además, poco exigente en requerimientos de información.

En esta publicación se muestra como los resultados de ejercicios realizados no tienen mayores dificultades para su comprensión y análisis.

1. INTRODUCCION

El estudio de las áreas de la periferia urbana (periurbanas, suburbanas o agropolitanas) es de especial interés en países en vías de desarrollo. Con tasas demográficas y migratorias en niveles altos, niveles de equipamento y cubrimiento en servicios públicos desiguales entre regiones y pobladores, y con inadecuadas inversiones en el campo, el crecimiento de las ciudades se da a velocidades preocupantes.

En la periferia el problema producido con los cambios en los usos del suelo es singular. La construcción de viviendas en las áreas más aptas para la explotación agropecuaria o en las montañas de pendiente alta o en suelos inadecuados, hace inaplazable planificar los asentamientos humanos en las ciudades.

En este sentido, hay avances importantes en los trabajos de Palacios (1982), Saldarriaga (1985), Planeación Metropolitana (1985), Area Metropolitana del Valle de Aburrá (1986) y Pérez y González (1987).

La presente investigación tiene por objeto primordial aportar metodologías que permitan ser usadas en el diseño de planes técnicos para el manejo de las áreas periféricas de la Metrópoli.

En lo referente al interés fundamental del estudio - usos del suelo periurbano - el modelo de Dinámica de Sistemas esbozado más adelante se aparta de los formulados en Forrester (1969), Alfred y Graham (1976) y Aracil (1983).

De esta manera, no se hace necesario la inclusión explícita de relaciones complejas de modelar, por ejemplo entre población, empleo, industria y comercio. Más bien, cuando es necesario se utilizan variables exógenas que bajo diversos escenarios permiten la evaluación de usos del suelo.

2. LA DINAMICA PERIURBANA

En la periferia de las ciudades los subsistemas urbanos coexisten con otros de índole suburbano o causirurales, y éstos en su conjunto interactúan con regiones vecinas urbanas y/o rurales.

La carestía de los suelos alrededor del centro de la Metrópoli ocasiona la demanda por las tierras de la periferia, lo cual produce la saturación (escasez) de éstas, obligando el desplazamiento de los límites urbanos a lugares todavía más apartados.

El estudio de mapas del crecimiento de la ciudad de Medellín permite la construcción de la Figura 1 que muestra las variables: Porcentaje de ocupación de un área de 16 kilómetros cuadrados alrededor del centro urbano (área interior); y porcentaje de ocupación, del total urbanizado, por fuera de esa área (área exterior). Resultados similares se obtuvieron haciendo variaciones en la extensión de la región central supuesta.

Las particularidades de la ciudad en cuestión se hacen visibles en la Figura 1. Hasta aproximadamente 1930 se presenta un crecimiento exponencial; posteriormente, debido al proceso de desarrollo urbano por fuera del Centro (creación de nuevos barrios) se produce un receso comparativo de la construcción en el corazón de la ciudad; finalmente, en los años sesenta y setenta se muestra una pequeña reactivación en el centro pero predominando niveles inferiores de la actividad edificadora.

De esta manera se exhiben algunos efectos ocasionados por el agotamiento de los suelos alrededor del centro de la ciudad sobre algunos procesos de la Dinámica Periurbana.

3. EL MODELO

En la construcción del modelo de la Dinámica Periurbana se consideraron con especial interés las variables de estado: Población, viviendas, área urbanizada, área agropecuaria y áreas vecinas; dándose un tratamiento desagregado a las zonas urbanas y rurales.

La representación de las diferentes componentes del sistema son conocidas en la ya literatura clásica de la Dinámica Urbana. No obstante, se introducen algunos elementos innovatorios en las relaciones de lo urbano con lo rural a través de la variable saturación territorial. En la Figura 2 se observan las estructuras causa-efecto de mayor importancia y se establece la conexión existente entre los diferentes usos del suelo. Puede verse una perfecta simetría entre las variables urbanas y rurales, con eje centrado en la variable Areas Vecinas (AV).

También, se sañala la relación de causalidad en los dos sentidos entre Población Urbana (PU) y Migración Urbana (MIGU), el efecto de la evolución poblacional sobre las Viviendas Urbanas requeridas (VU), la necesidad del sector de la construcción (VU) de nuevas Areas Urbanizadas (AUB), y la incidencia del crecimiento de los asentamientos humanos (AUB) y del Area Urbana disponible (AU) en la Saturación de los suelos urbanos (SATU).

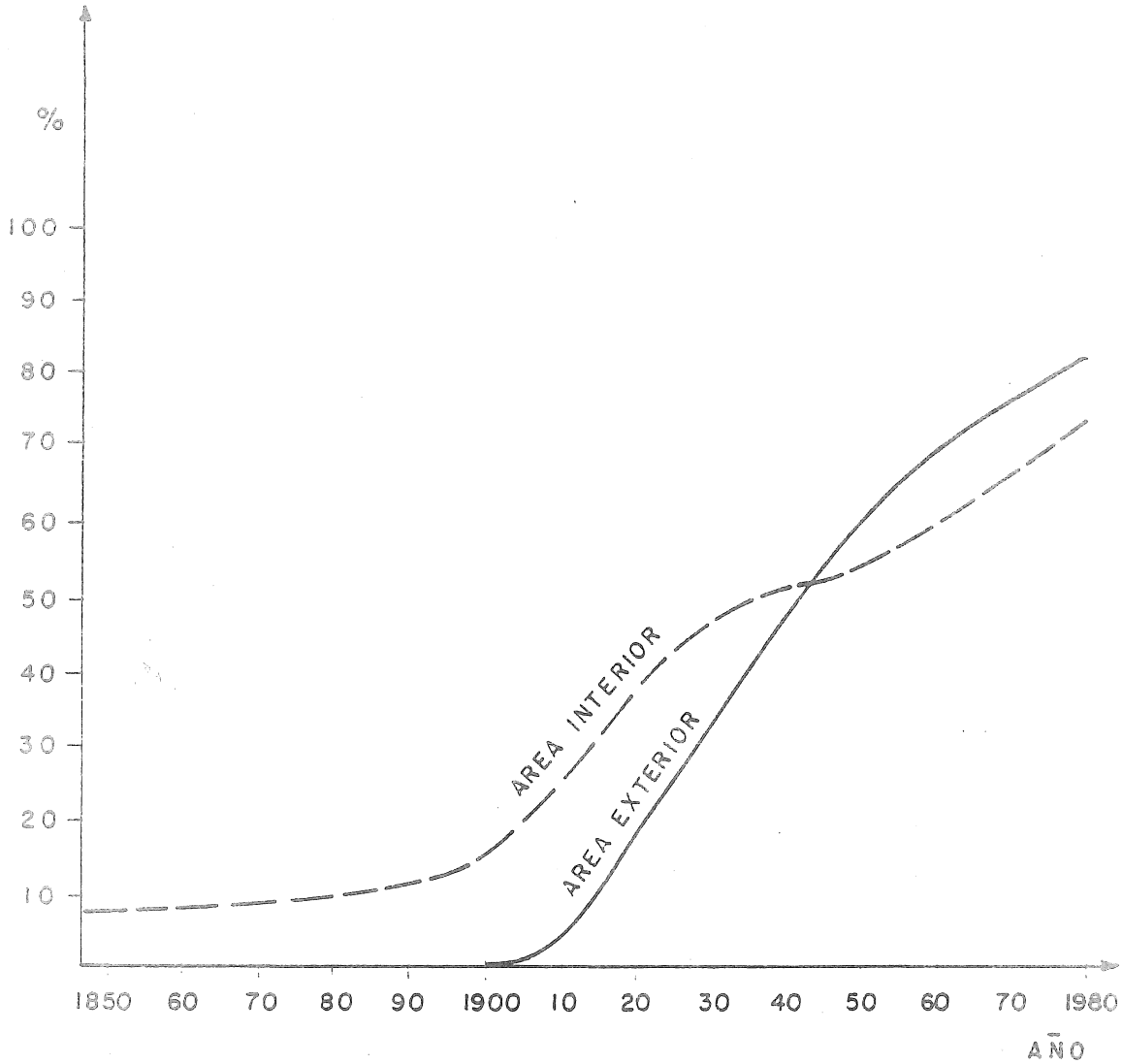

%
100
90
80
70
60
50
40
30
20
10
AREA INTERIOR
AREA EXTERIOR
1850
60
70
80
90
1900
10
20
30
40
50
60
70
1980
AÑO

FIGURA 1.

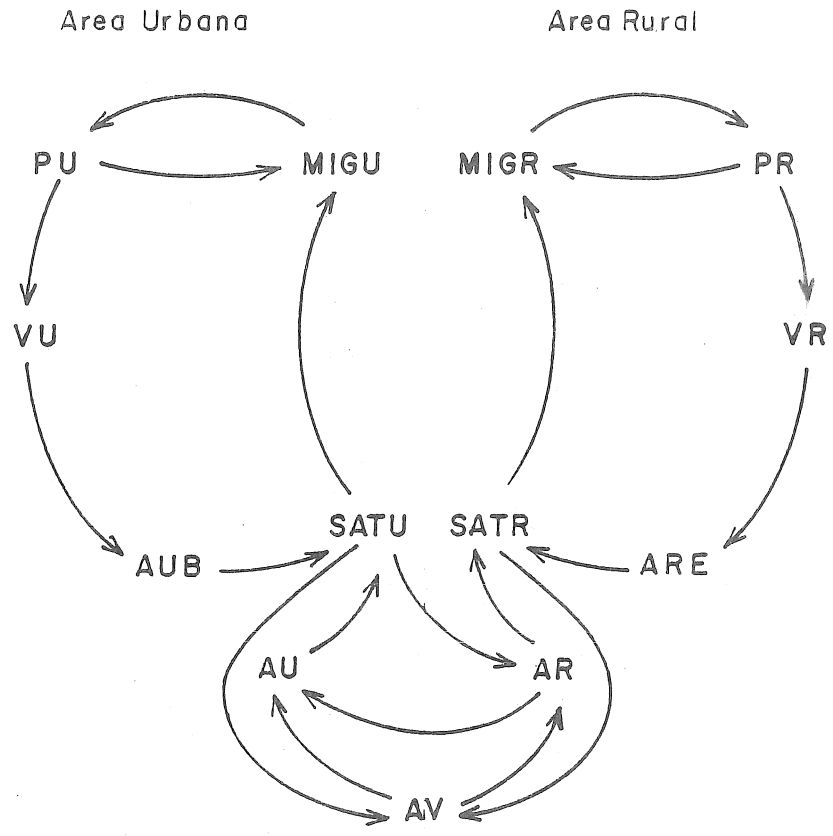

Area Urbana
Area Rural
PU
MIGU
MIGR
PR
VU
VR
SATU
SATR
AUB
ARE
AU
AR
AV

FIGURA 2.

Al mismo tiempo, el agotamiento de los terrenos urbanos (SATU) conlleva variaciones en los flujos migratorios (MIGU), en el Area Rural Metropolitana (AR) y en las Areas Vecinas (AV).

De manera similar, para el Area Rural la Figura 2 señala la conexión entre población (PR), migración (MIGR), vivienda (VR) y Area Rural Explotable (ARE).

La Saturación Rural (SATR), producida por el agotamiento del Area Rural disponible (AR) y del Area Rural Explotable (ARE), tiene efectos en los flujos migratorios rurales (MIGR) y en las Areas Vecinas (AV).

4. EVALUACION DEL MODELO

Evaluaciones preliminares del Modelo se realizaron en la región del Corazón del Area Metropolitana de Medellín.

La Tabla 1 muestra la adaptabilidad del modelo a una información histórica que presenta cambios drásticos, no completamente explicables, en sus tasas de crecimiento. Aún bajo estas condiciones, no se encuentran diferencias significativas entre los datos históricos y los arrojados por el Modelo para las variables población y vivienda.

	INFORMACION HISTORICA				RESULTADOS DEL MODELO	
	Población	%	Vivienda	%	Población	Vivienda
1981	3085		506		3085	506
1983	3368	9,2	612	20,9	3346	602
1985	3507	4,1	630	2,9	3546	614

TABLA 1. Area Urbana del Corazón.

En la Figura 3 se presenta un escenario de la evolución de las variables Población Urbana, Area Urbana, Area Rural Explotable y Area Vecina.

Para su construcción se utilizaron condiciones extremas en el crecimiento poblacional con el fin de destacar su efecto sobre la utilización de los espacios territoriales.

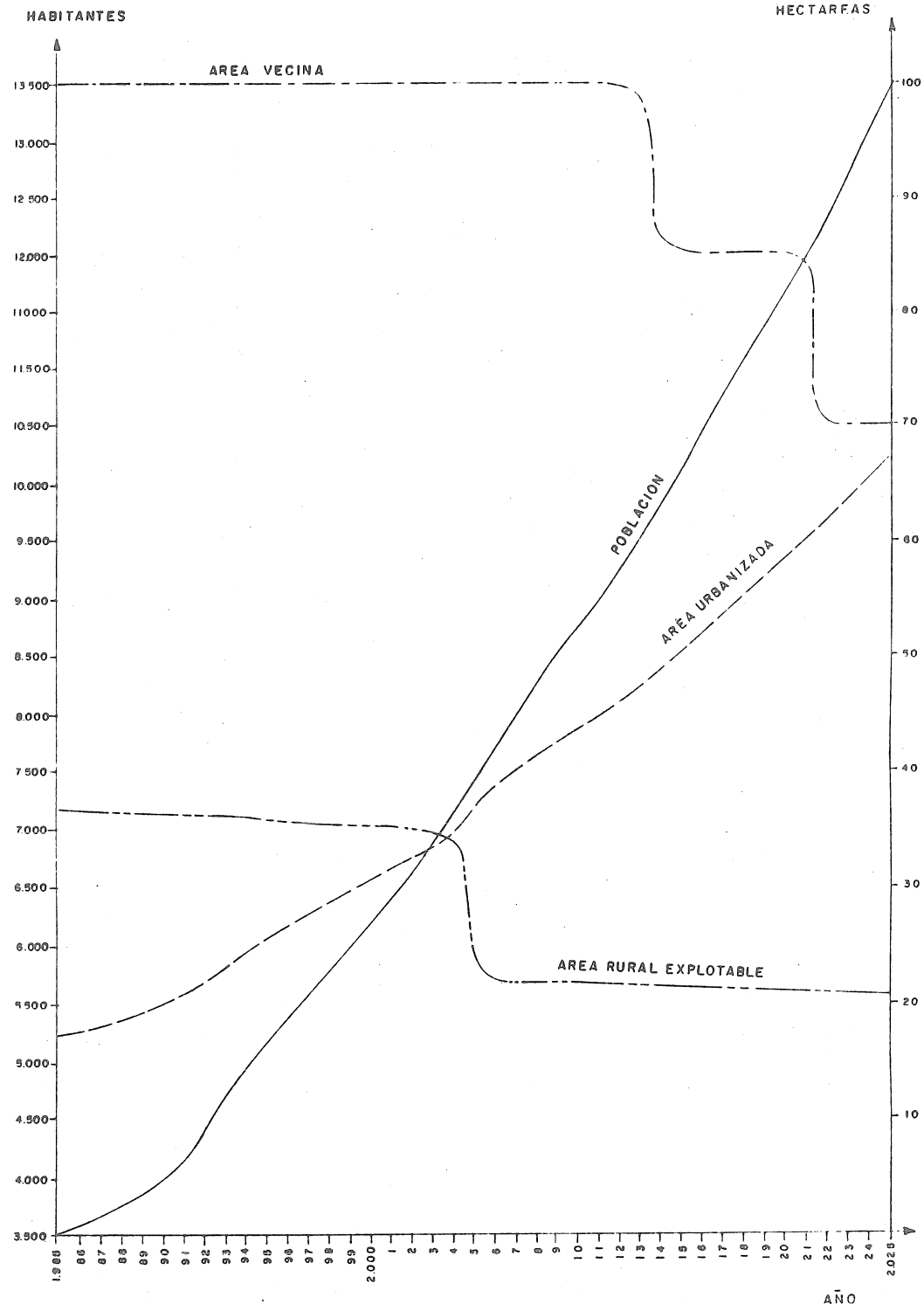

HABITANTES
HECTAREAS
AREA VECINA
POBLACION
AREA URBANIZADA
AREA RURAL EXPLOTABLE
AÑO

FIGURA 3.

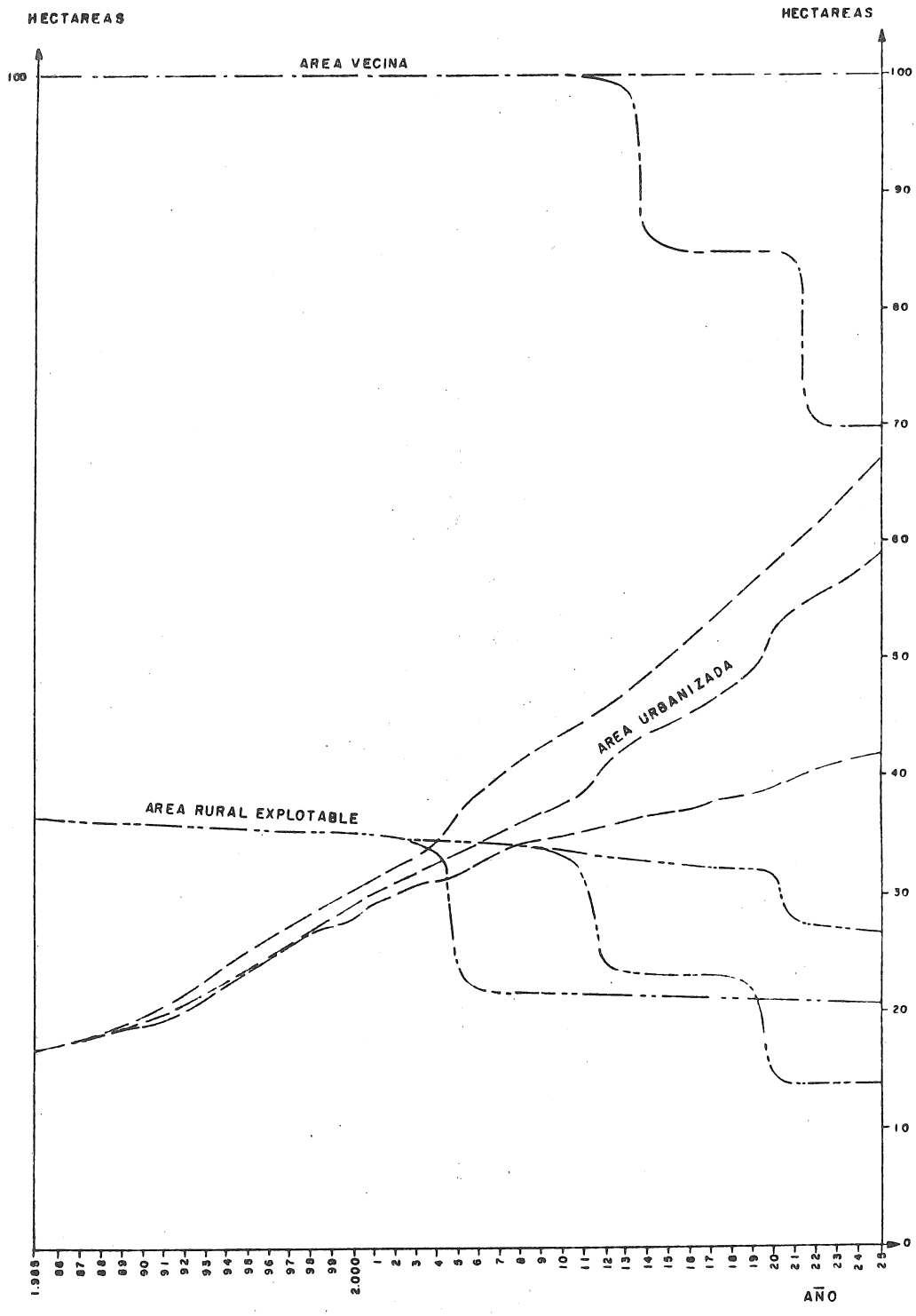

HECTAREAS
HECTAREAS
AREA VECINA
AREA URBANIZADA
AREA RURAL EXPLOTABLE
100
90
80
70
60
50
40
30
20
10
0
1.985
86
87
88
89
90
91
92
93
94
95
96
97
98
99
2.000
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
AÑO

FIGURA 4.

En la Figura 4 se contrastan otros escenarios con el de la Figura 3. Se consideran diversas situaciones como la disminución en las tasas de crecimiento poblacional y la variación en las políticas de urbanización de los espacios rurales. Los puntos de inflexión de los gráficos corresponden a efectos de saturación.

Los resultados preliminares son promisorios. La complementación del Modelo, actualmente en desarrollo, incluye la desagregación e incremento de variables en los sectores agropecuarios, educación, salud y recreación.

5. AGRADECIMIENTOS

El autor hace un reconocimiento especial a los ingenieros Alvaro González V. y María Elena Pérez S. por su importante contribución en la modelación y elaboración de programas, y al mismo tiempo agradece al Doctor Oscar Mesa sus interesantes sugerencias.

6. BIBLIOGRAFIA

AREA METROPOLITANA DEL VALLE DE ABURRA. 1986. Plan de Ordenamiento Territorial - Zona Norte.

ALFRED, L. and GRAHAM, A. 1976. Introduction to Urban Dynamics. Wright - Allen Press, Mass.

ARACIL, J. 1983. Introducción a la Dinámica de Sistemas. Alianza Editorial.

FORRESTER, J. 1969. Urban Dynamics. Mit Press.

PALACIOS, A. 1982. La Tierra Rural en el Desarrollo Metropolitano. El Area Agropolitana. Anotaciones sobre Planeación No.7. Universidad Nacional, Medellín.

PEREZ, M.E. y GONZALEZ, A. 1987. Simulación de Sistemas Dinámicos en la Utilización de los Suelos Periurbanos de Medellín. Trabajo de Grado. Facultad de Minas, Universidad Nacional, Medellín.

PLANEACION METROPOLITANA DE MEDELLIN. 1985. Plan de Desarrollo Metropolitano del Valle de Aburrá.

SALDARRIAGA, A. 1985. El espacio Rural en el Area Metropolitana de Medellín. Estudio de Caso del Corazón. Tesis de Magister. Postgrado en Planeación Urbana. Universidad Nacional, Medellín.

SISTEMA PARA GENERACION EN MICROCOMPUTADORES DE MATRICES Y REPORTES DE PROGRAMACION LINEAL

B. Brito; R. Calabrese; E. Correa; B. Lorenzo
Universidad Central de Venezuela
Facultad de Ciencias
Caracas - Venezuela

RESUMEN

En este trabajo se presenta el diseño y desarrollo de un sistema para generadores de matriz e informes (GIM) orientado a problemas de programación lineal de mediana escala en microcomputadoras, que facilita la captura de datos, la generación de la matriz correspondiente y la interpretación de datos y resultados.

El GIM fue diseñado en forma modular, de fácil comprensión y utilización por parte del usuario, además puede ser usado tanto en forma Interactiva como en Lote, basándose su desarrollo en los paquetes comerciales GAMMA 04 y OMNI implantados éstos para aplicaciones en Mainframe.

INTRODUCCION

El microcomputador ha pasado a ser un producto que día a día es más accesible al público consumidor. Esto a su vez ha generado la necesidad de producir una serie de aplicaciones comerciales (paquetes) que deben satisfacer las necesidades de éste público masivo; estos paquetes al igual que los micros se caracterizan en su mayoría por su fácil uso y manejo. Tenemos así que las dos grandes restricciones, como lo eran el capital y el personal capacitado para el manejo y uso de microcomputadores está desapareciendo.

En el área de Investigación de Operaciones nos encontramos con paquetes estadísticos que originalmente fueron creados para computadores (mainframe) como son el SPSS y el SIMSCRIPT; hay también en el mercado una serie de paquetes para hacer pronósticos como el MICRO-BJ, XTRAPOLATOR y NUMERICS, entre otros; inclusive con productos especializados en ciertos tópicos como generación y pruebas de números aleatorios (vease Sabbagt 10).

Por supuesto no podemos dejar de mencionar en éste trabajo los paquetes desarrollados para resolver Problemas de Programación Lineal (PPL). En las publicaciones y revistas del área nos encontramos con avisos promoviendo productos como LP88, LP/FORTRAN, MS/LP, MATHEMATICAL PROGRAMMING PACKAGE II, etc. Todos ellos ofrecen eficiencia y capacidad para resolver Problemas de Programación Lineal de cierta magnitud, como son problemas de 100x100 y 300x600.

La oferta de estos productos de Programación Lineal (PL) se justifica en parte por la tendencia actual en la industria de usar técnica de PL a fin de resolver Problemas de corte técnico, científico y gerencial como son los problemas de transporte, distribución obtención de mezclas óptimas en procesos productivos, finanzas e inversiones y asignación de recursos, entre otros. Ahora bien, la resolución de Problemas de Programación Lineal de gran escala como puede ser la determinación de la mezcla óptima a producir en una refinería, implica la generación de una matriz de coeficientes tecnológicos de gran magnitud. Esta matriz debe ser llenada tomando en cuenta la información numérica contenida en tablas, utilizadas para representar restricciones de caracter técnico, económico y político, así como también, relaciones estructurales y funcionales del problema. Además, una vez resuelto el problema debido a la necesidad de interpretar la información resultantes de la solución, que se encuentra en forma un tanto engorrosa para el usuario, se requiere de una serie de informes que permita un análisis de los resultados, facilitando así, la toma de decisiones.

Para aliviar la carga en el manejo de tanta información, surgen entonces en el mercado, paquetes comerciales conocidos como "GENERADORES DE MATRICES E INFORMES" para Problemas de Programación Lineal. Estos productos vienen a facilitar en alto grado no solo la captura de datos y generación de la matriz correspondiente, sino que además facilitan la interpretación de datos y re-

sultados. Entre los paquetes más conocidos se encuentran: el GAMMA 04, el OMNI/PDS 370 09 y el GENERADOR DE INFORMES MPSX370 08.

Sin embargo, estos productos fueron desarrollados para computadores grandes como son los de las series 370 y 4300 de IBM y la serie 6000 de BURROUGHS. Esto nos llevó a pensar en el desarrollo de un Sistema para Generadores de Matriz y Generadores de Informes para Problemas de Programación Lineal de mediana escala en microcomputadores. Estos generadores serán dos módulos que, junto con un tercero de búsqueda de la solución óptima conformarán un software destinado a resolver Problemas de Programación Lineal en micros.

La programación del paquete se realizó en forma modular y además tomando en cuenta el tipo de información que se ha de manejar se decidió usar el lenguaje "TURBO PASCAL" y el sistema fué implantado en una EPSON QX-10 (actualmente se implanta en PC y XT de IBM).

1.- DESCRIPCION GENERAL DEL SISTEMA GIM:

El sistema Generador de Matriz/Informe GIM fué desarrollado en forma modular, de fácil comprensión y utilización por parte de los usuarios. Este software fué diseñado para microcomputadores y puede ser usado tanto en forma interactiva como en lote. Consta de dos módulos: el primero para la generación de la matriz asociada a un problema de PL y el segundo módulo para la generación de informes asociados a la solución de Problemas de Programación Lineal.

A continuación se presenta el diagrama general del sistema GIM y la descripción de los módulos que lo conforman:

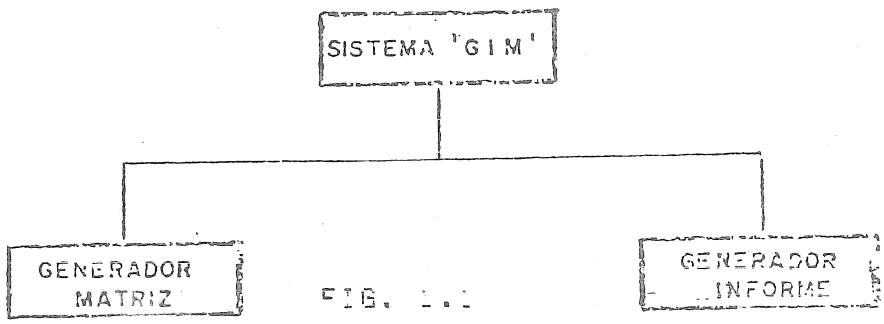


FIG. 1.1

FIG. 1.1

MODULO PARA LA GENERACION DE LA MATRIZ (MGM):

Este módulo es el encargado de generar la matriz a partir de los datos (tablas, listas, restricciones y variables de decisión) necesarios para la generación de la matriz bajo el formato de entrada MPS, de una manera natural y sencilla, por medio de un número mínimo de reglas y formatos.

En la fig. 1.2 se muestra el diagrama general de éste módulo:

FIG. 1.2

FIG. 1.2

MODULO PARA LA GENERACION DEL INFORME (MGI):

Este módulo es el encargado de generar los informes a partir del archivo solución (Producto de la resolución de un problema de PL) y de un conjunto sencillo de comandos.

En la fig. 1.3 se ilustra el diagrama general del MGI:

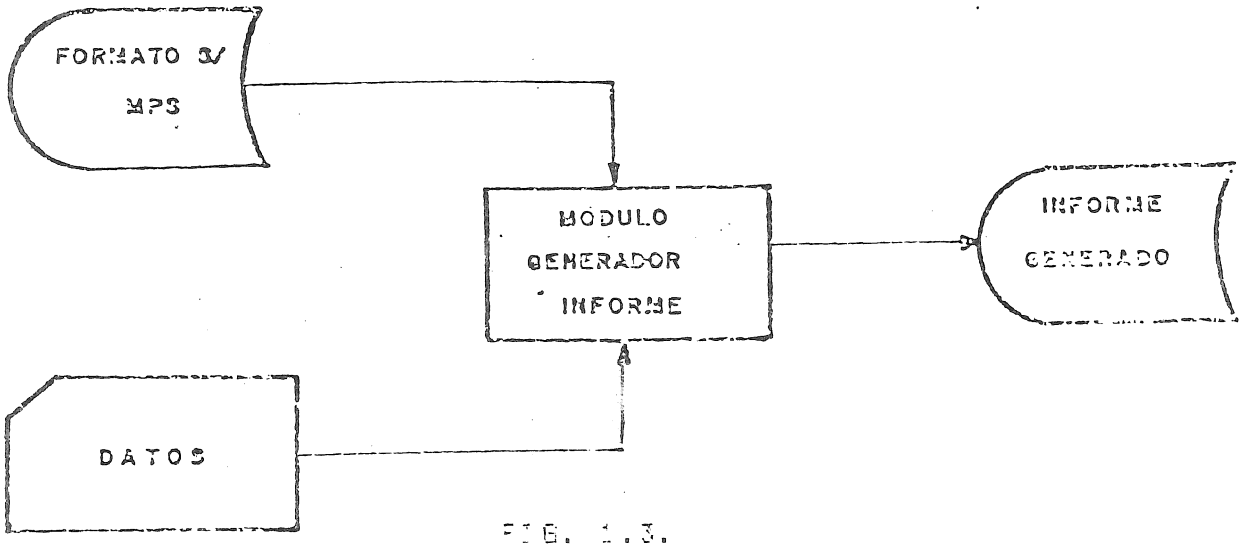


FIG. 1.3.

DIAGRAMA GENERAL DEL MODULO GENERADOR DE INFORME

Los datos de la fig. 1.2 y 1.3 son almacenados en archivos de acceso random. Los componentes de estos archivos son del tipo estático, descartando los tipos dinámicos, dado que la actualización de los mismos requieren de un proceso engorroso para mantener los datos existentes en la estructura y además, estas no permiten acceso directo a sus componentes.

2.- SISTEMA INTERACTIVO

El sistema interactivo GIM establece comunicación directa con el usuario por medio de un esquema de MENU. La principal razón de ello, es la necesidad de un producto amigable al analista/usuario, que los guien para realizar la captura de los datos de una manera totalmente transparente, sin que estos deban conocer la secuencia exacta de los pasos a ejecutar.

La siguiente figura muestra la configuración general del sistema GIM Interactivo.

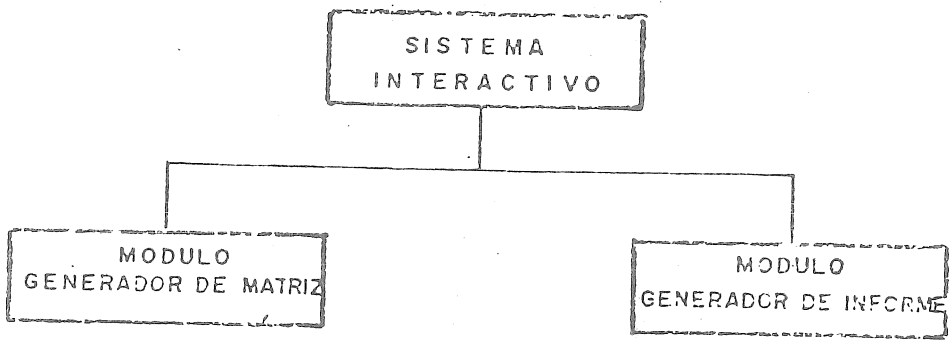


FIG. 1.4

FIG. 1.4

2.1- MODULO GENERADOR DE MATRIZ (MGM)-I

El módulo Generador de Matriz fué diseñado para el desarrollo, organización, generación y mantenimiento de matrices asociadas a Problemas de Programación Lineal. El generador agrupa los datos basándose en el uso de TABLAS para la colección y almacenamiento de los datos asociados al problema agrupados en estas tablas según su naturaleza y en el uso de LISTAS para el almacenamiento de información de cadenas de caracteres en forma mnemónica. Por medio de ésta clasificación, el MGM-I posee todos los datos agrupados en su forma lógica requeridos en la formulación de un problema de PL, los cuales serán accesados y manipulados en la generación de la matriz.

El MGM-I cumple las siguientes funciones:

a) Captura y validación de los datos asociados a un PPL, para lograr la generación de la matriz con datos válidos del problema.

b) Modificar y eliminar datos del problema, para brindarle al analista la flexibilidad y facilidad de corregir los datos numéricos y/o hacer cambios estructurales en un problema previamente definido.

c) Generar la matriz bajo el formato de entrada MPS, dado a que dicho formato es standard en la mayoría de los paquetes comerciales.

El MGM-I mantiene comunicación con el usuario por medio de un esquema de menú y para realizar sus funciones está subdividido en 5 submódulos que se describirán posteriormente, ellos son:

A.- Introducir Datos-Analista.

B.- Modificar Datos-Analista.

C.- Eliminar Datos-Analista.

D.- Introducir Datos-Usuario.

E.- Generación del formato de entrada MPS.

En la fig. 2.1 se muestra el diagrama general del Módulo Generador de Matriz Interactivo.

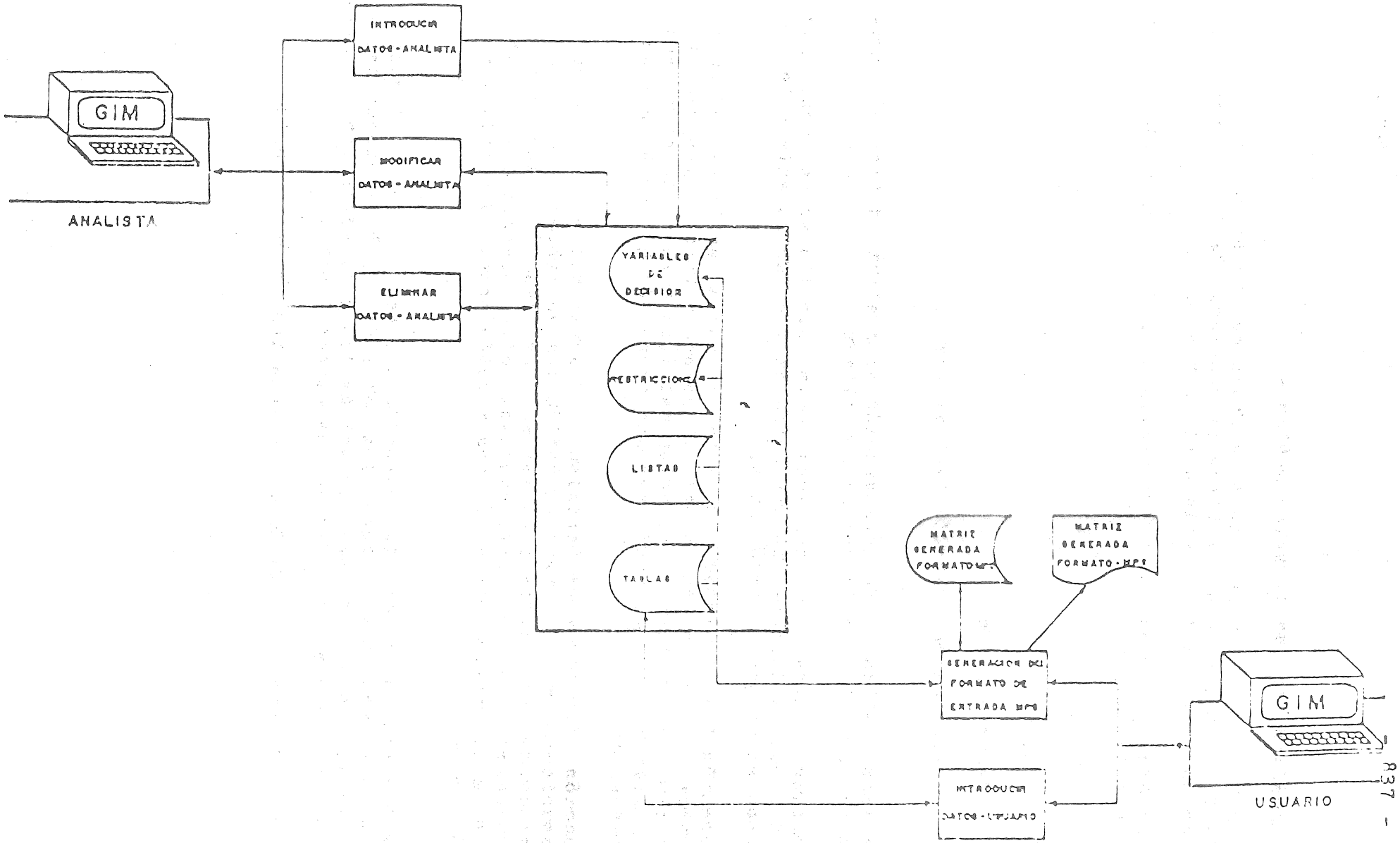


FIG. 2.1 DIAGRAMA GENERAL DEL MODULO GENERADOR DE MATRIZ - INTERACTIVO.

A. Submódulo Introducir Datos Análisis

Este submódulo se encarga de la captura de los datos requeridos en la formulación de un Problema de Programación Lineal para la generación de la Matriz asociada al problema.

El submódulo Introducir Datos-Analista está subdividido en 4 secciones de datos:

a) Tablas.

b) Listas.

c) Restricciones.

d) Variables de Decisión (actividades).

a) **Sección Tablas:** En esta sección el analista define el conjunto de tablas (o procesos) a utilizar.

b) **Sección Listas:** El analista define las listas ya sea, por medio de la información contenida en las Tablas existentes u otros elementos o componentes.

c) **Sección Restricciones:** En esta sección el analista define el nombre de la función objetivo, nombre y tipo de las restricciones del problema y el valor del lado derecho de cada restricción, el cual puede ser una expresión que será evaluada posteriormente.

d) **Sección Variables de Decisión:** Una vez definida la sección Restricciones el analista define las variables de decisión indicando su nombre y cuales son las restricciones con su expresión, la cual será evaluada posteriormente y este valor indica el coeficiente en la restricción asociado a cada variable de decisión.

El submódulo Introducir Datos-Analista a medida que interactúa con el analista para capturar los datos de las 4 secciones definidas previamente se encarga de validarlos y si estos son correctos se almacenan en los archivos correspondientes.

La fig. 2.2 muestra el flujo de información de éste submódulo

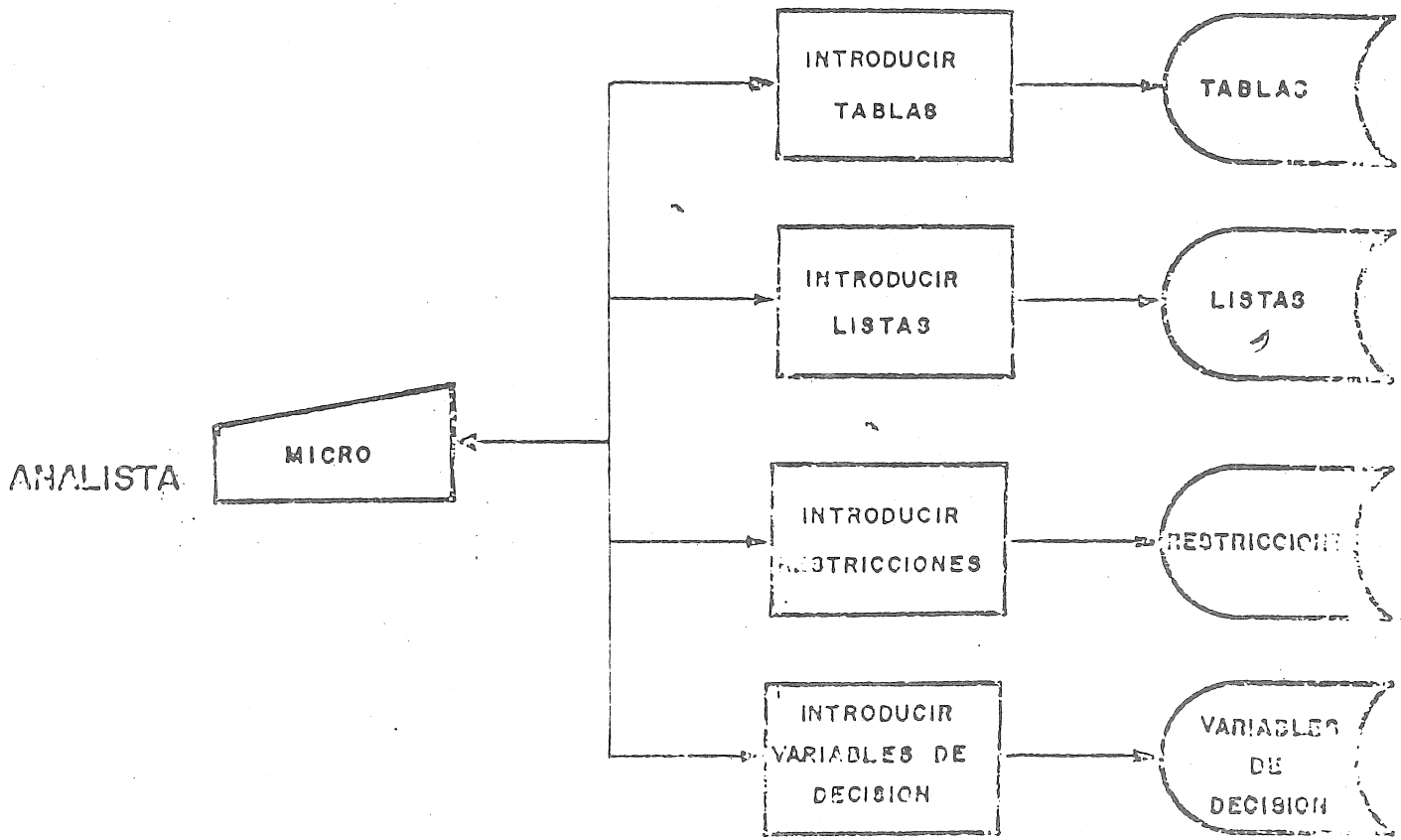


FIG. 2.2

SUBMODULO: INTRODUCIR DATOS - ANALISTA

B.- **Submódulo Modificador Datos-Analista**

Este submódulo se encarga de llevar a cabo las modificaciones que se requieran de los datos, por medio de un esquema de menú para la captura de la información a modificar, en los archivos de datos; abarcando el punto de inclusión de nuevos datos y efectuando la validación correspondiente. Este proceso se inicializa con la presentación de un menú Principal donde se contemplan todas las posibles alternativas de modificación sobre los datos, como son:

- Modificar Tablas
- Modificar Listas
- Modificar Restricciones
- Modificar Variables de Decisión

Donde para cada una de estas alternativas se realiza la búsqueda de la información que se desea modificar y se efectúa la sustitución.

El resultado de la ejecución de este submódulo son los archivos de datos ya actualizados.

En la fig. 2.3 se muestra el diagrama de éste submódulo.

C.- **Submódulo Eliminar Datos-Analista:**

Este submódulo se encarga de efectuar todas las eliminaciones que se requieran de los archivos de datos, la captura de la información a eliminar se realiza por medio de un esquema de menú. El usuario está en capacidad de escoger sobre las posibles alternativas de eliminación, hecho esto, se efectúa la búsqueda de la información a eliminar y se realiza la actualización de los archivos de datos.

En la fig. 2.4 se muestra el diagrama del submódulo Eliminar Datos-Analista.

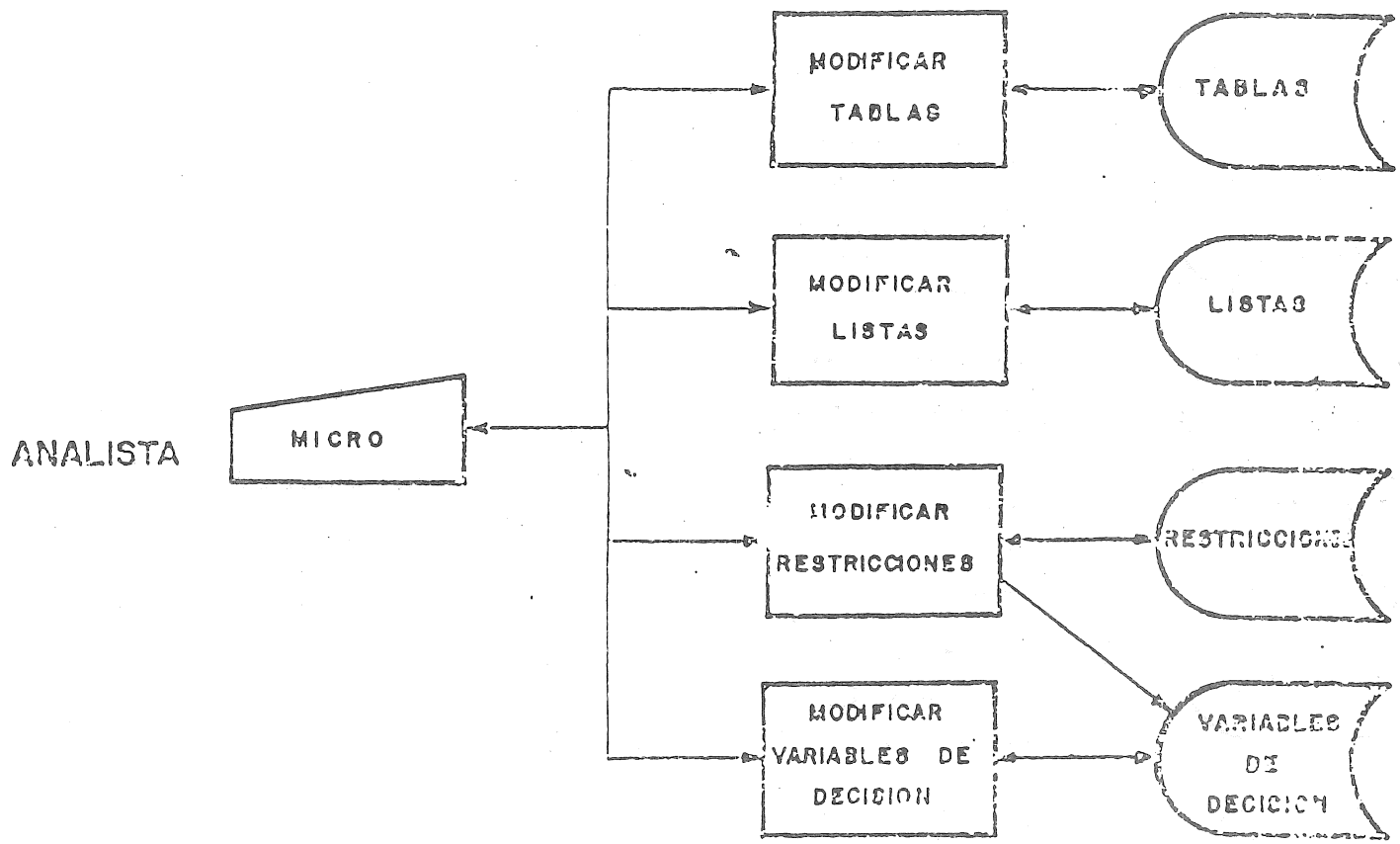


FIG. 2.3

SUBMODULO: MODIFICAR DATOS - ANALISTA

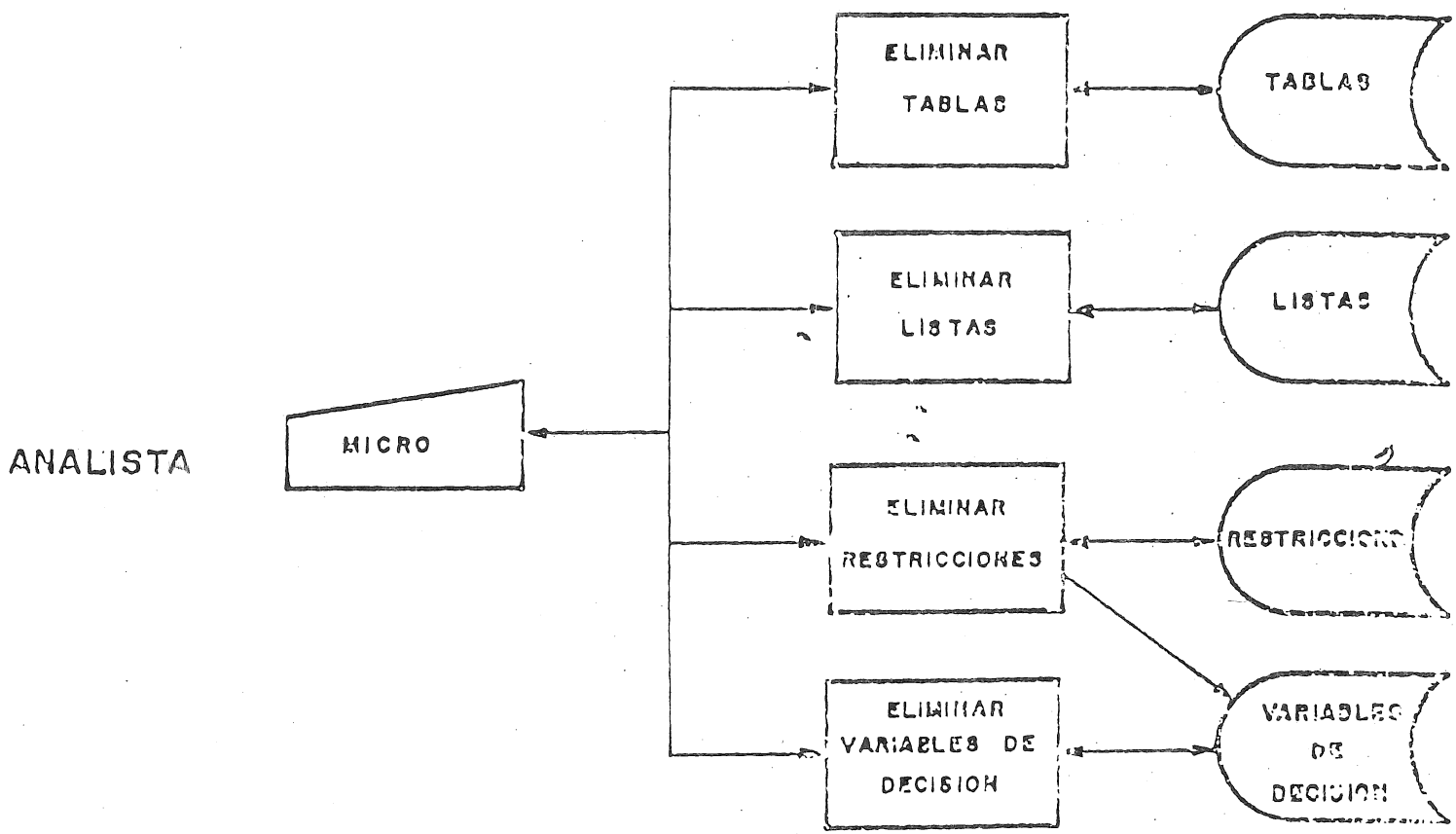


FIG. 2.4

SUBMODULO: ELIMINAR DATOS - ANALISTA

D.- **Submódulo Introducir Datos-Usuarios:**

Este submódulo se encarga de la captura y validación de la información numérica, de los distintos componentes pertenecientes a los procesos o tablas del Problema de Programación Lineal a resolver. Esta información queda registrada en el archivo de Tablas.

El usuario interactúa con este submódulo si el analista previamente ha realizado la formulación del problema.

La fig. 2.5 muestra el diagrama de este submódulo.

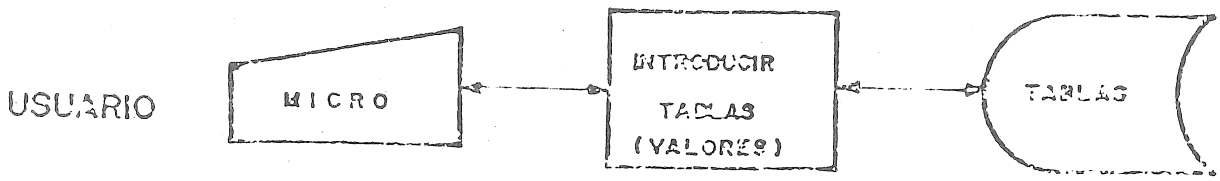


Fig. 2.5

FIG. 2.5

E.- **Submódulo Generación de Formato de entrada MPS:**

Este submódulo se encarga de la evaluación de las expresiones definidas en las secciones Restricciones y Variables de Decisión.

El resultado de este submódulo es un archivo con la matriz generada bajo el formato de entrada MPS.

La fig. 2.6 ilustra el diagrama de flujo de este submódulo.

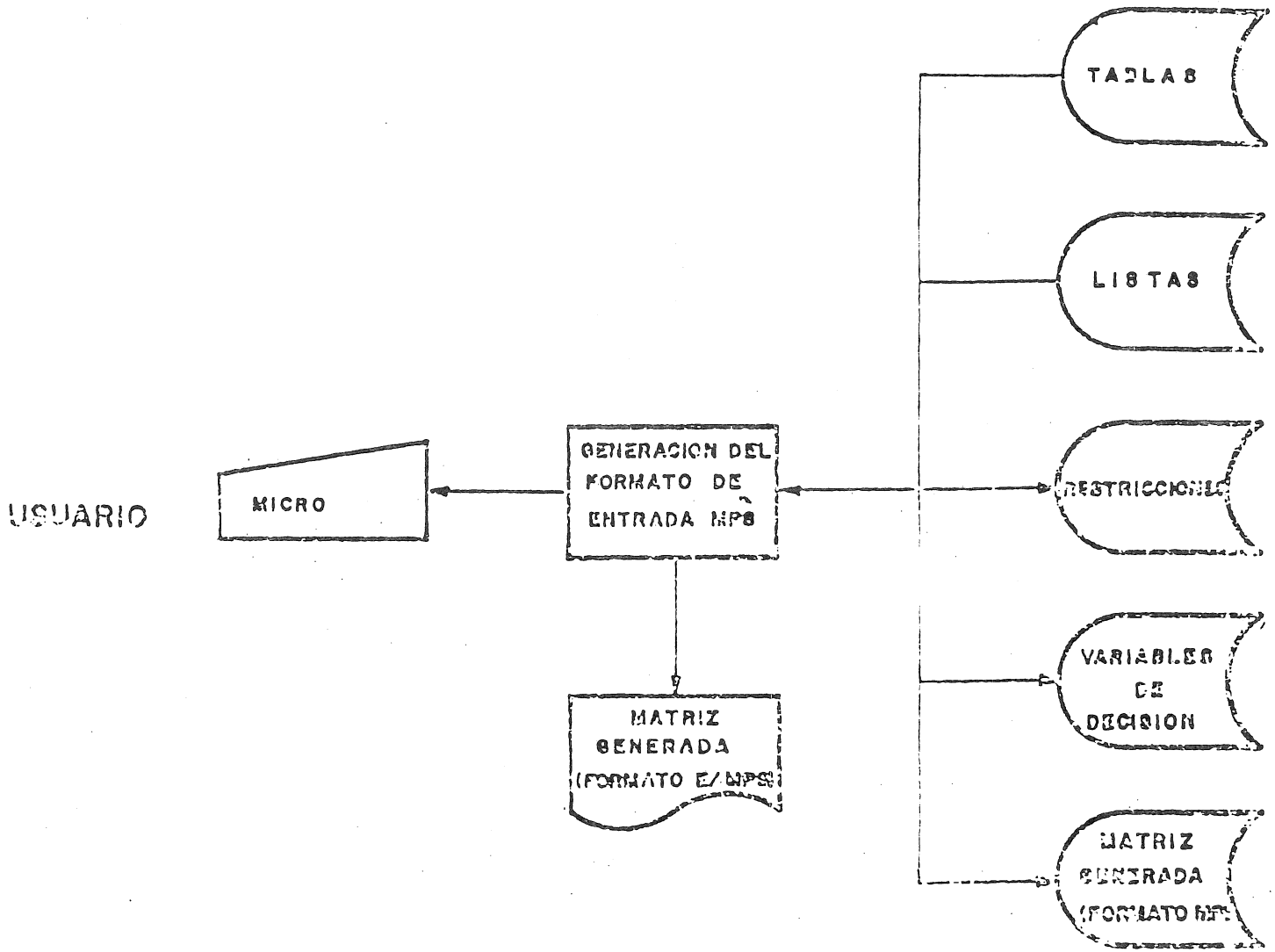


FIG. 2.6

SUBMODULO: GENERACION DEL FORMATO DE ENTRADA M.P.S.

2.2- MODULO GENERADOR DE INFORME (MGI-I9):

Con este software el usuario podrá diseñar informes, tomando los aspectos que sean de su mayor interés del archivo que contiene toda la información de la solución del problema (archivo con el formato de salida MPS). El usuario dispondrá de un conjunto sencillo de comandos para el acceso de la información y diseño del informe.

El MGI-I cumple con las siguientes funciones:

a) Captura y validación de los datos asociados al diseño de un informe, para lograr la generación de éste.

b) Modificar y eliminar datos del informe para brindarle al analista la flexibilidad y facilidad de corregir y/o hacer cambios en un informe previamente definido.

c) Generar un informe, según las especificaciones del analista.

El MGI-I para realizar sus funciones está subdividido en 4 módulos los cuales se describirán posteriormente.

A.- Introducir Datos-Informe.

B.- Modificar Datos-Informe.

C.- Eliminar Datos-Informea.

D.- Generador de Informe.

En la fig. 2.7 se muestra el flujo de información del Módulo Generador de Informe Interactivo.

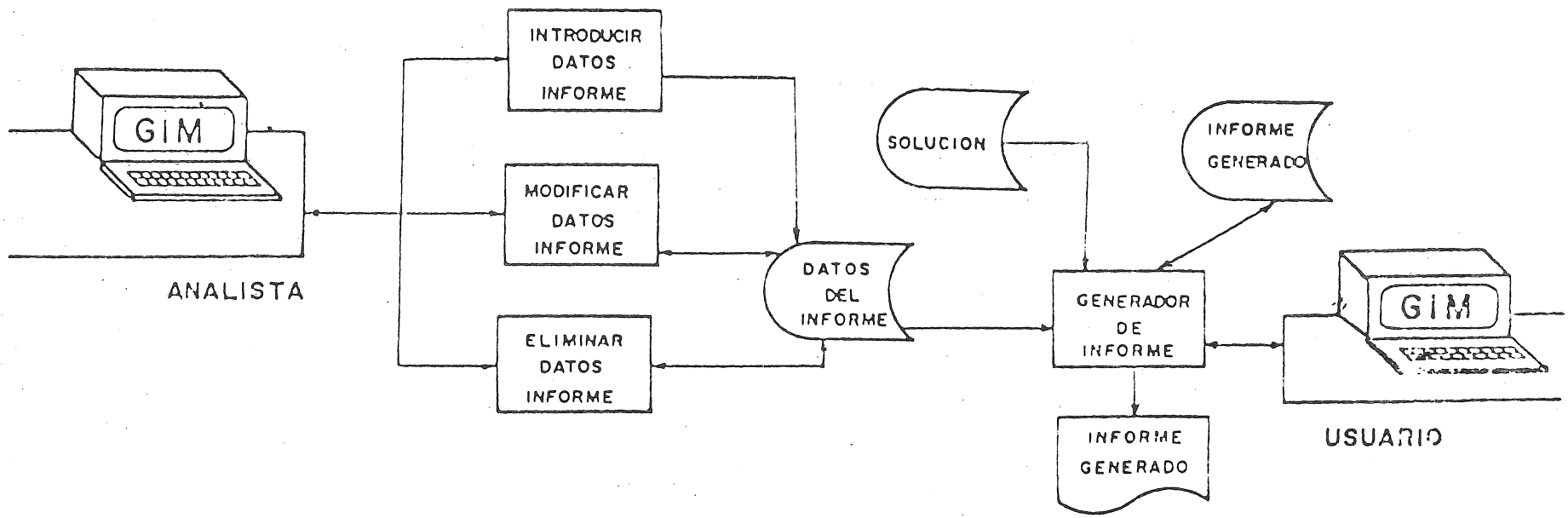


FIG. 2.7

MODULO GENERADOR DE INFORME - INTERACTIVO.

A.- Submódulo Introducir Datos-Informe

El objetivo primordial de este submódulo en la captura en forma interactiva de los datos para el diseño del informe.

Este submódulo crea el archivo que contendrá todos los datos suministrados por el analista quien puede definir rutinas, títulos, formatos, realizar cálculos y operaciones aritméticas con los valores de la solución, salto de páginas, etc.

B.- Submódulo Modificar Datos-Informe

Mediante este submódulo, el analista interactúa con el sistema para realizar las diferentes funciones de manipulación sobre los datos, que pueda necesitar en cualquier momento. Estas funciones son:

- Listar la información ya suministrada.
- Agregar datos al informe.
- Modificar un informe ya formulado.

La ejecución de este submódulo produce la actualización del archivo que contiene toda la información para la formulación del informe.

C.- Submódulo Eliminar Datos-Informe

Por medio de este submódulo, el analista interactúa con el sistema para eliminar datos del informe, facilitándole esta actividad con la opción de listar toda la información que posea el informe.

La ejecución de este submódulo produce la actualización del informe formulado.

D.- Submódulo Generador de Informe

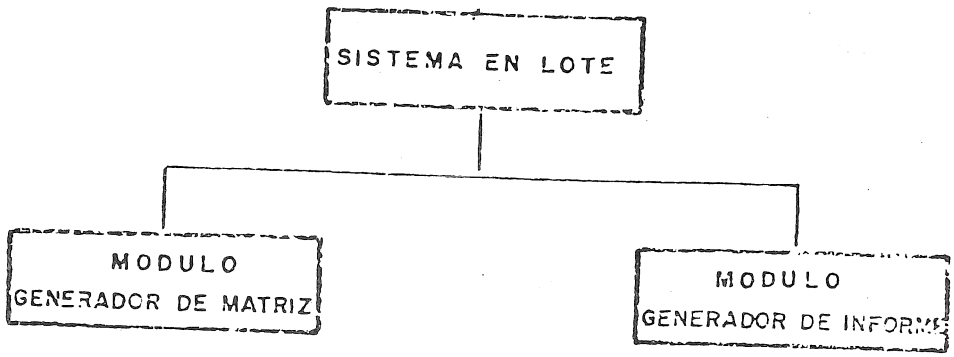
Este submódulo, recibe como entrada: el archivo con la información (válida) suministrada por el analista para la generación del informe y los archivos de solución en el formato de salida MPS.

Con esta información se genera el informe deseado por el usuario. Dicho informe se almacenará en un archivo y el usuario decidirá si desea listarlo por impresora y/o pantalla.

3.- SISTEMA EN LOTE

El sistema en lote, surge para aquellos analistas que prefieran definir los archivos de datos con información de las tablas, listas, restricciones y variables de decisión para la generación de la matriz e informes de la solución, sin necesidad de interactuar con el sistema.

El analista creará los archivos de datos del tipo txt (texto) a través de cualquier editor, por ejemplo: el editor de Turbo Pascal [06]. Utilizando los mismos formatos y las mismas secciones usados en el interactivo para definir el generador de matriz y el generador de informes. Si el analista desea hacer modificaciones sobre los datos, solo tiene que accesar estos archivos y actualizarlos con la información que desee, utilizando también el editor seleccionado.



CONCLUSIONES

El Generador de Matriz e Informes (GIM) en micros, que se presentó en este trabajo, reseulta ser un sistema que provee facilidades en la manipulación de datos para la generación de la matriz e informes de la solución del problema de Programación Lineal a resolver. El método de definición del modelo, la manipulación y mantenimiento de los datos, ofrecido por este sistema reduce el tiempo de desarrollo del modelo, así como también permite que las variables y restricciones pueden ser descritas en términos generales con un número pequeño de instrucciones y los datos que han sido modificados pueden ser rápidamente introducidos.

Por otra parte, el lenguaje que permite una modelación rápida es fácil de aprender; los informes de la solución son fácilmente formateados y descritos en términos generales independientes de los datos actuales y se generan informes completos o parciales.

Finalmente, por todos los aspectos mencionados anteriormente, a su sencillez y la comunicación sistema usuario "amigable" es de fácil uso, en comparación a muchos programas que corren en estructura "mainframe".

El GIM puede seguir desarrollándose específicamente en:

- Lograr que los informes puedan ser generados durante cualquier paso del desarrollo de la generación de la matriz (por ejemplo: Informes que puedan ser generados durante la entrada de datos o fases en la generación de la matriz).

- Obtener informes desde varios archivos de solución.

- Permitir usar listats para la definición de los nombres de las tablas.

- Incrementar el número de operadores condicionales a ser utilizados en la evaluación de expresiones aritméticas usadas en la definición de los coeficientes de la matriz y el vector de recursos.

PROGRAMA PARA LA SOLUCION DE MODELOS DE COLAS

Mauricio Zúñiga Castañeda
Departamento de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad de Los Andes
Apartado Aéreo 4976, Bogotá

Fernando Palacios Gómez
Departamento de Ingeniería Industrial
Facultad de Ingeniería
Universidad de Los Andes
Apartado Aéreo 4976, Bogotá
Agosto de 1987

RESUMEN

Los modelos de colas han sido utilizados exitósamente para analizar el desempeño de sistemas de computación y de comunicaciones, como función de su topología, de las distribuciones y disciplinas de servicio de los dispositivos, y del volumen y variedad de su carga. Por esta razón, para llevar a cabo el diseño y evaluación de ese tipo de sistemas, es útil una herramienta de solución de modelos de colas.

Este documento presenta un programa desarrollado para solucionar modelos generales de redes de colas, el cual utiliza la técnica de agregación, el análisis del valor medio y los resultados para las colas M/M/m y M/G/∞. Estas técnicas dan resultados exactos cuando el modelo es de tipo BCMP. Para modelos generales dan soluciones que pueden ser aproximadas, con un error generalmente pequeño.

El programa puede ser utilizado en los microcomputadores de la familia IBM y sus compatibles. Además el programa es fácilmente transportable pues ha sido desarrollado en el lenguaje C estándar.

Este documento también muestra un ejemplo de solución de modelos de sistemas de computación utilizando el programa presentado, estableciendo que el diseño y evaluación de este tipo de sistemas puede facilitarse con el uso del programa.

1. INTRODUCCION

Los modelos de colas han sido utilizados exitósamente para analizar sistemas de computación y de comunicaciones (Baske72), (Buzen78), (Klein76), (Kobay78), (Laven83), (Reise76) (Sauer80), (Sauer81) y (Wong78). El análisis con modelos de colas permite establecer el desempeño del sistema como función de su topología, de las distribuciones y disciplinas de servicio de los dispositivos, y del volumen y variedad de su carga.

Por esta razón, para llevar a cabo el diseño y evaluación de sistemas de gran complejidad, es altamente útil una herramienta de análisis de modelos de colas, la cual permite establecer la conveniencia de configuraciones arbitrarias y la eficiencia del sistema en función de sus características.

Este documento presenta un programa desarrollado para el análisis de sistemas de computación y de comunicaciones basados en modelos de colas. Las características de los modelos que soluciona son las siguientes:

[] Pueden ser modelos abiertos y cerrados

[] Para los modelos abiertos la tasa de llegada de clientes es Poisson e independiente del estado del modelo

[] Puede haber cualquier número de centros de servicio

[] La disciplina de los centros de servicio es FCFS, PS, IS ó LCFSPR

[] Los centros de servicio pueden estar arbitrariamente interconectados por la trayectoria de los saltos de los clientes. Las probabilidades de salto entre los centros son independientes del estado del modelo

[] En cada centro de servicio puede haber un número arbitrario de clases de clientes. Cada clase puede tener una probabilidad de salto particular a los diferentes centros de servicio

[] Los tiempos de servicio de los centros siguen una distribución cuya transformada de Laplace es una función racional propia. Entre otras, la exponencial, hiperexponencial e hipoexponencial son distribuciones de este tipo

[] Las tasas de servicio pueden ser función de la longitud de la cola en el centro, sin distinguir su composición

[] Puede haber subredes con población limitada

El programa ofrece una gran libertad en la definición del modelo para incluir los detalles relevantes del sistema bajo estudio.

El programa permite la definición del modelo y la consulta de la solución en forma interactiva. Para esto utiliza un sistema de menús que guían al usuario a lo largo de estas actividades.

El programa soluciona los modelos de colas utilizando principalmente la técnica de agregación y el análisis del valor medio (Chand74), (Chand78), (Reise80), (Sauer81) y (Sauer83). Estas técnicas dan resultados exactos para el modelo cuando éste es de tipo BCMP (Baske75). Para modelos generales pueden dar soluciones aproximadas con un error generalmente pequeño (Sauer81).

El programa puede ser utilizado en cualquier computador con microprocesador 8086/8088 y sistema operacional MS-DOS. El computador debe tener por lo menos 128K y una unidad de diskette. Esto incluye los microcomputadores de la familia IBM y sus compatibles. El programa es fácilmente transportable a otros equipos pues ha sido desarrollado en lenguaje C, utilizando el conjunto de instrucciones y funciones descritas en el texto de Kernighan y Ritchie (Kerni78).

2. MODELOS DE SISTEMAS DE COMPUTACION

Un sistema de computación consta de varias clases de recursos como son el procesador central, los procesadores de E/S, la memoria, los procesadores de comunicación y las líneas de comunicación, entre otros. Los trabajos entran al sistema y compiten por los recursos haciendo colas de espera para ser atendidos. Cuando termina el proceso de servicio, el trabajo puede entrar nuevamente a una cola de espera para ser atendido por otro recurso. Por esta razón, un sistema de computación puede ser modelado como una red de centros de servicio interconectados.

Hay cuatro disciplinas de servicio importantes para el modelaje de los recursos del sistema: FCFS ("First Come First Served"), PS ("Processor Sharing"), IS ("Infinite Server") y LCFSPR ("Last Come First Served Preemtive Resume").

En la disciplina FCFS los clientes (trabajos, tareas, procesos o paquetes de datos, dependiendo del modelo) se atienden en el orden de llegada. El centro atiende el requerimiento total del cliente antes de atender el siguiente en la cola.

La disciplina PS es el caso límite de la disciplina "Round Robin" si el costo de la interrupción del servicio se asume

nulo y el quantum se hace tender a cero. En esta disciplina todos los clientes de la cola reciben servicio simultáneamente. Cuando hay n clientes en la cola, cada uno recibe servicio a la 1/n parte de la tasa a la cual recibiría el servicio si estuviera solo.

En la disciplina IS el cliente es atendido cuando llega y no hay demora en la cola. Esto corresponde a un centro donde hay más servidores que clientes.

En la disciplina LCFSPR la cola de clientes se comporta como una pila. El último cliente en llegar es atendido en el acto. El cliente es atendido hasta terminar o hasta que la llegada de otro cliente interrumpa el servicio. Una vez se finaliza el servicio de un cliente, se continuaría prestando el servicio a los demás en la pila, en el orden inverso al de llegada.

La figura 2-1 presenta el modelo de servidor central (Baske72), (Klein76), (Kobay78), (Laven83), (Reise76) y (Sauer81). Los clientes o trabajos viajan cíclicamente haciendo uso alternado del procesador central y de los procesadores de E/S. El modelo es cerrado debido a que el número de trabajos en el sistema, N, es constante. El valor de esta constante es el nivel de multiprogramación del sistema.

El modelo supone que el salto de los trabajos entre los recursos es instantáneo. Sin embargo, con la disciplina IS se pueden modelar demoras o tiempos de propagación, si ello se requiriera.

El tiempo de servicio se asume que es caracterizado por una distribución exponencial con tasa media μ_m, m = 1, .., M. Es posible suponer distribuciones más complejas para representar el tiempo de servicio.

El procesador central es modelado con disciplina de servicio PS y los procesadores de E/S con disciplina de servicio FCFS.

En un momento dado, el estado del modelo es un vector cuyos elementos son el número de trabajos en cada uno de los procesadores. Las suposiciones hechas permiten estudiar la evolución del estado del modelo en el tiempo como un proceso de Markov. De hecho, los procesos markovianos juegan un papel central en la solución de los modelos de colas considerados en este trabajo.

La solución de este modelo tiene utilidad múltiple. Por ejemplo permite conocer, para diferentes niveles de multiprogramación, el porcentaje de tiempo que los procesadores están inactivos y el tiempo que deben esperar los trabajos para hacer uso de los procesadores. También el modelo puede emplearse para establecer qué recursos del

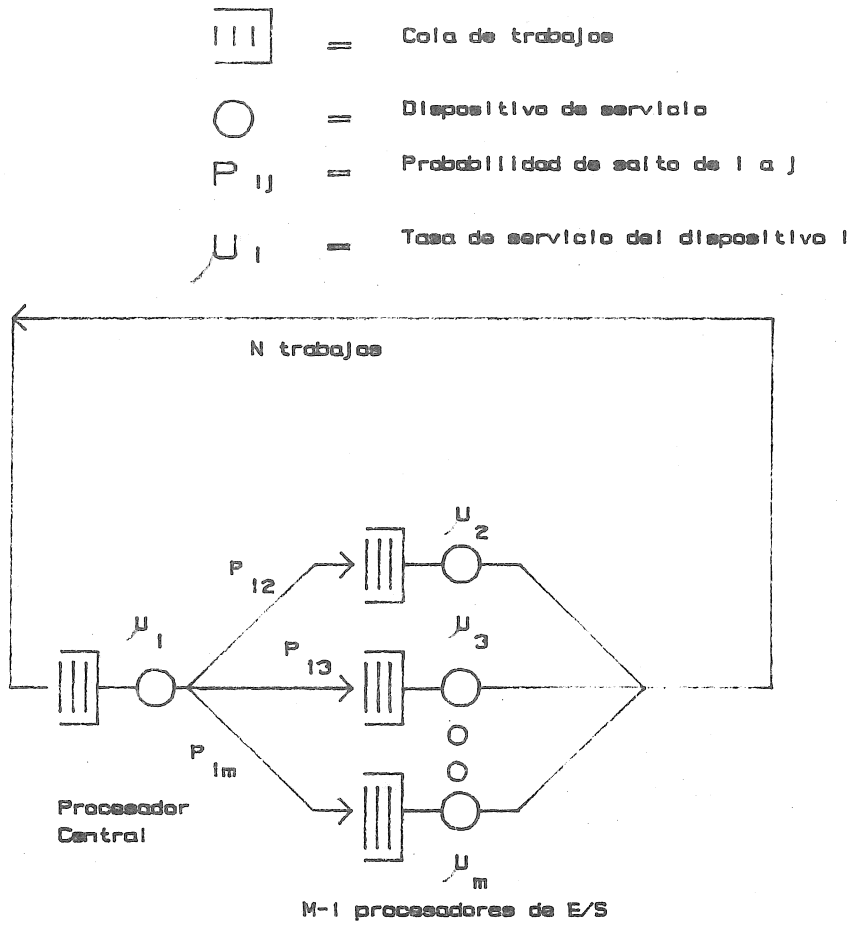


Figura 2 - 1 MODELO DE SERVIDOR CENTRAL. Los trabajos viajan cíclicamente alternando el uso del procesador central y de los procesadores de E/S.

sistema actúan como "cuello de botella".

Pueden formularse otros modelos con mayor nivel de sofisticación que reflejen detalles adicionales de la estructura y comportamiento de los sistemas reales. Igualmente pueden desarrollarse modelos que representen sistemas más complejos, tales como las redes de procesamiento centralizado y distribuido.

3. SOLUCION DE MODELOS POR AGREGACION

Una técnica de solución de modelos de colas es la agregación, llamada también descomposición. Mediante esta técnica se reemplaza una subred por un centro de servicio denominado compuesto. La figura 3-1 muestra un ejemplo de agregación.

Se ha demostrado que la agregación es exacta si el modelo tiene balance local (Chand74). Esto significa que cumpliendose dicha condición, se asegura que el centro de servicio compuesto interactúa de la misma forma que la subred con el resto de los centros de servicio del modelo.

El balance local es una propiedad que tienen ciertos procesos markovianos y conduce a que las soluciones del estado del modelo sean en forma de producto (Baske75):

$$P_i = (P_{i1} \; .. \; P_{iM})/G, \qquad 3\text{-}1$$

en donde P_i es la probabilidad del estado i del sistema; P_{im} es un factor determinado sólamente por el estado del centro de servicio m, m = 1, .., M; y G es la constante de normalización.

Todos los modelos BCMP satisfacen balance local.

Para calcular la tasa de servicio del centro de servicio compuesto, es necesario aislar la subred a agregar del resto de los centros de servicio del modelo. Para esto se conecta su salida directamente a su entrada (las líneas punteadas de la figura 3-1). Y, para cada población posible de clientes en la subred, N, se calcula su tasa de flujo de clientes ("throughput"), R(N). Esta tasa de flujo es, simplemente, la tasa de servicio del centro de servicio compuesto:

$$\mu(N) = R(N). \qquad 3\text{-}2$$

Para calcular R(N), existen varios algoritmos bastante eficientes que son aplicables si la subred satisface balance local (Buzen73), (Chand8Ø) y (Reise8Ø). Uno de los más importantes, en especial por sus buenas propiedades numéricas (Sauer83), es el análisis del valor medio.

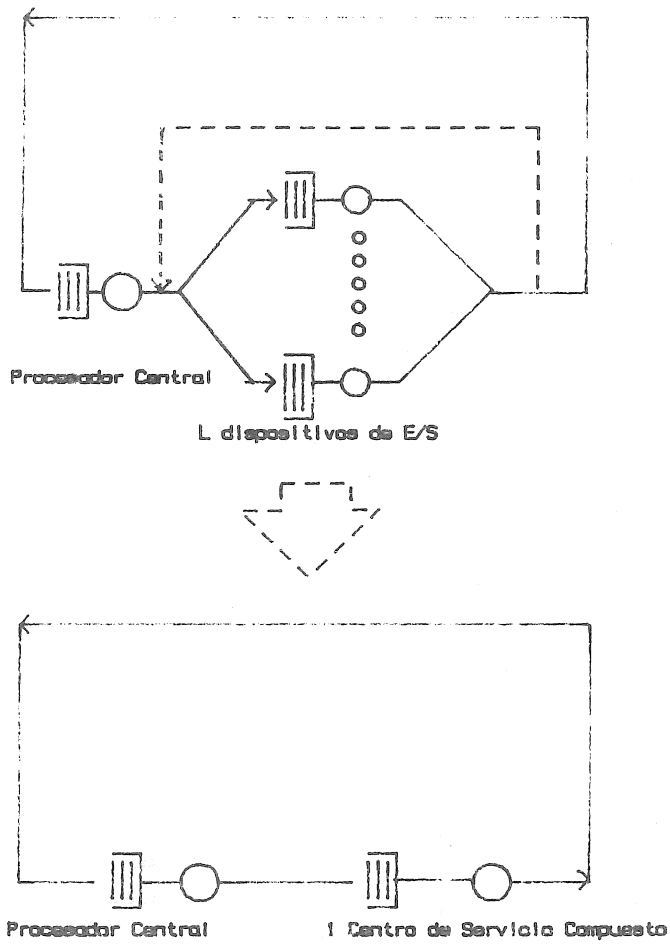


Figura 3 - 1 EJEMPLO DE AGREGACION. La subred de los dispositivos de E/S, es reemplazada por un centro de servicio compuesto con el mismo comportamiento.

El análisis del valor medio utiliza ecuaciones recursivas que relacionan el tiempo medio de espera en un centro de servicio, $Q_m(N)$, con la longitud media de su cola cuando el modelo tiene un cliente menos, $L_m(N-1)$ (Reise80). Por ejemplo, para el modelo de la figura 2-1, se tiene:

$$Q_m(N) = \mu_m \cdot (1 + L_m(N-1)); \qquad 3\text{-}3$$

y para centros de servicio con disciplina IS:

$$Q_m(N) = 1/\mu_m \qquad 3\text{-}4$$

Luego, aplicando las ecuaciones de relación entre:

I. La tasa de flujo en la red, R(N), con su población, N, y su tiempo medio de travesía, Q(N):

$$R(N) = N/Q(N) \qquad 3\text{-}5$$

establecida por la ley de Little (Little61);

II. El tiempo medio de travesía de la red con el tiempo de espera en los centros de servicio:

$$Q(N) = \sum_{m=1}^{M} r_m Q_m, \qquad 3\text{-}6$$

en donde r_m es la tasa relativa de flujo de clientes en el centro de servicio m, calculable con base en las probabilidades de salto de los clientes entre parejas de centros de servicio; y

III. La tasa de flujo de la red con las tasas de flujo, $R_m(N)$, y relativa de flujo r_m de cualquier centro de servicio m:

$$R(N) = R_m(N)/r_m; \qquad 3\text{-}7$$

se puede derivar la siguiente ecuación para el cálculo de la tasa de flujo de clientes en cada centro de servicio:

$$R_m(N) = r_m \frac{N}{\sum_{i=1}^{M} r_i Q_i(N)}, \qquad 3\text{-}8$$

Las longitudes medias de las colas se calculan aplicando nuevamente el resultado de Little:

$$L_m(N) = R_m(N).\ Q_m(N). \qquad 3\text{-}9$$

El análisis del valor medio implica un proceso iterativo y se inicia con $L_m(\emptyset) = \emptyset$, $m = 1, .., M$. De esta manera se pueden determinar $Q_m(n)$, $R_m(n)$ y $L_m(n)$ para $n = 1, .., N$.

Una ventaja importante de la agregación es su ecomomía en términos computacionales. Comparada con otras técnicas de solución de modelos de colas, su costo computacional es el menor.

Otra ventaja de la agregación es su aplicabilidad en modelos sin balance local. Tal es el caso de los modelos con centros de servicio pasivo. Un centro de servicio pasivo es aquél cuya tasa de servicio es determinada externamente por otros centros de servicio. La memoria es un caso típico de centro de servicio pasivo. En general la memoria permanece asignada en tanto el trabajo requiera servicio del procesador central y de las unidades de disco.

Para los modelos con centros de servcio pasivo, la agregación puede ser aproximada, siendo su error normalmente pequeño (Sauer81). Además, en muchas situaciones es superior a otras técnicas de solución de modelos de colas. El análisis numérico presenta problemas de control numérico y requiere la determinación exhaustiva de todos los estados del modelo, lo que es imposible cuando el número de estados no es finito. Finalmente y de otro lado, la simulación no produce necesariamente resultados más exactos que la agregación, además de que, en general, es computacionalmente más costosa.

4. PROGRAMA PARA LA SOLUCION DE MODELOS DE COLAS

Con la elaboración del programa se desea proveer una herramienta para el análisis de modelos de colas. Aunque se ha enmarcado la necesidad del programa en el diseño y evaluación de sistemas de computación, su utilidad se extiende a los sistemas en donde hay competencia por el uso de recursos.

El programa soluciona modelos abiertos y cerrados. Igualmente, no impone restricciones al número de centros de servicio que conforman el modelo ni a su modo de interconexión. Al mismo tiempo, permite poblaciones no homogêneas con lo cual el comportamiento probabilístico de los clientes del modelo puede ser diferente. Esto permite modelar sistemas de computacion en donde hay trabajos que

hacen uso de los recursos como el procesador central y las unidades de E/S, en forma marcadamente diferente.

El programa puede ser utilizado en cualquier computador con microprocesador 8086/8088 y sistema operacional MS-DOS. El computador debe tener por lo menos 128K y una unidad de diskette. El programa ha sido escrito en lenguaje C utilizando el conjunto de instrucciones y funciones descritas en el libro de Kernigham y Ritchie (Kerni78). Esto garantiza que el programa es transportable a otros equipos.

Para la solución del modelo el programa utiliza la técnica de agregación y el análisis del valor medio. La agregación de subredes se lleva a cabo con el fin de reducir la complejidad del modelo. Esto es indispensable cuando el modelo de colas no es de tipo BCMP. El análisis de subredes se hace con base en el análisis del valor medio. Además, para los modelos abiertos, el programa utiliza los resultados para las colas M/M/m y M/G/1.

La agregación, el análisis del valor medio y los resultados para las colas M/M/m y M/G/1 dan soluciones exactas cuando el modelo es de tipo BCMP. Para los modelos generales los resultados pueden ser aproximados; pero en general puede esperarse un error pequeño.

El programa permite la definición del modelo y la consulta de la solución en forma interactiva. Con este fin utiliza un sistema de menús que guían el uso de estas actividades.

Durante la definición del modelo de colas el programa verifica que las características de la red, dadas como entradas, sean válidas. Todos los errores se detectan en la fase de definición. La información de entrada que debe suministrarse al programa es la siguiente:

[] nombre del modelo, entrada opcional

[] número de centros de servicio del modelo

[] disciplina de servicio de cada centro

[] número de servidores de cada centro

[] número de clases de clientes del modelo

[] número de clientes o población del modelo, si éste es cerrado

[] probabilidades de salto entre centros de servicio, para cada clase de clientes

[] tasas de servicio de cada centro, para cada población de clientes posible en este

[] tasa de llegada de clientes, si el modelo es abierto

El programa crea un archivo en disco con las caracteristicas del modelo, de tal manera que puede utilizarse en diferentes sesiones de análisis.

Los parámetros calculados por el programa son:

[] población media en los modelos abiertos

[] utilización de la red de los modelos abiertos

[] utilización de cada centro de servicio

[] tiempo medio de travesia en los modelos abiertos

[] tiempo medio de ciclo en los modelos cerrados ciclicos

[] tasa de flujo de clientes de los modelos abiertos

[] tasa de flujo de clientes en cada centro de servicio

[] tiempo medio de espera en cada centro

[] longitud media de la cola en cada centro

[] probabilidad de todas las posibles longitudes de la cola en los centros

Detalles adicionales pueden ser encontrados en el manual del usuario del programa (Zuñig86).

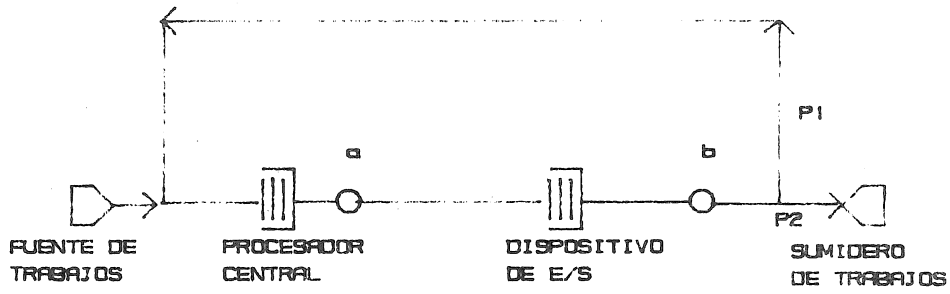
5. USO DE LOS MODELOS DE COLAS

Para ilustrar el uso de los modelos de colas y del programa desarrollado, se ha calculado el desempeño de los modelos de sistemas de computación mostrados en la figura 5-1.

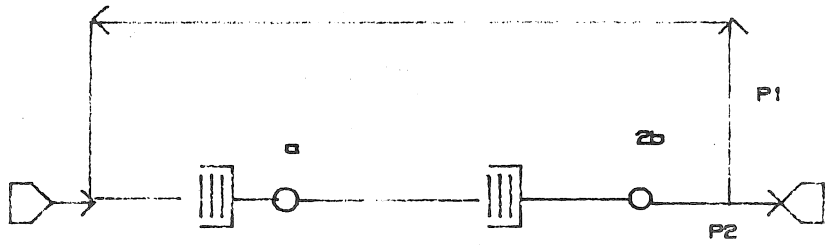
El primer modelo muestra un sistema al cual llegan trabajos con una tasa Poisson con media λ. Cada trabajo es atendido por el procesador central a una tasa de servicio exponencial con media a, y un dispositivo de E/S con tasa de servicio exponencial de media b. Los trabajos pueden requerir más de un ciclo de atención del procesador central y del dispositivo de E/S, ocurriendo esto con una probabilidad p_1; de otra forma, los trabajos salen del sistema luego de recibir servicio en el dispositivo de E/S, con una probabilidad p_2.

Supóngase que se desea aumentar la velocidad de servicio de E/S de este sistema de computación. Para esto se consideran las tres alternativas de la figura 5-1.

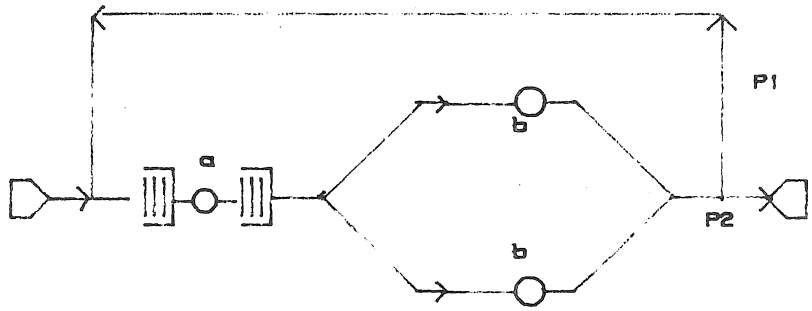
Sistema de computación



1ra alternativa



2da alternativa



3ra alternativa

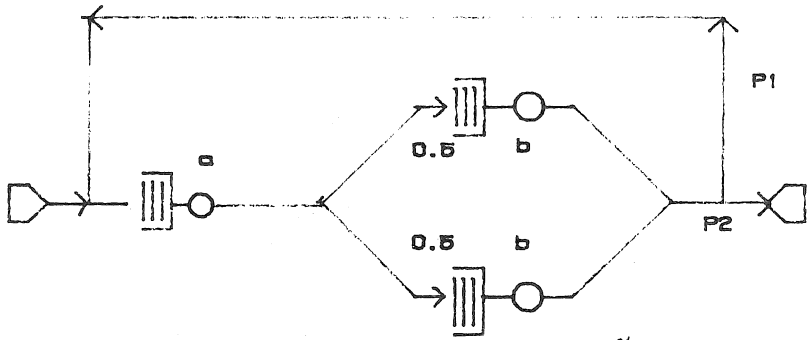


Figura 5 - 1 Alternativas para aumentar el desempeño de un sistema de computacion

En la primera se reemplazaria el dispositivo de E/S por otro con tasa de servicio 2b, dos veces más rápido que el dispositivo del sistema original.

En la segunda habria una cola de trabajos con dos dispositivos de E/S en paralelo, cada uno con tasa de servicio b. Los trabajos en la cola saltarian al primero de los dos dispositivos que se encuentre disponible.

En la tercera se tendrian en paralelo dos dispositivos con tasa de servicio b, los cuales serian escojidos por los trabajos en forma aleatoria con probabilidad 0,5.

En las tres alternativas las restantes caracteristicas del modelo original se mantienen.

Nótese que cada alternativa ofrece duplicar la velocidad del servicio de E/S; pero las dos últimas lo harian sólo en los casos en que se encuentren los dos dispositivos de E/S ocupados al mismo tiempo. Esta situación hace las alternativas similares.

Se desea saber:

- cuál de las tres alternativas daria el mejor aumento de productividad al sistema; y

- si el aumento de productividad justifica retener el procesador central.

Este problema ha sido estudiado en (Chand) y (Lipsk77).

El aumento de productividad de las alternativas de mejora del sistema, puede medirse de varias maneras, cada una de ellas expresando en forma diferente el hecho de que los trabajos serian atendidos más rápidamente. Algunas son las siguientes:

- la disminución del tiempo medio de espera de los trabajos para recibir el servicio de E/S;

- el aumento de la longitud de la cola en el procesador central; y

- el aumento de la utilización del procesador central.

Para el análisis se ha escojido la última pues, además de mostrar el aumento de la productividad del sistema, sirve de base para concluir la conveniencia de retener el procesador central.

La tabla 5-1 y la figura 5-2 muestran la utilización del procesador central para el modelo original y para las tres alternativas de mejora cuando b/a = 0,3. Esta relación corresponde a un sistema cuyo desempeño está limitado por el

b/a = 0,3

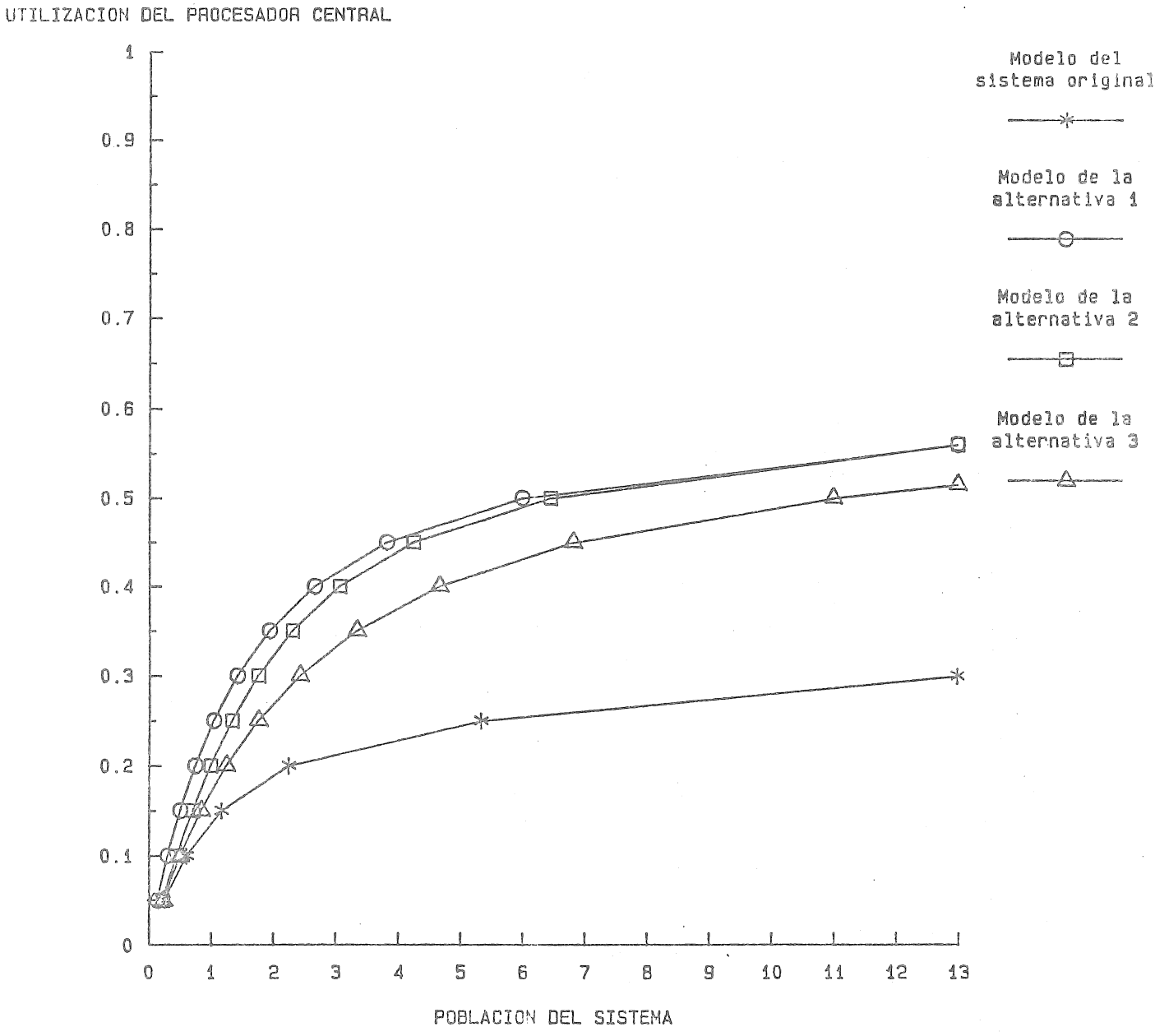


Figura 5-2. Utilizacion del procesador central como funcion de la poblacion del sistema

a = 1,0 b = 0,3

Tasa de Llegada	Utilización del Procesador Central	Población del Sistema	Población de la Alternativa 1	Población de la Alternativa 2	Población de la Alternativa 3
0.01	0.05	0.25263	0.14354	0.22046	0.23445
0.02	0.10	0.61111	0.31111	0.45397	0.51111
0.03	0.15	1.17647	0.50980	0.70980	0.84314
0.04	0.20	2.25000	0.75000	1.00000	1.25000
0.05	0.25	5.33331	1.04762	1.34174	1.76190
0.06	0.30		1.42857	1.76190	2.42857
0.07	0.35	--	1.93846	2.30688	3.33846
0.08	0.40	--	2.66666	3.06666	4.66666
0.09	0.45	--	3.81817	4.24674	6.81816
0.10	0.50	--	5.99998	6.45452	10.99995
0.11	0.55	--	12.22212	12.70038	22.22202
0.12	0.60	--			

Tabla 5-1. Valor de la utilización del procesador central para poblaciones del modelo del sistema y de sus tres alternativas de mejora de la figura 5-1. En la derivación de estos resultados se varió el valor de la tasa de llegada y se tomó $p_1 = 0,8$, $p_2 = 0,2$, a = 1,0 y b = 0,3

servicio del dispositivo de E/S.

La utilización se ha calculado para diferentes poblaciones en cada modelo. Para esto se han fijado los valores de la probabilidad de que el trabajo dé un nuevo ciclo en el sistema, p_1, en 0,8; y la probabilidad de que el trabajo salga del sistema, p_2, en 0,2. Y, al mismo tiempo, se ha variado el valor de la tasa de llegada de trabajos, λ, desde 0,01 hasta 0,12 con incremantos de 0,01. Esto ha permitido calcular la utilización del procesador central como función de una población variable en cada modelo.

Los mismos resultados podrian lograrse fijando el valor de λ y dando direrentes valores a p_1 y p_2, con igual efecto de variar la población en cada modelo.

La tabla 5-1 muestra el valor de λ utilizado para derivar cada utilización y las poblaciones en el modelo del sistema y en las alternativas correspondientes.

La tabla 5-1 y la figura 5-2 permiten concluir que la primera alternativa da mayor utilización al procesador central; y que sólo cuando las poblaciones de los modelos son altas, se logra la misma utilización de 0,6.

Esto significa que cuando el nivel de multiprogramación del sistema (representado por la población del modelo) no es alto, la elección de la alternativa apropiada es de mayor importancia. Tal es el caso de los sistemas en donde el tamaño de la memoria limita el nivel de multiprogramación.

De la misma forma la tabla 5-2 y la figura 5-3 muestran los resultados para b/a = 1. En este caso el sistema tendría balanceados los servicios del procesador central y del dispositivo de E/S. De allí se concluye que nuevamente la mejor alternativa es la primera. También que sólo cuando las poblaciones de los modelos son altas, se logra igual utilización. Por último que el desempeño del sistema mejora un porcentaje pequeño lo que sugiere la necesidad de un nuevo procesador central de mayor velocidad.

En suma, estos ejemplos muestran que:

[] para aumentar la velocidad de servicio de E/S en los sistemas de computación, es mejor utilizar dispositivos más rápidos que aumentar el número de dispositivos para una suma de velocidades igual;

[] la elección de la alternativa apropiada es crucial en sistemas con bajo nivel de multiprogramación; y

[] la mejora de la E/S puede conllevar el cambio del

a = 1,0 b = 1,0

Tasa de Llegada	Utilización del Procesador Central	Población del Sistema	Población de la Alternativa 1	Población de la Alternativa 2	Población de la Alternativa 3
0.01	0.05	0.10526	0.07827	0.10266	0.10391
0.02	0.10	0.22222	0.16374	0.21136	0.21637
0.03	0.15	0.35294	0.25755	0.32732	0.33863
0.04	0.20	0.50000	0.36111	0.45202	0.47222
0.05	0.25	0.66667	0.47619	0.58730	0.61905
0.06	0.30	0.85714	0.60504	0.73548	0.78151
0.07	0.35	1.07692	0.75058	0.89952	0.96270
0.08	0.40	1.33333	0.91667	1.08333	1.16667
0.09	0.45	1.63636	1.10850	1.29218	1.39883
0.10	0.50	2.00000	1.33333	1.53333	1.66666
0.11	0.55	2.44444	1.60153	1.81722	1.98084
0.12	0.60	2.99999	1.92857	2.15934	2.35714
0.13	0.65	3.71428	2.33862	2.58390	2.82010
0.14	0.70	4.66666	2.87179	3.13105	3.41025
0.15	0.75	5.99998	3.59999	3.87272	4.19999
0.16	0.80	7.99997	4.66665	4.95237	5.33332
0.17	0.85	11.33328	6.50577	6.70401	7.14490
0.18	0.90	17.99999	9.81812	10.12846	10.63630
0.19	0.95	37.99995	19.90449	20.22652	20.80925
0.20	1.00				

Tabla 5-2. Valor de la utilización del procesador central para poblaciones del modelo del sistema y de sus tres alternativas de mejora de la figura 5-1. En la derivación de estos resultados se varió el valor de la tasa de llegada y se tomó $p_1 = 0,8$, $p_2 = 0,2$, a = 1,0 y b = 1,0

b/a = 1,0

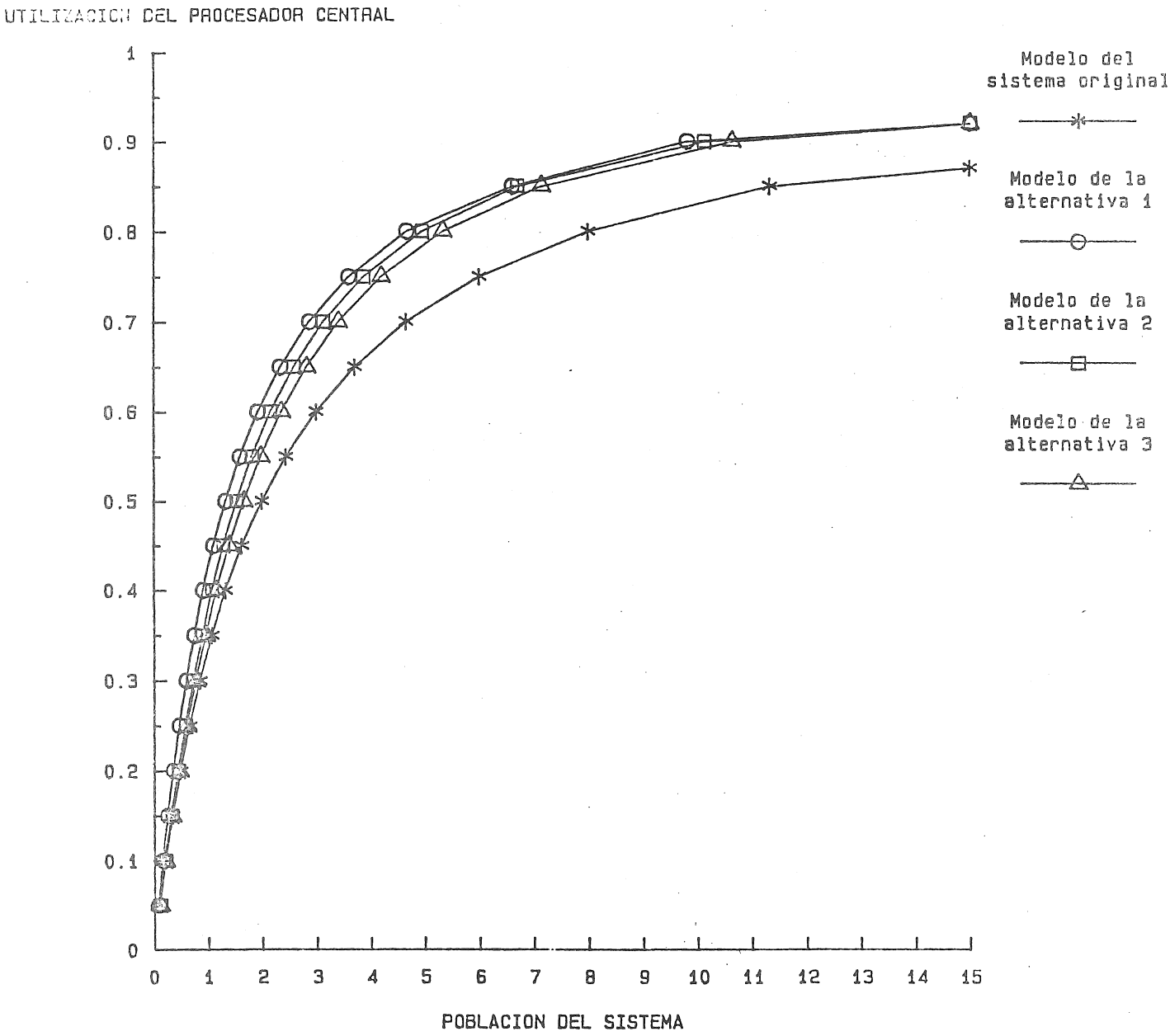


Figura 5-3. Utilizacion del procesador central como funcion de la poblacion del sistema

procesador central si el sistema tiene balanceadas las velocidades de estos servicios.

Los supuestos hechos para los modelos permiten generalizar los resultados a cualquier sistema de computación y a otros recursos como el procesador central.

En el análisis del procesador central se puede decir que en un sistema de computación con bajo nivel de multiprogramación, es mejor colocar un procesador más rápido que hacer multiprocesamiento con varios procesadores con una suma de velocidades equivalente, sin importar si hay sólo una cola de trabajos o si cada uno atendiende su propia cola. Esta conclusión se debe entender sólo en términos del desempeño del sistema y sin considerar otros aspectos como la confiabilidad de este.

Por último debe observarse que éste análisis no ha requerido hacer medición de parámetros de ningún sistema real. Sólo se ha necesitado un conjunto de supuestos que mantiene la generalidad de los resultados. Pero, de otro lado, en otras aplicaciones de esta metodología de análisis, podría ser necesaria la estimación de parámetros de sistemas reales para permitir su modelaje mediante redes de colas.

6. CONCLUSIONES

Se ha presentado un programa para llevar a cabo el análisis de sistemas de computación y de comunicaciones basados en modelos de colas. Al mismo tiempo, se ha dado un ejemplo de evaluación de modelos de sistemas de computación utilizando el programa presentado. Finalmente, se ha mostrado que el programa puede ser de gran utilidad en el diseño y evaluación de sistemas, en particular de los sistemas de computación.

7. AGRADECIMIENTOS

Los autores quieren agradecer de una manera muy especial a la Facultad de Ingeniería de la Universidad de Los Andes, por el apoyo dado a este trabajo.

8. BIBLIOGRAFIA

(Baske72) Baskett, F. y Palacios Gómez, F. Processor Sharing in a Central Server Queueing Model of Multiprogramming wiht Applications. Proc. Sixth Annual Princeton Conference on Information Science and Systems,

Princeton U., Princeton, N.J. Marzo 1972, 598-603.

(Baske75) Baskett, F.; Chandy, K.M.; Muntz, R.R. y Palacios-Gómez, F. Open, Closed and Mixed Networks of Queues whit Different Classes of Customers. J.ACM 22,2 (abril 1975), 248-260.

(Buzen73) Buzen, J.P. Computational Algorithms for Closed Queueing Networks whit Exponential Servers. C.ACM 16,9 (septiembre 1973), 527-531.

(Buzen78) Buzen, J.P. A Queueing Network Model of MVS. Comp. Surveys 10,3 (septiembre 1978), 319-331.

(Chand) Chandy, K.M. Queueing Models in Computer Systems. Fotocopia. Sin fecha.

(Chand75) Chandy, K.M.; Herzog, V. y Woo, L. Parametrics Analysis of Queueing Network Models. RC4730, IBM Reseach, Yorktown Heights, New York. Febrero, 1974. IBM J. Res. Dev. 19,1 (enero 1975), 36-42.

(Chand78) Chandy, K.M. y Sauer, C.H. Approximate Analysis of Queueing Networks Models of Computing Systems. Comp. Surveys 10,3 (septiembre 1978), 281-317.

(Chand80) Chandy, K.M. y Sauer, C.H. Computational Algorithms for Product Form Queueing Networks. C.ACM 23,10 (octubre 1980), 573-583.

(Kerni78) Kernigham, B.W. y Ritchie, D.M. The C Programming Language. Prentice Hall, Englewood Cliffs, N.J. 1978.

(Klein76) Kleinrock, L. Queueing Systems. Vol. II: Computer Applications. Wiley. New York. 1976.

(Kobay78) Kobayashi, H. System Design and Perfomance Analysis Using Analytic Models. En Current Trends in Programming Methodology, Vol III: Software Modeling and its Impact on Perfomance. K.M. Chandy y R.T. Yeh (Eds.), Prentice Hall, Inc., Englewood Cliffs, N.J., 1978, 72-114.

(Laven83) Notes and Reports in Computer Science and Applied Mathematics, Vol IV: Computer Perfomance Modeling Handbook. S.S. Lavenberg (Ed.), Academic Press, Inc., New York, N.Y., 1983.

(Lipsk77) Lipsky, L. y Church, J.D. Applications of a Queueing Network Model for a Computer System. Comp. Surveys 9,3 (septiembre 1977), 205-221.

(Little61) Little, J.D.C. A Proof of the Formula $L = \lambda W$. Op. Res. 9 (1961), 383-387.

(Reise76) Interactive Modeling of Computer Systems. IBM Syst. J. 12,4 (abril 1976), 309-327.

(Reise80) Reiser, M. y Lavenberg S.S. Mean Value Analysis of Closed Multi Chains Queueing Networks. J.ACM 27,2 (abril 1980), 313-322. Corrigendum J.ACM 28,3 (julio 1981), 629.

(Sauer80) Sauer, C.H.; MacNair, E.A. y Salza, S. A Lenguage for Extended Queueing Network Models. IBM J. Res. Dev. 24,6 (noviembre 1980), 747-755.

(Sauer81) Sauer, C.H. y Chandy K.M. Computer Systems Perfomance Modeling. Prentice Hall, Englewood Cliffs, N.J., 1981.

(Sauer83) Sauer, C.H. Computational Algorithms for State Dependent Queueing Networks. ACM Trans. on Comp. Syst. 1,1 (febrero 1983), 67-92. Corrigendum ACM Trans. on Comp. Syst. 1,4 (noviembre 1983), 369.

(Wong78) Wong, J.W. Queueing Network Modeling of Computer Communication Networks. Comp. Surveys 10,3 (septiembre 1978), 343-351.

(Zuñig86) Zúñiga, M. Modelos de Redes de Colas. Tesis de Máster. Depto. Ing. Sist. y Comp. U. de Los Andes, Bogotá. 1986.

TIEMPO, SISTEMA PARA ANALISIS GRAFICO DE SERIES DE TIEMPO

F. Da Silva, E. Granados , B. Lorenzo

Universidad Central de Venezuela
Escuela de Computaciòn
Apartado 47002, Caracas
Venezuela

Resumen

El sistema TIEMPO, es una herramienta computacional diseñada para el anàlisis gràfico de series de tiempo, provee los modelos de regresiòn, tendencias polinomiales, suavizamiento y tendencias cìclicas. Opera en un microcomputador IBM-PC o compatible, y posee un alto grado de "amigabilidad al usuario", una presentaciòn gràfica avanzada, uso del producto LOTUS 1-2-3, para captura de los datos de la serie, y un modulo completo de manejo de archivos, incluyendo una interfaz con el sistema operativo MS-DOS.

1.- Introducciòn.

El estudio y aplicaciones de las series de tiempo, se hace presente en muchas àreas de las ciencias actuales; en economìa, en medicina, en el estudio del medio ambiente, Etc.. Una serie de tiempo, no es mas que una secuencia de observaciones ordenadas respecto al tiempo en que fueron colectadas. En nuestros dìas, el auge de los equipos microcomputadores hace deseable poseer una herramienta, fàcil de manejar, que permita al interesado efectuar la modelaciòn de series de tiempo, liberandolo de las tareas de càlculo y graficaciòn de resultados; no asì de la interpretaciòn de los mismos y de la posible toma de decisiones en base a èllos.

Los estudios para modelar series de tiempo no son nuevos (Anderson (1971), Anderson y Singpurwalla (1980), Da Silva, Granados y Lorenzo (1986)). La forma de determinar los paràmetros del modelo y los posibles test de hipòtesis estan perfectamente determinados; lo novedoso es su implantaciòn en un sistema, con facilidades de graficaciòn de los modelos considerados, que pueda funcionar en microcomputadores.

El presente trabajo, tiene como objetivo describir el sistema TIEMPO (sistema diseñado para permitir la modelaciòn, y graficaciòn correpondiente, de series de tiempo usando un microcomputador IBM-PC o compatible). En la secciòn 2, se describen los modelos implantados. La secciòn 3, describe el sistema desde el punto de vista del usuario. En la secciòn 4, se detallan los aspectos computatcionales relevantes del sistema. La secciòn 5, muestra los resultados obtenidos al usar el sistema

sobre un caso de estudio. Finalmente se presentan las conclusiones y posibles lineas futuras de investigaciòn.

2.- Anàlisis de Series de Tiempo.

Como se mencionò anteriormente, una serie de tiempo no es mas que una secuencia de observaciones ordenadas respecto al tiempo en que fueron colectadas. Matemàticamente, una serie de tiempo, es una secuencia de observaciones $y_1, y_2, \ldots, y_T$, de las variables aleatorias $Y_1, Y_2, \ldots, Y_T$, donde Y_t, es el valor nûmerico de cierto fenòmeno de interès en el tiempo t. Si biên esta definiciòn es bastante general, las series a estudiar deben poseer las siguientes caracteristicas:

- Datos recolectados a intervalos regulares y discretos del tiempo.

- Valores reales, no limitados a un conjunto finito de valores.

- Ausencia de cambios demasiado abruptos.

Para estudiar las series de tiempo, se usa el siguiente modelo genêrico:

$$Y_t = f(t) + U_t \qquad , t = 1, \ldots, T$$

donde,

- T es el nûmero de observaciones,

- f(t), es la parte determinìstica, y

- U_t , es la parte aleatoria donde se incluyen los errores de medida y la influencia de otros factores no cuantificables, por ejemplo, el factor humano.

En este modelo se asumen las siguientes hipòtesis:

a) El tiempo solo intirviene en f(t), esto quiere decir:

   - $E(U_t) = 0$,

   - $VAR(U_t) = \sigma^2$, y

   - las variables U_t , son independientes entre si.

b) f(t) posee coficientes desconocidos a determinar.

Este modelo genêrico, es la base de los siguientes modelos (incluidos en el sistema TIEMPO) :

- Modelo de Regresiòn:

$$f(t) = B_1 Z_{1t} + \ldots + B_p Z_{pt}$$

donde, los B_i, son parâmetros desconocidos; y los Z_i, funciones conocidas del tiempo (Z_{it}, es el valor de Z_i en el tiempo t).

Se proveen test de hipòtesis para determinar si un parâmetro es relevante al modelo o no.

- Modelo Tendencias Polinomiales:

$$f(t) = A_0 \Theta_{0T}(t) + \ldots + A_p \Theta_{pT}(t)$$

donde, los A_i, son parâmetros a determinar, y los Θ_{iT}, polinomios ortogonales que dependen de T.

Se proveen test de hipòtesis para determinar si el grado del polinomio es el adecuado o no.

- Modelo Suavizamiento:

Este modelo es una variaciòn del anterior, y se basa en modelar la serie, con un polinomio, por intervalos.

$$f_t(s) = f(t + s) = B_0 + B_1 s + \ldots + B_p s^p \quad , \quad s = 0, \pm 1, \ldots, \pm m$$

m = longitud del intervalo

p = grado del polinomio

de donde,

$$f(t) = B_0 = \sum_{s=-m}^{m} C_m Y_{t+m}$$

satisfaciendose,

$$\sum_{s=-m}^{m} C_m = 1 \qquad y \qquad C_m = C_{-m}$$

El objetivo es encontrar el estimador de B_0, o equivalentemente los C_m.

- Modelo Tendencias Ciclicas:

$$f(t) = A_0 + \sum_{j=1}^{q} [A(k_j)\,\mathrm{sen}(2\pi k_j t/T) + B(k_j)\cos(2\pi k_j t/T)]$$

cuando T es impar, o

$$f(t) = A_0 + \sum_{j=1}^{q} [A(k_j)\,\mathrm{sen}(2\pi k_j t/T) + B(k_j)\cos(2\pi k_j t/T)] + (-1)^t A_{T/2}$$

cuando T es par.

Donde los valores k_1, k_2, ..., k_q, son un subconjunto propio del conjunto (1, 2, ..., (T - 1)/2), si T es impar, o del conjunto (1, 2, ..., T/2 - 1), si T es par. Y los valores A_0, $A(k_j)$, $B(k_j)$ y $A_{T/2}$, parámetros a determinar.

Se proveen test de hipótesis para determinar si los parámetros $A(k_j)$ y $B(k_j)$ son relevantes en el modelo o no.

Para una descripción detallada de cada uno de estos modelos, véase Da Silva, Granados y Lorenzo (1986).

3.- TIEMPO, Sistema para Modelar Series de Tiempo.

El sistema TIEMPO, fue diseñado para obtener los parámetros de los modelos descritos en la sección anterior, en ningun caso se toma decisiones en base a ellos. Por ejemplo, si no se satisface una hipótesis se elimina el parámetro, Etc.; este trabajo se deja al usuario que es la persona que puede dar interpretación a los resultados.

El sistema TIEMPO, puede dividirse en tres areas:

a) Interacción con el usuario,

b) Presentación de resultados,

c) Facilidades adicionales.

3.1.- Interacción con el usuario:

Uno de los objetivos del sistema TIEMPO, es ser "amigable al usuario", esto significa, entre otras cosas, mantener siempre informado al usuario de lo que pasa en el sistema, decirle que sucederá cuando ejecute un comando, y requerirle la menor cantidad de información posible para ejecutar un comando.

La introducción de comandos se hace de la manera siguiente, la primera linea de la pantalla se reserva para los menús y la segunda para la descripción de la opción del menú seleccionada. Las opciones de los menús se selccionan a través del movimiento de los cursores y se ejecutan pulsando la tecla <ENTER>. Para usuarios avanzados, con presionar la tecla correspondiente a la primera letra en mayúscula de una opción del menú, se ejecuta el comando asociado. Ver Figura 1.

La última linea de la pantalla, se reserva para informar al usuario del estado del sistema. Las restantes, son el àrea de trabajo del sistema y se describen en la proxima sección.

Este tipo de introducción de comandos, se mantiene a lo largo de todo el sistema, tanto si el usuario desea especificar valores o una función. Para este último caso, se diseñó una màquina de estado finito, que indica al usuario cuales son los posibles elementos que pueden conformar una función. Ver Figura 2.

Un aspecto importante dentro de la interacción con el usuario, es la especificación de los valores de la serie. Dado que una serie de tiempo, puede ser representada como una tabla, donde las filas son los valores de las variables en un instante de tiempo y las columnas las variables en si de la serie; se usa el producto LOTUS 1-2-3, para capturar estos valores. Esto significa, que desde el sistema TIEMPO se hace la interfaz directa a este último producto, el cual será presentado en el punto 4.2 de este trabajo. Cada serie se almacena en un archivo independiente, lo que permite al usuario tener tantas series como desee.

3.2.- Presentación de resultados:

Si bièn una tabla, es ideal para almacenar los valores de la serie, por el contrario no lo es para tomar decisiones en forma ràpida. Para esto es preferible un gràfico de la serie y de los resultados obtenidos con el modelo. TIEMPO, se basa en esta premisa y mantiene en las lineas tercera a penúltmita de la pantalla, un gràfico de la serie, y en el caso de haberse procesado un modelo, de sus resultados. A su vez, se provee un modulo de manejo de gràficos, el cual permite al usuario manipular los mismos, para mejor visualización, impresión y poner los titulos que desee. Ver Figura 3.

El esquema de graficación, se mantiene para la presentación de los resultados estadísticos, en este caso se grafica la variable aleatoria correspondiente y los valores de los estadísticos. Ver Figura 4. En caso de quererse, los valores numèricos del modelo y las aproximaciones; los mismos pueden ser visualizados e impresos. Ver Figura 5.

3.3- Facilidades adicionales:

Un sistema no es completo, si no permite al usuario efectuar todas las funciones que el espera realizar con su objeto de trabajo, para este sistema el objeto son series de tiempo y se representan como archivos. Por lo tanto, se provee un modulo de manejo de archivos genèrico, el cual permite copiar, eliminar, borrar y renombrar archivos; asi como otros funciones del sistema operativo MS-DOS, incluyendo una interfaz directa con el mismo. Ver Figura 6.

4.- Aspectos computacionales del sistema TIEMPO.

4.1.- Componentes del sistema:

El sistema, fuè desarrollado en el lenguaje Turbo Pascal, sus componentes se muestran en la figura 7; en la misma se observa la serie de archivos y programas que se emplean.

El programa ITIEMPO, es el inicializador del archivo de configuraciòn del sistema (TIEMPO.CNF), en este archivo se almacena el medio ambiente donde se ejecutarà el sistema TIEMPO.

El programa TIEMPO, esta compuesto por los siguientes archivos: TIEMPO.COM, TIEMPO.001 a TIEMPO.006 (englobados en la Figura 7, bajo el nombre de TIEMPO), que son los modulos ejecutables, en los mismos se usa la tecnica de solapamiento. El archivo TIEMPO.COM, es el modulo de control y los restantes los archivos de solapamiento. El modulo GRAPHIX (Turbo Tool Graphix) està incluido dentro de los archivos anteriores. Los archivos ERROR.MSG, 8x8.FON y 4x6.FON, son los archivos para el modulo GRAPHIX, los dos ultimos son los archivos donde se almacenan las definiciones de los caràcteres gràficos. El archivo TIEMPO.AYU, contiene los textos de ayuda para el usuario, estos son descripciones de cada uno de los modelos implantados y de los paràmetros que debe especificar el usuario para que sean procesados; esta informaciòn se muestra al presionar la tecla F1 (Ver Figura 8). El manejador PRTSC.SYS (provisto con el equipo), es usado para imprimir los gràficos.

Los archivos de datos son los siguientes: Los de extensiòn .DIF, que almacenan los valores de la serie de tiempo, los de extensiòn .REG, .POL, .SUA y .CIC, que almacenan los resultados del ùltimo modelo de regresiòn, tendencias polinomiales, suavizamiento y tendencias cìclicas, aplicados a la serie de tiempo. De esta forma el usuario puede guardar los resultados de un modelo particular y recuperarlos, de forma automàtica, la pròxima vez que modele la serie. Existe uno de estos archivos por cada archivo de extension .DIF.

Los elementos del producto LOTUS 1-2-3, son: El programa 123.EXE, que ejecuta el manejo de la hoja electronica (tablas de la serie dentro de TIEMPO); los programas WKSDIF.EXE y DIFWKS.EXE, que se encargan de la transaformaciòn del formato de los archivos que emplea LOTUS al formato de archivos empleado por TIEMPO; y el archivo 123.CNF que contiene la configuraciòn del medio ambiente para LOTUS.

4.2.- Elementos relevantes:

a) Intefaz con LOTUS 1-2-3:

Ejecutar un programa foraneo a Turbo Pascal desde èste, no se

puede hacer en forma directa, si se desea retomar el control de ejecución al finalizar el programa foraneo. Para lograrlo, se requirió incluir una rutina en lenguaje de máquina (Figura 9), que se encarga de salvar los registros de medio ambiente de ejecución del sistema, invocar al programa foraneo, y finalmente restaurar los registros de ejecución dandose control al Run-Time system de Turbo Pascal.

A su vez, leer el formato de datos empleado por LOTUS para manejar sus archivos es practicamente imposible, debido a la falta de documentación sobre el mismo. A fin de obtener la información de estos archivos, se efectua una transaformación del formato LOTUS al formato DIF (Data Interchange Format) el cual es descrito en Kalish y Mayer (1981) , esto es posible por la presencia de los programas DIFWKS.EXE y WKSDIF.EXE.

El último problema que se enfrenta al realizar una interfaz de este tipo es: como decir al programa 123.EXE que tome como archivo de trabajo, el archivo que desee modificar el usuario. LOTUS posee la capacidad de cargar en forma automática el archivo de datos AUTO123.WKS (los archivos de extension WKS, se encuentran en formato LOTUS) que se encuentre en su directorio de defecto. El sistema TIEMPO, transforma el archivo DIF al archivo AUTO123.WKS y cambia el directorio de defecto de LOTUS (almacenado en el archivo 123.CNF) al directorio donde reside este archivo.

b) Impresión de gráficos:

La impresión de gráficos desde el modulo GRAPHIX, se restringe a un grupo pequeño de impresoras. Sin embargo, MS-DOS prove el manejador PRTSC.SYS que copia el contenido de la pantalla a la impresora que esté conectada al equipo. Este manejador se carga en la inicialización del sistema (si se incluye en el archivo CONFIG.SYS de MS-DOS) y luego puede ser usado desde cualquier lenguaje de programación. Este es el esquema que emplea TIEMPO.

c) Manejo de teclado y pantalla:

El manejo de teclado y pantalla, se realiza a través de rutinas especiales que controlan su funcionamiento, entre ellas tenemos: captura de un carácter, video reverso de una parte de la pantalla, validación de carácter y texto, Etc..

5.- Estudio de casos.

El presente ejemplo es tomado de Anderson y Singpurwalla(1980).

El consumo anual de carne en los Estados Unidos durante el período 1919 - 1941, se muestra en la Tabla 1.

Tabla 1

Período de tiempo 1918 + t	Consumo anual de carne y_t
1	171.5
2	167.0
3	164.5
4	169.3
5	179.4
6	179.2
7	172.6
8	170.5
9	168.6
10	164.7
11	163.0
12	162.1
13	160.2
14	161.2
15	165.8
16	163.5
17	146.7
18	160.2
19	156.8
20	156.8
21	165.4
22	174.7
23	178.7

La forma como se introducen los datos usando LOTUS, se muestra en la Figura 10.

Los resultados obtenidos con el sistema TIEMPO, se muestran en las Figuras 11 a 32. Las Figuras 11 a 17, muestran los resultados obtenidos al aplicar el modelo de regresión con la función $f(t) = B_1 + B_2 t * sen(t) + B_3 t^2$. Las Figuras 18 a 21, al aplicar el modelo tendencias polinomiales con un polinomio de grado 3. Las Figuras 22 a 24, al aplicar el modelo de suavizamiento en un intervalo de longitud 3 con un polinomio de grado 2. Y finalmente, las Figuras 25 a 32, al aplicar el modelo tendencias ciclicas con el conjunto $(k_1, k_2, k_3) = (3, 6, 9)$.

6.- Concluisones.

A pesar de que el trabajo no introduce nada nuevo en el plano teórico, el trabajo computacional en si es totalmente novedoso. Introducir algun modelo de Series de Tiempo nuevo o mejorar alguno de los existentes es un trabajo más complicado. Generalmente, las innovaciones teóricas son para casos muy particulares o son el resultado de una vasta experiencia en el Análisis de Series de Tiempo.

El trabajo, aunque se trata de matemàticas aplicadas a la computaciòn, se relaciona con otras àreas de esta ciencia, por ejemplo :

- Con el àrea de Programaciòn de Sistemas: Se interactua con el sistema operativo MS-DOS.

- Con el àrea de los Lenguajes de Programaciòn: En el modulo de Regresiòn se da la posibilidad de editar una funciòn.

- Con el area de Computaciòn Gràfica: Todo el manejo y presentaciòn gràfica del sistema està relacionado con esta àrea.

En lo que corresponde a futuros desarrollos, los resultados obtenidos, inducen a estudiar la implantaciòn de nuevos modelos, tales como filtros de Kalmar y suavizamiento exponencial; tambièn, se espera mejorar la capacidad de graficaciòn. restringida actualmente a 100 observaciones de la serie. Y probablemente, abandonar el uso de LOTUS y diseñar una herramienta de captura solo para las observaciones de la serie.

Agradecimientos

Los autores desean expresar su agradecimiento a los profesores Euripides Montagne y Eliecer Correa del Escuela de Computaciòn, Universidad Central de Venezuela. Por su ayuda en la correcciòn de este trabajo.

Referencias

Anderson, T. W. (1971), "The Statistical Analysis of Time Series", John Wiley and Sons Inc..

Anderson, T. W. and Singpurwalla, N. D. (1980), "Methods and Applications of Time Series Analysis", Deparment of Statistics Stanford University.

Da Silva, F. , Granados, E. R. y Lorenzo, B. (1986), "TIEMPO - Sistema para la modelaciòn de Series de Tiempo", Trabajo especial de grado Licenciatura en Computaciòn, Escuela de Computaciòn, Facultad de Ciencias, Universidad Central de Venezuela.

Kalish C. and Mayer M. (1981), "DIF : A format for data exchange between Applications Programs", BYTE Noviembre 1981.

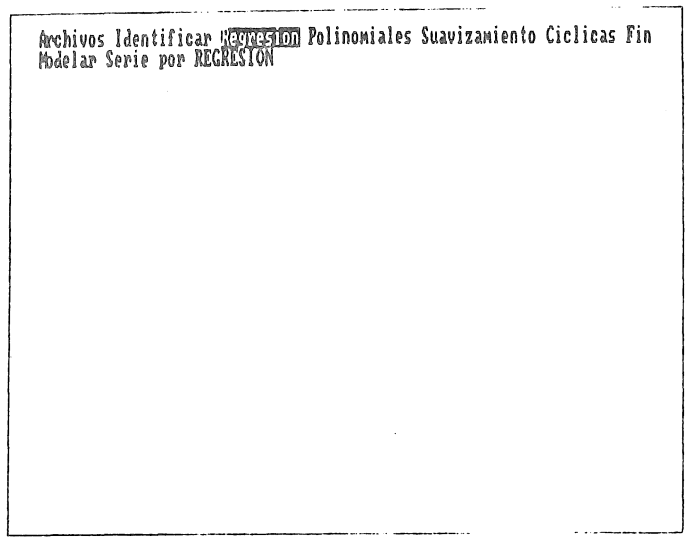


Figura 1. Introducción de comandos

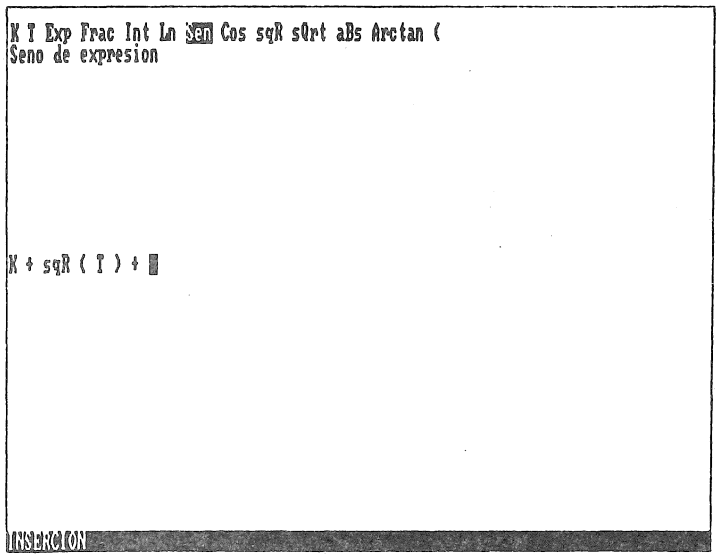


Figura 2. Especificación de una función

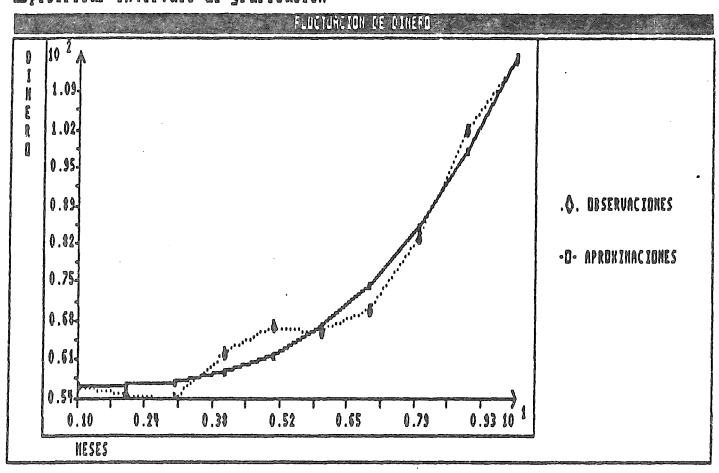


Figura 3. Presentación de resultados

Figura 4. Presentación de estadísticos

Imprimir Fin
Imprimir estadísticos

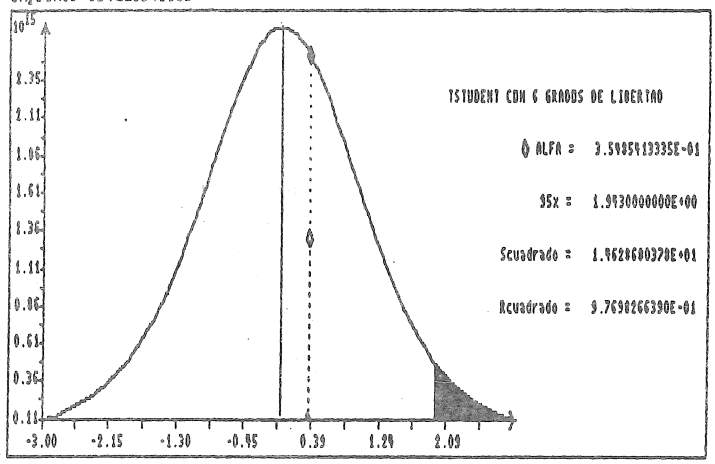


Figura 5. Presentación numérica de resultados

Imprimir Siguiente Anterior Fin
Imprimir valores

i	Best(i)	i	Best(i)	i	Best(i)
1	1.0303507445E+00	2	-1.2083234415E+00	3	5.5972488838E+01
4	7.5810889427E-02				

Figura 6. Menú para manejar archivos

Crear Editar Driver Path forMatear cOpiar Renombrar Borrar directorio DOS Fin
Ejecutar comando de Dos

Figura 7. Componentes del sistema TIEMPO

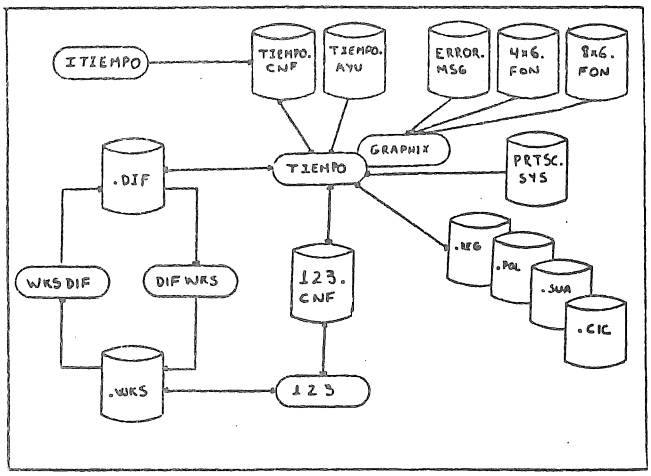


Figura 8. Pantalla inicial de ayuda

```
                        T I E M P O

El  sistema  TIEMPO  realiza  el  proceso  de  MODELACION  de  una
SERIE DE TIEMPO.
Una  SERIE  DE  TIEMPO,  se  define  como  una  secuencia  de observaciones
ordenadas  respecto  al  tiempo  en  que  fueron  colectadas.  Es decir, una
secuencia  de observaciones  (y1, y2, ..., yT)  de las variables aleatorias
(Y1, Y2, ..., YT). Cada  Yt  es el valor de  cierto fenomeno en el tiempo t.
El sistema  TIEMPO  permite la modelacion de una Serie de Tiempo, usando los
siguientes modelos:

        1- REGRESION                    2- TENDENCIAS POLINOMIALES

        3- SUAVIZAMIENTO                4- TENDENCIAS CICLICAS

          Presione el numero del modelo para obtener su descripcion
```

Figura 9. Rutina para ejecutar un programa foraneo desde Turbo Pascal

```
Function Execute(SegAddPath,OfsAddPath,SegAddParam,OfsAddParam
  : Integer) : Boolean;
{$$$
```

Figura 11. Funciones para el modelo de regresión

```
T I E M P O                    MODULO DE IMPRESION              ARCHIVO : DATPOL

REGRESION                      EXPRESIONES ARITMETICAS

NUMERO  EXPRESION

  1       1.0000000000E+00
  2     T ^ Sen ( T )
  3     sqR ( T )
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
```

Figura 12. Resultado gráfico del modelo de regresión

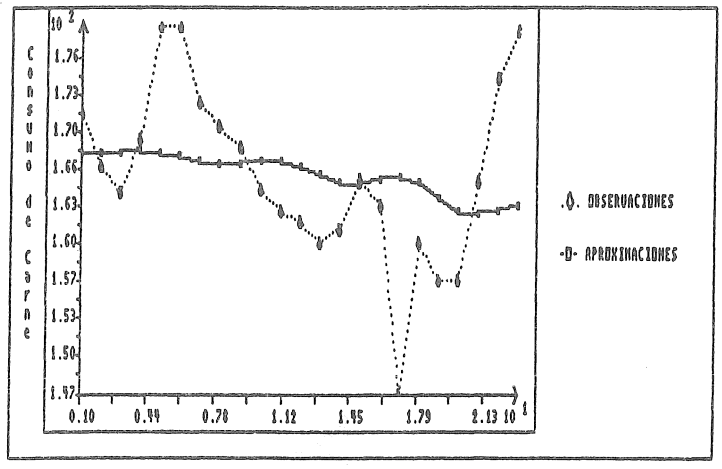


Figura 13. Aproximaciones obtenidas con regresión

```
T I E M P O                    MODULO DE IMPRESION              ARCHIVO : DATPOL

REGRESION                      ESTIMADORES DE LOS Y(t)

ELEMENTOS DEL MODELO : 1 2 3

    t        Yest(t)          t        Yest(t)          t        Yest(t)

    1   1.6815940493E+02      2   1.6807826079E+02      3   1.6809438900E+02
    4   1.6819191954E+02      5   1.6818370553E+02      6   1.6790943758E+02
    7   1.6745563187E+02      8   1.6712848743E+02      9   1.6715639142E+02
   10   1.6741027654E+02     11   1.6744298191E+02     12   1.6698756427E+02
   13   1.6612322234E+02     14   1.6541233728E+02     15   1.6530673562E+02
   16   1.6569228563E+02     17   1.6592561983E+02     18   1.6540813733E+02
   19   1.6419139102E+02     20   1.6300092481E+02     21   1.6259494765E+02
   22   1.6302226680E+02     23   1.6350338090E+02
```

Figura 14. Parámetros del modelo de regresión

```
T I E M P O                    MODULO DE IMPRESION              ARCHIVO : DATPOL

REGRESION                      ESTIMADORES DE LOS B(i)

ELEMENTOS DEL MODELO : 1 2 3

    i        Best(i)          i        Best(i)          i        Best(i)

    1   1.6821227178E+02      2  -5.0059433672E-02      3  -1.0743291319E-02
```

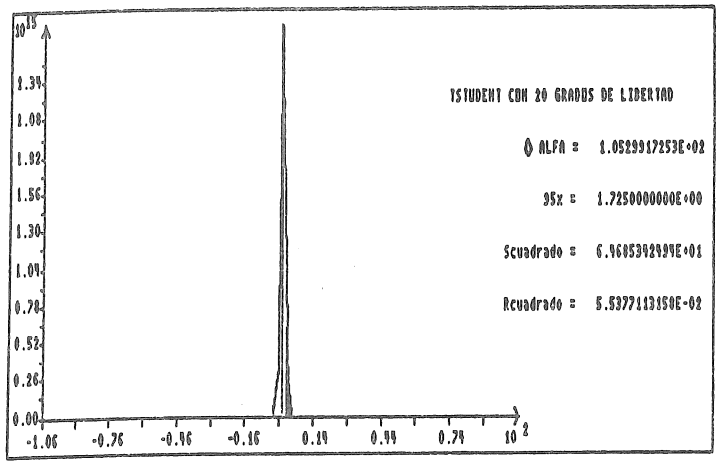


Figura 15. Estadístico de B_1

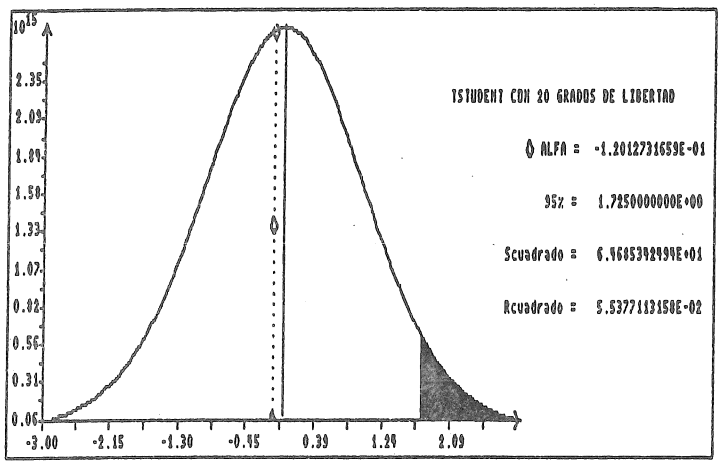


Figura 16. Estadístico de B_2

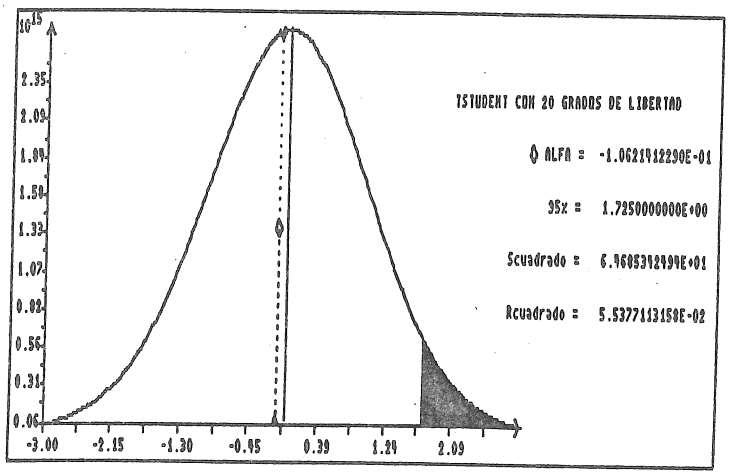


Figura 17. Estadístico de B_3

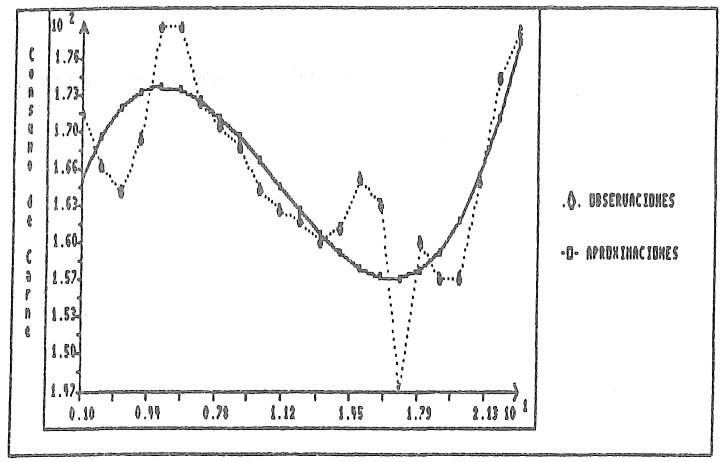


Figura 18. Resultado gráfico del modelo tendencias Polinomiales

Figura 19. Aproximaciones obtenidas con tendencias polinomiales

T I E M P O — MODULO DE IMPRESION — ARCHIVO : DATPOL

TENDENCIAS POLINOMIALES — ESTIMADORES DE LOS Y(t) — GRADO POLINOMIO : 3

t	Yest(t)	t	Yest(t)	t	Yest(t)
1	1.6580468227E+02	2	1.6945588325E+02	3	1.7192677757E+02
4	1.7335011076E+02	5	1.7385862833E+02	6	1.7358507579E+02
7	1.7266219867E+02	8	1.7122274247E+02	9	1.6939943272E+02
10	1.6732507492E+02	11	1.6513233460E+02	12	1.6295403727E+02
13	1.6092286843E+02	14	1.5917159362E+02	15	1.5783295835E+02
16	1.5703970812E+02	17	1.5692458845E+02	18	1.5762034487E+02
19	1.5925972288E+02	20	1.6197546801E+02	21	1.6590032576E+02
22	1.7116704165E+02	23	1.7790836120E+02		

Figura 20. Parámetros del modelo tendencias polinomiales

T I E M P O — MODULO DE IMPRESION — ARCHIVO : DATPOL

TENDENCIAS POLINOMIALES — ESTIMADORES DE LOS A(p) — GRADO POLINOMIO : 3

p	Aest(p)	p	Aest(p)	p	Aest(p)
0	1.6619130435E+02	1	-3.7905138341E-01	2	7.3574251834E-02
3	2.2124252559E-02				

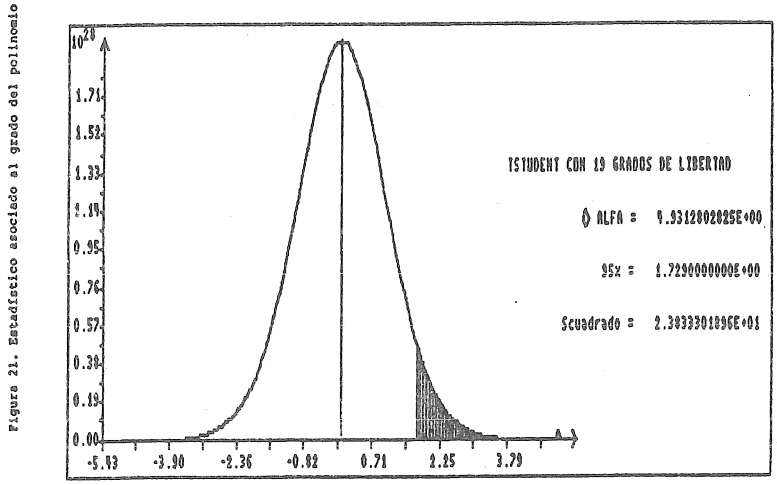


Figura 21. Estadístico asociado al grado del polinomio

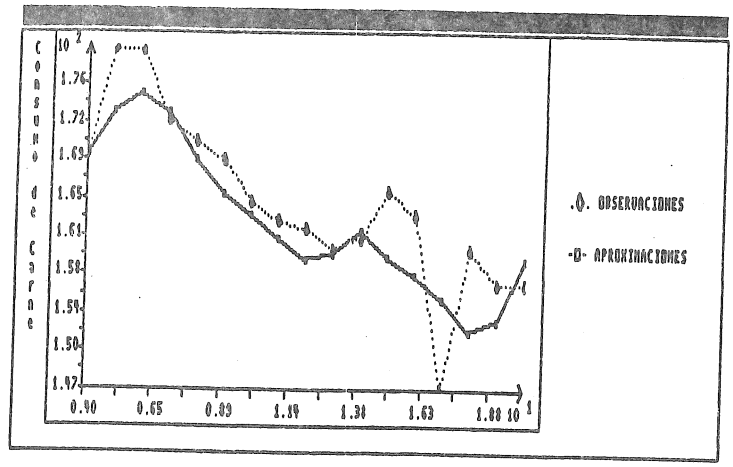


Figura 22. Resultado gráfico modelo suavizamiento

Figura 23. Aproximaciones obtenidas con el modelo suavizamiento

T I E M P O — MODULO DE IMPRESION — ARCHIVO : DATPOL

SUAVIZAMIENTO ESTIMADORES DE LOS Y(t) GRADO POLINOMIO : 2 LONG. INT. : 3

t	Yest(t)	t	Yest(t)	t	Yest(t)
4	1.6932942177E+02	5	1.7334064626E+02	6	1.7507602041E+02
7	1.7330425170E+02	8	1.6880068027E+02	9	1.6540153061E+02
10	1.6348622449E+02	11	1.6112942177E+02	12	1.5913741496E+02
13	1.5954408843E+02	14	1.6198622449E+02	15	1.5928146258E+02
16	1.5761428571E+02	17	1.5529625850E+02	18	1.5263197279E+02
19	1.5330255102E+02	20	1.5923860544E+02		

Figura 24. Parámetros del modelo suavizamiento

T I E M P O — MODULO DE IMPRESION — ARCHIVO : DATPOL

SUAVIZAMIENTO ESTIMADORES DE LOS Cs GRADO POLINOMIO : 2 LONG. INT. : 3

s	Cs	s	Cs	s	Cs
0	3.3163265306E-01	1	2.8401360544E-01	2	1.4115646258E-01
3	-9.6938775510E-02				

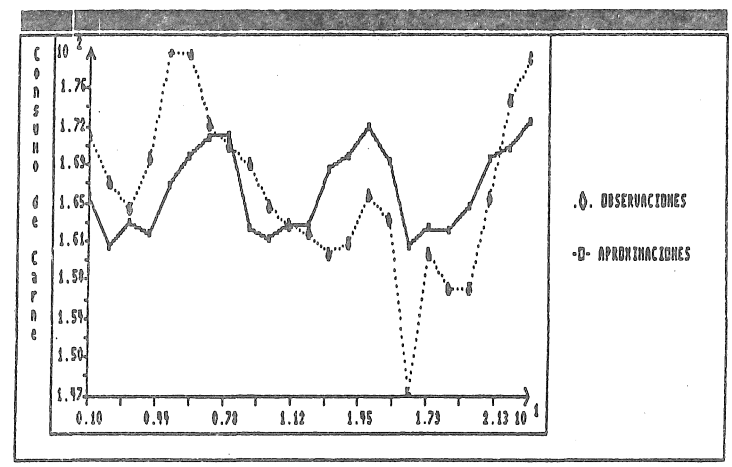


Figura 25. Resultado gráfico modelo tendencias cíclicas

Figura 26. Aproximaciones obtenidas modelo tendencias cíclicas

T I E M P O — MODULO DE IMPRESION — ARCHIVO : DATPOL

TENDENCIAS CICLICAS — ESTIMADORES DE LOS Y(t) — NUMERO TERMINOS SERIE : 3

t	Yest(t)	t	Yest(t)	t	Yest(t)
1	1.6567129354E+02	2	1.6079168755E+02	3	1.6319735020E+02
4	1.6217879168E+02	5	1.6658098448E+02	6	1.6936068540E+02
7	1.7122544434E+02	8	1.7155863432E+02	9	1.6269333999E+02
10	1.6166392300E+02	11	1.6293024076E+02	12	1.6293371965E+02
13	1.6822281582E+02	14	1.6952483934E+02	15	1.7235001351E+02
16	1.6899066292E+02	17	1.6097869394E+02	18	1.6271438009E+02
19	1.6234953534E+02	20	1.6458388264E+02	21	1.6910522210E+02
22	1.7011874990E+02	23	1.7267369766E+02		

Figura 27. Parámetros modelo tendencias cíclicas

T I E M P O — MODULO DE IMPRESION — ARCHIVO : DATPOL

TENDENCIAS CICLICAS — ESTIMADORES DE LOS A(Kq) — NUMERO TERMINOS SERIE : 3

Kq	A(Kq)	Kq	A(Kq)	Kq	A(Kq)
0	1.6619130435E+02	3	4.0194945615E+00	6	1.2307214074E+00
9	1.2321773443E+00				

Figura 28. Parámetros modelo tendencias cíclicas

T I E M P O — MODULO DE IMPRESION — ARCHIVO : DATPOL

TENDENCIAS CICLICAS — ESTIMADORES DE LOS B(Kq) — NUMERO TERMINOS SERIE : 3

Kq	B(Kq)	Kq	B(Kq)	Kq	B(Kq)
3	-3.2506303632E+00	6	-4.8028713901E-01	9	1.0000458432E+00

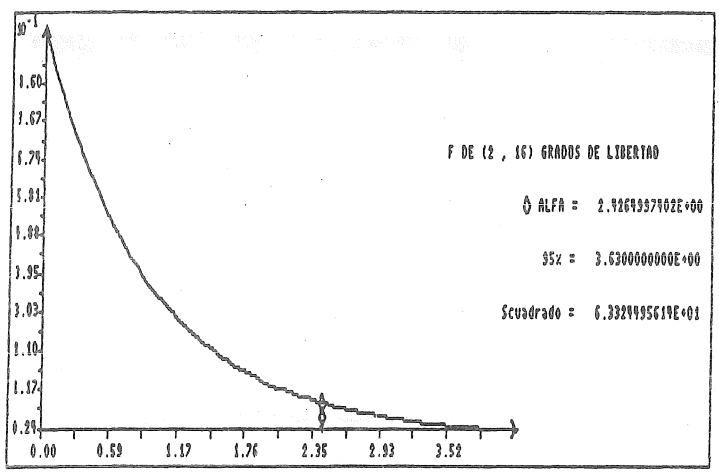


Figura 29. Estadísticos asociados a $A(k_1)$ y $B(k_1)$

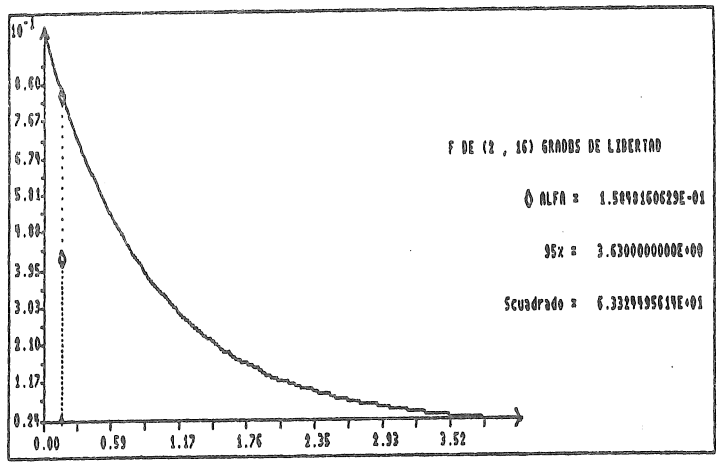


Figura 30. Estadísticos asociados a $A(k_2)$ y $B(k_2)$

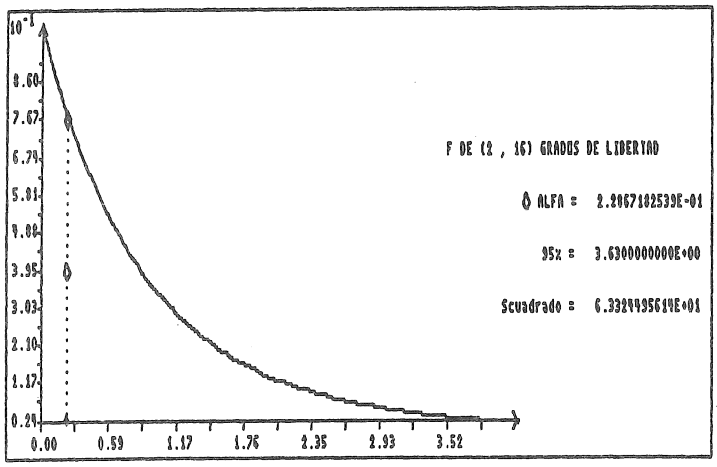


Figura 31. Estadísticos asociados a $A(k_3)$ y $B(k_3)$

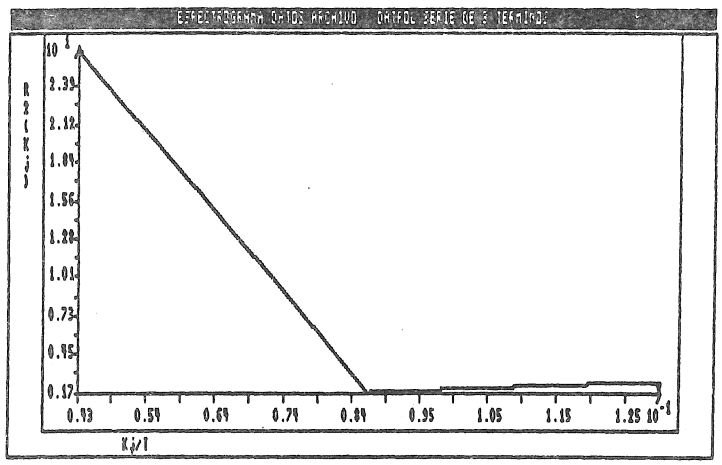


Figura 32. Espectrograma del modelo tendencias cíclicas

Evaluación y Predicción del Desempeño de Sistemas Computacionales

Ignacio Casas
Francisco Aurtenechea

Depto. de Ciencia de la Computación
Escuela de Ingeniería
Universidad Católica de Chile
Casilla 6177 (143), Santiago, Chile

Resumen

El desempeño de un sistema computacional puede verse afectado por un gran número de decisiones que deben ser tomadas durante las distintas etapas de análisis, desarrollo y puesta en marcha del sistema. Debido a la complejida de las alternativas (trade-offs) de dimensionamiento, adquisición, diseño e implementación, es difícil predecir el impacto de las decisiones en el futuro desempeño del sistema. En este artículo se presenta una introducción a las técnicas de evaluación del desempeño de sistemas computacionales basadas en modelos analíticos de redes de espera. Como una aplicación práctica de estas técnicas, se describe un modelo específico para un sistema computacional en operación. Se presentan los resultados de validación del modelo contra mediciones efectuadas durante un período de observación de una semana. La validación del modelo indica un nivel de precisión adecuado para los objetivos del estudio. Se presenta también un estudio de proyecciones donde se utiliza el modelo para predecir el efecto de cambios tanto del sistema como de la carga de trabajo.

1. INTRODUCCION

El desempeño de un sistema computacional puede verse afectado por un gran número de decisiones que deben ser tomadas durante las distintas etapas de análisis, desarrollo y puesta en marcha del sistema. Debido a la complejida de las alternativas (trade-offs) de dimensionamiento, adquisición, diseño e implementación, es difícil predecir el impacto de las decisiones en el futuro desempeño del sistema. De igual manera, es difícil planificar la forma en que debe crecer la capacidad de procesamiento del sistema, en función de la evolución de la demanda de servicio (nuevas aplicaciones, aumento de la tasa de llegada de transacciones, etc.).

Actualmente, se están desarrollando nuevas técnicas de modelación, basadas en modelos analíticos de redes de espera, que pueden ser utilizadas para ayudar en la evaluación y predicción del desempeño de sistemas computacionales ([Grah78], [Saue81], [Lazo84], [Casa86]). Estas herramientas y metodologías sirven de apoyo al experto en evaluación y selección de equipamiento. Con estos modelos, el desempeño de un sistema computacional se evalúa en términos de tiempos de respuesta para transacciones típicas, "throughput" o cantidad de procesos por unidad de tiempo (en modalidad batch), utilización de centros de servicio (CPU, discos, canales, etc.) y largo de colas en los distintos centros.

Uno de los aspectos más importantes de esta metodología es la capacidad de predecir el impacto en el desempeño del sistema de eventuales modificaciones al hardware (por ej., cambiar a

una CPU más rápida o agregar una segunda CPU, agregar un nuevo canal y redistribuir los discos) y variaciones de la carga (por ej., incremento de la tasa de llegada de transacciones, procesamiento de nuevas aplicaciones y redistribución de archivos en los discos). Al mismo tiempo, la evaluación de un sistema en funcionamiento permite la identificación del "cuello de botella" (unidad más cargada que impide un mayor nivel de procesamiento en el sistema), la utilización de las unidades y la identificación de los tipos de procesos que consumen más recursos, entre otros.

En este artículo se describe un modelo de desempeño para un sistema computacional en operación, su validación y utilización para la predicción del desempeño ante eventuales cambios al sistema y a la carga de trabajo. El sistema computacional estudiado consta de un computador DEC-1091 con 10.7 MBytes de memoria principal, 7.8 Gbytes de memoria secundaria, 2 procesadores de comunicaciones, 187 terminales y 98 impresoras distribuidos a lo largo de todo Chile.

En la sección 2 se entrega una breve introducción a los modelos de desempeño basados en redes de espera y su utilización para la predicción del desempeño. En la sección 3 se describe, en líneas generales, el sistema computacional en cuestión y se define para él un modelo base de desempeño. Se presentan estadísticas de utilización de CPU y discos, tasas de llegada de transacciones y consumo de recursos por procesos, obtenidas de mediciones efectuadas por los monitores de software del sistema. En el mismo capítulo se indica como estas estadísticas se utilizan para caracterizar la carga de trabajo en el sistema y producir los datos de entrada al modelo.

En la sección 4 se presenta la validación del modelo, comparando las mediciones de utilización de CPU y discos contra las estimaciones del modelo. En la sección 5 se presenta un análisis de modificaciones donde utilizamos el modelo para predecir el desempeño del sistema frente a variaciones de tasas de llegada de transacciones, cambio de CPU y cambios en la modalidad de procesos.

2. CONSTRUCCION Y USO DE UN MODELO DE DESEMPEÑO

En esta sección se describen los componentes básicos de un modelo de desempeño y su relación con los componentes de un sistema computacional de producción. En seguida, se indica como se obtienen los distintos parámetros del modelo a partir de información de monitoreo obtenida durante un período de observación del sistema. Finalmente, se indica como se utiliza el modelo para la predicción del desempeño frente a eventuales variaciones de la carga de trabajo y variaciones de los componentes del sistema.

2.1 Componentes de un modelo de red espera

Un modelo de red de espera consiste de los siguientes componentes:

(1) *centros de servicio* , que representan a los procesadores (CPUs) y dispositivos, tales como discos, canales y controladores, que al ser compartidos por distintos procesos producen congestión, formándose colas de espera;

(2) *clases de procesos* , que representan las tareas (transacciones o jobs) con características similares de consumo de recursos; y

(3) *dominios* , que indican la competencia por recursos (como la memoria principal) de las transacciones dentro de una clase, y entre distintas clases.

La definición de un modelo de red de espera, de un sistema computacional dado, incluye la identificación de todos los centros de servicio, las clases de procesos y sus respectivos dominios, junto con los valores de los parámetros asociados a cada componente. Estos parámetros incluyen las características de servicio de los centros y la carga de trabajo impuesta por las clases de procesos en cada centro de servicio.

Utilizando un software de evaluación de redes de espera (por ej: MAP [Lazo84]) un estudio de modelación del desempeño de un sistema computacional en operación, comprende tres etapas, a decir:

(1) creación de un modelo base,
(2) validación del modelo base, y
(3) análisis de modificaciones o proyecciones

Un modelo base debe representar fielmente al sistema actual con una carga de trabajo promedio para uno o más intervalos de observación (turnos) considerado para las mediciones del sistema. La validación del modelo base implica el asegurar que las medidas de rendimiento que arroja el modelo sean correctas.

El análisis de modificaciones se refiere al uso del modelo para la predicción (o proyección) del desempeño del sistema ante eventuales variaciones de la carga de trabajo y del sistema mismo (hardware y software). A continuación describimos estas tres etapas.

2.2 Creación de un modelo base

Para crear un modelo base, se siguen los siguientes pasos:

(1) medición del sistema computacional, mediante un monitor de sistema, durante períodos de carga significativa;
(2) definición de los componentes del modelo (centros de servicio, clases de procesos y dominios de memoria), identificando aquellos procesos que tienen un mayor consumo de recursos;
(3) utilización de las mediciones entregadas por el monitor del sistema para dar valores a los parámetros de los componentes del modelo.

Medición del sistema

Las mediciones del sistema, obtenidas por medio de un monitor, deben ser efectuadas en un período significativo que incluya una buena muestra de todos los tipos de procesos. Típicamente, se considera un período de una o dos horas de duración cuando la carga alcanza valores peak.

Las mediciones deben incluir estadísticas de uso de CPU y discos por cada tipo de proceso, número de ejecuciones de cada tipo de proceso, utilización global de CPU y discos, tiempos de respuesta por proceso, número de terminales activos y tiempo de retardo entre interacciones, y razónes de "swapping" y "paging" por proceso, entre otras.

Identificación de centros de servicio

Generalmente, los centros de servicio que se definen en el modelo corresponden a las unidades de CPU, discos (incluyendo canales, controladores y strings) y terminales activos del sistema real. Para la definición de los centros de servicio que representan CPUs se debe indicar si la disciplina de encolamiento es normal (por ej., FIFO o time-sharing) o si utiliza prioridades. Para los centros que representan discos se asume una disciplina FIFO. Para los centros que representan terminales se define un tiempo de retardo promedio entre interacciones que ingresa el usuario.

Cuando se desea determinar con precision la congestión producida por los accesos a las unidades de disco, es necesario que la estructura de canales y controladores, que conecta los discos con la(s) CPU(s), sea representada en detalle en el modelo.

Definición de clases de procesos

Para la definición de las clases de procesos se deben identificar y representar aquellos procesos que consumen la mayor cantidad de recursos. Una clase de procesos está constituida por jobs que tienen un origen común y que poseen consumos de servicio similares. Para cada clase de procesos se define el consumo promedio (segundos) que una instancia del proceso impone en la CPU y en cada disco, en base a las mediciones del monitor.

Las clases de proceso, las cuales representan la carga de trabajo en el sistema, pueden ser de tipo transacción, batch o time-sharing ([Lazo84]). Una clase de proceso tipo transacción puede ser un modulo de software (como por ej. un conjunto de rutinas para procesar una cédula de identidad, o un editor de texto) con una cierta tasa de llegada (número de ejecuciones por unidad de tiempo medidas durante el período de observación). Una clase de proceso tipo batch representa procesos cuya única limitante es el nivel de multiprogramación impuesto por el sistema. Una clase tipo time-sharing representa una carga de trabajo impuesta por un número de terminales activos con un tiempo de retardo promedio entre interacciones.

En un estudio de modelación típico, se representan como clases de procesos en el modelo sólo aquellos jobs que son más frecuentemente utilizados y/o que consumen la mayor cantidad de recursos. Aquellos otros procesos que individualmente no son muy significativos, son agrupados en una clase "otros" con un consumo promedio que representa la suma de los consumos individuales. Se entiende que el interés del estudio es observar cuidadosamente aquellos procesos críticos que imponen una mayor carga en el sistema.

Identificación de dominios de memoria

La cantidad disponible de memoria principal impone al menos dos grandes limitaciones al desempeño del sistema: un nivel máximo de multiprogramación y un nivel de overhead producido por paging y swapping. Ambas limitaciones pueden ser representadas en un modelo de desempeño.

La limitación en el nivel de multiprogramación del sistema se representa por medio de dominios de memoria. Un dominio tiene asociada una capacidad que indica el número de jobs que pueden estar residentes simultaneamente en memoria. Una vez que un dominio está lleno, otros jobs que llegan requiriendo memoria tiene que esperar en una cola de memoria asociada con ese dominio. En el modelo se puede representar un número arbitrario de dominios, de tal modo que cada clase de proceso se define como asociada a uno (y sólo uno) de estos dominios.

El segundo efecto de una memoria principal limitada se incluye en el modelo por medio de la especificación de razónes de paging y swapping para cada clase de proceso, según las mediciones entregadas por el monitor del sistema.

2.3 Validación del modelo base

Una vez completado el proceso anterior, la definición de componentes y sus parámetros son ingresados al software de evaluación de redes de espera para obtener estimaciones de desempeño tales como el "throughput" (rendimiento del sistema) y tiempos de respuesta para las distintas clases de transacciones.

Enseguida, las estimaciones del modelo son comparadas con las mediciones recogidas por el monitor del sistema durante el período de observación. El modelo base se considera validado una vez que las estimaciones proporcionadas por el modelo coinciden con las mediciones reales observadas directamente en el sistema.

La construcción del modelo base validado, además de ser la etapa previa a la de predicción del desempeño, es un excelente ejercicio en simismo. Durante esta primera etapa se puede observar las características de las transacciones más frecuentemente utilizadas y aquellas que consumen una mayor cantidad de recursos, la utilización de los centros de servicio, la unidad "cuello de botella" en el sistema, etc.

2.4 Análisis de modificaciones

Una vez que se obtiene un modelo base validado, la etapa de análisis de modificaciones (proyección) involucra efectuar cambios en le definición de componentes del modelo y sus parámetros y obtener del software de modelación las estimaciones del futuro desempeño (por ej., tiempos de respuestas) correspondientes. Las modificaciones al modelo utilizando el software de modelación son fáciles de realizar y las mediciones estimadas son entregadas en cuestión de segundos.

Típicamente, en esta etapa se responden preguntas tales como, ¿cuál es el efecto en el desempeño del sistema de:

- cambiar la CPU por una x veces más rápida que la actual?
- agregar una segunda CPU?
- agregar un nuevo canal al subsistema de Entrada/Salida?
- aumentar x veces la memoria principal?
- agregar n nuevas líneas de terminales?
- redistribuir los carga en los discos?
- aumentar la tasa de llegada de un cierto tipo de proceso?
- procesar una cierta carga batch en otro turno de trabajo?
- definir nuevas prioridades de procesamiento?

Para facilitar la etapa de proyección de este estudio, se ha construído un software front-end para la preparación de parámetros de entrada para el software de evaluación (MAP). Este software permite especificar en forma muy simple y rápida la carga en los archivos (operaciones de I/O) del sistema en estudio y la distribución de archivos en los distintos discos.

3. DEFINICION DEL MODELO BASE DEL SISTEMA EN ESTUDIO

En esta sección entregamos una breve descripción del sistema computacional en estudio y definimos el modelo base de desempeño que lo representa. Junto a esto, identificamos la carga de trabajo en el sistema y su parametrización en el modelo.

3.1 Descripción general del sistema computacional

El sistema computacional bajo estudio está compuesto de un procesador DEC-1091, con 10.7 Mbytes de memoria principal y provee de acceso a aproximadamente 200 terminales.

El almacenamiento secundario está compuesto por 3 discos RP06 (176 Mb cada uno), y 14

discos RP20 (483 Mb cada uno). Los discos RP06 están conectado a un canal de acceso, y los discos RP20 tienen dos canales de acceso, cada uno administrado independientemente por un controlador (uno controla el acceso a 8 discos RP20 y el otro controla el acceso a 6 discos RP20).

Los procesos que interactúan en el sistema computacional consisten fundamentalmente de procesos de tipo transaccional, administrados por un sistema controlador de mensajes. Este sistema permite el control de las comunicaciones entre la red de terminales y los programas de aplicación. Entre estos últimos cabe destacar a los programas de procesamiento de mensajes (MPP) que son los que demandan el mayor uso de recursos dentro del sistema. Además, existen otros utilitarios que realizan labores de administración y apoyo a otros sistemas.

3.2 Definición del modelo de red de espera

En la primera etapa de este estudio de evaluación de desempeño se ha construido un modelo abierto (transaccional) que no incluye explícitamente el manejo de terminales (ver figura 3.1). Sin embargo, se realizaron las mediciones necesarias (tiempo de respuesta a los terminales, throughput del controlador de líneas, etc), para la modelación de la red de terminales, en una segunda fase de este estudio.

En el modelo base de desempeño del sistema computacional se representaron explícitamente, como componentes de hardware, la CPU (DEC-1091), los 3 discos RP06 y los 14 discos RP20. Junto a esto se incluyeron los 3 canales con sus respectivos controladores y strings.

Es importante notar que en el modelo no se han incluído dominios de memoria, dado que los procesos que interactúan en este sistema computacional no están limitados por memoria principal. Cada proceso ocupa una partición específica de memoria, la que se mantiene mientras el proceso esté activo.

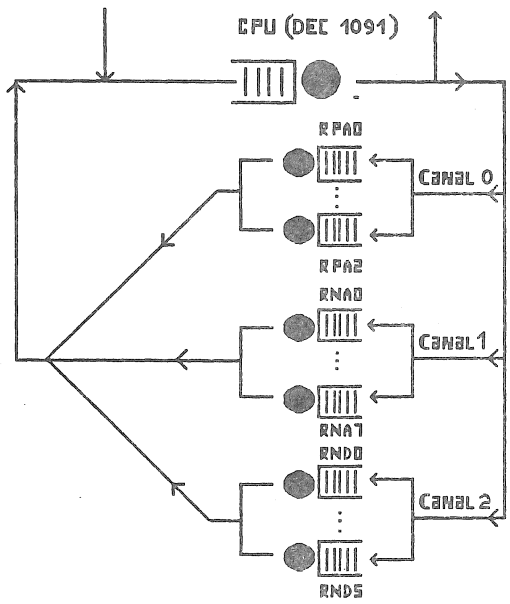


Figura 3.1 Modelo de Red de Espera para el Sistema en Estudio

En la modelación se caracterizó en forma desagregada la demanda de servicio y tasa de llegada sólo de aquellas clases de procesos que en la actualidad aportan más de un 90% de utilización de los distintos centros de servicio. En este caso se consideraron 7 tipos de procesos MPP, llamados Canmpp, Leempp, Rvmmpp, Filmpp, Cidmpp, Matmpp y Defmpp. Los procesos MPP restantes fueron agrupados y representados como una sóla clase llamada Otrmpp con una tasa de llegada y consumo promedio, de tal modo de representar el consumo del resto de los procesos.

Cabe hacer notar que existen otros procesos que están interactuando en el sistema, tales como subprocesos de algunas clases MPP y otros programas administrativos que no fueron detectados por el monitor de programas pero si son considerados por el monitor de utilización de CPU y discos. Estos procesos son representados en el modelo como una clase adicional, denominada clase BACKG.

3.3 Estadísticas de uso del sistema

Las estadísticas de uso del sistema computacional, se refieren a datos inherentes al sistema computacional (cantidad de transacciones, tiempos de respuesta, demanda de servicio, etc.), obtenidos ya sea mediante observación directa (p.ej. cronómetro), a través de monitores de software o en algunos casos mediante referencia de catálogos (p.ej. tiempos de acceso a disco). Estos datos constituyen la base para la implementación del modelo del sistema y su posterior validación.

En relación al perfil de uso del sistema, se cuantificó por separado la carga de trabajo de cinco períodos del día, que representan los distintos perfiles de carga (turnos de trabajo) a que está sometido el sistema (ver tabla 3.1).

Todas las clases de procesos observados durante la semana de mediciones son claramente de tipo transaccional. Es decir, sus cargas en el sistema se pueden representar en base a tasas de llegada y demandas de servicio en la CPU, discos y canales.

	TURNO 1	TURNO 2	TURNO 3	TURNO 4	TURNO 5
Horario	23:30-05:30	07:00-13:00	13:00-17:30	17:30-19:00	19:00-23:00
Tiempo (segs)	(21600)	(21600)	(16200)	(5400)	(21600)

Tabla 3.1 Períodos a Considerar en la Modelación

3.4 Porcentaje de uso de CPU y overhead

Además de caracterizar la carga de trabajo que ejercen las distintas clases de procesos en el sistema, se midió el perfil de carga global a que está sometido el sistema, en términos de utilización de CPU, durante cada turno. Adicionalmente, se midió el overhead inherente a la actividad del sistema operativo (porcentaje de CPU utilizado por el sistema operativo).

El porcentaje de uso de CPU es un dato muy útil para la validación del modelo. Por otro lado, conociendo el overhead del sistema, se tienen básicamente dos opciones: considerar esta carga como sometida por una clase especial (la clase overhead) o bien distribuir el overhead porcentualmente, de acuerdo al uso de CPU, en cada una de las clases de procesos consideradas en la modelación. En esta modelación se procedió a considerar la segunda opción ya que aparte de ser más representativo de la realidad, facilita el proceso involucrado durante la fase de análisis de modificaciones en el modelo.

En las figuras 3.2 y 3.3 se muestra la utilización de CPU y el overhead soportado por el sistema (a intervalos de media hora) para los turnos 2 y 3. Estos valores y el monitoreo (WR) de uso de CPU por los programa MPP (ver punto 4.6) proporcionan antecedentes para estimar la demanda de CPU de la clase BACKG.

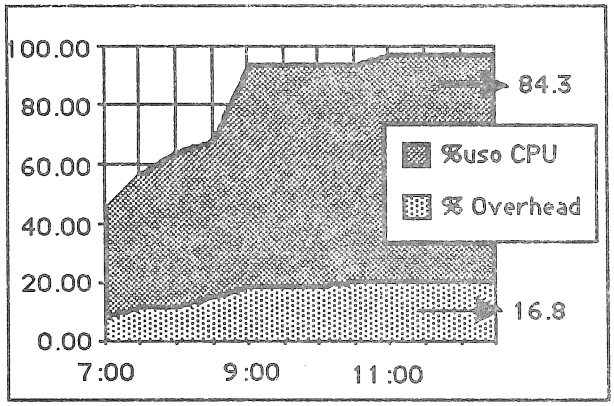


Figura 3.2 Utilizacion CPU - Turno 2

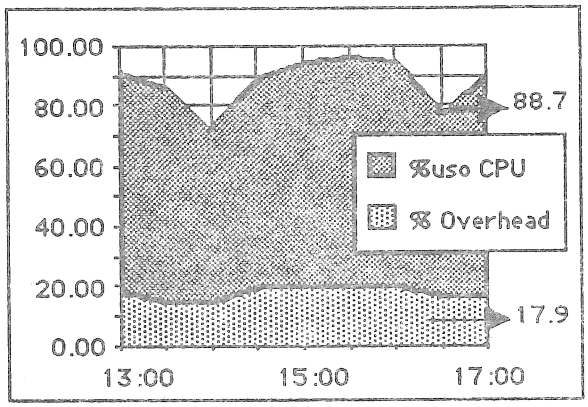


Figura 3.3 Utilización CPU - Turno 3

3.5 Tasa de llegada de transacciones

La tasa de llegada es una medida de la cantidad de requerimientos (transacciones) que llegan al sistema por unidad de tiempo. Para estimar este parámetro se midió el total de transacciones de cada clase de procesos, en cada uno de los turnos.

Si TOTR(p,t) es el total de transacciones de la clase de proceso p en el turno t, y T(t) la cantidad de tiempo del turno t, entonces la tasa de llegada promedio, en transacciones por segundo, para la clase de proceso p en el turno t está dada por:

$$Tlle\ (p,t) = TOTR\ (p,t) / T(t) \qquad (3.1)$$

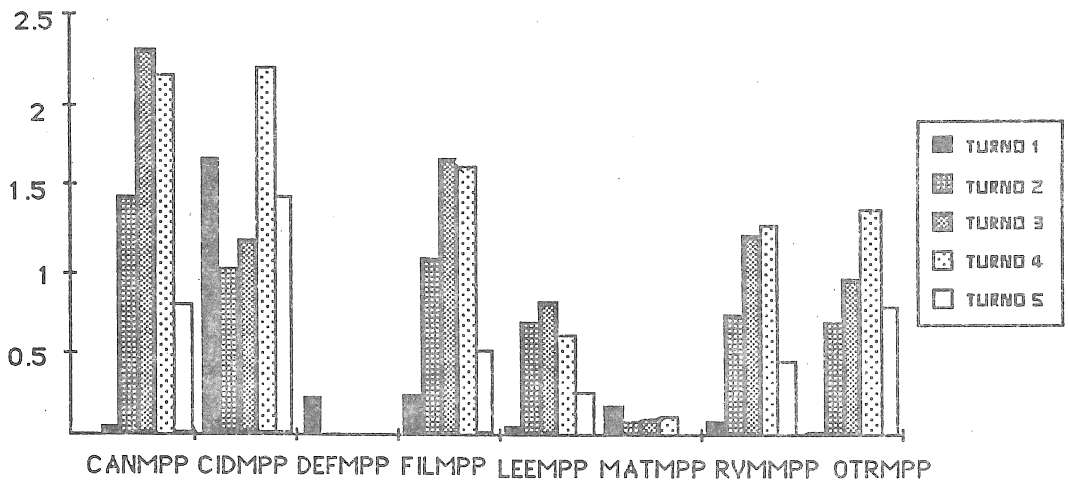


Figura 3.4 Tasa de Llegada (Transacciones/seg.) por clase de proceso

En la figura 3.4 se ilustran los valores de Tlle, obtenidos directamente del monitor de programas. Estos valores se incorporan directamente como datos de entrada al modelo.

3.6 Demanda de servicio en los discos

La demanda de servicio que ejerce una clase de proceso p, en un disco d, para un turno t, está dada por:

$$Ddsk(p,d,t) = FIO(p,d,t) * Tac(d,t) \quad (3.2)$$

Donde, FIO(p,d,t) es la fracción de accesos al disco d realizados por una instancia promedio de la clase de procesos p, en el turno t; y Tac(d,t) es el tiempo promedio de acceso al disco d, para el turno t. Nótese que Tac es dependiente del turno, dado que el perfil de requermientos a los discos es diferente en cada turno, lo que influye en la fracción del número de seeks por I/O. En fecto, Tac está dado por:

$$Tacc\ (d,t) = \mu(d,t) * T_{seek}(d) + T_{lat}(d) + T_{trans}(d) \quad (3.3)$$

Donde, $\mu(d,t)$ es el número de seeks por I/O en el disco d para el turno t, $T_{seek}(d)$ y $T_{lat}(d)$ son los tiempos de seek y latencia del disco d respectivamente, y $T_{trans}(d)$ es el tiempo de transferencia de un bloque de datos entre el disco d y la memoria. El valor de μ se midió directamente mediante monitores especiales (es normalmente igual a 1 en los discos RNA y fluctua entre 0.3 y 0.6 para los discos RPA). Por otro lado, FIO está dado por:

$$FIO(p,d,t) = f(p,d,t) * [TOTIO(p,t) / TOTR(p,t)] \quad (3.4)$$

Donde, TOTIO(p,t), TOTR(p,t) y $f(p,d,t)$ corresponden al total de accesos a disco, al total de transacciones y al porcentaje del total de accesos a disco que le corresponden al disco d, respectivamente, de la clase p en el turno t.

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEENOM	MATMPP	RVMMPP	OTRMPP
RPA0	5.0	52.0	0.0	6.0	10.0	95.0	22.0	42.0
RPA1	3.0	33.0	0.0	5.0	7.0	49.0	12.0	34.0
RPA2	5.0	62.0	0.0	7.0	17.0	104.0	11.0	39.0
RNA0	1.0	12.0	4.0	2.0	4.0	10.0	0.0	22.0
RNA1	3.0	6.0	10.0	1.0	4.0	11.0	36.0	10.0
RNA2	3.0	12.0	0.0	19.0	76.0	0.0	0.0	0.0
RNA3	6.0	30.0	96.0	18.0	71.0	46.0	8.0	27.0
RNA4	3.0	11.0	1.0	0.0	0.0	0.0	15.0	0.0
RNA5	2.0	28.0	1.0	2.0	8.0	25.0	15.0	36.0
RNA6	75.0	21.0	75.0	32.0	12.0	195.0	0.0	35.0
RNA7	9.0	21.0	0.0	13.0	7.0	0.0	0.0	0.0
RND0	12.0	10.0	95.0	1.0	5.0	139.0	15.0	9.0
RND1	2.0	91.0	2.0	2.0	7.0	20.0	0.0	26.0
RND2	2.0	0.0	0.0	0.0	0.0	0.0	38.0	0.0
RND3	3.0	15.0	31.0	4.0	11.0	91.0	0.0	22.0
RND4	4.0	5.0	2.0	1.0	4.0	10.0	8.0	9.0
RND5	12.0	3.0	22.0	12.0	2.0	86.0	0.0	13.0

Tabla 3.2 Demanda de Servicio (milisegs) en cada disco (Turno 3)

Los valores de TOTIO se obtuvieron directamente a traves de mediciones, y los valores de f fueron estimados en base al conocimiento del perfil de acceso a los archivos por cada clase de proceso y la localización de estos archivos en los discos del sistema (no se contó al momento de la modelación con un monitor que permitiera obtener directamente el factor f*TOTIO).

Por otro lado, y a fin de disponer de un elemento de validación de las estimaciones anteriores, se midió directamente, en forma independiente de cada clase de proceso, el número total de accesos a cada uno de los discos en cada turno.

Para facilitar el ingreso de datos, modificaciones al perfil de acceso a los discos, y la obtención de datos de validación, se construyó un software front-end, que además estima la demanda de servicio (Ddsk(p,d,t)) en los discos. En la tabla 3.2 se muestran estos valores, sólo para el turno 3. No se incluye la clase BACKG, cuya demanda de servicio fue calculada por el software front-end y modificada por MAP para incluir el tiempo de contención en los canales.

3.7 Demanda de Servicio en la CPU

La demanda de servicio promedio en la CPU que ejerce una instancia de una clase de proceso p, en el turno t, se estimó como:

$$Dcpu(p,t) = [Tcpu(p,t) + Tovh(p,t)]/ TOTR(p,t) \qquad (3.5)$$

En que, Tcpu(p,t) y Tovh(p,t) corresponden al tiempo útil de CPU y al overhead de CPU respectivamente, que requiere, en promedio, la clase de proceso p durante el turno t. TOTR(p,t) es el total de transacciones de la clase de proceso p en el turno t (ver punto 3.5).

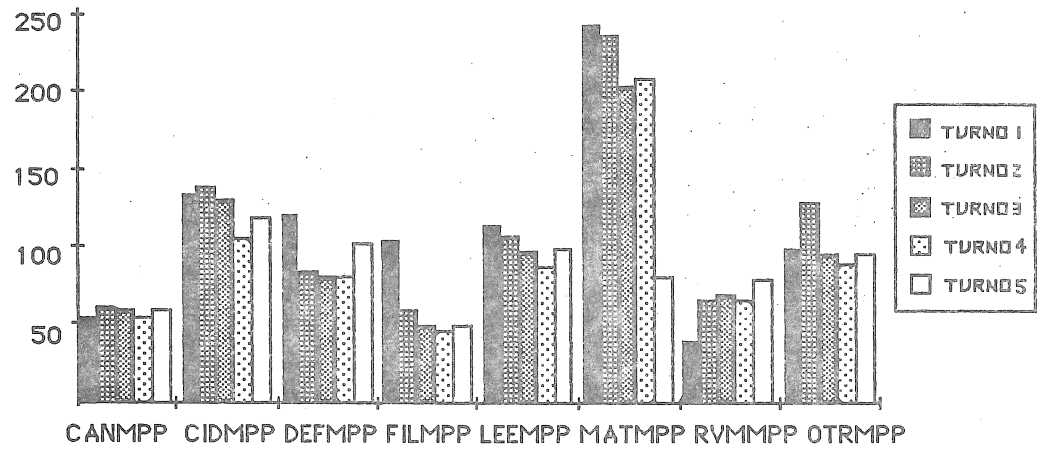


Figura 3.5 Demanda Total de CPU en milsegundos

El valor de Tcpu se midió directamente mediante monitores. El valor de Tovh se calculó como el porcentaje promedio de overhead del turno (ver figuras 3.2 y 3.3) multiplicado por la cantidad de tiempo del turno. En la figura 3.5se ilustran los valores correspondientes a la ecuación 3.5.

Las demandas de servicio indicadas en la figura 3.5, junto con las demandas de servicio en los discos de la sección anterior, y la carga en el sistema (tasas de llegada) son los parámetros (de entrada

para el software de modelación) que representan la carga de trabajo de los programas MPP. El consumo de CPU de la clase BACKG ha sido estimado en base a la diferencia entre el total de uso de CPU entregado por el monitor de CPU y el total de uso de CPU entregado por el monitor de programas. En forma similar se ha estimado el consumo de disco de la misma clase BACKG.

4. VALIDACION DEL MODELO BASE

En esta sección presentamos las estimaciones de desempeño obtenidas de la evaluación (por medio del software MAP) del modelo descrito en la sección 3. Se discuten, en este artículo, sólo las estimaciones de desempeño para los turnos 3 y 4. (En [Casa87] se presenta una discusión completa de este estudio de modelación.) Junto a esto, se indican los resultados de validación del modelo frente a medidas de uso de CPU y accesos a las distintas unidades de disco.

En la sección anterior se indicó que mediante un software front-end, y mediciones de acceso a disco, se realizó una validación de las estimaciones de accesos a disco por cada una de las clases de transacciones. En el turno 3, las estimaciones de acceso a discos tienen un margen de error menor al 5%. En los otros turnos el margen de error es menor al 20%. Esta precisión de las estimaciones de accesos a disco es considerada adecuada, considerando los objetivos que persigue el estudio. Con este margen de error, podemos esperar estimaciones de desempeño del software de modelación cercanas a la realidad.

En las tablas 4.1 y 4.2 se muestran los resultados de desempeño más importantes del software de modelación, para los turnos 3 y 4 respectivamente.

El primer informe de desempeño "Response Times" en la tabla 4.1, indica para cada (instancia de) transacción, el tiempo de respuesta en segundos. Nótese que el mayor tiempo de respuesta es para MATMPP y CIDMPP. Este resultado era esperado puesto que ambos tipos de procesos consumen más recursos que los otros procesos. Es importante destacar que estos tiempos de respuesta no incluyen el delay en el sistema de time-sharing, sino que sólo el tiempo de residencia en el sistema central (DEC-1091).

El informe de desempeño "Device Utilizations" (tabla 4.1) muestra el porcentaje de utilización de la CPU y discos durante el turno 3. El informe indica una utilización total de la CPU del 88.9%. Esta estimación puede compararse con las mediciones entregadas por el monitor de CPU (ver figura 4.3), que indican una utilización del 88.7%. Se tiene entonces un error de menos del 1%, lo cual es indicativo de la precisión del modelo. La misma precisión se repite para los otros cuatro turnos. La utilización de la CPU, que indican tanto el modelo como el monitor de CPU, muestran que el sistema está cercano a la saturación.

En el mismo informe se puede apreciar que la utilización de los discos es bastante baja con respecto a la utilización de la CPU. Definitivamente, el "cuello de botella" del sistema es la CPU. El disco más cargado es el RNA6, el cual tiene aproximadamente un 33% de utilización. Le siguen los discos RPA0, RPA3, RNA3 y RND1 todos con menos de un 25% de utilización. Por consiguiente, cualquier redistribución de archivos en los discos, con el fin de uniformar la carga en el subsistema de I/O, no tendría mayor efecto en el desempeño del sistema debido a la gran contención producida en la CPU actual. En todo caso, una vez que se haya cambiado la CPU por una más rápida, se podría obtener una mejor utilización del subsistema de discos haciendo una redistribución de archivos. El software front-end puede ser utilizado para evaluar distintas distribuciones de archivos en discos.

El informe "Device Queue Lengths" (tabla 4.1) muestra el largo promedio (número de transacciones) de las colas de espera en cada centro de servicio. Naturalmente, la cola más larga se produce en la CPU. El informe "Device Residence Times" muestra los tiempos promedio de residencia (en segundos) para cada tipo de transacción en cada centro de servicio. El tiempo de residencia en un centro de servicio incluye tanto el tiempo de servicio como el de espera en la cola.

RESPONSE TIME:

CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVIMPP	OTRMPP	BACKG
0.7278	1.6793	1.1434	0.6058	1.1749	2.9361	0.8339	1.2487	0.9491

DEVICE UTILIZATIONS

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVIMPP	OTRMPP	BACKG	TOTAL
CPU	0.1358	0.2058	0.0006	0.0822	0.0775	0.0203	0.0834	0.0875	0.1960	0.8890
RPA0	0.0116	0.0823	0.0000	0.0099	0.0080	0.0095	0.0264	0.0390	0.0564	0.2432
RPA1	0.0069	0.0521	0.0000	0.0083	0.0056	0.0049	0.0144	0.0315	0.0141	0.1377
RPA3	0.0115	0.0978	0.0000	0.0115	0.0135	0.0103	0.0131	0.0361	0.0120	0.2059
RNA0	0.0023	0.0190	0.0000	0.0033	0.0032	0.0010	0.0000	0.0204	0.0020	0.0512
RNA1	0.0070	0.0095	0.0001	0.0017	0.0032	0.0011	0.0433	0.0093	0.0040	0.0792
RNA2	0.0069	0.0190	0.0000	0.0314	0.0605	0.0000	0.0000	0.0000	0.0000	0.1178
RNA3	0.0139	0.0474	0.0008	0.0297	0.0565	0.0046	0.0096	0.0250	0.0040	0.1914
RNA4	0.0069	0.0174	0.0000	0.0000	0.0000	0.0000	0.0160	0.0000	0.0020	0.0443
RNA5	0.0046	0.0442	0.0000	0.0033	0.0064	0.0025	0.0180	0.0333	0.0000	0.1124
RNA6	0.1734	0.0331	0.0006	0.0528	0.0095	0.0194	0.0000	0.0324	0.0060	0.3273
RNA7	0.0208	0.0332	0.0000	0.0215	0.0056	0.0000	0.0000	0.0000	0.0020	0.0831
RND0	0.0277	0.0158	0.0008	0.0016	0.0040	0.0138	0.0179	0.0083	0.0020	0.0918
RND1	0.0046	0.1432	0.0000	0.0033	0.0055	0.0020	0.0000	0.0240	0.0000	0.1827
RND2	0.0046	0.0000	0.0000	0.0000	0.0000	0.0000	0.0454	0.0000	0.0000	0.0500
RND3	0.0069	0.0236	0.0002	0.0066	0.0087	0.0090	0.0000	0.0203	0.0000	0.0755
RND4	0.0092	0.0079	0.0000	0.0016	0.0032	0.0010	0.0096	0.0063	0.0000	0.0408
RND5	0.0277	0.0047	0.0002	0.0198	0.0016	0.0085	0.0000	0.0120	0.0000	0.0745

DEVICE QUFUE LENGTHS:

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVIMPP	OTRMPP	BACKG	TOTAL
CPU	1.2235	1.8547	0.0058	0.7403	0.6986	0.1829	0.7513	0.7885	1.7663	8.0119
RPA0	0.0152	0.1077	0.0000	0.0130	0.0104	0.0124	0.0345	0.0510	0.0738	0.3180
RPA1	0.0080	0.0601	0.0000	0.0095	0.0064	0.0056	0.0166	0.0363	0.0162	0.1587
RPA3	0.0145	0.1225	0.0000	0.0145	0.0169	0.0130	0.0165	0.0452	0.0151	0.2581
RNA0	0.0024	0.0199	0.0000	0.0035	0.0033	0.0010	0.0000	0.0213	0.0021	0.0536
RNA1	0.0075	0.0102	0.0001	0.0018	0.0034	0.0012	0.0465	0.0100	0.0043	0.0851
RNA2	0.0078	0.0214	0.0000	0.0354	0.0681	0.0000	0.0000	0.0000	0.0000	0.1326
RNA3	0.0171	0.0582	0.0009	0.0365	0.0694	0.0056	0.0118	0.0307	0.0049	0.2352
RNA4	0.0072	0.0181	0.0000	0.0000	0.0000	0.0000	0.0187	0.0000	0.0021	0.0461
RNA5	0.0052	0.0495	0.0000	0.0037	0.0071	0.0028	0.0201	0.0373	0.0000	0.1258
RNA6	0.2560	0.0489	0.0009	0.0780	0.0141	0.0286	0.0000	0.0478	0.0089	0.4832
RNA7	0.0226	0.0360	0.0000	0.0233	0.0060	0.0000	0.0000	0.0000	0.0022	0.0900
RND0	0.0304	0.0173	0.0008	0.0018	0.0044	0.0151	0.0197	0.0091	0.0023	0.1008
RND1	0.0056	0.1748	0.0000	0.0040	0.0068	0.0024	0.0000	0.0293	0.0000	0.2230
RND2	0.0048	0.0000	0.0000	0.0000	0.0000	0.0000	0.0476	0.0000	0.0000	0.0525
RND3	0.0075	0.0255	0.0003	0.0071	0.0094	0.0097	0.0000	0.0219	0.0000	0.0814
RND4	0.0096	0.0082	0.0000	0.0017	0.0033	0.0010	0.0099	0.0086	0.0000	0.0424
RND5	0.0298	0.0051	0.0002	0.0213	0.0017	0.0092	0.0000	0.0129	0.0000	0.0802

DEVICE RESIDENCE TIMES

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVIMPP	OTRMPP	BACKG
CPU	0.5317	1.1806	0.7300	0.4506	0.8832	1.8474	0.6308	0.8561	0.8832
PPA0	0.0066	0.0685	0.0000	0.0079	0.0132	0.1252	0.0290	0.0554	0.0369
RPA1	0.0035	0.0382	0.0000	0.0058	0.0081	0.0568	0.0139	0.0394	0.0081
PPA3	0.0063	0.0780	0.0000	0.0088	0.0214	0.1308	0.0138	0.0491	0.0075
RNA0	0.0011	0.0126	0.0042	0.0021	0.0042	0.0105	0.0000	0.0232	0.0011
RNA1	0.0033	0.0065	0.0109	0.0011	0.0043	0.0119	0.0391	0.0109	0.0022
RNA2	0.0034	0.0136	0.0000	0.0215	0.0861	0.0000	0.0000	0.0000	0.0000
RNA3	0.0074	0.0371	0.1186	0.0222	0.0877	0.0568	0.0099	0.0333	0.0025
RNA4	0.0031	0.0115	0.0010	0.0000	0.0000	0.0000	0.0157	0.0000	0.0010
RNA5	0.0023	0.0315	0.0011	0.0023	0.0090	0.0281	0.0169	0.0405	0.0000
RNA6	0.1112	0.0311	0.1112	0.0475	0.0178	0.2892	0.0000	0.0519	0.0044
RNA7	0.0098	0.0229	0.0000	0.0142	0.0076	0.0000	0.0000	0.0000	0.0011
RND0	0.0132	0.0110	0.1046	0.0011	0.0055	0.1530	0.0165	0.0099	0.0011
RND1	0.0024	0.1113	0.0024	0.0024	0.0086	0.0245	0.0000	0.0318	0.0000
RND2	0.0021	0.0000	0.0000	0.0000	0.0000	0.0000	0.0400	0.0000	0.0000
RND3	0.0032	0.0162	0.0335	0.0043	0.0119	0.0984	0.0000	0.0238	0.0000
RND4	0.0042	0.0052	0.0021	0.0010	0.0042	0.0104	0.0083	0.0094	0.0000
RND5	0.0130	0.0032	0.0238	0.0130	0.0022	0.0929	0.0000	0.0140	0.0000
TOTAL	0.7278	1.6792	1.1434	0.6058	1.1749	2.9361	0.8339	1.2487	0.9491

CONTROLLER UTLIZATIONS:

CONTROL-0: 0.22; CONTROL-1: 0.24 CONTROL-2: 0.12

Tabla 4.1 RESULTADOS DE LA MODELACION PARA EL SISTEMA ACTUAL -TURNO -3

RESPONSE TIME:

CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG
0.6393	1.3983	1.1640	0.5639	1.0556	3.0192	0.8141	1.1378	1.8688

DEVICE UTILIZATIONS:

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG	TOTAL
CPU	0.1157	0.2331	0.0006	0.0751	0.0500	0.0216	0.0831	0.1202	0.1920	0.8915
RPA0	0.0325	0.0932	0.0000	0.0129	0.0070	0.0115	0.0275	0.0391	0.0263	0.2500
RPA1	0.0216	0.0598	0.0000	0.0096	0.0046	0.0062	0.0200	0.0323	0.0050	0.1592
RPA3	0.0344	0.1104	0.0000	0.0144	0.0110	0.0138	0.0137	0.0537	0.0100	0.2614
RNA0	0.0000	0.0177	0.0000	0.0016	0.0012	0.0005	0.0000	0.0174	0.0040	0.0424
RNA1	0.0043	0.0066	0.0001	0.0000	0.0012	0.0006	0.0462	0.0054	0.0010	0.0654
RNA2	0.0065	0.0221	0.0000	0.0273	0.0387	0.0000	0.0000	0.0000	0.0010	0.0955
RNA3	0.0086	0.0088	0.0008	0.0192	0.0277	0.0022	0.0087	0.0201	0.0020	0.0982
RNA4	0.0194	0.0199	0.0000	0.0000	0.0000	0.0000	0.0187	0.0000	0.0050	0.0629
RNA5	0.0043	0.0375	0.0000	0.0016	0.0040	0.0027	0.0187	0.0134	0.0060	0.0883
RNA6	0.0495	0.0353	0.0007	0.0385	0.0063	0.0207	0.0000	0.0268	0.0030	0.1807
RNA7	0.0151	0.0331	0.0000	0.0176	0.0017	0.0000	0.0000	0.0000	0.0000	0.0676
RND0	0.0193	0.0154	0.0008	0.0016	0.0029	0.0081	0.0087	0.0094	0.0070	0.0733
RND1	0.0150	0.2246	0.0000	0.0112	0.0127	0.0080	0.0000	0.0776	0.0060	0.3552
RND2	0.0043	0.0000	0.0000	0.0000	0.0000	0.0000	0.0460	0.0000	0.0000	0.0503
RND3	0.0022	0.0132	0.0003	0.0016	0.0029	0.0091	0.0000	0.0174	0.0040	0.0507
RND4	0.0065	0.0044	0.0000	0.0016	0.0012	0.0005	0.0087	0.0040	0.0000	0.0269
RND5	0.0086	0.0022	0.0002	0.0176	0.0000	0.0074	0.0000	0.0040	0.0000	0.0401

DEVICE QUEUE LENGTHS:

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG	TOTAL
CPU	1.0665	2.1483	0.0060	0.6918	0.4611	0.1994	0.7657	1.1082	1.7696	8.2164
RPA0	0.0427	0.1228	0.0000	0.0170	0.0092	0.0152	0.0363	0.0515	0.0346	0.3291
RPA1	0.0255	0.0705	0.0000	0.0114	0.0055	0.0073	0.0236	0.0381	0.0059	0.1877
RPA3	0.0464	0.1486	0.0000	0.0194	0.0148	0.0185	0.0184	0.0722	0.0135	0.3519
RNA0	0.0000	0.0184	0.0000	0.0017	0.0012	0.0005	0.0000	0.0181	0.0042	0.0441
RNA1	0.0046	0.0071	0.0001	0.0000	0.0012	0.0007	0.0491	0.0057	0.0011	0.0695
RNA2	0.0071	0.0243	0.0000	0.0300	0.0426	0.0000	0.0000	0.0000	0.0011	0.1051
RNA3	0.0095	0.0098	0.0009	0.0212	0.0306	0.0024	0.0096	0.0222	0.0022	0.1085
RNA4	0.0206	0.0211	0.0000	0.0000	0.0000	0.0000	0.0198	0.0000	0.0053	0.0669
RNA5	0.0047	0.0410	0.0000	0.0018	0.0044	0.0029	0.0204	0.0146	0.0066	0.0964
RNA6	0.0601	0.0429	0.0008	0.0467	0.0077	0.0251	0.0000	0.0326	0.0037	0.2197
RNA7	0.0161	0.0354	0.0000	0.0188	0.0018	0.0000	0.0000	0.0000	0.0000	0.0721
RND0	0.0208	0.0166	0.0009	0.0017	0.0031	0.0087	0.0094	0.0101	0.0076	0.0788
RND1	0.0232	0.3476	0.0001	0.0173	0.0196	0.0124	0.0000	0.1201	0.0093	0.5496
RND2	0.0045	0.0000	0.0000	0.0000	0.0000	0.0000	0.0483	0.0000	0.0000	0.0528
RND3	0.0023	0.0139	0.0003	0.0017	0.0030	0.0095	0.0000	0.0183	0.0042	0.0532
RND4	0.0066	0.0045	0.0000	0.0016	0.0012	0.0005	0.0069	0.0041	0.0000	0.0275
RND5	0.0089	0.0023	0.0002	0.0183	0.0000	0.0077	0.0000	0.0042	0.0000	0.0416

DEVICE RESIDENCE TIMES

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG
CPU	0.4977	0.9769	0.7465	0.4332	0.8018	1.9354	0.6175	0.6295	1.7696
RPA0	0.0199	0.0558	0.0000	0.0106	0.0159	0.1475	0.0292	0.0385	0.0346
RPA1	0.0119	0.0321	0.0000	0.0071	0.0095	0.0713	0.0190	0.0255	0.0059
RPA3	0.0216	0.0676	0.0000	0.0122	0.0257	0.1798	0.0149	0.0541	0.0135
RNA0	0.0000	0.0084	0.0042	0.0010	0.0021	0.0052	0.0000	0.0136	0.0042
RNA1	0.0021	0.0032	0.0118	0.0000	0.0021	0.0064	0.0396	0.0043	0.0011
RNA2	0.0033	0.0111	0.0000	0.0188	0.0740	0.0000	0.0000	0.0000	0.0011
RNA3	0.0044	0.0044	0.1164	0.0133	0.0532	0.0233	0.0078	0.0166	0.0022
RNA4	0.0096	0.0096	0.0011	0.0000	0.0000	0.0000	0.0160	0.0000	0.0053
RNA5	0.0022	0.0186	0.0011	0.0011	0.0077	0.0285	0.0164	0.0110	0.0066
RNA6	0.0281	0.0195	0.1012	0.0293	0.0134	0.2439	0.0000	0.0244	0.0037
RNA7	0.0075	0.0161	0.0000	0.0118	0.0032	0.0000	0.0000	0.0000	0.0000
RND0	0.0097	0.0076	0.1133	0.0011	0.0054	0.0841	0.0076	0.0076	0.0076
RND1	0.0108	0.1581	0.0077	0.0108	0.0341	0.1209	0.0000	0.0899	0.0093
RND2	0.0021	0.0000	0.0000	0.0000	0.0000	0.0000	0.0390	0.0000	0.0000
RND3	0.0011	0.0063	0.0348	0.0011	0.0053	0.0927	0.0000	0.0137	0.0042
RND4	0.0031	0.0021	0.0021	0.0010	0.0021	0.0051	0.0072	0.0031	0.0000
RND5	0.0042	0.0010	0.0240	0.0115	0.0000	0.0750	0.0000	0.0031	0.0000
TOTAL	0.6393	1.3983	1.1640	0.5639	1.0556	3.0192	0.8141	1.1378	1.8688

CONTROLLER UTILIZATIOS:

CONTROL-0: 0.25 CONTROL-1: 0.17 CONTROL-2: 0.14

Tabla 4.2 RESULTADOS DE LA MODELACION PARA EL SISTEMA ACTUAL -TURNO 4 -

Como era de esperarse, para todas las transacciones el mayor tiempo de residencia es en la CPU.

En el informe "Controler Utilizations" de la tabla 4.1 se muestran las cifras de utilización de los controladores. La mayor utilización (de 24%) se produce en el EJE1 que controla los discos RNA. Sin embargo, un 24% de utilización muestra que el canal y controlador correspondiente no producen mayor contención en los accesos a los discos RNA. De todos modos, se podría redistribuir mejor la carga en los discos, y la asignación de discos a canales, para obtener una utilización más uniforme de los 3 ejes de canales y controladores.

En la tabla 4.2 se puede apreciar que, al igual que para el turno 3, la utilización de los discos en el turno 4 es mucho menor que la utilización de la CPU. Los discos más utilizados en este turno son el RPA0, RPA3 y el RND1 con una utilización menor que 36%.

5. ANALISIS DE MODIFICACIONES Y PROYECCIONES

En esta sección se discuten las estimaciones de desempeño obtenidas, por medio del software de modelación, para algunas modificaciones tanto a los componentes del modelo como a los parámetros que representan la carga de trabajo en el sistema. Nuevamente, se discuten las predicciones de desempeño sólo para el turno 3, por ser este el turno más crítico en cuanto a saturación de la CPU. En todo caso, las predicciones para el turno 3 también son aplicables para los turnos 2 y 4, por ser estos tres turnos de características muy similares en cuanto a la carga de trabajo que presentan.

Cambio a Una CPU Más Rápida

En esta sección se observa el desempeño que tendría el sistema computacional si se cambia la CPU actual por una dos veces más rápida. Todos los demás componentes del sistema permanecen sin variación, al igual que la carga de trabajo en el sistema.

En tabla 5.1 se muestra el informe obtenido del software de modelación, para los mismos parámetros de desempeño descritos en la tabla 4.1.

En el primer informe de desempeño ("Response Times"), se puede observar una reducción de los tiempos de respuesta del orden del 65%. Por otro lado, el informe de desempeño ("Device Utilizations") indica una utilización total de la CPU del 44.45%. Esto representa una reducción del 50% de la utilización original de la CPU, tal como se esperaba.

Aumento de la Tasa de Llegada de la clase CIDMPP

En la tabla 5.2 se muestra el informe entregado por el software de modelación con los principales parámetros de desempeño, para el turno 3, con una tasa de llegada, para la clase CIDMPP, 1.5 veces mayor que la original. Todos los otros parámetros del modelo, incluyendo la CPU, permanecen sin modificación tal como son en el sistema real.

En el informe de desempeño "Response Times" se observa que los tiempos de respuesta para las distintas transacciones aumentan considerablemente. Esto se debe a que el sistema actual se encuentra muy próximo a la saturación.

En el informe "Device Utilizations" se muestra una utilización total de la CPU del 99.19%. Esta cifra indica que el sistema ya no puede tolerar más carga y lo más probable es que entre en estado

RESPONSE TIME:

CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG
0.2492	0.6165	0.4864	0.2002	0.3799	1.2732	0.2661	0.4781	0.1542

DEVICE UTILIZATIONS:

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG	TOTAL
CPU	0.0679	0.1029	0.0003	0.0411	0.0366	0.0101	0.0417	0.0437	0.0980	0.4423
RPA0	0.0110	0.0833	0.0000	0.0099	0.0080	0.0095	0.0261	0.0390	0.0564	0.2432
RPA1	0.0069	0.0521	0.0000	0.0083	0.0056	0.0049	0.0144	0.0315	0.0141	0.1377
RPA3	0.0115	0.0978	0.0000	0.0115	0.0135	0.0103	0.0131	0.0361	0.0120	0.2059
RNA0	0.0023	0.0190	0.0000	0.0033	0.0032	0.0010	0.0000	0.0204	0.0020	0.0512
RNA1	0.0070	0.0095	0.0001	0.0017	0.0032	0.0011	0.0437	0.0093	0.0040	0.0792
RNA2	0.0069	0.0190	0.0000	0.0314	0.0605	0.0000	0.0000	0.0000	0.0000	0.1178
RNA3	0.0139	0.0474	0.0008	0.0297	0.0565	0.0046	0.0096	0.0250	0.0040	0.1914
RNA4	0.0069	0.0174	0.0000	0.0000	0.0000	0.0000	0.0160	0.0000	0.0020	0.0443
RNA5	0.0046	0.0442	0.0000	0.0033	0.0064	0.0025	0.0180	0.0333	0.0000	0.1124
RNA6	0.1734	0.0331	0.0006	0.0528	0.0095	0.0194	0.0000	0.0324	0.0060	0.3273
RNA7	0.0208	0.0332	0.0000	0.0215	0.0056	0.0000	0.0000	0.0000	0.0020	0.0831
RND0	0.0127	0.0156	0.0006	0.0016	0.0040	0.0156	0.0137	0.0063	0.0020	0.0918
RND1	0.0046	0.1432	0.0000	0.0033	0.0055	0.0020	0.0000	0.0240	0.0000	0.1827
RND2	0.0046	0.0000	0.0000	0.0000	0.0000	0.0000	0.0454	0.0000	0.0000	0.0500
RND3	0.0069	0.0236	0.0002	0.0066	0.0087	0.0090	0.0000	0.0203	0.0000	0.0755
RND4	0.0092	0.0079	0.0000	0.0016	0.0032	0.0010	0.0096	0.0083	0.0000	0.0408
RND5	0.0277	0.0047	0.0002	0.0198	0.0016	0.0085	0.0000	0.0120	0.0000	0.0745

DEVICE QUEUE LENGTHS:

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG	TOTAL
CPU	0.1222	0.1852	0.0006	0.0739	0.0659	0.0183	0.0750	0.0788	0.1764	0.8002
RPA0	0.0152	0.1077	0.0000	0.0130	0.0104	0.0124	0.0345	0.0510	0.0738	0.3180
RPA1	0.0060	0.0601	0.0000	0.0095	0.0064	0.0056	0.0166	0.0363	0.0162	0.1587
RPA3	0.0145	0.1225	0.0000	0.0145	0.0169	0.0130	0.0165	0.0452	0.0151	0.2581
RNA0	0.0024	0.0199	0.0000	0.0035	0.0033	0.0010	0.0000	0.0213	0.0021	0.0535
RNA1	0.0075	0.0102	0.0001	0.0018	0.0034	0.0012	0.0465	0.0100	0.0043	0.0851
RNA2	0.0078	0.0214	0.0000	0.0354	0.0681	0.0000	0.0000	0.0000	0.0000	0.1326
RNA3	0.0171	0.0582	0.0009	0.0365	0.0694	0.0056	0.0118	0.0307	0.0049	0.2352
RNA4	0.0072	0.0181	0.0000	0.0000	0.0000	0.0000	0.0167	0.0000	0.0021	0.0461
RNA5	0.0052	0.0495	0.0000	0.0037	0.0071	0.0028	0.0201	0.0373	0.0000	0.1258
RNA6	0.2560	0.0489	0.0009	0.0780	0.0141	0.0286	0.0000	0.0478	0.0089	0.4632
RNA7	0.0226	0.0360	0.0000	0.0233	0.0060	0.0000	0.0000	0.0000	0.0022	0.0900
RND0	0.0304	0.0173	0.0008	0.0018	0.0044	0.0151	0.0197	0.0091	0.0022	0.1008
RND1	0.0056	0.1748	0.0000	0.0040	0.0068	0.0024	0.0000	0.0293	0.0000	0.2230
RND2	0.0048	0.0000	0.0000	0.0000	0.0000	0.0000	0.0476	0.0000	0.0000	0.0525
RND3	0.0075	0.0255	0.0003	0.0071	0.0094	0.0097	0.0000	0.0219	0.0000	0.0814
RND4	0.0096	0.0082	0.0000	0.0017	0.0033	0.0010	0.0099	0.0086	0.0000	0.0424
RND5	0.0298	0.0051	0.0002	0.0213	0.0017	0.0092	0.0000	0.0129	0.0000	0.0802

DEVICE RESIDENCE TIMES

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG
CPU	0.0531	0.1179	0.0729	0.0450	0.0352	0.1845	0.0630	0.0855	0.0882
RPA0	0.0066	0.0685	0.0000	0.0079	0.0132	0.1252	0.0299	0.0554	0.0369
RPA1	0.0035	0.0382	0.0000	0.0055	0.0081	0.0566	0.0159	0.0394	0.0081
RPA3	0.0063	0.0780	0.0000	0.0088	0.0214	0.1308	0.0138	0.0491	0.0075
RNA0	0.0011	0.0126	0.0042	0.0021	0.0042	0.0105	0.0069	0.0232	0.0011
RNA1	0.0033	0.0065	0.0109	0.0011	0.0043	0.0119	0.0391	0.0109	0.0022
RNA2	0.0034	0.0136	0.0000	0.0215	0.0661	0.0000	0.0000	0.0000	0.0000
RNA3	0.0074	0.0371	0.1186	0.0222	0.0877	0.0568	0.0099	0.0333	0.0025
RNA4	0.0031	0.0115	0.0010	0.0000	0.0000	0.0000	0.0157	0.0000	0.0010
RNA5	0.0023	0.0315	0.0011	0.0023	0.0090	0.0281	0.0169	0.0405	0.0000
RNA6	0.1112	0.0311	0.1112	0.0475	0.0178	0.2892	0.0000	0.0519	0.0044
RNA7	0.0098	0.0229	0.0000	0.0142	0.0076	0.0000	0.0000	0.0000	0.0011
RND0	0.0132	0.0110	0.1046	0.0011	0.0055	0.1530	0.0165	0.0099	0.0011
RND1	0.0024	0.1113	0.0024	0.0024	0.0086	0.0245	0.0000	0.0318	0.0000
RND2	0.0021	0.0000	0.0000	0.0000	0.0000	0.0000	0.0460	0.0000	0.0000
RND3	0.0032	0.0162	0.0335	0.0043	0.0119	0.0984	0.0000	0.0238	0.0000
RND4	0.0042	0.0052	0.0021	0.0010	0.0042	0.0104	0.0093	0.0094	0.0000
RND5	0.0130	0.0032	0.0238	0.0130	0.0022	0.0929	0.0000	0.0140	0.0000
TOTAL	0.2492	0.6165	0.4864	0.2002	0.3799	1.2732	0.2661	0.4781	0.1542

CONTROLLER UTILIZATIONS

CONTROL-0 0.22 CONTROL-1 0.24 CONTROL-2 0.12

Tabla 5.1 CAMBIO DE LA CPU POR UNA 2 VECES MAS RAPIDA – TURNO 3 –

RESPONSE TIME:

CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVNMPP	OTRMPP	BACKG
7.5193	16.773	10.469	6.3513	12.455	22.553	8.8903	12.1921	12.223

DEVICE UTILIZATIONS:

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVNMPP	OTRMPP	BACKG	TOTAL
CPU	0.1358	0.3087	0 0006	0.0822	0 0775	0 0203	0 0834	0.0875	0 1960	0 9919
RPA0	0.0116	0 1237	0 0000	0.0100	0.008	0.0095	0 0265	0.0391	0 0565	0.2848
RPA1	0.0070	0 0784	0 0000	0 0083	0 0056	0 0049	0 0144	0 0316	0 0141	0 1642
RPA3	0.0116	0.1469	0.0000	0.0116	0 0135	0.0103	0.0132	0.0361	0 0121	0.2552
RNA0	0 0023	0.0285	0 0000	0 0033	0 0032	0 0010	0 0000	0 0204	0 0020	0.0607
RNA1	0.0070	0.0143	0.0001	0.0017	0 0032	0.0011	0.0434	0 0093	0.0040	0.0841
RNA2	0 0069	0 0285	0 0000	0 0314	0 0605	0 0000	0 0000	0.0000	0 0000	0.1274
RNA3	0.0139	0.0711	0.0008	0 0298	0.0565	0 0046	0 0096	0.0250	0.0040	0.2153
RNA4	0.0070	0 0261	0.0000	0 0000	0 0000	0 0000	0 0180	0 0000	0.0020	0.0531
RNA5	0 0046	0.0664	0 0000	0 0033	0.0064	0 0025	0 0180	0.0334	0.0000	0.1346
RNA6	0 1735	0 0498	0 0006	0 0529	0 0095	0.0194	0 0000	0 0324	0.0060	0.3441
RNA7	0.0209	0.0498	0.0000	0.0215	0 0056	0.0000	0.0000	0.0000	0 0020	0.0998
RND0	0 0277	0 0237	0.0008	0 0016	0.0040	0 0138	0 0179	0.0083	0.0020	0 0998
RND1	0.0046	0.2150	0.0000	0.0033	0.0056	0.0020	0 0000	0.0240	0 0000	0.2544
RND2	0.0046	0.0000	0 0000	0.0000	0 0000	0 0000	0 0454	0 0000	0.0000	0 0501
RND3	0.0069	0 0355	0.0002	0.0066	0.0087	0 0090	0.0000	0.0203	0.0000	0.0874
RND4	0.0092	0.0118	0.0000	0.0016	0.0032	0.0010	0.0096	0.0083	0.0000	0.0448
RND5	0.0277	0.0071	0.0002	0.0198	0.0016	0.0085	0.0000	0.0120	0.0000	0.0769

DEVICE QUEUE LENGTHS:

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVNMPP	OTRMPP	BACKG	TOTAL
CPU	16.8364	38.2841	0.0804	10.1880	9.6135	2.5169	10.3393	10.8508	24.3073	123.016
RPA0	0 0162	0 1730	0.0000	0 0139	0 0112	0 0133	0 0370	0 0546	0 0791	0.3984
RPA1	0.0083	0.0938	0.0000	0.0099	0.0067	0 0059	0.0172	0.0378	0.0169	0.1965
RPA3	0 0155	0 1972	0 0000	0 0155	0 0182	0 0139	0 0177	0 0485	0 0162	0.3427
RNA0	0 0025	0 0303	0.0000	0 0035	0 0034	0.0011	0 0000	0 0217	0 0021	0.0647
RNA1	0 0076	0 0156	0.0001	0 0018	0 0035	0 0012	0 0474	0 0102	0 0044	0 0918
RNA2	0.0080	0.0326	0 0000	0 0360	0.0694	0 0000	0 0000	0 0000	0.0000	0.1460
RNA3	0.0177	0.0907	0 0010	0 0379	0.0720	0 0058	0 0122	0 0319	0.0051	0 2744
RNA4	0.0073	0.0276	0.0000	0.0000	0.0000	0.0000	0.0190	0.0000	0.0021	0.0561
RNA5	0 0054	0 0768	0 0000	0 0038	0 0074	0.0029	0 0208	0.0386	0.0000	0.1555
RNA6	0.2646	0.0759	0.0009	0.0806	0.0146	0.0296	0.0000	0.0494	0.0092	0.5248
RNA7	0 0232	0 0554	0.0000	0.0239	0 0062	0.0000	0 0000	0 0000	0 0022	0.1108
RND0	0 0308	0 0263	0.0008	0.0018	0.0044	0 0153	0 0199	0 0092	0.0022	0.1109
RND1	0.0062	0 2883	0.0000	0 0044	0 0074	0.0027	0 0000	0.0322	0 0000	0.3412
RND2	0.0049	0.0000	0.0000	0.0000	0.0000	0.0000	0.0478	0.0000	0.0000	0.0527
RND3	0.0076	0.0389	0.0003	0.0072	0.0096	0.0099	0 0000	0.0223	0.0000	0.0957
RND4	0.0097	0.0124	0.0000	0.0017	0.0033	0.0010	0.0100	0.0087	0.0000	0.0469
RND5	0.0300	0.0077	0.0002	0.0214	0.0017	0 0093	0.0000	0.0130	0.0000	0.0834

DEVICE RESIDENCE TIMES:

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVNMPP	OTRMPP	BACKG
CPU	7.3170	16.2462	10 0453	6 2008	12.1536	25.4234	8.6812	11.7816	12.1536
RPA0	0 0071	0 0734	0 0000	0 0065	0 0141	0 1342	0 0311	0 0593	0 0395
RPA1	0.0036	0 0398	0.0000	0.0060	0.0084	0.0591	0.0145	0.0410	0.0084
RPA3	0 0067	0.0837	0.0000	0 0094	0 0229	0.1404	0 0148	0 0526	0 0081
RNA0	0.0011	0.0129	0 0043	0.0021	0.0043	0.0107	0 0000	0.0236	0.0011
RNA1	0 0033	0 0066	0 0110	0.0011	0 0044	0 0121	0 0396	0 0110	0 0022
RNA2	0.0035	0 0138	0 0000	0 0219	0.0877	0 0000	0 0000	0 0000	0 0000
RNA3	0 0077	0 0585	0 1231	0 0231	0 0910	0 0590	0 0103	0 0346	0 0026
RNA4	0 0032	0 0117	0 0011	0 0000	0 0000	0 0000	0 0160	0 0000	0 0011
RNA5	0 0023	0 0326	0 0012	0 0027	0 0095	0 0291	0 0174	0 0419	0 0000
RNA6	0 1150	0 0322	0 1150	0 0491	0 0184	0 2990	0 0000	0 0537	0 0046
RNA7	0 0101	0 0235	0 0000	0 0145	0 0078	0 0000	0 0000	0 0000	0 0011
RND0	0 0134	0 0112	0 1059	0 0011	0 0056	0 1550	0 0167	0 0100	0 0011
RND1	0 0027	0 1223	0 0027	0 0027	0 0094	0 0269	0 0000	0 0350	0 0000
RND2	0 0021	0.0000	0.0000	0.0000	0.0000	0.0000	0 0402	0.0000	0.0000
RND3	0 0033	0 0165	0 0341	0 0044	0 0121	0 1001	0 0000	0 0242	0 0000
RND4	0 0042	0 0053	0 0021	0 0011	0 0042	0 0105	0 0084	0 0095	0.0000
RND5	0 0131	0 0033	0 0239	0 0131	0 0022	0 0935	0 0000	0 0141	0 0000
TOTAL	7.5193	16.7734	10.4698	6.3513	12.4556	26.5530	8.8903	12.1921	12.2235

CONTROLLER UTILIZATIONS:

CONTROL-0 0.26 CONTROL-1 0.27 CONTROL-2 0.14

Tabla 5.2 TASA LLEGADA DE CLASE CIDMPP SE AUMENTA EN 1.5 - TURNO 3 -

RESPONSE TIME:

CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG
3.7438	2.3805	1.3740	2.6974	6.0952	1.9220	2.7325	2.4662

DEVICE UTILIZATIONS:

	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG	TOTAL
CPU	0.4116	0.0006	0.0822	0.0775	0.0203	0.0834	0.0874	0.1960	0.9591
RPA0	0.1652	0.0000	0.0100	0.0080	0.0095	0.0265	0.0391	0.0566	0.3149
RPA1	0.1047	0.0000	0.0083	0.0056	0.0040	0.0144	0.0316	0.0141	0.1837
RPA3	0.1960	0.0000	0.0116	0.0135	0.0104	0.0132	0.0361	0.0121	0.2929
RNA0	0.0379	0.0000	0.0033	0.0032	0.0010	0.0000	0.0204	0.0020	0.0678
RNA1	0.0190	0.0001	0.0017	0.0032	0.0011	0.0433	0.0093	0.0040	0.0817
RNA2	0.0379	0.0000	0.0314	0.0605	0.0000	0.0000	0.0000	0.0000	0.1298
RNA3	0.0947	0.0008	0.0297	0.0564	0.0046	0.0096	0.0250	0.0040	0.2248
RNA4	0.4034	0.0000	0.0000	0.0000	0.0000	0.0180	0.0000	0.0020	0.0548
RNA5	0.0685	0.0000	0.0033	0.0064	0.0025	0.0180	0.0333	0.0000	0.1519
RNA6	0.0664	0.0006	0.0529	0.0095	0.0194	0.0000	0.0324	0.0060	0.1873
RNA7	0.0664	0.0000	0.0215	0.0056	0.0000	0.0000	0.0000	0.0020	0.0954
RND0	0.0315	0.0008	0.0016	0.0040	0.0138	0.0179	0.0083	0.0020	0.0800
RND1	0.2864	0.0000	0.0033	0.0055	0.0020	0.0000	0.0240	0.0000	0.3212
RND2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0454	0.0000	0.0000	0.0454
RND3	0.0473	0.0002	0.0066	0.0087	0.0090	0.0000	0.0203	0.0000	0.0923
RND4	0.0158	0.0000	0.0016	0.0032	0.0010	0.0096	0.0083	0.0000	0.0395
RND5	0.0095	0.0002	0.0198	0.0016	0.0066	0.0000	0.0120	0.0000	0.0516

DEVICE QUEUE LENGTHS:

	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG	TOTAL
CPU	10.0582	0.0158	2.0075	1.8943	0.4959	2.0373	2.1381	4.7896	23.4366
RPA0	0.2411	0.0000	0.0145	0.0117	0.0139	0.0387	0.0571	0.0826	0.4595
RPA1	0.1283	0.0000	0.0102	0.0069	0.0060	0.0177	0.0387	0.0173	0.2250
RPA3	0.2772	0.0000	0.0164	0.0191	0.0147	0.0186	0.0511	0.0171	0.4142
RNA0	0.0407	0.0000	0.0035	0.0034	0.0011	0.0000	0.0219	0.0022	0.0728
RNA1	0.0207	0.0001	0.0018	0.0035	0.0012	0.0472	0.0101	0.0044	0.0890
RNA2	0.0436	0.0000	0.0361	0.0695	0.0000	0.0000	0.0000	0.0000	0.1491
RNA3	0.1222	0.0010	0.0383	0.0728	0.0059	0.0124	0.0322	0.0052	0.2900
RNA4	0.0368	0.0000	0.0000	0.0000	0.0000	0.0190	0.0000	0.0021	0.0579
RNA5	0.1043	0.0000	0.0039	0.0075	0.0029	0.0212	0.0393	0.0000	0.1791
RNA6	0.0817	0.0007	0.0651	0.0117	0.0239	0.0000	0.0399	0.0074	0.2304
RNA7	0.0734	0.0000	0.0237	0.0062	0.0000	0.0000	0.0000	0.0022	0.1055
RND0	0.0343	0.0008	0.0018	0.0043	0.0150	0.0195	0.0090	0.0022	0.0870
RND1	0.4220	0.0000	0.0048	0.0082	0.0029	0.0000	0.0353	0.0000	0.4733
RND2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0476	0.0000	0.0000	0.0476
RND3	0.0521	0.0003	0.0073	0.0096	0.0100	0.0000	0.0224	0.0000	0.1017
RND4	0.0164	0.0000	0.0017	0.0033	0.0010	0.0100	0.0087	0.0000	0.0411
RND5	0.0100	0.0002	0.0209	0.0017	0.0090	0.0000	0.0127	0.0000	0.0544

DEVICE RESIDENCE TIMES:

	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMMPP	OTRMPP	BACKG
CPU	3.2012	1.9794	1.2218	2.3948	5.0095	1.7106	2.3215	2.3948
RPA0	0.0767	0.0000	0.0089	0.0146	0.1402	0.0345	0.0620	0.0413
RPA1	0.0408	0.0000	0.0062	0.0087	0.0606	0.0146	0.0421	0.0087
RPA3	0.0882	0.0000	0.0100	0.0242	0.1460	0.0157	0.0555	0.0085
RNA0	0.0130	0.0043	0.0022	0.0043	0.0108	0.0000	0.0237	0.0011
RNA1	0.0066	0.0110	0.0011	0.0044	0.0121	0.0396	0.0110	0.0022
RNA2	0.0139	0.0000	0.0220	0.0878	0.0000	0.0000	0.0000	0.0000
RNA3	0.0389	0.1245	0.0233	0.0920	0.0596	0.0104	0.0350	0.0026
RNA4	0.0117	0.0011	0.0000	0.0000	0.0000	0.0160	0.0000	0.0011
RNA5	0.0332	0.0012	0.0024	0.0095	0.0296	0.0176	0.0427	0.0000
RNA6	0.0260	0.0928	0.0396	0.0148	0.2413	0.0000	0.0433	0.0037
RNA7	0.0234	0.0000	0.0145	0.0078	0.0000	0.0000	0.0000	0.0011
RND0	0.0109	0.1037	0.0011	0.0055	0.1517	0.0164	0.0098	0.0011
RND1	0.1343	0.0030	0.0030	0.0103	0.0295	0.0000	0.0384	0.0000
RND2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0400	0.0000	0.0000
RND3	0.0166	0.0343	0.0044	0.0122	0.1006	0.0000	0.0243	0.0000
RND4	0.0052	0.0021	0.0010	0.0042	0.0105	0.0084	0.0094	0.0000
RND5	0.0032	0.0233	0.0127	0.0021	0.0911	0.0000	0.0136	0.0000
TOTAL	3.7438	2.3805	1.3740	2.6974	6.0952	1.9220	2.7325	2.4662

CONTROLLER UTLIZATIONS:

CONTROL-0 0.29 CONTROL-1 0.24 CONTROL-2 0.14

Tabla 5.3 SE ELIMINA CANMPP Y SE AUMENTA TASA LLEGADA CIDMPP AL DOBLE
- TURNO 3 -

RESPONSE TIME:

CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMIMPP	QTRMPP	BACKG
0.4546	1.3959	0.7795	0.3767	0.7331	2.2286	0.4998	0.9425	0.4481

DEVICE UTILIZATIONS:

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMIMPP	QTRMPP	BACKG	TOTAL
CPU	0.0679	0.5145	0.0003	0.0411	0.0388	0.0101	0.0417	0.0437	0.0980	0.8561
RPA0	0.0118	0.4175	0.0000	0.0101	0.0081	0.0096	0.0268	0.0395	0.0572	0.5806
RPA1	0.0071	0.2660	0.0000	0.0084	0.0057	0.0050	0.0147	0.0321	0.0144	0.3534
RPA3	0.0117	0.4948	0.0000	0.0117	0.0137	0.0105	0.0133	0.0365	0.0122	0.6043
RNA0	0.0023	0.0954	0.0000	0.0033	0.0032	0.0010	0.0000	0.0205	0.0020	0.1278
RNA1	0.0070	0.0481	0.0001	0.0017	0.0032	0.0011	0.0437	0.0094	0.0041	0.1184
RNA2	0.0070	0.0953	0.0000	0.0316	0.0608	0.0000	0.0000	0.0000	0.0000	0.1947
RNA3	0.0139	0.2381	0.0008	0.0299	0.0567	0.0046	0.0096	0.0251	0.0040	0.3828
RNA4	0.0070	0.0874	0.0000	0.0000	0.0000	0.0000	0.0181	0.0000	0.0020	0.1145
RNA5	0.0047	0.2223	0.0000	0.0033	0.0064	0.0025	0.0181	0.0335	0.0000	0.2908
RNA6	0.1743	0.1666	0.0006	0.0531	0.0096	0.0195	0.0000	0.0326	0.0061	0.4622
RNA7	0.0209	0.1668	0.0000	0.0216	0.0056	0.0000	0.0000	0.0000	0.0020	0.2170
RND0	0.0278	0.0791	0.0008	0.0017	0.0040	0.0139	0.0130	0.0084	0.0020	0.1556
RND1	0.0046	0.7174	0.0000	0.0033	0.0056	0.0029	0.0000	0.0240	0.0000	0.7569
RND2	0.0046	0.0000	0.0000	0.0000	0.0000	0.0000	0.0456	0.0000	0.0000	0.0503
RND3	0.0070	0.1187	0.0002	0.0066	0.0088	0.0091	0.0000	0.0204	0.0000	0.1708
RND4	0.0093	0.0396	0.0000	0.0017	0.0032	0.0010	0.0096	0.0084	0.0000	0.0727
RND5	0.0278	0.0238	0.0002	0.0199	0.0016	0.0086	0.0000	0.0121	0.0000	0.0939

DEVICE QUEUE LENGTHS:

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVMIMPP	QTRMPP	BACKG	Total
CPU	0.4718	3.5759	0.0023	0.2855	0.2694	0.0705	0.2897	0.3041	0.6811	5.9502
RPA0	0.0271	0.9631	0.0000	0.0232	0.0187	0.0222	0.0618	0.0912	0.1320	1.3393
RPA1	0.0106	0.3987	0.0000	0.0126	0.0085	0.0075	0.0220	0.0482	0.0215	0.5296
RPA3	0.0287	1.2128	0.0000	0.0286	0.0335	0.0256	0.0326	0.0895	0.0299	1.4812
RNA0	0.0026	0.1085	0.0000	0.0038	0.0036	0.0011	0.0000	0.0233	0.0023	0.1454
RNA1	0.0079	0.0539	0.0001	0.0019	0.0036	0.0012	0.0491	0.0105	0.0046	0.1329
RNA2	0.0086	0.1174	0.0000	0.0389	0.0749	0.0000	0.0000	0.0000	0.0000	0.2398
RNA3	0.0224	0.3821	0.0012	0.0480	0.0911	0.0074	0.0154	0.0403	0.0065	0.6144
RNA4	0.0078	0.0980	0.0000	0.0000	0.0000	0.0000	0.0203	0.0000	0.0023	0.1284
RNA5	0.0065	0.3109	0.0000	0.0046	0.0089	0.0035	0.0253	0.0469	0.0000	0.4066
RNA6	0.3203	0.3061	0.0011	0.0976	0.0176	0.0358	0.0000	0.0598	0.0111	0.8495
RNA7	0.0265	0.2114	0.0000	0.0274	0.0071	0.0000	0.0000	0.0000	0.0026	0.2749
RND0	0.0327	0.0930	0.0009	0.0019	0.0047	0.0163	0.0212	0.0098	0.0024	0.1829
RND1	0.0188	2.9223	0.0001	0.0134	0.0226	0.0081	0.0000	0.0979	0.0000	3.0832
RND2	0.0049	0.0000	0.0000	0.0000	0.0000	0.0000	0.0477	0.0000	0.0000	0.0526
RND3	0.0063	0.1421	0.0003	0.0079	0.0105	0.0109	0.0000	0.0244	0.0000	0.2045
RND4	0.0099	0.0424	0.0000	0.0018	0.0034	0.0011	0.0103	0.0089	0.0000	0.0778
RND5	0.0305	0.0260	0.0002	0.0218	0.0017	0.0094	0.0000	0.0132	0.0000	0.1028

DEVICE RESIDENCE TIMES:

	CANMPP	CIDMPP	DEFMPP	FILMPP	LEEMPP	MATMPP	RVIIMPP	QTRMPP	BACKG
CPU	0.2050	0.4552	0.2815	0.1738	0.3406	0.7124	0.2433	0.3301	0.3406
RPA0	0.0118	0.1226	0.0000	0.0141	0.0236	0.2240	0.0519	0.0990	0.0660
RPA1	0.0046	0.0508	0.0000	0.0077	0.0108	0.0754	0.0185	0.0523	0.0106
RPA3	0.0125	0.1544	0.0000	0.0174	0.0423	0.2590	0.0274	0.0971	0.0149
RNA0	0.0012	0.0138	0.0046	0.0023	0.0046	0.0115	0.0000	0.0253	0.0012
RNA1	0.0034	0.0069	0.0114	0.0011	0.0046	0.0126	0.0412	0.0114	0.0023
RNA2	0.0037	0.0149	0.0000	0.0237	0.0947	0.0000	0.0000	0.0000	0.0000
RNA3	0.0097	0.0486	0.1557	0.0292	0.1151	0.0746	0.0130	0.0438	0.0032
RNA4	0.0034	0.0125	0.0011	0.0000	0.0000	0.0000	0.0170	0.0000	0.0011
RNA5	0.0028	0.0396	0.0014	0.0028	0.0113	0.0353	0.0212	0.0509	0.0000
RNA6	0.1392	0.0390	0.1392	0.0594	0.0223	0.3619	0.0000	0.0650	0.0056
RNA7	0.0115	0.0269	0.0000	0.0167	0.0090	0.0000	0.0000	0.0000	0.0013
RND0	0.0142	0.0118	0.1125	0.0012	0.0059	0.1646	0.0176	0.0107	0.0012
RND1	0.0082	0.3720	0.0082	0.0082	0.0286	0.0818	0.0000	0.1063	0.0000
RND2	0.0021	0.0000	0.0000	0.0000	0.0000	0.0000	0.0401	0.0000	0.0000
RND3	0.0036	0.0181	0.0374	0.0048	0.0133	0.1098	0.0000	0.0265	0.0000
RND4	0.0043	0.0054	0.0022	0.0011	0.0043	0.0108	0.0086	0.0097	0.0000
RND5	0.0133	0.0033	0.0243	0.0133	0.0022	0.0950	0.0000	0.0144	0.0000
TOTAL	0.4546	1.3959	0.7795	0.3767	0.7331	2.2286	0.4998	0.9425	0.4481

CONTROLLER UTILIZATIONS:

CONTROL-0: 0.56 CONTROL-1: 0.45 CONTROL-2 0.30

Tabla 5.4 SE DUPLICA VELOCIDAD DE LA CPU Y TASA DE LLEGADA DE CIDMPP AUMENTA 5 VECES - TURNO 3 -

de saturación (tiempos de respuesta "infinitos") en varias opurtunidades durante el turno.

Se puede concluir, entonces, que el sistema actual, sin modificación, no es capaz de tolerar un aumento de los procesos CIDMPP más allá de un 30 a 40%.

Traslado de Procesos CANMPP

En tabla 5.3 se muestra el informe entregado por el software de modelación con la definición del modelo y carga de trabajo, para el turno 3, donde se ha eliminado la clase CANMPP y se ha aumentado al doble la tasa de llegada de la clase CIDMPP (por ej. los procesos de CANMPP han sido trasladados a un computador de respaldo). (Se supone que los procesos CANMPP son trasladados al computador de respaldo de la institución.)

En el informe de desempeño "System Performance Measures" (listado 5.3) se observa que los tiempos de respuesta ("Response Time") para las distintas transacciones ha aumentado con respecto a los del sistema real (tabla 4.1). Sin embargo, estos tiempos de respuesta se mantienen dentro de un rango aceptable.

En el informe "Device Utilizations" se muestra una utilización total de la CPU del 95.9%. A pesar de encontrarse la CPU próxima a la saturación, estas cifras indican que es posible aumentar la ejecución de procesos CIDMPP hasta aprox. un 80%, manteniendo tiempos de respuesta relativamente razonables.

Aumento de Tasa de Llegada de CIDMPP con CPU más Rápida

En la tabla 5.4 se muestra el informe entregado por el software de modelación con la definición del modelo, para el turno 3, considerando una CPU dos veces más rápida, y una clase CIDMPP con tasa de llegada 5 veces mayor que la del sistema real. Todos los demás parámetros del modelo se mantienen igual que en el modelo original.

En el informe de desempeño "System Performance Measures" (listado 5.4.b) se observa que los tiempos de respuesta ("Response Time") para las distintas transacciones son similares a los tiempos de respuesta del turno 3 original (listado 5.1).

En el informe "Device Utilizations" se muestra una utilización total de la CPU del 85.6%. Esta utilización es muy similar a la del sistema original. Estas cifras indican que el sistema computacional, con una CPU dos veces más rápida que la actual, podría procesar aproximadamente 5 veces más transacciones CIDMPP manteniendo un desempeño similar al del sistema actual.

6. CONCLUSIONES

En este trabajo se ha presentado una aplicación de la tecnología de modelos de redes de espera a un sistema computacional en operación. Para este efecto, se ha utilizado el paquete comercial de modelación MAP ([Lazo84]) y se ha implementado un software front-end que facilita la definición de la carga de trabajo, la estimación de la demanda de servicio en el subsistema de entrada/salida y la validación del modelo.

La validación del modelo entrega un margen de error del orden de 1% para la estimación de la utilización de la CPU en distintos turnos de procesamiento, y entre 4 y 20% para la estimación de accesos al subsistema de entrada/salida. Este nivel de precisión se considera adecuado para los objetivos del estudio. Las mediciones por medio del monitor de sistema y las estimaciones del modelo indican que la CPU es el "cuello de botella" del sistema, con una utilización mayor de 80%.

El modelo fué utilizado para predecir el desempeño del sistema ante eventuales variaciones del hardware (específicamente el cambio de CPU) y variaciones en la carga de trabajo, arrojando resultados que están de acuerdo con el desempeño esperado.

REFERENCIAS

[Casa86] Casas I.
"Prophet: A Layered Analytical Model for Performance Prediction of Database Systems", Computer Systems Research Institute, University of Toronto, Tech. Rep. CSRI-180, Toronto, Canada, 1986.

[Casa87] Casas I., Aurtenechea F.
"Modelación y Evaluación del Sistema Computacional de una Empresa Chilena", Depto. de Ciencia de Computación, Escuela de Ingeniería, Univ. Católica de Chile, Informe Técnico IT87-2, Santiago, Chile, 1987.

[Grah78] Graham G.S. (ed.)
"Queueing Network Models of Computer System Performance", ACM Comp. Surveys Special Issue #10.3, Sept. 1978.

[Lazo84] Lazowska E.D., Zahorjan J., Graham G.S., Sevcik K.C.
"Quantitative System Performance - Computer System Analysis Using Queueing Network Models", Prentice Hall Inc., New Jersey, 1984.

[Saue81] Sauer C.H., Chandy K.M.
"Computer Systems Performance Modeling", Prentice-Hall Inc., Englewood Cliffs, N.J., 1981.

SIMCOL: UNA HERRAMIENTA PARA SIMULAR MODELOS DE REDES DE COLAS

Federico R. Echavarría R.
Departamento de Ingeniería Industrial
Facultad de Ingeniería
Universidad de Los Andes
Apartado Aéreo 4976, Bogotá

Fernando Palacios Gómez
Departamento de Ingeniería Industrial
Facultad de Ingeniería
Universidad de Los Andes
Apartado Aéreo 4976, Bogotá

RESUMEN

El propósito de este documento es presentar un programa desarrollado para simular sistemas que se pueden modelar como redes de colas abiertas o cerradas. Dicho programa permite definir las características de la red con base en el número y tipos de clientes, en los centros de servicio y en los arcos. Los clientes son definidos de manera global. Los centros de servicio pueden ser de recurso activo o pasivo. Los centros de servicio activo pueden tener las siguientes disciplinas: Fifo, Lifo, Aleatorio, Fifo con Preferencia y sin Interrupción, Fifo con Preferencia y con Interrupción, LCFSPR, PS, RR con Interrupción y con Preferencia, RR sin Interrupción y con Preferencia, RR sin Interrupción y sin Preferencia. Los arcos definen el paso de los clientes de un centro a otro, el cual se hace de acuerdo con unas probabilidades de salto o ramificación y puede o no tener duración. Para definir la duración de un servicio, la duración del viaje de propagación entre dos nodos y el lapso entre llegadas de dos clientes (este último para redes abiertas) se pueden utilizar las siguientes distribuciones: Constante, Uniforme, Normal, Exponencial, Erlang, Hypoexponencial, Hiperexponencial, Discreta Diseñada Por El Usuario y Red Exponencial.

El programa funciona bajo el método de próximo evento y está escrito en lenguaje Fortran, utilizando la versión del computador personal de la IBM.

INTRODUCCION

La importancia de la simulación en el mundo actual es ampliamente reconocida. Según Pritsker (5,p.5), en los paises desarrollados la simulación y los métodos estadísticos son las técnicas de Investigación de Operaciones y Administración más utilizadas por la industria y el gobierno. Dentro de las múltiples ventajas de la simulación, la más importante es su gran generalidad. Además un simulador, como laboratorio de experimentación, presenta las siguientes ventajas con respecto a la experimentación directa:

- Si el sistema propuesto no está implementado actualmente, las alternativas se pueden evaluar sin la necesidad de construír el sistema.

- Si el sistema ya está implementado, las diferentes alternativas pueden ser evaluadas sin perturbar el sistema. Esto es importante ya que en muchos casos la experimentación directa es costosa, insegura y toma mucho tiempo.

En particular, este trabajo presenta un simulador de sistemas que se pueden modelar como redes de colas, el cual ha sido llamado SIMCOL. Al desarrollar esta herramienta, siempre se tuvieron los siguientes objetivos en mente: facilidad de uso tanto en la entrada de datos (configuración de la red) como en la consulta de resultados y razonables facilidades para la detección de errores.

Aunque existen métodos analíticos y heurísticos que en la mayoría de los casos son más eficientes que la simulación, esta última presenta las siguientes ventajas sobre los primeros:

- Para evaluar una red de colas mediante métodos analíticos y heurísticos se requiere una profundidad teórica mayor que la que se requiere para evaluarla con un simulador.

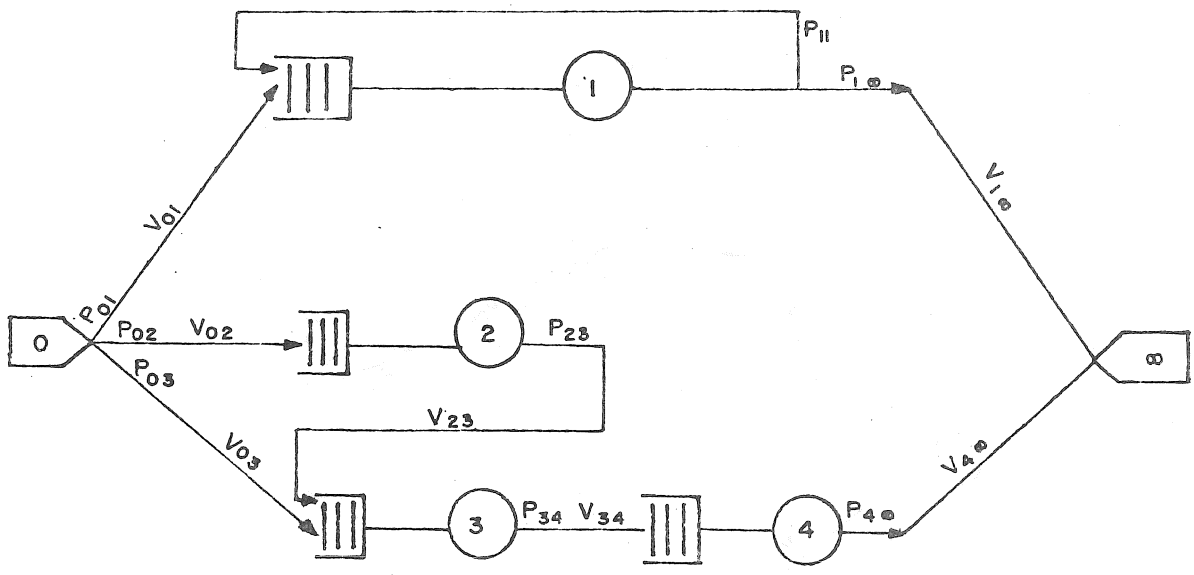
- Los métodos analíticos existentes en algunos casos son bastante más difíciles de emplear y no son tan generales como la simulación.

1. DESCRIPCION GENERAL

El programa SIMCOL es una herramienta mediante la cual se pueden simular modelos de redes de colas con ciertas características.

Una red de colas esta constituída por centros de servicio, clientes y arcos. Los centros de servicio están caracterizados por la disciplina de servicio, el número de recursos del centro, la capacidad del centro y el tiempo o duración del servicio. Los clientes viajan a través de los centros en busca de un servicio. El paso de los clientes de un centro a otro, el cual puede tener o no duración, se hace a través de un arco y de acuerdo a unas probabilidades de salto o ramificación. El programa simula dos tipos de modelos en general: modelos de redes de colas abiertas y cerradas.

RED ABIERTA

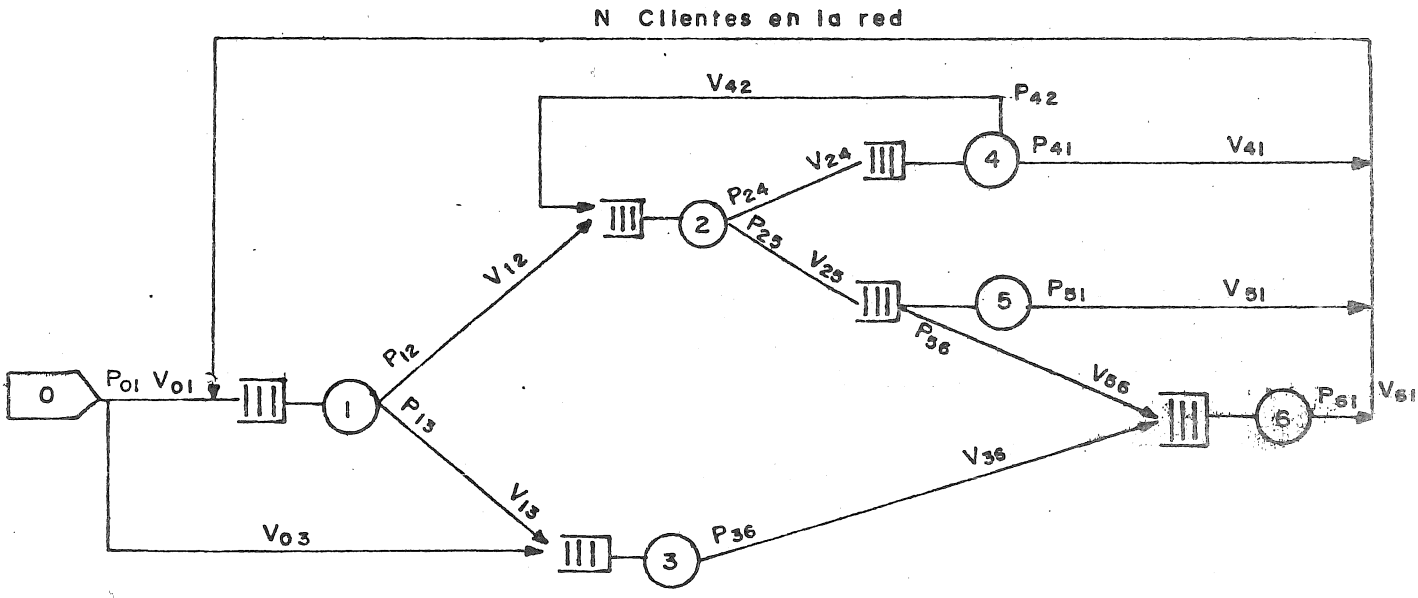


donde:

	Fuente externa
	Salida de la red
	Centro de servicio
Pij	Probabilidades de ramificaciòn entre el centro i y j
Vij	Propagaciòn o viaje entre el centro i y j

Figura 1-1

RED CERRADA



donde:

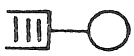
	Fuente interna donde son colocados todos los clientes al principio de la simulaciòn.
	Centro de servicio
Pij	Probabilidades de ramificaciòn entre el centro i y j
Vij	Propagaciòn o viaje entre el centro i y j

Figura 1-2

Los modelos de redes de colas abiertas se caracterizan por el hecho de que los clientes son creados mediante una fuente externa, viajan a través de la red y finalmente salen en algún lugar de esta (figura 1-1).

Por otro lado, los modelos de colas de redes cerradas se caracterizan porque los clientes no entran ni salen de la red, sino que permanecen en ella indefinidamente. De lo anterior se deduce que la red mantiene una población constante de clientes a través del tiempo (figura 1-2).

Mediante el simulador SIMCOL se pueden obtener los siguientes parámetros en cada uno de los centros de la red, tanto para los clientes en general como para los tipos de clientes en particular:

- Thruput o número de personas que completaron el servicio por unidad de tiempo.

- Media y varianza del número de clientes en el centro.

- Tiempo promedio de los clientes en el centro.

- Media y varianza del número de clientes en cola.

- Tiempo promedio de los clientes en la cola.

- Media y varianza del número de servidores activos o utilización de los servidores del centro de servicio.

- Histograma del número de clientes en el centro. Para obtener dicho histograma, el usuario debe especificar el número de celdas, el límite superior de la primera celda y el tamaño de cada celda.

Es importante anotar que las medias y las varianzas mencionadas anteriormente son con respecto al tiempo (time averages) y son calculadas mediante las siguientes fórmulas:

$$\hat{x}_T = \frac{\int_0^T x(t)\,dt}{T}$$

$$\hat{s}^2_T = \frac{\int_0^T x^2(t)\,dt}{T} - \hat{x}^2_T$$

donde:

$\hat{x}_T$: media
$\hat{s}^2_T$: varianza
x(t): curva en función del tiempo
T : duración total de la(s) simulacion(es).

Además para redes abiertas el programa calcula los siguientes parámetros para la red en general, tanto para los clientes en general como para los distintos tipos de clientes en particular:

- Thruput o número de clientes por unidad de tiempo que completaron su viaje a través de la red.

- Tiempo promedio de permanencia de los clientes en la red.

- Histograma del tiempo promedio de los clientes en la red. Para obtener este histograma, el usuario debe especificar el número de celdas del histograma, el límite superior de la primera celda y el ancho o intervalo de tiempo de cada celda.

1.1. Elementos Constitutivos de una Red

1.1.1. Clientes

Una red de colas puede tener diferentes tipos de clientes. Los tipos de clientes se diferencian unos de otros mediante ciertas propiedades, las cuales son fijas a través de la red. Estas propiedades son con respecto a las disciplinas y a las distribuciones de servicio, a las probabilidades de ramificación y a la duración de propagación entre centros de servicio. Además, para redes abiertas estas características incluyen la distribución del tiempo entre llegadas de clientes de un mismo tipo a la red, y para redes cerradas la población total de cada tipo en la red.

1.1.2. Centros de Servicio

Un centro de servicio tiene dos componentes: una cola y un conjunto de recursos. Una cola es un grupo de clientes que esperan o reciben un servicio. Dependiendo del tipo de recurso que posean los centros de servicio estos se dividen en activos o pasivos.

1.1.2.1. Centros de Recurso Activo

Se caracterizan primordialmente por el hecho de que la duración del servicio depende directamente del recurso. El recurso de un centro activo es lo que normalmente se denomina servidor. Además, el recurso está ligado al centro y no al cliente, es decir, una vez finalizado el servicio en dicho centro, el cliente pierde el recurso y sale del centro.

En los centros de recurso activo los clientes son servidos de acuerdo a una disciplina. SIMCOL tiene a disposición del usuario las siguientes disciplinas:

1.1.2.1.1. Fifo

Bajo esta disciplina, los clientes son servidos en orden de llegada al centro.

1.1.2.1.2. Lifo

Bajo esta disciplina, el último cliente en llegar es el primero en ser servido. Por lo tanto, al terminar un servicio, si hay clientes en la cola, el próximo en ser atendido es el último en haber llegado al centro.

1.1.2.1.3. Aleatorio

Al terminar un servicio, si hay clientes en la cola, el servidor escoge un cliente de ésta aleatoriamente para ser el siguiente en ser atendido. La escogencia se hace de manera que todos los clientes en la cola tengan la misma probabilidad de ser escogidos.

Es importante anotar que en las tres disciplinas anteriores no existe interrupción, es decir, una vez asignado un servidor a un cliente, este último es atendido en su totalidad.

1.1.2.1.4. Fifo Con Preferencia y Sin Interrupción

En esta disciplina los clientes son organizados en subcolas de acuerdo a una prioridad, la cual es propia de cada tipo de cliente (figura 1.1.2.1-1). Dentro de cada subcola, los clientes son organizados de acuerdo a la disciplina Fifo. Al finalizar un servicio, el primer cliente de la subcola no vacía con mayor prioridad es servido.

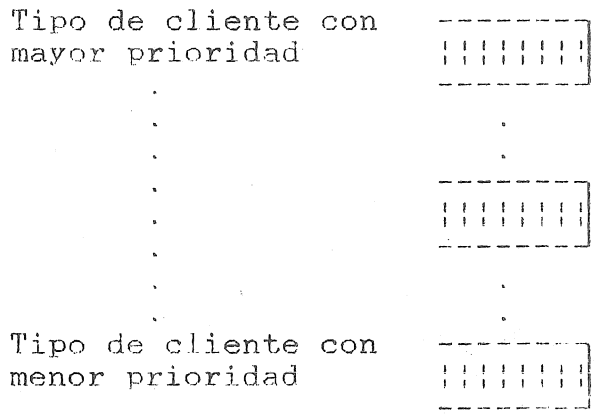


Figura 1.1.2.1-1

1.1.2.1.5. Fifo Con Preferencia y Con Interrupción

Esta disciplina tiene las mismas características que la disciplina anterior (Fifo Con Preferencia y Sin Interrupción). Sin embargo, cuando llega a la red un cliente de mayor prioridad que alguno de los que están siendo servidos en ese momento, éste interrumpe al

cliente con la menor prioridad, el cual vuelve a la cabeza de la cola, y el de mayor prioridad empieza a ser atendido. Si existen dos o más clientes con la menor prioridad y que están siendo servidos, el último cliente que entró a ser servido, es el que es interrumpido.

1.1.2.1.6. LCFSPR

En esta disciplina, en todo momento los clientes que están siendo servidos son los últimos n en arribar al centro de servicio, donde n es el número de servidores. Además, al terminar un servicio, si hay clientes en la cola, el próximo en ser atendido es el último, de aquellos que están en la cola, en haber llegado al centro. Por lo tanto, esta disciplina es equivalente a una Lifo con interrupción, donde la mayor prioridad la tiene el último cliente en llegar al centro.

1.1.2.1.7. PS(Process Sharing)

Bajo esta disciplina, todos los clientes del centro de servicio son atendidos simultáneamente. Esta disciplina es equivalente a que los servidores se desdoblaran en tantos servidores como clientes en el centro, disminuyéndose su eficiencia proporcionalmente. Sin embargo, cuando el número de servidores es mayor que el número de clientes en el centro, la eficiencia de los servidores permanece invariable. Para determinar la duración del tiempo de servicio se utiliza la siguiente formula:

$$D = L * X / MIN(L,K)$$

donde:

L: Número de personas en el centro de servicio.
K: Número de servidores.
X: Duración del servicio si se le asignara un servidor al cliente
D: Duración del servicio si en el centro hay K servidores atendiendo a L clientes.

Es importante anotar que para el caso en que el número de personas en el centro sea menor o igual al número de servidores ($L \leq K$) se da que D=X.

1.1.2.1.8. RR(Round Robin) Con Interrupción y Con Preferencia

Esta disciplina tiene las mismas características que la disciplina número 5 (Fifo Con Preferencia y Con Interupción). Sin embargo, para cada clase o tipo de cliente, se establece un tope en la duración de cada servicio, el cual se llama "time slice" o "lapso" de servicio. Por lo tanto, al finalizar un servicio, si el cliente ha completado su servicio requerido, éste sale del centro, de lo contrario vuelve al final de su respectiva subcola. Además, el próximo cliente en ser servido después de dicha terminación es la

cabeza de la subcola no vacía con mayor prioridad.

1.1.2.1.9. RR Sin Interrupción y Con Preferencia

Esta disciplina tiene las mismas características que la disciplina anterior (RR Con Interrupción y Con Preferencia) a excepción del hecho de que un servicio nunca es interrumpido al llegar un cliente de mayor prioridad.

1.1.2.1.10. RR Sin Interrupción y Sin Preferencia

Esta disciplina tiene las mismas características que la disciplina anterior (RR Sin Interrupción y Con Preferencia) a excepción del hecho de que los clientes son organizados en la cola de acuerdo a la disciplina Fifo.

1.1.2.2. Centros de Recurso Pasivo

Estos centros de servicio se caracterizan primordialmente por el hecho de que la duración de la utilización del recurso no depende directamente del recurso asignado en ese centro, sino de los recursos activos de los centros por los cuales pasa el cliente entre el momento de asignación y liberación del recurso pasivo. De lo anterior se deduce que el recurso pasivo, aunque es asignado en un centro de servicio específico, se liga al cliente durante su viaje a través de otros centros de servicio entre los momentos de asignación y liberación de dicho recurso. Otra característica importante de los centros de recurso pasivo es el hecho de que un cliente puede requerir uno o más recursos pasivos; a diferencia del activo en el cual se requiere máximo un recurso para lograr el servicio.

En la cola de los centros de recurso pasivo, los clientes son organizados de acuerdo a la disciplina Fifo, y allí estarán todos aquellos clientes para los cuales su requerimiento de recurso es mayor que la cantidad de recurso disponible en ese momento.

Para mayor ilustración, a continuación se presentan dos ejemplos de recurso activo y pasivo. El primer ejemplo es en el área de sistemas de computación (Figura 1.1.2.2-1). En dichos sistemas el CPU es un recurso activo y la memoria es un recurso pasivo. Al llegar un job (cliente) a dicho sistema, éste debe primero obtener memoria para luego ser procesado en el CPU. La duración del job en el CPU depende de este último directamente, en cambio el tiempo que el job mantiene la memoria asignada no depende de ésta sino del CPU (recurso activo del centro a través del cual viaja el job entre los momentos de asignación y liberación del recurso pasivo).

El segundo ejemplo es en el área de administración hospitalaria y específicamente en el manejo de un centro de urgencias (figura 1.1.2.2-2). En dicho sistema el recurso activo son los doctores y el recurso pasivo son las salas de operación. Así, al llegar un paciente que requiera operación, el doctor no podrá hacerlo hasta que no haya disponible una sala. El tiempo que permanece ocupada la sala no depende de ésta directamente, sino de la relación doctor-

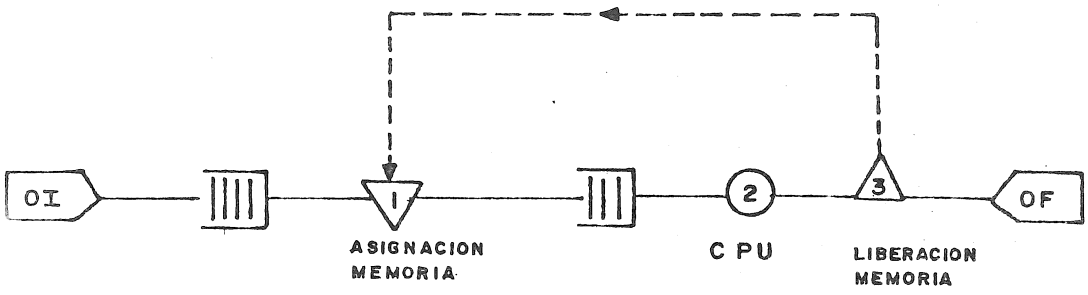


Figura 1.1.2.2-1

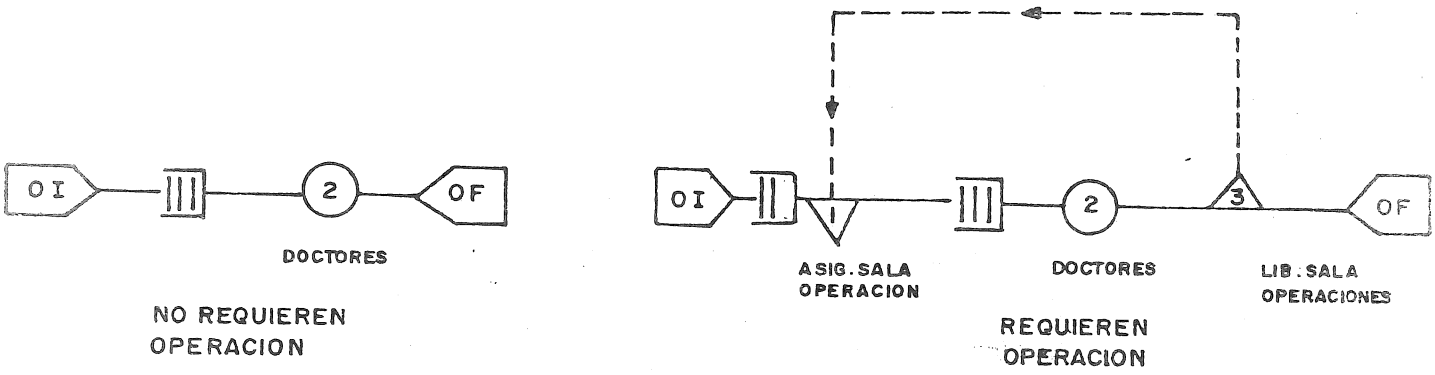


Figura 1.1.2.2-2

paciente. Por otro lado, aquellos tipos de pacientes que no requieran operación recorrerán el trayecto 0I-2-0F.

1.1.3. Topologia De La Red

Se define como la manera en que se intercomunican los diferentes centros de servicio en una red. El programa SIMCOL permite definir la topologia de las redes por tipo de cliente, en las cuales éstos saltan de un centro o nodo a otro de acuerdo a unas probabilidades que se denominan probabilidades de ramificación o de salto. Además de estas probabilidades, el programa permite al usuario asignarle una duración a ese salto, el cual se denomina viaje de propagación entre centros de servicio. El ejemplo más típico de este viaje de propagación es una banda transportadora en un proceso de producción, y es equivalente a un centro de servicio con infinitos servidores.

Es importante anotar que para la definición de la topología de la red, la fuente de creación (para ambos tipos de redes) es considerada como un nodo desde el cual los clientes pueden saltar hacia otros centros de servicio, pero no pueden saltar desde otros centros hacia ésta. Por otro lado, para redes abiertas, el punto de salida de la red es considerado como un nodo hacia el cual los clientes pueden saltar desde otros centros de servicio, pero no pueden saltar de éste hacia otros centros de servicio.

1.1.4. Distribuciones de Probabilidad

Para terminar la explicación de los elementos constitutivos de una red, sólo falta añadir que la duración de un servicio, la duración del viaje de propagación entre dos nodos y el lapso entre llegadas de dos clientes (este último para redes abiertas) son definidos mediante una distribución de probabilidad. SIMCOL pone a disposición del usuario las siguientes distribuciones: Constante, Uniforme, Normal, Exponencial, Erlang, Hypoexponencial, Hiperexponencial, Discreta Diseñada Por El Usuario y Red Exponencial.

La mayoría de estas distribuciones son conocidas en el medio en el que se trabaja, por lo tanto sólo se explicarán aquellas sobre las cuales existe poca bibliografía, las cuales son:

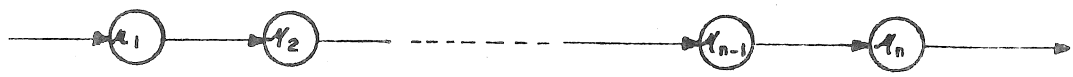
- Hypoexponencial: Se define como n exponenciales en serie y se representa de acuerdo a la figura 1.1-1.

- Hiperexponencial: Se define como n exponenciales en paralelo y se representa de acuerdo a la figura 1.1-2.

- Red Exponencial: Una red exponencial de n etapas se puede representar de acuerdo a la figura 1.1-3.

Además, para facilidad del usuario se añdió un caso especial de la distribución constante para especificar la duración del viaje de propagación entre nodos. Ese caso especial es la distribución con duración igual a cero o sin duración.

HYPOEXPONENCIAL

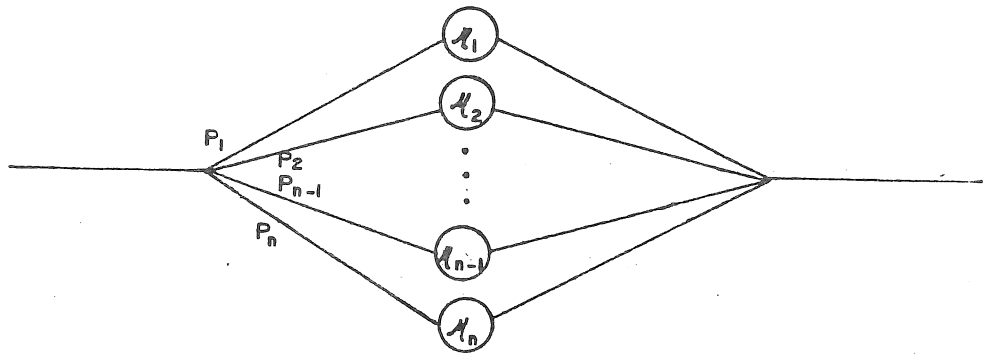


donde:

i: Media de la exponencial asociada con la etapa i

Figura 1.1-1

HIPEREXPONENCIAL



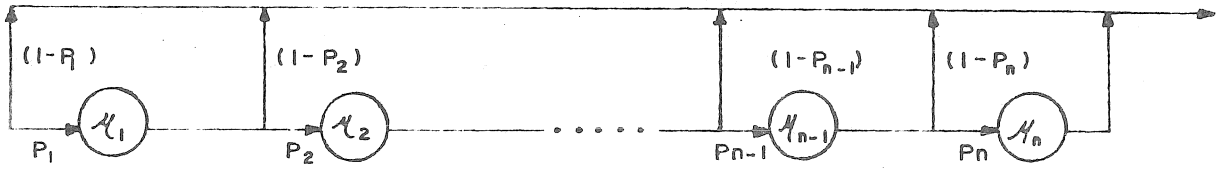
donde:

Pi: Probabilidad de ocurrencia de la exponencial i

i: Media de la exponencial asociada con la etapa i

Figura 1.1-2

RED EXPONENCIAL

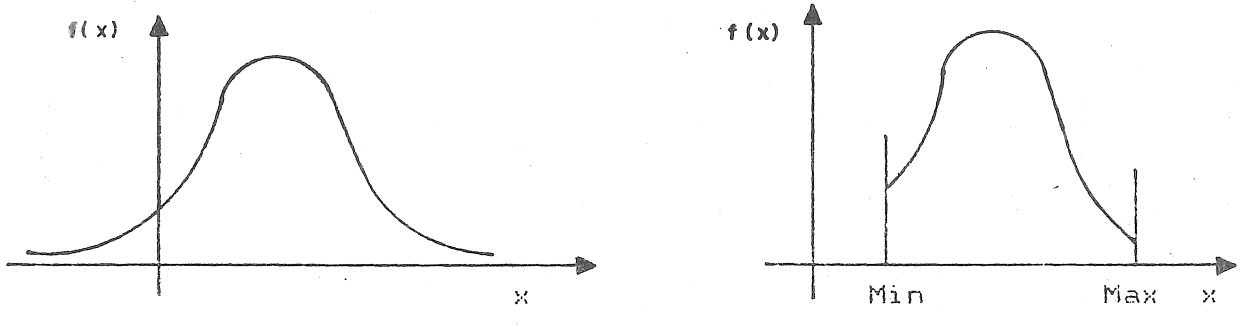


donde:

Pi: Probabilidad de entrar a la exponencial número i.

i: Media de la exponencial asociada con la etapa i

Figura 1.1-3



Original

Acomodada

Figura 1.1-4

1.1.4.1. Tipo de Truncación

Es importante anotar que todas las distribuciones, a excepción de la constante, la uniforme y la discreta, son acomodadas entre un máximo y un mínimo, establecidos por el usuario, de la siguiente manera: si el valor obtenido de la distribución es menor que el mínimo, el programa devuelve el mínimo; y si el valor es mayor que el máximo, el programa devuelve el máximo. El menor valor posible de ese mínimo es cero y el mayor valor posible de ese máximo es infinito. En la figura 1.1.4. se muestra una distribución en su forma original y en su forma acomodada.

1.2. Explicación Del Programa SIMCOL

El programa SIMCOL es gobernado por el método de simulación llamado próximo evento. Bajo este método, el reloj del tiempo simulado avanza en cada oportunidad una cantidad igual a la que hay hasta que ocurre el próximo evento.

Este proceso de avance del reloj ocurre hasta que al menos uno de los dos siguientes eventos tiene lugar:

- El tiempo del próximo evento es mayor que el tiempo de duración de la simulación.

- El número de clientes que salen de la red alcanza un número dado, el cual es especificado por el usuario, si la red es abierta; si la red es cerrada, la simulación termina cuando el número de clientes servidos en un nodo, especificado por el usuario, alcanza un valor dado, el cual también es especificado por el usuario.

Para terminar esta breve explicación del programa SIMCOL, es importante mencionar que el programa fue escrito en lenguaje FORTRAN, utilizando la versión del computador personal de la IBM.

2. EJEMPLO

El proposito de esta sección es presentar un ejemplo de una red de colas cerrada con recurso pasivo y los resultados obtenidos mediante SIMCOL. En la figura 3.2.1-1 se muestra el dibujo de la red de este ejemplo. Esta red cerrada es un modelo de un sistema de computación en el cual hay 10 usuarios en igual número de terminales. Dadas las propiedades de estos clientes, estos se han agrupado en dos grupos: tipo de cliente 1 y 2, con 7 y 3 clientes respectivamente. En este sistema, cada usuario piensa por un momento, teclea el comando y espera la respuesta. Una vez recibida la respuesta, el usuario repite este ciclo. El tiempo para pensar y teclear tiene una distribución exponencial con media 3 segundos para el tipo de cliente 1 y 3.5 segundos para el tipo de cliente 2. Cada comando requiere en promedio 10 ciclos de proceso CPU-I/O. Para poder entrar a dichos ciclos de proceso CPU-I/O, debe haber una asignación de 1 o 2 particiones de memoria para los tipos de clientes 1 y 2 respectivamente, de las 6 particiones de memoria que hay inicialmente. La disciplina del CPU es PS y tiene una tasa de servicio exponencial con media 50 y 70 ms. (milisegundos) para los

tipos de clientes 1 y 2 respectivamente. Cada disco o dispositivo I/O es escogido con una probabilidad igual a .5 y tiene una disciplina Fifo. La tasa de servicio de cada disco es exponencial con media 60 y 30 ms. para los tipos 1 y 2 respectivamente. Esta red se simuló durante 111741 milisegundos.

3.CONCLUSIONES

Un simulador de redes de colas que sea al mismo tiempo de amplio alcance y fácil de usar es un aporte importante al proceso de difusión de métodos de análisis con una amplia aplicabilidad en diversos campos tales como sistemas de computación, telecomunicaciones, producción, etc.

4. Bibliografía

4.1. Barkakati, N. (1986). TCL-A FORTRAN-Based Pointer Threaded Command Language for Simulation. SIMULATION 46:4, 160-163.

4.2. Clark, R. (1985). A Pseudorandom Number Generator. SIMULATION. 45:5, 252-255.

4.3. Hooper, J. (1986). Strategy-related Characteristics of Discrete Event Languages and Models. SIMULATION 46:4, 153-159.

4.4. Kleijnen, J. (1986). Selecting Random Number Seeds in Practice. SIMULATION 47:1, 15-17.

4.5. Pritsker, A. (1984). Introduction to Simulation and SLAM II. West Lafayette: System Publishing Corporation.

4.6. Torn, A. (1985). Simulation Nets, a Simulation Modeling and Validation Tool. SIMULATION 45:2, 71-75.

4.7. Sauer, C y Chandy, M. (1981). Computer Systems Performance Modeling. Englewood Cliffs: Prentice-Hall, Inc.

4.8. Zuñiga, M. (1984). Modelos de Redes de Colas: Análisis y Programación de Métodos Aproximados de Solución. Bogotá: Universidad de Los Andes.

EJEMPLO: RED CERRADA CON RECURSO PASIVO

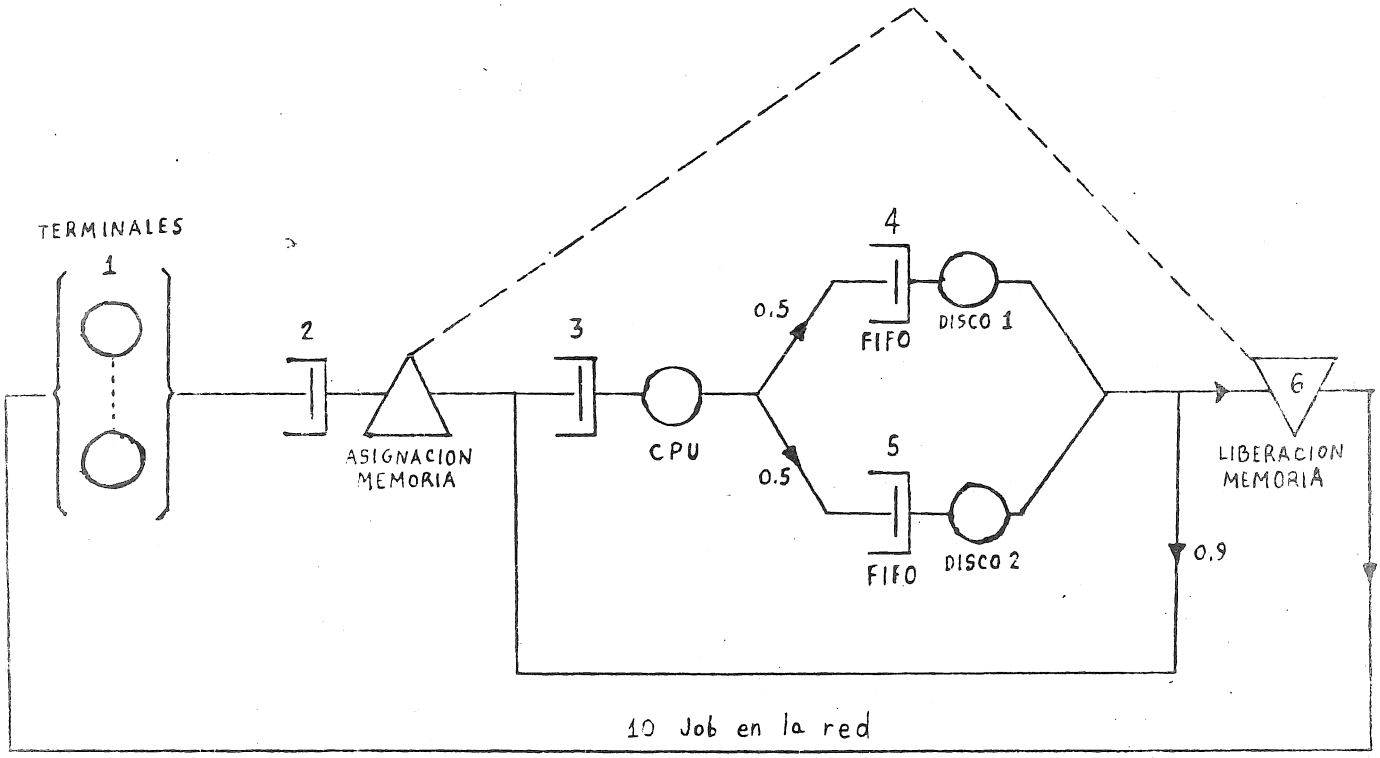


FIGURA 2 - 1

ESTADISTICAS DE LOS CENTROS DE SERVICIO

CENTRO DE SERVICIO NUMERO 1 (FIFO) CLIENTES EN GENERAL

THRUPUT .16306E-02

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.50199E+01	.43708E+01	.30695E+04
COLA	.00000E+00	.00000E+00	.00000E+00

NUMERO DE SERVIDORES ACTIVOS	MEDIA	VARIANZA
	.50199E+01	.43708E+01

HISTOGRAMA DEL NUMERO DE CLIENTES EN EL CENTRO

FRECUENCIA RELATIVA	FRECUENCIA ACUMULADA	LIMITE SUPERIOR DE LA CELDA	0 20 40 60 80 100
			+ + + + + + + + + + + +
.0053	.0053	.00000E+00	+
.0321	.0374	.10000E+01	+*C
.0823	.1197	.20000E+01	+**** C
.1320	.2517	.30000E+01	+******** C
.7483	1.0000	INFINITO	+**********C
			+ + + + + + + + + + + +
			0 20 40 60 80 100

CENTRO DE SERVICIO NUMERO 1 (FIFO) TIPO DE CLIENTE NUMERO 1

THRUPUT .12046E-02

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.36112E+01	.25123E+01	.29906E+04
COLA	.00000E+00	.00000E+00	.00000E+00

NUMERO DE SERVIDORES ACTIVOS	MEDIA	VARIANZA
	.36112E+01	.25123E+01

CENTRO DE SERVICIO NUMERO 1 TIPO DE CLIENTE NUMERO 2

THRUPUT .42599E-03

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.14087E+01	.95320E+00	.32926E+04
COLA	.00000E+00	.00000E+00	.00000E+00

NUMERO DE SERVIDORES ACTIVOS	MEDIA	VARIANZA
	.14087E+01	.95320E+00

CENTRO DE SERVICIO NUMERO 2 (ASIG. REC.PAS.) CLIENTES EN GENERAL

THRUPUT .16297E-02

	LONGITUD		TIEMPO
	MEDIA	VARIANZA	PROMEDIO
COLA	.10493E+01	.16179E+01	.64379E+03

NUMERO DE SERVIDORES	MEDIA	VARIANZA
ACTIVOS	.50142E+01	.21218E+01

HISTOGRAMA DEL NUMERO DE CLIENTES EN EL CENTRO

FRECUENCIA RELATIVA	FRECUENCIA ACUMULADA	LIMITE SUPERIOR DE LA CELDA	0 20 40 60 80 100
.4823	.4823	.00000E+00	+**********C +
.2054	.6877	.10000E+01	+**********C +
.1605	.8482	.20000E+01	+********C +
.0998	.9480	.30000E+01	+******C +
.0520	1.0000	INFINITO	+***C
			0 20 40 60 80 100

CENTRO DE SERVICIO NUMERO 2 (ASIG. REC. PAS.) TIPO DE CLIENTE NUMERO 1

THRUPUT .12037E-02

	LONGITUD		TIEMPO
	MEDIA	VARIANZA	PROMEDIO
COLA	.54138E+00	.86936E+00	.44967E+03

NUMERO DE SERVIDORES	MEDIA	VARIANZA
ACTIVOS	.28474E+01	.15233E+01

CENTRO DE SERVICIO NUMERO 2 (ASIG. REC. PAS.) TIPO DE CLIENTE NUMERO 2

THRUPUT .42599E-03

	LONGITUD		TIEMPO
	MEDIA	VARIANZA	PROMEDIO
COLA	.50790E+00	.50139E+00	.11923E+04

NUMERO DE SERVIDORES	MEDIA	VARIANZA
ACTIVOS	.21668E+01	.16759E+01

TABLA 2-1

CENTRO DE SERVICIO NUMERO 3 (PS) CLIENTES EN GENERAL

THRUPUT .15945E-01

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.26174E+01	.22028E+01	.16415E+03

NUMERO DE SERVIDORES ACTIVOS	MEDIA	VARIANZA
	.89313E+00	.95445E-01

HISTOGRAMA DEL NUMERO DE CLIENTES EN EL CENTRO

FRECUENCIA RELATIVA	FRECUENCIA ACUMULADA	LIMITE SUPERIOR DE LA CELDA	0 20 40 60 80 100
			+ + + + + + + + + + + +
.1069	.1069	.00000E+00	+*****C +
.1446	.2515	.10000E+01	+******** C +
.1920	.4435	.20000E+01	+********** C +
.2332	.6766	.30000E+01	+************ C +
.3234	1.0000	INFINITO	+**************** C
			+ + + + + + + + + + + +
			0 20 40 60 80 100

CENTRO DE SERVICIO NUMERO 3 (PS) TIPO DE CLIENTE NUMERO 1

THRUPUT .11759E-01

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.17893E+01	.16405E+01	.15216E+03

NUMERO DE SERVIDORES ACTIVOS	MEDIA	VARIANZA
	.59199E+00	.11619E+00

CENTRO DE SERVICIO NUMERO 3 (PS) TIPO DE CLIENTE NUMERO 2

THRUPUT .41856E-02

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.82812E+00	.48163E+00	.19785E+03

NUMERO DE SERVIDORES ACTIVOS	MEDIA	VARIANZA
	.30115E+00	.85113E-01

TABLA 2-1

CENTRO DE SERVICIO NUMERO 4 (FIFO) CLIENTES EN GENERAL

THRUPUT .80006E-02

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.65052E+00	.93600E+00	.81304E+02
COLA	.24096E+00	.40963E+00	.30118E+02

NUMERO DE SERVIDORES ACTIVOS	MEDIA	VARIANZA
	.40956E+00	.24182E+00

HISTOGRAMA DEL NUMERO DE CLIENTES EN EL CENTRO

FRECUENCIA RELATIVA	FRECUENCIA ACUMULADA	LIMITE SUPERIOR DE LA CELDA	0 20 40 60 80 100
.5904	.5904	.00000E+00	+**********C +
.2517	.8421	.10000E+01	+**********C +
.0995	.9416	.20000E+01	+******C +
.0391	.9807	.30000E+01	+**C+
.0193	1.0000	INFINITO	+*C
			0 20 40 60 80 100

CENTRO DE SERVICIO NUMERO 4 (FIFO) TIPO DE CLIENTE NUMERO 1

THRUPUT .59226E-02

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.52381E+00	.67861E+00	.88435E+02
COLA	.17505E+00	.26268E+00	.29557E+02

NUMERO DE SERVIDORES ACTIVOS	MEDIA	VARIANZA
	.34875E+00	.22712E+00

CENTRO DE SERVICIO NUMERO 4 (FIFO) TIPO DE CLIENTE NUMERO 2

THRUPUT .20780E-02

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.12672E+00	.12313E+00	.60979E+02
COLA	.65910E-01	.67078E-01	.31718E+02

NUMERO DE SERVIDORES ACTIVOS	MEDIA	VARIANZA
	.60805E-01	.57108E-01

TABLA 2-1

CENTRO DE SERVICIO NUMERO 5 (FIFO) CLIENTES EN GENERAL

THRUPUT .79407E-02

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.66292E+00	.93365E+00	.83444E+02
COLA	.24657E+00	.40284E+00	.31012E+02

NUMERO DE SERVIDORES	MEDIA	VARIANZA
ACTIVOS	.41635E+00	.24300E+00

HISTOGRAMA DEL NUMERO DE CLIENTES EN EL CENTRO

FRECUENCIA RELATIVA	FRECUENCIA ACUMULADA	LIMITE SUPERIOR DE LA CELDA	0 20 40 60 80 100
			+ + + + + + + + + + + +
.5837	.5837	.00000E+00	+**********C +
.2520	.8356	.10000E+01	+**********C +
.1047	.9403	.20000E+01	+******C +
.0410	.9813	.30000E+01	+**C+
.0187	1.0000	INFINITO	+*C
			+ + + + + + + + + + + +
			0 20 40 60 80 100

CENTRO DE SERVICIO NUMERO 5 (FIFO) TIPO DE CLIENTE NUMERO 1

THRUPUT .58340E-02

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.53435E+00	.68345E+00	.91562E+02
COLA	.18155E+00	.26627E+00	.31089E+02

NUMERO DE SERVIDORES	MEDIA	VARIANZA
ACTIVOS	.35280E+00	.22833E+00

CENTRO DE SERVICIO NUMERO 5 (FIFO) TIPO DE CLIENTE NUMERO 2

THRUPUT .21067E-02

	LONGITUD MEDIA	LONGITUD VARIANZA	TIEMPO PROMEDIO
SISTEMA	.12857E+00	.12771E+00	.60964E+02
COLA	.65020E-01	.67718E-01	.30799E+02

NUMERO DE SERVIDORES	MEDIA	VARIANZA
ACTIVOS	.63549E-01	.59510E-01

TABLA 2-1

CENTRO DE SERVICIO NUMERO 6 (LIB. REC. PAS.) CLIENTES EN GENERAL

THRUPUT .16252E-02

CENTRO DE SERVICIO NUMERO 6 (LIB. REC. PAS.) TIPO DE CLIENTE NUMERO 1

THRUPUT .12001E-02

CENTRO DE SERVICIO NUMERO 6 (LIB. REC. PAS.) TIPO DE CLIENTE NUMERO 2

THRUPUT .42509E-03

TABLA 2-1

REDES

Y

SISTEMAS DISTRIBUIDOS

PROTOCOLOS DE CONVERSION PARA EL INTERCAMBIO DE INFORMACION ENTRE BASES DE DATOS BIBLIOGRAFICAS HETEROGENEAS.

JOSE ABASOLO PRIETO
Grupo de Investigación SINBAD
Depto. de Ingeniería de Sistemas
Universidad de los Andes
Apartado aéreo 4976
Bogotá-Colombia.

RESUMEN:

Se presenta una solución al problema de cómo brindar transparencia hacia la heterogeneidad a los usuarios de una Base de Datos distribuida heterogénea con información bibliográfica. La solución resuelve el problema de traducción entre diferentes representaciones de datos adoptando un formato puente a través del cual debe pasar toda conversión.

1. DESCRIPCION GENERAL.

El intercambio de información entre Bases de Datos bibliográficas automatizadas que no siguen los mismos estándares de clasificación y que son manejadas por diversos paquetes de 'software' crea la necesidad de conversión de datos entre los diferentes formatos y estándares utilizados.

Una alternativa de solución al problema de la heterogeneidad sería desarrollar convertidores entre cada par de Bases de Datos. Esta solución se descarta por su complejidad: a cada nueva Base de Datos que se quiera integrar al sistema habría que desarrollarle convertidores con cada una de las Bases ya existentes.

La alternativa correcta consiste en adoptar un **Modelo Global** (formato y estándar de clasificación) que sirva de puente entre los otros. Bajo este esquema solo se requiere un convertidor entre el Modelo Global y cada Base de Datos perteneciente al sistema [CERI 84].

El presente artículo describe la situación actual y la proyección futura del proyecto **"Protocolos de Traducción bibliográfica desde/hacia el Formato Común de Comunicación Colombiano (FCCC)"** [ABAS 87] en el cual se han desarrollado protocolos de conversión que utilizan un Modelo Global y permiten el intercambio de información entre Bases de Datos bibliográficas automatizadas en microcomputadores IBM PC XT/AT y manejadas por uno cualquiera de los siguientes tres paquetes de 'software': ISIS, SCIMATE ó SCIB. Dichos protocolos han sido desarrollados en el marco de un convenio entre la Universidad de los Andes y el Fondo Colombiano de Investigaciones Científicas y Proyectos Especiales "Francisco José de Caldas" (Colciencias).

Una característica importante de los protocolos es que los programas que los soportan constan de un gran núcleo, **independiente de los paquetes de 'software'** que se estén utilizando, más unos componentes muy sencillos específicos a cada uno de esos paquetes. La integración al sistema de intercambio de Bases de Datos bibliográficas manejadas por paquetes distintos a los tres mencionados antes - por ejemplo NOTIS, LIBRUNAM, etc.- requerirá únicamente del desarrollo de los componentes que tengan en cuenta los detalles específicos del paquete en cuestión. Dichos componentes representan menos del 10% del sistema total, y generalmente son muy sencillos de desarrollar.

Otro aspecto importante de los protocolos es que son **independientes del tipo de información manejada**: en lugar de registros bibliográficos se podrían manejar historias clínicas, información de catastro, inventario de recursos naturales, etc., siempre y cuando se disponga de un Modelo Global para cada uno de esos tipos de información. Este aspecto asegura un gran campo de aplicación del proyecto "FCCC".

Este artículo se ha organizado de la siguiente manera:

En la sección 2 se describen las características principales, relacionadas con el problema de intercambio, de los tres paquetes de 'software' contemplados inicialmente en el proyecto "FCCC": SCIB, SCIMATE e ISIS.

La sección 3 presenta el FCCC (Formato Común de Comunicación Colombiano), utilizado como Modelo y formato puente para facilitar las conversiones.

En la sección 4 se describe un formato de entrada/salida a/de los convertidores, llamado Formato Interno, que fué necesario definir para poder independizar esos convertidores de los paquetes de 'software' utilizados.

La sección 5 explica en forma general el procedimiento que debe seguirse para poder integrar una nueva Base de Datos al sistema.

En la sección 6 se presentan los protocolos actuales de intercambio de información utilizando diskettes como sistema de comunicación.

En las secciones 7 y 8 se presentan las extensiones previstas para los protocolos así como algunas conclusiones.

2.CARACTERISTICAS DE LOS MANEJADORES DE LAS BASES DE DATOS LOCALES.

La clasificación, estructuración y manipulación de los datos de una Base bibliográfica dependen del paquete de 'software' escogido para manejarla. En las siguientes secciones se describen los aspectos clasificación y estructuración para los tres paquetes incluidos actualmente en el proyecto "FCCC".

2.1. SCIB.

Todas las Bases de Datos bibliográficas administradas por el paquete SCIB (Sistema Colombiano de Información Bibliográfica) utilizan como estándar de clasificación el formato MARC (Machine Readable Cataloging) diseñado por la Biblioteca del Congreso de los Estados Unidos, con algunas modificaciones. Bajo dicho forma-

to los elementos de información se agrupan en campos de longitud fija ó variable. Cada campo tiene una etiqueta que identifica el tipo de elemento de información que aparece en dicho campo. Un campo puede ser repetible y contener a su vez subcampos. Cada subcampo es identificado por un conjunto de caracteres especiales, y puede ser repetible.

La etiqueta de un campo se complementa con dos indicadores que en algunos casos sirven para especificar en más detalle el tipo de información contenida en el campo, y en otros casos se utilizan para facilitar procesos administrativos tales como producción de fichas catalográficas, indización, etc.

Ejemplo: en SCIB, el título de una obra se codifica en el campo con etiqueta 245. Dicho campo puede contener dos subcampos que corresponden a subtítulo y mención de autoría. El subcampo subtítulo se identifica con el caracter especial ':' mientras que el subcampo mención de autoría se identifica con el caracter '/' si es la primera vez que aparece y con ';' para los demás. Un ejemplo de cómo se codificaría un campo título es:

24510EDGAR WALLACE : THE MAN WHO MADE HIS NAME

El 10 que viene después de la etiqueta 245 representa los valores que toman los indicadores en este caso específico. No hay menciones de autoría asociadas a este título.

SCIB permite importar un archivo de texto para integrarlo a una Base de Datos local cuando dicho archivo de texto se ajusta a un formato específico que refleja el modelo SCIB. Igualmente una Base de Datos SCIB puede exportar un archivo de texto en ese formato.

2.2 SCIMATE.

Las Bases de Datos manejadas por SCIMATE no están obligadas a seguir un estándar de clasificación específico. El formato al que se deben ajustar es el siguiente:

Para una misma Base de Datos se pueden definir varias máscaras ('templates') para los registros. Cada registro incluye un identificador de la máscara que utiliza. Un registro es una agrupación de campos no repetibles de longitud variable. El máximo número de campos en un registro es 20 y la longitud máxima de un registro es 1894 caracteres. Cada campo tiene una etiqueta que identifica el tipo de información que contiene dicho campo. Un campo puede contener subcampos identificados por un conjunto de caracteres especiales, pero esta subdivisión es transparente para SCIMATE.

Ejemplo:

En una Base de Datos particular se pueden tener registros bibliográficos correspondientes a monografías, seriadas o audiovisuales. Los datos que describen una monografía no son los mismos que describen una seriada o un audiovisual. Se pueden tener tres máscaras : una para monografías, una para seriadas y otra para audiovisuales.

La máscara para monografías podría tener los siguientes campos: título con etiqueta **TI**; autor personal con etiqueta **AP**; autor institucional con etiqueta **AI**; editor con etiqueta **ED**; y fecha de publicación con etiqueta **FP**. El campo autor personal puede contener dos subcampos: en el uno se coloca el apellido y en el otro el nombre. El subcampo apellido se identifica porque aparece al comienzo del campo. El subcampo nombre se precede por una coma y un espacio. Ejemplo:

AP GARCIA MARQUEZ, GABRIEL

Para SCIMATE es transparente el hecho de que después de la coma venga el nombre: todo lo que venga en el campo con etiqueta AP es el autor personal, sin importarle cómo se subdivide la información al interior.

SCIMATE permite importar/exportar archivos de texto hacia/desde una Base de Datos SCIMATE, siempre y cuando los archivos de texto se ajusten a un formato específico que refleja el modelo SCIMATE.

2.3 ISIS.
Las Bases de Datos manejadas por ISIS tampoco están obligadas a seguir un estándar de clasificación específico. El formato que impone es el siguiente:

Un registro es una agrupación de campos repetibles de longitud variable pero con un máximo definido. Cada campo tiene una etiqueta que identifica el tipo de información que contiene dicho campo. Un campo puede contener subcampos repetibles identificados por un caracter especial y una letra. Los subcampos son manejados por ISIS. Adicionalmente, un campo o subcampo puede contener otros subcampos repetibles identificados por unos caracteres especiales, pero estos últimos subcampos son transparentes para ISIS: solo las personas responsables de la Base de Datos conocerán de su existencia.

Ejemplo:

En una Base de Datos particular el título y sus menciones de autoría se van a codificar en el campo con etiqueta **10**. Cada mención de autoría será un subcampo con identificador ^a. Un ejemplo de codificación sería:

10Asterix in Switzarland^atext by Goscinny^adrawings by Underzo^atranslated by Anthea Bell and Dereck Hockridge

Si además del título hubiera subtítulo, éste se podría poner después del título, separado por dos puntos (:). En este caso habría dos subcampos -título y subtítulo- que no serían manejados como tales por ISIS, pero que serían interpretados como subcampos por el responsable de la Base de Datos.

ISIS permite importar/exportar archivos en formato ISO 2709 hacia/desde una Base de Datos ISIS.

3. MODELO GLOBAL: EL FCCC.

Como Modelo Global se escogió una adaptación del CCF de Unesco [SIMM 84].

El CCF -The Common Communication Format- es un formato que se ajusta a los estándares ISO y fué diseñado, entre otras razones, para permitir el intercambio de registros bibliográficos entre grupos de Bibliotecas. Los estándares de clasificación que establece son diferentes a los de MARC.

En CCF un registro bibliográfico consta de información de control, un directorio, y un conjunto de campos formados por subcampos. Campos y subcampos tienen identificadores que indican la información que contienen. Unos y otros pueden ser repetibles y los textos que contienen son de longitud variable. Un campo puede tener indicadores que especifican en más detalle el tipo de información contenida. Un registro puede dividirse en segmentos, y se pueden establecer encadenamientos entre dichos segmentos ó entre campos de un segmento. Los valores posibles de algunos subcampos tales como nivel bibliográfico, idioma, rol, tipo de material, etc., han sido codificados.

La adaptación del CCF al caso colombiano, que fué bautizada con el nombre de Formato Común de Comunicación Colombiano -FCCC-, simplificó el formato de los registros bibliográficos, eliminando directorios, por considerar que en un formato de transporte es importante minimizar el volumen de datos transportado y que la información de directorios es relevante para el almacenamiento eficiente de datos pero no para el transporte. También se suprimieron algunos campos y/o subcampos y se añadieron otros. Se eliminaron los indicadores, la segmentación y los encadenamientos. Las tablas de códigos para ciertos subcampos también fueron modificadas.

Un registro FCCC tiene la siguiente estructura:

- Separador de registro
- Nivel bibliográfico
- Etiqueta de campo
- Identificador de subcampo
- Texto de subcampo
- [Separador de subcampo]*
- [Identificador de subcampo
- [Texto de subcampo
- [Separador de campo]*
- [Etiqueta de campo
- [Identificador de subcampo
- [Texto de subcampo
- [Separador de subcampo]*
- [Identificador de subcampo
- [Texto de subcampo

Para ilustrar los conceptos de campo y subcampo en el FCCC tomemos como ejemplo el título de una obra y sus menciones de autoría: Se codifican en el campo con etiqueta 200. El título propiamente dicho es el subcampo con identificador @A. Las menciones de autoría, repetibles, constituyen cada una un subcampo con identificador @B. Un ejemplo de esta codificación sería:

200@AAsterix in Switzerland@Btext by Goscinny@Bdrawings by Underzo@Btranslated by Anthea Bell and Dereck Hockridge

Dado que el FCCC debe poder cambiar para adaptarse a las nuevas necesidades de intercambio de información, se desarrolló un módulo que permite definirlo y editarlo dinámicamente. La responsabilidad de la definición y de las modificaciones ha sido centralizada en Colciencias, quién será la encargada de distribuir las versiones actualizadas del formato a los responsables de las Bases de Datos que participen en el programa de intercambio.

4. FORMATO INTERNO.

La solución adoptada para independizar los programas convertidores de los paquetes de 'software' utilizados para manejar las Bases de Datos locales, fué introducir un formato interno que incluyera las características importantes de los formatos utilizados por SCIB, ISIS y SCIMATE: cualquier registro producido por alguno de esos paquetes es pasado a formato interno antes de entrar al convertidor hacia FCCC; a la inversa, todo registro en FCCC es convertido a formato interno antes de pasarse al formato del paquete manejador de la Base de Datos que lo va a recibir.

En formato interno un registro bibliográfico se estructura de la siguiente manera:

- Separador de registro
- Máscara (vacio si no hay máscara)
- Separador de etiqueta
- Etiqueta
- Separador de subcampo
- Identificador de subcampo (vacio si no hay subcampo)
- Separador de texto
- Texto de subcampo

Para cada uno de los posibles paquetes manejadores de las Bases de Datos locales que van a intercambiar información es necesario desarrollar programas especiales que conviertan del formato de exportación/importación de cada uno de ellos hacia/desde el formato interno. Esta conversión es puramente sintáctica.

5. PASOS INICIALES PARA QUE UNA BASE DE DATOS SE INTEGRE AL SISTEMA DE INTERCAMBIO.

Para que una Base de Datos pueda integrarse al sistema de intercambio es necesario que su responsable defina cómo y dónde se codifican en su Base de Datos cada uno de los elementos de información del Modelo Global (FCCC). Este proceso de definición lo llamaremos **especificación de equivalencias.**

5.1 ESPECIFICACION DE EQUIVALENCIAS Y EJEMPLO.

5.1.1 ESPECIFICACION DE EQUIVALENCIAS.

La especificación de equivalencias se hace interactivamente, utilizando un módulo que para cada elemento del FCCC pide la siguiente información:

```
MASCARA:          ETIQUETA:       SUBCAMPO:
DELIMITADORES INICIALES(POSICION):
DELIMITADORES FINALES(LONGITUD):
VALOR:
DIRECCION DE LA CONVERSION:
-------------------------------------------------------
condición:
ETIQUETA:         SUBCAMPO:
POSICION(DELIMITADORES INICIALES):
LONGITUD(DELIMITADORES FINALES):
VALOR:
```

Si el paquete local es SCIMATE, que permite múltiples máscaras para una misma Base de Datos, hay que dar el nombre de la **máscara** donde se encuentra el elemento de información. Si el elemento se puede encontrar en varias máscaras, se dará una equivalencia por cada una de ellas. Si el paquete local no maneja múltiples máscaras, MASCARA se deja en blanco.

A continuación se debe dar la **etiqueta** del campo donde se encuentra el elemento. Si se puede dar en **más de** un campo pero **éstos** se excluyen mutuamente, se colocan las etiqueta de esos campos, separados por '|'. Si el elemento se puede encontrar en varios campos no excluyentes, se da una equivalencia por campo.

Si el paquete local es ISIS, que permite el manejo explícito de subcampos, y el elemento de información se encuentra al interior de un subcampo, se coloca el identificador de dicho **subcampo**. Si no se está en ISIS ó el elemento no se encuentra en un subcampo, SUBCAMPO se deja en blanco.

Luego se especifica entre qué conjuntos de caracteres especiales (**delimitadores iniciales y finales**) se puede encontrar el elemento en cuestión. En lugar de caracteres especiales se puede especificar una posición fija y una longitud.

En caso de que el elemento esté codificado en la Base de Datos local, se debe especificar qué código (**valor**) le corresponde. Si son varios, se debe especificar una equivalencia para cada uno.

En **Dirección de la Conversión** se indica si la equivalencia es válida únicamente en la dirección formato local->formato puente, si es válida solo en la dirección formato puente->formato local, ó si es válida en ambas direcciones.

En algunos casos un dato que se encuentra en un campo, entre ciertos delimitadores, corresponde a un elemento del FCCC si y solo si en algún otro campo (que puede ser el mismo) aparece(n) cierto(s) valor(es). Esto se especificará en forma de **condición**, indicando la localización y valores posibles de los datos condicionantes.

Una vez especificadas las equivalencias, éstas serán compiladas y guardadas como tablas específicas de esa Base de Datos. Dichas tablas se utilizarán posteriormente cada vez que se vaya a enviar ó recibir información por intercambio. Las equivalencias pueden ser modificadas y recompiladas en cualquier momento.

5.1.2 EJEMPLO.

Para el elemento de información "fecha de copyright normalizada" del FCCC, en una Base de Datos manejada por SCIB la especificación de equivalencia será:

```
MASCARA:        ETIQUETA:008     SUBCAMPO:
DELIMITADORES INICIALES(POSICION):6
DELIMITADORES FINALES(LONGITUD):4
VALOR:
DIRECCION DE LA CONVERSION:3 (formato local <->formato puente)
-----------------------------------------------------------------
condición:
ETIQUETA:008     SUBCAMPO:
POSICION(DELIMITADORES INICIALES):1
LONGITUD(DELIMITADORES FINALES):1
VALOR:c
```

En la equivalencia se está diciendo que la fecha de copyright normalizada, cuando aparece, se encuentra en el campo con etiqueta 008 en la posición fija 6, y tiene una longitud de 4 caracteres. Sinembargo, es posible que lo que se encuentre allí no sea la fecha de copyright normalizada sino otra fecha. Para cerciorarse de de ésto hay que mirar la posición 1 del mismo campo: si allí hay una c, entonces la fecha que está en la posición 6 es efectivamente la deseada. La equivalencia debe operar en ambas direcciones.

5.2 ESPECIFICACIONES ADICIONALES.

Dado que el formato puente maneja tablas de códigos para idioma del documento y rol de un autor, el responsable de la Base de Datos que se desea integrar al sistema de intercambio deberá indicar, utilizando unos módulos del sistema, la equivalencia entre los códigos que él maneja y los códigos del formato puente.

Además de especificar equivalencias, el responsable de cada Base de Datos manejada por SCIMATE ó ISIS deberá hacer unas definiciones adicionales:

- Para las Bases de Datos en SCIMATE se deberá indicar el orden en que pueden aparecer los campos al interior de cada máscara . Si un registro muy largo se puede partir en dos ó más, hay que indicar el campo que se replica para hacer el encadenamiento lógico entre esos registros;

- Para las Bases de Datos en ISIS se deberán especificar los identificadores de los campos con modo de inversión 2 ó 3.

Para SCIB se deben especificar los valores por defecto que se deben dar a los indicadores que acompañan a cada etiqueta cuando se vaya a recibir información.

6. PROTOCOLO ACTUAL DE INTERCAMBIO.

A falta de una red de comunicaciones de gran alcance que interconecte las Bases de Datos que participan en el proyecto FCCC, el intercambio de información entre Bases de Datos se está haciendo actualmente a través de diskettes.

El protocolo que se sigue es el siguiente:

A) SOLICITUD Y ENVIO DE INFORMACION (figura 1):

1. El responsable de una Base de Datos solicita información a su homólogo en otra Base de Datos. Esta solicitud puede ser verbal ó escrita.

2. El responsable de la Base de Datos a quien se solicita la información hace la consulta utilizando el lenguaje del paquete local (ISIS, SCIMATE ó SCIB). El conjunto de registros producidos se deja en un archivo tipo exportación (por ejemplo utilizando la opción de exportar archivos en formato ISO 2709 de ISIS).

3. Se utiliza un módulo específico al paquete local para convertir el archivo del formato de exportación al formato interno estándar.

4. Se utilizan los módulos que convierten el archivo en formato interno a otro archivo en el formato puente (FCCC). Para hacer la conversión, los módulos consultarán las tablas de equivalencias de la Base de Datos de la cual se extrajo la información. Como en un mismo computador puede haber varias Bases de Datos, cada una con sus tablas de equivalencias propia, los módulos preguntarán el nombre de la Base de Datos específica.

5. Una vez convertido al formato puente, el archivo será copiado a diskettes.

6. Los diskettes son enviados al responsable que hizo la solicitud de registros bibliográficos.

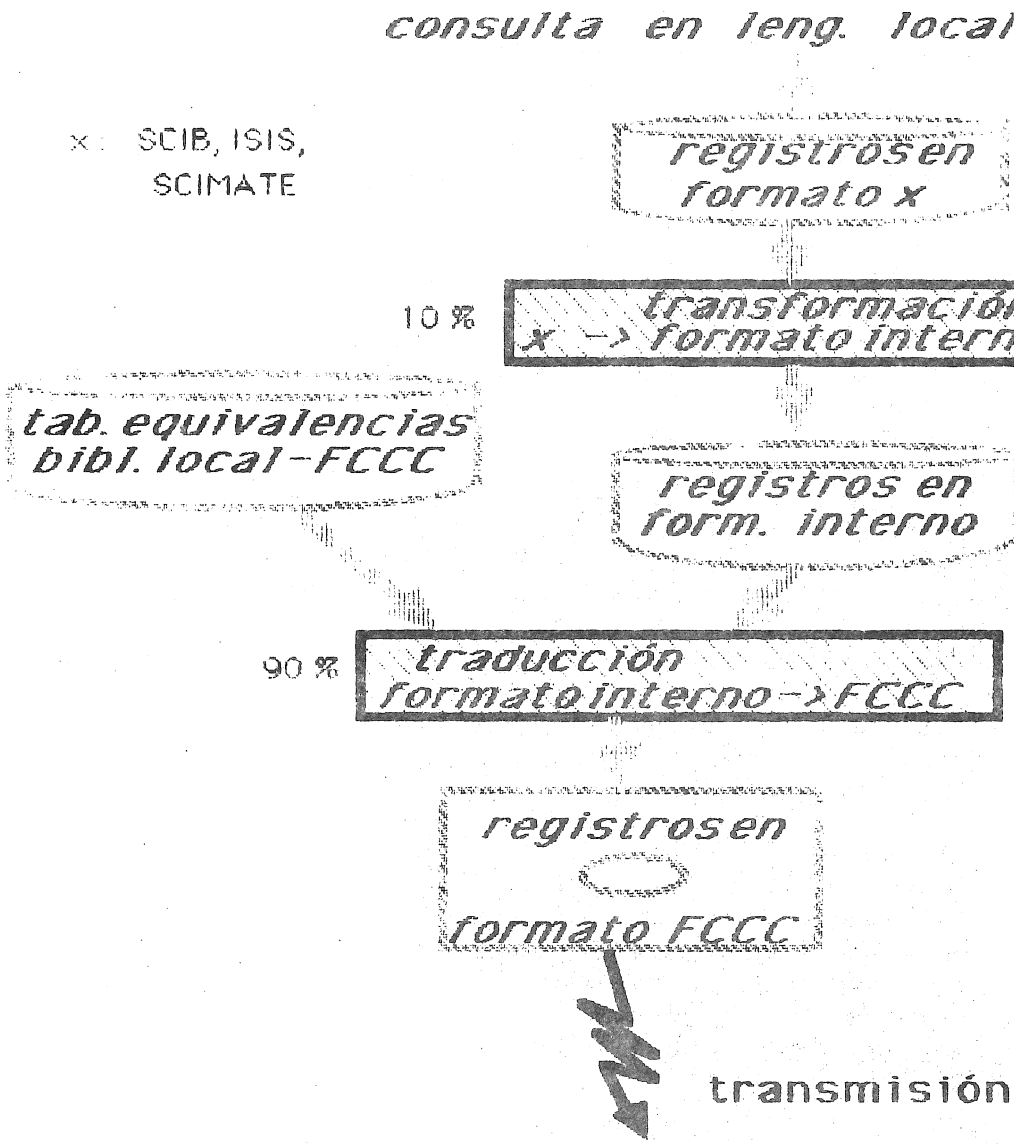


FIG. 1 : SOLICITUD Y ENVIO DE INFORMACION

B) RECEPCION Y EVENTUAL INTEGRACION DE LA INFORMACION (figura 2):

7. El responsable que hizo la solicitud copia en disco duro el archivo que recibió en diskettes.

8. Se utilizan los módulos que convierten el archivo en formato puente (FCCC) a otro archivo en formato interno estándar. Para hacer la conversión los módulos consultarán las tablas de equivalencia de la Base de Datos local a la cual, eventualmente, se integrarían los registros recibidos.

9. Se utiliza un módulo específico al paquete local (ISIS, SCIB ó SCIMATE) para convertir el archivo en formato interno a otro archivo en el formato de importación del paquete local.

10. El responsable que hizo la solicitud examina el archivo recibido, ya en su formato local, lo edita y, eventualmente, lo integra parcial o totalmente a su Base de Datos.

7. DESARROLLOS FUTUROS.

Se está empezando a trabajar en convertidores FCCC<->CCF que permitan intercambiar información bibliográfica con otros paises que utilicen el CCF como formato de transporte. También se planea desarrollar versiones de los programas del proyecto "FCCC" para máquinas diferentes de IBM PC XT/AT.

Una vez instalada la red pública de transmisión de datos colombiana -COLDAPAQ- se piensa montar un servicio de consulta en línea a un conjunto de Bases de Datos Distribuidas Heterogéneas que ofrezca transparencia tanto a la heterogeneidad de las Bases de Datos participantes como a la localización física de los datos. La información bibliográfica estará en una de esas Bases de Datos Distribuidas. Otras Bases podrán contener información estadística, de salud, factual, etc. Cada consulta será enrutada automáticamente a la Base de Datos correspondiente.

Para poder montar tal sistema de consulta en línea será necesario definir un lenguaje de consulta global, así como resolver problemas ligados a la confidencialidad de los datos, tarificación de los servicios, etc.

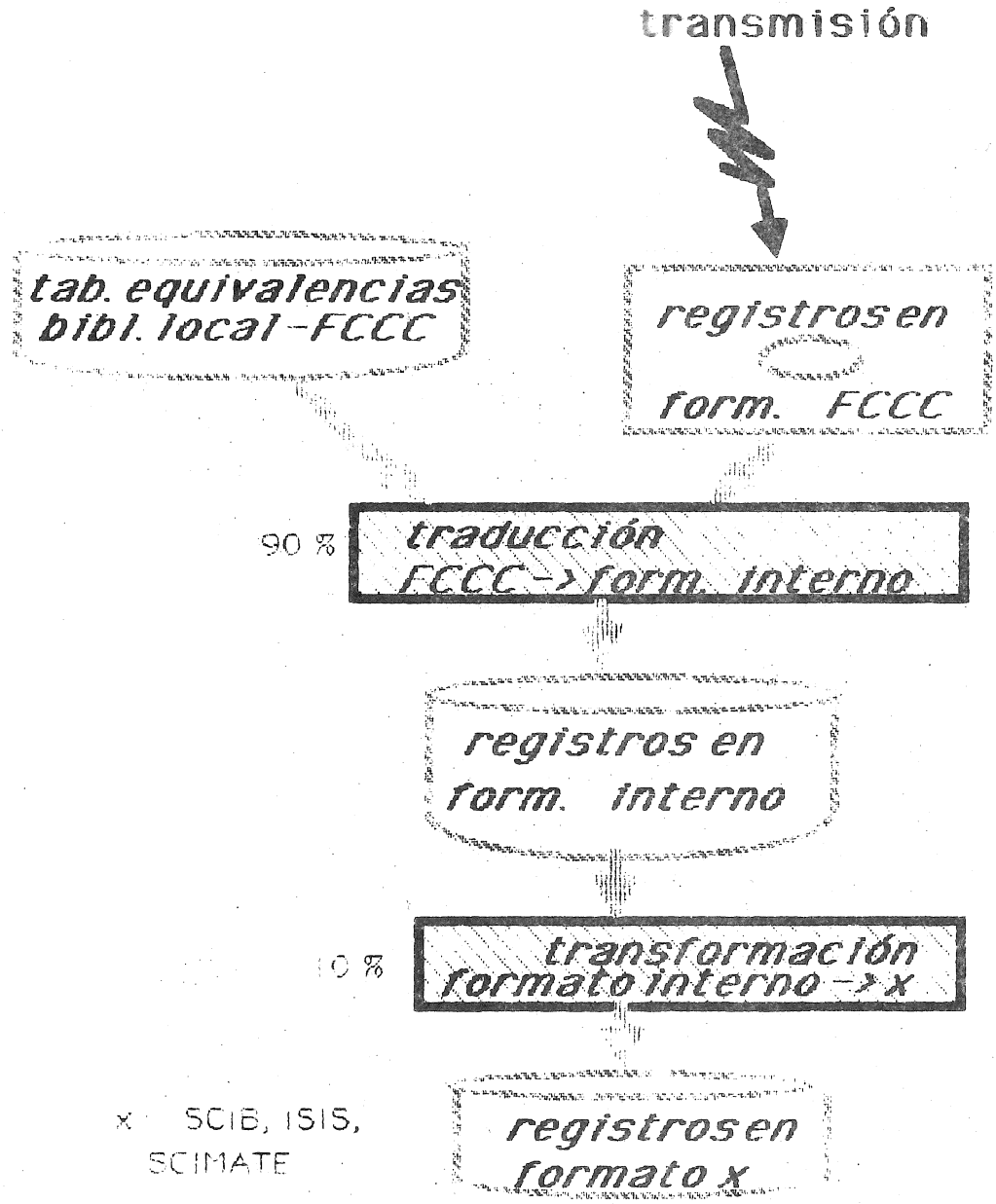


FIG. 2 : RECEPCION Y EVENTUAL INTEGRACION DE LA INFORMACION

8. CONCLUSIONES.

Los protocolos presentados en este artículo resuelven, para el caso específico de la información bibliográfica, el problema de cómo brindar transparencia hacia la heterogeneidad a los usuarios de una Base de Datos Distribuida heterogénea.

Esos protocolos podrían utilizarse también en la integración de Bases de Datos no bibliográficas, siempre y cuando exista un Modelo Global para el tipo de información que se esté integrando.

En futuros desarrollos se espera resolver el problema de cómo brindar transparencia a la localización física de los datos, integrando , por tipo de información, muchas Bases de Datos automatizadas que existen en el país, utilizando como sistema de comunicación la red pública de transmisión de datos colombiana COLDAPAC.

9. BIBLIOGRAFIA.

[ABAS 87] Abásolo, José; Franky, Ma. Consuelo; Muñoz Adriana
"Protocolos de traducción bibliográfica desde/hacia el Formato Común de Comunicación Colombiano (FCCC)"
Informe final preliminar proyecto 1204-03-098-85
Colciencias-Universidad de los Andes. Junio 1987.

[CERI 84] Ceri, Stefano; Pelagatti, Giuseppe
"Distributed Databases: Principles and Systems"
McGraw-Hill, 1984

[SIMM 84] Simmons, Peter; Hopkinson, Alan
"CCF: The Common Communication Format"
United Nations Educational, Scientific and Cultural Organization, 1984.

TESEO : AMBIENTE DE PROGRAMACION DE SISTEMAS TRANSACCIONALES DISTRIBUIDOS

MARIA CONSUELO FRANKY

Grupo de investigación SINBAD
Dpto. de Ingeniería de Sistemas
Universidad de los Andes
Apartado aéreo 4976
Bogotá - Colombia

RESUMEN :

Se presenta un sistema que permite al diseñador de una Base de Datos Distribuida especificar el esquema de fragmentación y distribución de los datos. Además le permite programar las transacciones que podrán ser solicitadas por los usuarios finales; esta programación se hace en términos de la base de datos global, tal como se haría para una base de datos centralizada. A partir de las especificaciones del diseñador, el sistema genera el protocolo de ejecución de las transacciones en un ambiente distribuido asegurando atomicidad y control de concurrencia. Este Ambiente de Programación de Sistemas Transaccionales Distribuidos "TESEO" es entonces un Manejador de Bases de Datos Distribuidas que permite que el diseñador se despreocupe totalmente de las técnicas de procesamiento distribuido y pueda concentrarse en los aspectos propios de su aplicación.

1. INTRODUCCION

TESEO: Ambiente de Programación de Sistemas Transaccionales Distribuidos es un sistema en desarrollo en la Universidad de los Andes, actualmente en fase de prototipo, cuyo propósito es automatizar la especificación e implementación de aplicaciones transaccionales que van a operar en ambientes distribuidos (i.e. redes locales o redes geográficamente dispersas).

TESEO permite al diseñador especificar el esquema de fragmentación y de distribución de los datos que maneja la aplicación. Le permite además programar las transacciones que desea ofrecer a los usuarios finales. Para programar transacciones el diseñador no requiere conocer técnicas de procesamiento distribuido, solo las técnicas usuales del manejo de bases de datos.

La implementación de una aplicación específica a través de TESEO implica 3 etapas : en la etapa de *especificación* el diseñador especifica los esquemas de la base de datos distribuida y programa las transacciones; en la etapa de *generación* el sistema genera un programa que permite ejecutar esas transacciones en un ambiente distribuido; finalmente en la etapa de *ejecución* los usuarios finales solicitan transacciones desde los distintos sitios que conforman el ambiente distribuido.

El presente artículo está dividido en 8 partes : a continuación de esta introducción, la parte 2 presenta las características actuales del sistema considerado como Sistema Manejador de Bases de datos Distribuidas. La parte 3 presenta los módulos en que está dividido el sistema. Las partes 4 y 5 explican cómo el diseñador interactúa con el sistema para especificar los esquemas global, de fragmentación y de distribución de los datos y los algoritmos de las transacciones. La parte 6 explica la generación del programa que implementa el protocolo de ejecución distribuida de lastransacciones a partir de las especificaciones del diseñador. En las partes 7 y 8 se presentan las extensiones previstas para este sistema así como algunas conclusiones.

2. CARACTERISTICAS ACTUALES DEL SISTEMA

TESEO es un Sistema Manejador de Bases de Datos Distribuidas (SMBDD) con las siguientes características (en referencia a la arquitectura de B.D.D. presentada en *[Cer 84]*):

ESQUEMA GLOBAL :

La Base de datos Global sigue el modelo Relacional; el diseñador debe entonces especificar las relaciones globales que va a manejar la aplicación.

ESQUEMA DE FRAGMENTACION :

El sistema permite al diseñador fragmentar las relaciones globales de forma horizontal simple, es decir, por agrupación de tuplas que según el valor de sus atributos deben estar juntas para la aplicación.

ESQUEMA DE DISTRIBUCION :

El sistema permite al diseñador asignar cada fragmento a uno o varios sitios del ambiente distribuido. Para efectos de actualización una de las copias se considera "primaria" y las demás "secundarias".

SISTEMA MANEJADOR DE BASE DE DATOS LOCAL :

El sistema asume **homogeneidad** de los SMBD locales y por lo tanto no requiere Esquemas de Traducción. En la versión actual se implementó para cada sitio un SMBD sencillo capaz de realizar las operaciones básicas de consulta y actualización de fragmentos locales. Estas operaciones permiten insertar, modificar y eliminar una tupla de un fragmento local, así como seleccionar y proyectar las tuplas de un fragmento que cumplen con determinada condición.

Se ha previsto que en el futuro TESEO se acople con diversos SMBD comerciales disponibles para los sitios del ambiente distribuido.

NIVEL DE TRANSPARENCIA OFRECIDA AL DISEÑADOR :

Una vez que el diseñador especifica los Esquemas Global, de Fragmentación y de Distribución, puede programar transacciones en términos de las relaciones globales, es decir, de forma transparente a la fragmentación y distribución de datos.

En la versión descrita en este artículo el lenguaje que utiliza el diseñador para programar transacciones es **Turbo-Pascal** con servicios provistos por TESEO para manipular relaciones globales.

FUNCIONES AUTOMATICAS INCLUIDAS EN LA APLICACION GENERADA :

El sistema TESEO incorpora en la aplicación generada las siguientes funciones :

a- asignación de estampillas a las transacciones que se ejecutan asegurando un orden total
b- selección de fragmentos que participan en una consulta
c- distribución de instrucciones de cada transacción a los sitios participantes
d- atomicidad y recuperación de transacciones bajo un protocolo de validación
e- actualización consistente de todas las copias de un fragmento
f- control de concurrencia de transacciones utilizando candados y listas de espera para asegurar la serialización legal de transacciones
g- prevención de abrazos mortales

3. MODULOS DEL SISTEMA

La Figura 1 de la página siguiente muestra los módulos componentes del Ambiente de Programación de Sistemas Transaccionales Distribuidos TESEO.

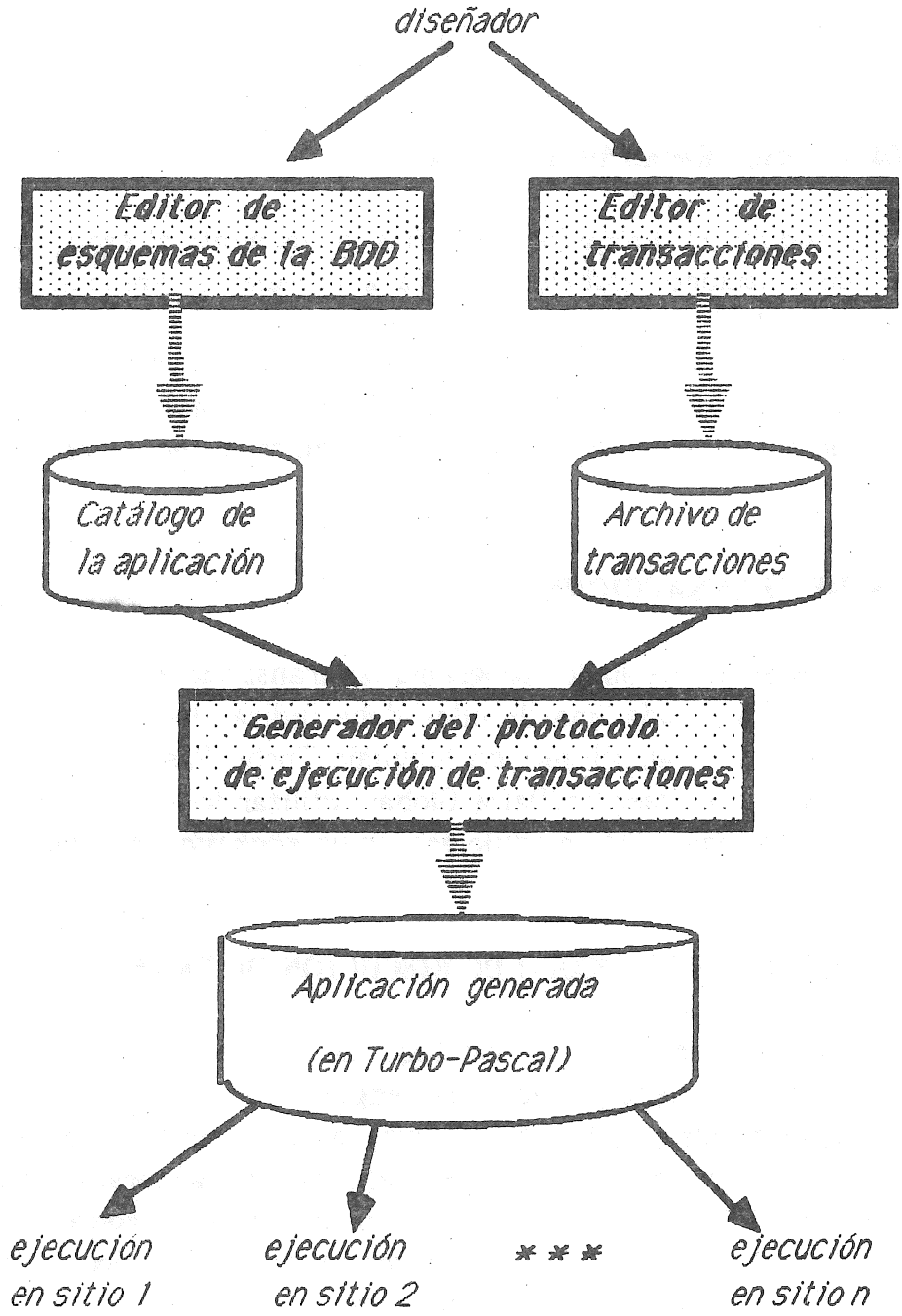


Figura 1 : MODULOS DE TESEO

Una explicación breve de los módulos es la siguiente :

Módulo **EDITOR DE ESQUEMAS DE LA B.D.D.** :

A través de este módulo el diseñador especifica inicialmente el esquema global de la B.D.D., es decir, las relaciones globales que va a manejar la aplicación. En seguida especifica el esquema de fragmentación de esas relaciones dando fórmulas de selección de tuplas para constituir fragmentos. Por último especifica el esquema de distribución asignando cada fragmento a uno o varios sitios lógicos. Este módulo genera el **catálogo** de la aplicación que posteriormente se va a duplicar en cada uno de los sitios.

Módulo **EDITOR DE TRANSACCIONES** :

A través de este módulo el diseñador programa las transacciones que quiere ofrecer a los usuarios finales. Para cada una de ellas debe indicar los parámetros que debe suministrar el usuario y el algoritmo en lenguaje Turbo-Pascal. Los algoritmos se expresan en términos de la Base de Datos Global utilizando los servicios del sistema para manipular las relaciones. Este módulo genera un **archivo de transacciones.**

Módulo **GENERADOR DEL PROTOCOLO DE EJECUCION DE TRANSACCIONES:**

Este módulo toma como entrada el catálogo de la aplicación y el archivo de transacciones y genera un programa fuente en lenguaje Turbo-Pascal, con la lógica para poder ejecutar transacciones en ambiente distribuido asegurando atomicidad y control de concurrencia. En la versión actual de TESEO el programa generado incluye un nivel de comunicación específico para una red local de microcomputadores conectados en anillo; en esta red se asume que a cada microcomputador (sitio físico) corresponde un único sitio lógico. En el futuro se planea generalizar el nivel de comunicación para que el diseñador pueda especificar la interfaz de TESEO con los servicios de comunicación que ofrezca la red específica en la cual va a operar la aplicación.

Para tener el sistema en funcionamiento, debe compilarse el programa resultante y ponerse en ejecución en cada uno de los sitios.

4. ESPECIFICACION DE ESQUEMAS DE LA BASE DE DATOS DISTRIBUIDA

Cuando se programa una aplicación con TESEO, el diseñador debe empezar por definir el **Esquema Global** especificando las relaciones globales que va a manejar la aplicación: para cada relación debe indicar su nombre, sus atributos y los tipos correspondientes; también debe indicar cúal es la llave (uno o más atributos). Por ejemplo, en una aplicación bancaria se especificaría la relación *CUENTAS* así :

atributos de *CUENTAS* :

código	: tipo	*entero*	*(llave)*
saldo	:	*real*	
propietario	:	*cadena*	
saldo inicial	:	*real*	

A continuación el diseñador debe especificar el **Esquema de Fragmentación** indicando para cada relación cómo deben agruparse las tuplas en fragmentos a través de fórmulas de selección. El objetivo aquí es agrupar tuplas que deben estar juntas desde el punto de vista de la aplicación; sin embargo, se sabe que hacerlo de manera óptima es un problema muy complejo de optimización que no tiene solución única *[Ceri-84]*.

La versión actual del sistema permite únicamente fragmentación horizontal simple de cada relación. Por ejemplo, para la relación *CUENTAS* se podrían indicar los siguientes fragmentos :

$CUENTAS_1$	:	tuplas con *saldo inicial* $\geq$ 1.000.000
$CUENTAS_2$	:	tuplas con (*saldo inicial* $<$ 1.000.000) y (*código* $\leq$ 100)
$CUENTAS_3$	:	tuplas con (*saldo inicial* $<$ 1.000.000) y (100 $<$ *código* $\leq$ 200)
$CUENTAS_4$	:	tuplas con (*saldo inicial* $<$ 1.000.000) y (*código* $>$ 200)

En la versión actual del sistema es reponsabilidad del diseñador asegurar una

fragmentación correcta de cada relación, es decir, cada tupla debe quedar asignada exactamente a un solo fragmento. De esta forma los fragmentos son disyuntos y su unión es igual a la Base de Datos Global.

En el futuro se piensan incluir esquemas más generales de fragmentación, como es la fragmentación horizontal derivada, la vertical y combinaciones de las anteriores.

Finalmente el diseñador debe especificar el **Esquema de Distribución** de cada relación, asignando cada fragmento a uno o a varios sitios lógicos del ambiente distribuido en el cual va a ejecutarse la aplicación. Por ejemplo, los fragmentos de *CUENTAS* podrían ser asignados así :

$CUENTAS_1$: $sitio_1$, $sitio_2$, $sitio_3$, $sitio_4$
$CUENTAS_2$: $sitio_2$
$CUENTAS_3$: $sitio_3$
$CUENTAS_4$: $sitio_4$

En el caso de un fragmento replicado su primera copia se denomina "**primaria**" y las demás "**secundarias**".

Una vez especificados por el diseñador los Esquemas Global, de Fragmentación y de Distribución, el sistema genera el **Catálogo de la aplicación** que posteriormente se replicará en cada uno de los sitios de ejecución. Este catálogo permitirá primordialmente la localización de datos que participan en una transacción solicitada por el usuario final.

5. ESPECIFICACION DE TRANSACCIONES

En la versión del sistema descrita en este artículo el lenguaje ofrecido al diseñador para programar transacciones es **Turbo-Pascal**.

El diseñador debe comenzar por enumerar las transacciones que podrán ser solicitadas por los usuarios finales : para cada una debe indicar nombre, parámetros y tipos. Por ejemplo, en la aplicación bancaria se podrían ofrecer, entre otras, las siguientes

transacciones:

transacción *ABRIR CUENTA* :
parámetros *nuevo-código* : tipo *entero*
nombre-de-propietario : tipo *cadena*
saldo-al-comenzar : tipo *real*

transacción *TRANSFERENCIA* :
parámetros *cuenta-fuente* : tipo *entero*
cuenta-destino : tipo *entero*
cantidad : tipo *real*

Conocidas las diferentes transacciones, el sistema TESEO incluye en la aplicación generada un "despachador de comandos" *[Toro 87]* que se encargará de ofrecer al usuario un menú con las diversas transacciones disponibles. En tiempo de ejecución, cada vez que el usuario final seleccione una transacción, el despachador de comandos pedirá los parámetros respectivos, activará la transacción respectiva y posteriormente presentará al usuario los resultados de ésta.

En un futuro se prevee brindar al diseñador la posibilidad de escoger entre divesos tipos de despachadores *[Toro 87]* para diversos tipos de terminales, e inclusive, permitir que el usuario programe su propio despachador.

Una vez enumeradas las transacciones, el diseñador debe especificar el algoritmo en Turbo-Pascal de cada una de ellas : el diseñador programa las transacciones en términos de las relaciones globales como si la Base de Datos fuera centralizada, despreocupándose aquí de la fragmentación y distribución de los datos. Para lograr esta transparencia, la programación de la manipulación de relaciones globales debe hacerse únicamente a través de procedimientos "**servicios**" provistos por el sistema TESEO . Estos servicios y sus parámetros son básicamente los siguientes :

- Servicio **Insertar** **(nombre de relación, valor de atributos)**
  Genera el código para que en tiempo de ejecución se inserte una nueva tupla en la relación nombrada. En ejecución la aplicación generada deducirá entonces el fragmento de la relación en el cual debe insertarse la nueva tupla y se encargará de hacer la inserción en cada una de las copias de ese fragmento.

  Es importante resaltar que el diseñador especifica el valor de cada atributo indicando la constante, variable o parámetro que contiene el valor. El nombre la relación debe ser una constante.

- Servicio **Eliminar** **(nombre de relación, valor de llave)**
  Genera el código para que en tiempo de ejecución se elimine la tupla correspondiente a la llave especificada de la relación nombrada. En ejecución la aplicación generada eliminará la tupla de todas las copias del fragmento al cual pertenece. En forma similar al servicio anterior, el diseñador especifica el valor de la llave indicando la constante, variable o parámetro que contiene tal valor; el nombre de la relación debe ser una constante.

- Servicio **Modificar** **(nombre de relación, valor de llave, nombres de atributos que cambian, nuevos valores de atributos)**
  Genera el código para que en tiempo de ejecución se modifique la tupla de la relación nombrada correspondiente a la llave especificada. Así, en ejecución la aplicación generada modificará la tupla especificada en todas las copias del fragmento al cual pertenece. El nombre de la relación y los nombres de los atributos que se quieren modificar deben ser constantes mientras que el valor de la llave y los nuevos valores de los atributos pueden ser constantes, variables o parámetros.

- Servicio **Selección-Proyección** **(nombre de relación, fórmula de selección, atributos de proyección, nombre de archivo respuesta)**
  Genera el código para que en tiempo de ejecución en el archivo respuesta se copien las tuplas de la relación nombrada que cumplen la fórmula de selección (dada en forma normal conjuntiva). En ejecución la aplicación generada deducirá los fragmentos a los cuales pertenecen las tuplas y realizará la selección en una copia (i.e. sitio) de cada uno de esos fragmentos. Hecho ésto, la aplicación generada efectuará la proyección sobre los atributos nombrados y los almacenará en el archivo de respuesta.

  Si la fórmula de selección es nula el servicio equivale a una proyección simple; de manera análoga, si los atributos de proyección son nulos el servicio equivale a una selección simple.

A continuación se da como ejemplo el algoritmo dado por el diseñador para la transacción *TRANSFERENCIA* mencionada al comienzo de la sección 4. Se observará el uso del caracter " / " para separar los distintos valores que pueden ir en un parámetro, por ejemplo, los nombres de los atributos de proyección en un servicio Selección_Proyección, o los atributos que conforman la llave en un servicio Modificar.

```
transacción TRANSFERENCIA  ( cuenta_fuente  : integer;
                             cuenta_destino : integer ;
                             cantidad       : real);

    Var      respuesta                 : file of real;
             saldo_actual, nuevo_saldo : real;

    Procedure Avisar ( texto : string );
    begin
          writeln; writeln; writeln;
          writeln ( '          ', texto )
    end Avisar;

begin
    Selección_proyección ( CUENTAS, código = cuenta_fuente, /saldo/, respuesta ) ;
    read ( respuesta, saldo_actual );
    If   saldo_actual ≥ cantidad  then
         begin
            nuevo_saldo := saldo_actual - cantidad ;
            Modificar ( CUENTAS, /cuenta_fuente/, /saldo/, /nuevo_saldo/);
            Selección_proyección ( CUENTAS, código = cuenta_destino,
                                   /saldo/, respuesta );
            read ( respuesta, saldo_actual) ;
            nuevo_saldo := saldo_actual + cantidad ;
            Modificar ( CUENTAS, /cuenta_destino/, /saldo/,/nuevo_saldo/ )
         end
    else Avisar ( ' transacción rechazada : saldo insuficiente ')
end
fin_transacción
```

Como se puede observar en el ejemplo anterior, las transacciones pueden incluir procedimientos locales (como *Avisar)* ; sin embargo, el sistema impone como restricción que estos procedimientos no incluyan servicios de manipulación de las relaciones globales.

6. GENERACION DEL PROTOCOLO DE EJECUCION DE TRANSACCIONES PARA AMBIENTE DISTRIBUIDO

Una vez que el diseñador ha interactuado con TESEO para especificar los esquemas de la B.D.D. y las transacciones, el sistema invoca el **Generador del Protocolo de Ejecución de Transacciones.** Este módulo toma como entrada el catálogo de la aplicación y el archivo de transacciones, producidos por los módulos anteriores, y genera la aplicación que habrá de ser ejecutada en cada uno de los sitios del ambiente distribuido.

6.1 CARACTERISTICAS DEL PROTOCOLO DE EJECUCION DE TRANSACCIONES

Cuando se genera la aplicación distribuida cada una de las transacciones que había especificado el diseñador es descompuesta automáticamente en diversos trozos. Estos trozos están básicamente delimitados por las invocaciones a los servicios de manipulación de la base de datos global. Los diversos trozos obtenidos y las invocaciones a los servicios de la base de datos son entonces "sumergidos" dentro de un esquema de ejecución que asegura los siguientes aspectos:

ATOMICIDAD Y RECUPERACION DE TRANSACCIONES :

Una vez que una transacción comienza a ejecutarse por demanda de un usuario final se asegura su **atomicidad**, es decir, sus efectos se realizarán o bien totalmente (si no hay fallas del sistema), o bien ninguno de ellos (en caso falla). Para lograr esta atomicidad cada transacción se ejecuta siguiendo un **protocolo de validación en 2 etapas** : de manera general, en una primera etapa la transacción reserva los datos que necesita y construye "listas de intención" en cada uno de los sitios participantes, incluyendo en esas listas las modificaciones que se desean hacer a los datos locales; en la segunda etapa la transacción modifica efectivamente los datos y los libera. Este protocolo permite entonces anular los efectos de la transacción por simple destrucción de las listas de intenciones si ocurre una falla que interrumpa la transacción durante su primera etapa *[Le Lann 81]*.

En el caso de que algunos de los datos modificados por una transacción sean fragmentos con copias se aplica una adaptación del protocolo de *[Stonebraker 79]* : durante la primera etapa se reservan los fragmentos primarios y se construyen listas de

intenciones en los sitios correspondientes; durante la segunda etapa se difunden las intenciones a los fragmentos secundarios y se modifican efectivamente. Cuando un sitio estuvo aislado al recuperarse debe actualizar sus fragmentos secundarios pidiendo las modificaciones faltantes a los sitios donde se encuentran los fragmentos primarios.

Para consultar un fragmento que participa en una Selección_Proyección se reserva el fragmento primario; adicionalmente, si existe un fragmento secundario a nivel local, éste también se reserva y la consulta se realiza sobre el fragmento local.

CONTROL DE CONCURRENCIA DE TRANSACCIONES :

En la aplicación generada se asegura el control de concurrencia de transacciones distribuidas que tratan de accesar los mismos datos en uno o varios sitios. Para ello se sigue un protocolo de control parcialmente distribuido que incluye :

* uso de **estampillas** para identificar transacciones (y sus mensajes correspondientes) asegurando un orden total, según el método de *[Lamport 78]*.
* uso de **candados** de reservación y **listas de espera** en cada sitio para serializar de forma consistente las transacciones que quieren accesar los mismos datos; se logran así las propiedades definidas por *[Eswaran 76]* : serialización "legal" de transacciones "bien-formadas" y a "dos-fases".
* **prevención de abrazos mortales** entre transacciones aplicando el método "herir o esperar" de *[Rosenkratz 78]* el cual implica anular una transacción que durante su primera etapa entra en conflicto con otra transacción de estampilla menor.

SIMULACION DE PROCESOS PARALELOS:

La aplicación generada implementa el protocolo de ejecución de transacciones a través de **varios procesos por sitio**, encargados de asegurar la atomicidad y control de concurrencia entre transacciones. En la versión actual TESEO genera aplicaciones para Sistemas Operacionales monoproceso (CP/M, MS-DOS); en consecuencia, el programa generado para cada sitio contiene un núcleo de control que da turnos sucesivos a los diversos procesos de acuerdo a los mensajes que van llegando al sitio. Más específicamente, la programación de estos procesos se hizo siguiendo una metodología derivada de las redes gráficas de Nutt, la cual se presenta ampliamente en *[Franky 86]*.

INCLUSION DE UN NIVEL DE COMUNICACION SIMPLE :

El programa generado incluye un nivel de comunicación simple que permite intercambiar mensajes entre sitios lógicos asociados a microcomputadores conectados en topología de anillo. En un futuro se planea que TESEO disponga de diversas librerías de comunicación que se adapten a diferentes estándares y topologías de redes.

6.2 PROCESOS GENERADOS

El programa generado para cada sitio corresponde a la ejecución simultánea de 3 procesos que aseguran los objetivos del protocolo de ejecución de transacciones; estos procesos se sincronizan por mensajes utilizando el nivel de comunicación para enviar y recibir mensajes, ya sea a nivel local (procesos dentro de un mismo sitio) o a nivel remoto (procesos dentro de sitios diferentes).

La figura 2 de la siguiente página ilustra la relación entre procesos de un sitio y las principales estructuras de datos que manejan. Las funciones principales de los procesos son las siguientes :

PROCESO DESPACHADOR :

Está encargado de realizar la **interfaz con el usuario y de coordinar cada transacción** demandada por él.

El proceso Despachador muestra por pantalla el Menú de Transacciones de la aplicación, detecta la clase de transacción que solicita el usuario, captura los parámetros correspondientes y va invocando sucesivamente a los diferentes trozos en que fue descompuesta la transacción. Cuando se invoca alguno de los servicios de manipulación de las relaciones globales (i.e. al final de cada trozo) el Despachador envía el mensaje local correspondiente al proceso Monitor y se queda esperando el mensaje de respuesta para poder proseguir con la ejecución de otro trozo.

Adicionalmente, cuando el proceso Despachador termina los trozos de una transacción difunde el mensaje de validación a los sitios participantes a fin de iniciar la segunda etapa de la transacción en la cual deben modificarse efectivamente los datos. En caso de falla, también es el proceso Despachador el encargado de difundir el mensaje de anulación.

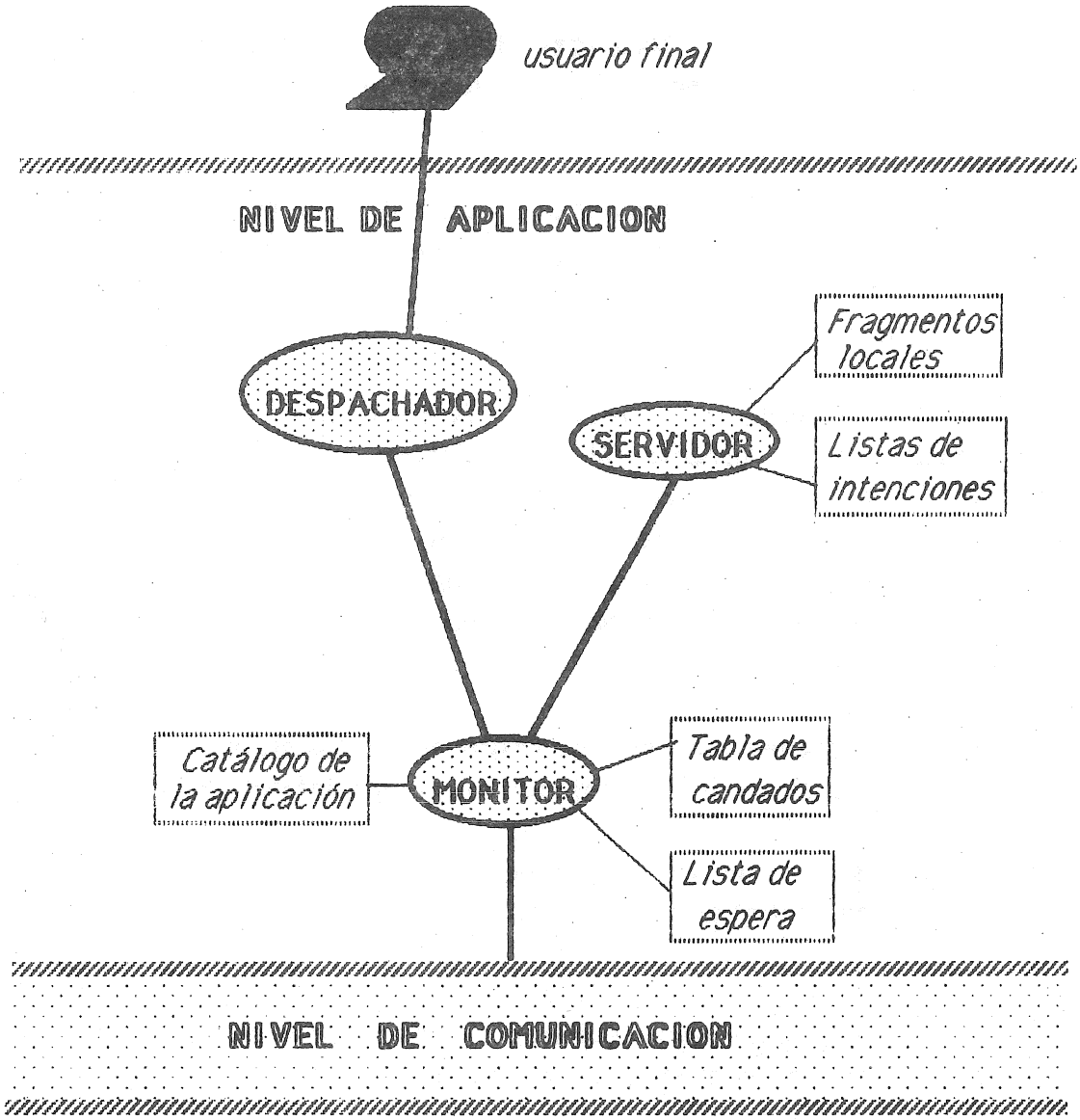


Figura 2 : **PROCESOS GENERADOS Y ESTRUCTURAS DE DATOS QUE MANEJAN**

PROCESO MONITOR :

Este proceso está encargado de **localizar los datos** que intervienen en cada transacción y de **controlar la concurrencia** entre transacciones que quieren accesar datos locales.

Para realizar la primera de estas funciones el proceso Monitor maneja el Catálogo de la Aplicación, el cual representa los esquemas global, de fragmentación y de distribución de los datos de la aplicación. Esta información permite deducir los sitios que deben participar en la resolución de un servicio invocado por una transacción.

Así por ejemplo, cuando el proceso Despachador invoca el servicio *Selección_Proyección*, el proceso Monitor deberá recibir el mensaje local correspondiente. Con base en la información que lleva este mensaje (i.e. la fórmula de selección) y en el Catálogo de la Aplicación el proceso Monitor deberá deducir los sitios participantes. A estos sitios participantes el proceso Monitor envía entonces un mensaje externo *Selección_Proyección* destinado a los procesos Monitores correspondientes.

Posteriormente cuando lleguen mensajes de respuesta de los sitios participantes, en el sitio origen de la transacción se irán reuniendo las tuplas seleccionadas en un archivo local. Finalmente el proceso Monitor pasará un mensaje local de respuesta al proceso Despachador para avisarle que ya se recibieron los datos que había solicitado, y permitirle así que pueda continuar su ejecución.

Para realizar la segunda función relativa al control de concurrencia, el proceso Monitor maneja una Tabla de Candados para registrar el estado de reservación de los datos locales y así poder detectar conflictos de concurrencia. Siguiendo con el ejemplo, cuando un proceso Monitor recibe un mensaje externo *Selección_Proyección* examina la Tabla de Candados para ver si el fragmento local sobre el cual se quiere hacer la selección ya está reservado por otra transacción en modo conflictivo : si no hay conflicto el proceso Monitor registra en la Tabla la nueva reservación del fragmento y envía un mensaje local al proceso Servidor para que éste efectúe el servicio; la respuesta que el proceso Servidor entregue será enviada por el proceso Monitor al sitio origen de la transacción.

En caso de conflicto de concurrencia, el proceso Monitor coloca el mensaje externo de petición de servicio en una Lista de Espera, postergando la entrega del mensaje al proceso Servidor para cuando terminen las transacciones que originan el conflicto. Sin embargo, si una de esas transacciones posee estampilla mayor que la transacción que

está solicitando el servicio, el Monitor provocará la anulación de la primera con el fin de prevenir un abrazo mortal (método "herir o esperar" de *[Rosenkratz 78]*).

PROCESO SERVIDOR :

Este proceso está encargado de **accesar los fragmentos de datos locales**, **manejar las listas de intenciones** y asegurar la **actualización consistente de copias de fragmentos.**

Cuando este proceso Servidor recibe un mensaje de servicio *Selección_Proyección* selecciona tuplas del fragmento local correspondiente y conforma con ellas un archivo de respuesta que el proceso Monitor local deberá transmitir al sitio origen de la transacción. Cuando recibe un mensaje de servicio *Insertar, Eliminar* o *Modificar* registra la intención correspondiente en la Lista de Intenciones, y únicamente acualiza efectivamente los datos cuando recibe el mensaje de Validación de una transacción : solo en ese momento realiza las intenciones correspondientes a esa transacción.

Para el caso de fragmentos replicados, es el proceso Servidor el encargado de difundir las intenciones relativas a un fragmento primario local hacia los sitios donde estén localizadas sus copias secundarias. Esta difusión ocurre después de haber actualizado el fragmento primario. Para asegurar la consistencia mutua entre copias de un fragmento, el proceso Servidor de un sitio que se recupera después de una falla debe solicitar las intenciones acumuladas por el fragmento primario de cada una de los fragmentos secundarios que posea.

6.3 DESCOMPOSICION DE LOS ALGORITMOS DE TRANSACCIONES

En términos generales la programación de los procesos **Monitor** y **Servidor** se hizo enunciando las acciones que se debían realizar cada vez que se recibiera un mensaje de petición de servicio (*Insertar, Eliminar, Modificar* o *Selección_Proyección)* , o un mensaje de respuesta a un servicio.

A diferencia de lo anterior, la programación del proceso **Despachador** debe seguir en cada caso la lógica del algoritmo de la transacción escogida por el usuario, y al mismo tiempo debe generar un mensaje de petición de servicio (para enviar al Monitor) cada vez que en la transacción se accesa la base de datos global. Esta programación plantea entonces el problema de descomponer el algoritmo de cada transacción en una secuencia de "trozos" que serán ejecutados uno a uno, en forma intercalada con el

envío de mensajes de petición de servicios y la recepción de mensajes de respuestas. Más exactamente en cada turno de ejecución el proceso Despachador recibe y procesa el mensaje de respuesta al servicio que solicitó en el turno anterior, ejecuta un trozo según la secuencia establecida, y finaliza el turno enviando al Monitor un mensaje de solicitud de servicio.

La descomposición del algoritmo de cada transacción plantea adicionalmente el problema de la preservación del estado del proceso Despachador cada vez que pierda el turno de ejecución al enviar un mensaje de petición de servicio al Monitor, es decir, al final de cada trozo. En particular, si se está ejecutando un trozo correspondiente a una parte de una estructura de control condicional o iterativa (i.e. IF o WHILE) y se encuentra una petición de servicio, se debe poder registrar en qué estado queda la ejecución para que el siguiente trozo pueda retomarlo después de la recepción y tratamiento del mensaje de respuesta al servicio solicitado.

En el estado actual de TESEO se permite que una transacción invoque servicios de manipulación de la B.D.D. desde una estructura de control condicional o iterativa situada a nivel de anidamiento cero pero no lo permite desde estructuras de control anidadas a niveles mayores.

El siguiente ejemplo ilustra el algoritmo de una transacción y su descomposición en trozos realizada por TESEO. El algoritmo contiene además de instrucciones simples de Turbo-Pascal, una estructura condicional y una estructura iterativa que contienen invocaciones de servicios :

```
transacción T (parámetros .....)
   declaraciones .... ;
begin
   instrucción a ;
   Selección_Proyección (parámetros...) ;
   instrucción b ;
   If condición1  then
        begin
           instrucción c ;
           Modificar (parámetros....) ;
           instrucción d
        end
   else
        begin
            instrucción e
        end  ;
```

```
    instrucción f ;
    While condición2 do
    begin
        instrucción g ;
        Insertar (parámetros....) ;
        instrucción h ;
    end ;
end
fin_transacción
```

Descomposición de la transacción *T* en trozos :

trozo 1 de *T* :

```
begin
    instrucción a ;
    enviar (Monitor, "Selección_Proyección", parámetros ....)
end
```

trozo 2 de *T*:

```
begin
    instrucción b ;
    var1 := condición1 ;
    If var1 then
        begin
            instrucción c ;
            enviar (Monitor, "Modificar", parámetros....)
        end
    else
        begin
            instrucción e ;
            enviar (Despachador, "nulo" )
        end
end
```

trozo 3 de T :

```
begin
   If  var1  then
        begin
            instrucción d
        end ;
   enviar (Despachador, "nulo")
end
```

trozo 4 de *T* :

```
begin
    instrucción f
    enviar (Despachador, "nulo" )
end
```

trozo 5 de *T* :

```
begin
   var2 := condición2 ;
   If  condición2  then
        begin
            instrucción g ;
            enviar  (Monitor, "Insertar", parámetros.....)
        end
   else
        begin
            núm_trozo  :=  núm_trozo + 1 ;
            enviar (Despachador, "nulo" )
        end
end
```

trozo 6 de *T* :

```
begin
    instrucción  h ;
    num_trozo  :=  núm_trozo - 2
    enviar (Despachador, "nulo" )
end ;
```

Aclaraciones :

Para facilitar la descomposición del algoritmo de la transacción, el sistema impone un fin de trozo antes de empezar una estructura de control condicional o iterativa, y al final de las mismas. Cuando ese fin de trozo no coincide con una petición de servicio se genera un envío de mensaje "nulo" dirigido al mismo proceso Despachador para permitir que este proceso pueda retomar el control cuando se le de turno de ejecución (de acuerdo a la metodología de programación que se siguió *[Franky 86]*, se otorga turno de ejecución a un proceso sólo si existe un mensaje destinado a él).

Por otra parte, el proceso Despachador maneja una variable estática *num_trozo* que indica cúal es el trozo que se debe ejecutar al recibir control, una vez que haya tratado el último mensaje de respuesta recibido (en caso del mensaje "nulo" ese tratamiento es también nulo). Teniendo en cuenta que el proceso Despachador incrementa en 1 la variable *num_trozo* antes de ejecutar el trozo indicado, es necesario alterar el valor de esa variable en los trozos correspondientes a una estructura iterativa *While* a fin de seguir la lógica de la estructura (i.e. poder salir del *While* cuando la condición ya no se cumple, o poder retornar al trozo que evalúa la condición para realizar un nuevo ciclo del *While*).

7. EXTENSIONES PREVISTAS PARA EL SISTEMA

La primera versión del sistema TESEO fue implementada en el trabajo de tesis de pregrado de *[Alvarado y Muñoz 86]* . Posteriormente el trabajo de *[Quintero 87]* amplió el protocolo de ejecución de transacciones para incluir el manejo de copias de fragmentos, aspecto no considerado en la versión inicial.

Además se han hecho otros 3 trabajos de tesis que desarrollan aspectos adicionales que no fueron descritos en este artículo :

- El trabajo de *[Morales 87]* implementó el manejo dinámico del catálogo de la aplicación : se trata aquí de ofrecer al diseñador un conjunto de transacciones para modificar dinámicamente el esquema global, de fragmentación o de distribución de una aplicación existente. Este tipo de transacciones permite no solamente la actualización del catálogo (que está replicado en todos los sitios del sistema), sino

también, la migración de datos de un sitio a otro, o la replicación de los datos de un fragmento en otro sitio. Para ejecutar estas transacciones del sistema se aplicó un protocolo adaptado del propuesto por *[Ellis 77]*.

El trabajo de *[Pedraza 87]* añade a la versión inicial del sistema un manejo flexible del control de concurrencia : teniendo en cuenta los diferentes servicios de manipulación de relaciones globales, el sistema maneja candados de reservación no solo a nivel de fragmento sino también a nivel de tupla, elevando así la concurrencia efectiva. Para ello se utilizan los candados de "intención" propuestos por *[Gray 75]*.

Además se ofrece al diseñador 3 opciones para escoger la técnica de tratamiento de abrazos mortales :

* prevención según la técnica "herir o esperar" de *[Rosenkratz 78]* expuesta en este artículo
* prevención por "timeouts" *[Ceri 84]*
* detección por construcción de un grafo global en un sitio "central", con posibilidad de que otro sitio asuma ese rol cuando el primero se aisla (basado en *[Stonebraker 79]*).

El sistema tiene entonces la posibilidad de generar 3 protocolos diferentes de ejecución de transacciones. Para que el diseñador pueda evaluar qué tan acertado es escoger determinado protocolo para su aplicación, el sistema lleva algunas estadísticas sencillas cuando se ejecutan las transacciones (como número de mensajes intercambiados entre sitios).

El trabajo de *[Levy 87a, 87b]* añade a la versión inicial del sistema un lenguaje relacional tipo SQL *[Chamberlin 76]* para que el diseñador pueda expresar más cómodamente los algoritmos de las transacciones. Los algoritmos se pueden entonces expresar en *Turbo-Pascal* de forma transparente a los esquemas de fragmentación y distribución permitiendo además la manipulación de las relaciones globales a través de las siguientes instrucciones relacionales :

INSERT INTO relación
FROM tabla

SELECT atributos *INTO* tabla
FROM relaciones
WHERE fórmula_de_selección

UPDATE relación
SET atributo$_1$ = expresión$_1$, ..., atributo$_n$ = expresión$_n$
WHERE fórmula_de_selección

DELETE relación
WHERE fórmula_de_selección

En este proyecto se preprocesa el texto de los algoritmos de transaciones para transformar cada instrucción relacional en un conjunto de invocaciones a los servicios de más bajo nivel que ofrece la versión inicial del sistema. Se incluye también una optimización de la consulta derivada de cada *SELECT*, *UPDATE* o *DELETE* mediante una combinación de las transformaciones presentadas en *[Ceri-84]*.

Los trabajos realizados alrededor del sistema TESEO son todavía prototipos de experimentación, pues están implementados en microcomputadores con serias limitaciones de memoria, conectados por una red local muy simple y poco eficiente. A corto plazo se planea implementar TESEO en una red local *Ethernet* de equipos *Vax*. Se espera así ofrecer en un mediano plazo un sistema eficiente que pueda ser usado en el diseño de aplicaciones destinadas a organizaciones de tamaño considerable.

A más largo plazo se espera incluir en TESEO las siguientes ampliaciones :

* Encadenar TESEO con diversos SMBD locales, disponibles a nivel comercial, que realicen de forma eficiente el rol del proceso Servidor descrito en este artículo.
* Disponer de diversas librerías de comunicación que permitan adaptar TESEO a diferentes topologías y protocolos de redes.
* Disponer de un repertorio amplio de Despachadores (i.e. manejadores de la interfaz con el usuario) correspondientes a diferentes esquemas de interacción, diversos tipos de pantallas, o bien, permitir que el Diseñador componga su propio Despachador.

8. CONCLUSIONES

El **Ambiente de Programación de Sistemas Transaccionales Distribuidos TESEO** expuesto en este artículo es un ambiente de desarrollo de gran utilidad para el diseñador de aplicaciones sobre Bases de Datos Distribuidas. El diseñador no está obligado a conocer las técnicas de procesamiento distribuido, y puede entonces diseñar las aplicaciones de la misma forma como lo haría para una Base de Datos Centralizada.

Teniendo en cuenta que en Colombia entra a funcionar próximamente la Red Nacional de Datos *COLDAPAQ* se hace necesario en nuestro medio no solo dominar una nueva tecnología de comunicación, sino también reconceptualizar la forma de diseñar y planear sistemas de información para las organizaciones.

El proyecto TESEO se enmarca dentro del objetivo de apoyar y facilitar ese cambio tecnológico fundamental. La meta final es aún más ambiciosa : se desea ofrecer, a través de *COLDAPAQ*, un ambiente completo de diseño, evaluación, simulación, prototipificación y programación de Sistemas de Información Distribuidos.

En la actualidad la implementación de la siguiente etapa de TESEO en la red *Ethernet* de equipos *Vax* depende de la obtención de financiación para este proyecto.

BIBLIOGRAFIA

[Alvarado y Muñoz 86]: Armando Alvarado y Adriana Muñoz,
"Sistema Generalizado de Transacciones Distribuidas sobre Bases de Datos Homogéneas", tesis de pregrado en Ingeniería de Sistemas ISC-86-I-9, Universidad de los Andes, Sept. 1986.

[Ceri 84]: S. Ceri, G. Pelagatti,
"Distributed Databases : principles and systems", McGraw-Hill, 1984.

[Chamberlin 76]: D.D. Chamberlin et al.,
"SEQUEL 2 : A unified approach to Data Definition, Manipulation and Control", IBM Journal of Research and Development, Vol. 20 nº 6, Nov. 1976.

[Ellis 77]: C.A. Ellis,
"Consistency and Correctness of Duplicate Database Systems", Proc. of 6th Symposium on Operating Systems Principles (ACM), Nov. 1977.

[Eswaran 76]: K.P. Eswaran et al.,
"The notions of consistency and predicate locks in a database system", CACM Vol.19 nº 11, Nov. 1976.

[Franky 86]: M. Consuelo Franky,
"Una metodología y ambiente de Programación de Sistemas Distribuidos a partir de redes de Nutt", XII Conferencia Latinoamericana de Informática (CLEI), Montevideo - Uruguay, Nov. 1986.

[Gray 75]: J.N. Gray et al.,
"Granularity of locks and degrees of consistency in a shared data base", R.J. 1654 IBM Research Laboratory, San José - California, Sept. 1975.

[Lamport 78]: L. Lamport,
"Time, clocks and the ordering of events in a distributed system", CACM Vol. 21 nº 7, July 1978.

[Le Lann 81]: G. Le Lann,
"A distributed system for real-time transaction processing", IEEE Computer Magazine, Vol. 14 nº 2, Feb. 1981.

[Levy 87a]: Perla Levy,
"Sistema Generalizado de Transaciones Distribuidas : Compilador de expresiones de lenguaje relacional", tesis de pregrado en Ingeniería de Sistemas ISC-87-I-23, Universidad de los Andes, Sept. 1987.

[Levy 87b]: Perla Levy,
"Compilador de expresiones de lenguaje relacional para el sistema TESEO", XIII-Conferencia Latinoamericana de Informática (CLEI), Bogotá - Colombia, Nov. 1987.

[Morales 87]: Luis Germán Morales,
"Manejo Dinámico de catálogos para el Sistema Generalizado de Transacciones sobre BDD", tesis de pregrado en Ingeniería de Sistemas ISC-86-II-20, Universidad de los Andes, Enero 1987.

[Pedraza 87]: Gilberto Pedraza G.,
"Control de concurrencia eficiente para el Sistema Generalizado de Transacciones sobre BDD", tesis de pregrado en Ingeniería de Sistemas ISC-86-II-24, Universidad de los Andes, Feb. 1987.

[Quintero 87]: José Ignacio Quintero A.,
"Sistema Generalizado de Transacciones Distribuidas : protocolo para datos parcialmente replicados", tesis de pregrado en Ingeniería de Sistemas ISC-86-II-29, Universidad de los Andes, Feb. 1987.

[Rosenkratz 78]: D.J. Rosenkratz et al.,
"System level concurrency control for distributed database systems", ACM Transactions on Database Systems, Vol. 3 nº 2, June 1978.

[Stonebraker 79]: M. Stonebraker,
"Concurrency control and consistency of multiple copies of data in distributed INGRES", IEEE Transactions on Software Eng., Vol. SE-5 nº 3, May 1979.

[Toro 87]: Víctor M. Toro,
"Especificación, Diseño y Prototipificación de Sistemas Interactivos", XIII-Conferencia Latinoamericana de Informática (CLEI), Bogotá - Colombia, Nov. 1987.

COMPILADOR DE EXPRESIONES DEL LENGUAJE RELACIONAL PARA EL SISTEMA TESEO

PERLA LEVY

Grupo de Investigación SINBAD
Dpto. de Ingeniería de Sistemas
Universidad de los Andes
Apartado Aéreo 4976
Bogotá-Colombia

RESUMEN

Tener las facilidades de un lenguaje relacional para manipular una Base de Datos Distribuida es muy importante. En este artículo se presenta un lenguaje relacional inspirado en SQL para el sistema TESEO, y las transformaciones que debe tener cada una de las instrucciones de este lenguaje para quedar en términos de los servicios ofrecidos por el sistema TESEO para manejar Bases de Datos Distribuidas.

Estas transformaciones incluyen optimizaciones al generar los planes de consulta, con lo cual se reduce el tiempo de respuesta de las operaciones aplicadas sobre una Base de Datos Distribuida.

1. INTRODUCCION

Este artículo describe la implementación del módulo COMPILADOR DE EXPRESIONES DEL LENGUAJE RELACIONAL para el sistema TESEO [Franky 87]. Los objetivos de este proyecto realizado como tesis de pregrado, son ofrecerle al diseñador de una Base de datos distribuida(BDD) un lenguaje relacional que le permita manipular la base de datos global(BD), y realizar algunas optimizaciones en la generación de planes de consulta a partir de instrucciones del lenguaje relacional.

El lenguaje relacional ofrecido al diseñador facilita el acceso a la BD: permite hacer consultas, modificaciones, etc., que involucren varias relaciones al tiempo y ofrece operadores como join, proyección, unión y producto cartesiano.

El módulo Compilador de TESEO se aplica en el contexto de BDD con las siguientes características:

- Bases de datos que sigan el modelo relacional.

- Definición estática de relaciones y de los esquemas de fragmentación y distribución.

- Fragmentación horizontal simple de las relaciones; sin replicación.

  El diseñador de la aplicación debe manejar las relaciones globales únicamente a traves del lenguaje relacional descrito en este articulo.

  El sistema está orientado hacia la compilación de transacciones especificadas por el diseñador, que pueden ser solicitadas con diversos parámetros por los usuarios finales.

2 - DESCRIPCION GENERAL

Debido a que el módulo **Compilador** de TESEO debe trabajar en forma coordinada con los módulos existentes del sistema TESEO, a continuación se dara una breve explicación de estos últimos. La descripción completa se encuentra en [Franky 87] y posteriormente se presentarán las ampliaciones hechas a los módulos existentes a fin de implementar el módulo **Compilador**.

2.1 MODULOS INICIALES DEL SISTEMA TESEO

La primera versión del sistema TESEO está conformada por dos módulos que deben ejecutarse en dos etapas sucesivas y que son:

A.- MODULO ESPECIFICADOR

El objetivo de este módulo es ofrecerle al diseñador las facilidades necesarias para especificar tanto los esquemas global, de fragmentación y de distribución de la base de datos, como las operaciones y los algoritmos de cada transacción (ver definición de esquemas de BDD en [Ceri 84]). Esto se hace con el fin de generar tablas y archivos que seran consultados tanto por el sistema (para efectuar verificaciones y validaciones), como por la aplicación (en ejecución).

El diseñador utiliza este módulo para:

- Especificar los sitios lógicos sobre los cuales se va a ejecutar la aplicación.
- Definir las relaciones globales de la base de datos.
- Especificar la fragmentación y distribución de relaciones.
- Definir y especificar las transacciones que podrán ser solicitadas por los usuarios finales.

B.- MODULO GENERADOR

En este módulo el sistema toma los archivos definidos por el diseñador en el módulo especificador y, a partir de ellos, va generando el programa que simula los procesos concurrentes que conforman la aplicación distribuida.

En esta fase ocurren una serie de verificaciones con respecto a las transacciones definidas, para detectar errores como:

- Transacciones mal estructuradas (Begin - End).
- Utilización de variables locales y de parámetros que no han sido definidos.

- Utilización de relaciones, llaves o atributos que no han sido definidos o que sus tipos no corresponden.

- Utilización inadecuada de los servicios ofrecidos por el sistema en uno o varios de sus parámetros.

- Existencia de algunos archivos utilizados.

2.2 - AMPLIACIONES AL MODULO ESPECIFICADOR

En el módulo Especificador de la primera versión de TESEO, el diseñador especifica las transacciones en lenguaje Turbo-Pascal utilizando los siguientes **servicios** del sistema, para manejar las relaciones globales:

- **Insertar** (relacion)

  Permite adicionar una tupla a relación. La llave principal e info constituyen los valores de todos los atributos y se separan por '/' así:

  /atr1/atr2/...../atrn/

- **Eliminar** (relacion, llave-ppal)

  Borra la tupla con **llave-ppal** de la relación especificada, es la función inversa de insertar.

- **Modificar** (relacion, llave, atributos que cambian, info)

  Modifica los atributos con la nueva información de la tupla con llave de la relación especificada.

- **Seleccione-Proyeccion**(relacion,formula,proyeccion,archivo_resultado)

  Selecciona de **relación** aquellas tuplas que cumplen la **fórmula** de selección, devolviendo en **archivo_resultado** la proyección de esas tuplas los atributos especificados en **proyección**.

Respecto a la primera versión de TESEO el compilador permite al diseñador utilizar además de los servicios mencionados, un lenguaje relacional inspirado en SQL [Chamberlin 76] con las siguientes instrucciones para manipular la base de datos global.

a.- SELECT atributos INTO tabla
FROM relaciones globales
WHERE fórmula

Significado:
Seleccione las tuplas de las **relaciones globales** que cumplan con la **fórmula**, proyéctelas sobre **atributos** y guardelas en **tabla**.

b.- INSERT INTO relación
FROM tabla

Inserte las tuplas de **tabla** en la **relación**.

c.- UPDATE relación
SET atr1=<exp1>, atr2=<exp2>, ... , atrn=<expn>
WHERE fórmula

Actualice en **relación** las tuplas que cumplan con la **fórmula** de la siguiente manera: en el **atr-i** coloque la **<exp-i>**.

d.- DELETE relación
WHERE fórmula

Borre de **relación** las tuplas cumplan con la **fórmula**.

Para las instrucciones especificadas anteriormente se asume que:

tabla es una relación temporal local.

<EXPI> ::= <CONSTANTE> / <PARAMETRO>

<ATRIBUTOS> ::= <RELACION> <.> <ATRIBUTO> /
<RELACION> <.> <*>

<FORMULA> ::= <ATOMO> / <ATOMO> <Y> <FORMULA>

<ATOMO> ::= <ATRIBUTO> <OPREL> <ATRIBUTO> /
<ATRIBUTO> <OPREL> <CONSTANTE> /
<ATRIBUTO> <OPREL> <PARAMETRO>

<OPREL> ::= [<, >, =, <=, >=, <>]

Solo se acepta 'Y' como operador lógico de la fórmula porque una instrucción que tenga un 'O' en su fórmula de selección, se puede expresar como la unión de 2 tablas resultantes de un SELECT, UPDATE, etc.

En conclusión, la implementación del módulo compilador de TESEO permite que el diseñador programe las transacciones de su aplicación en lenguaje Turbo-Pascal pudiendo manipular las relaciones globales a través del lenguaje relacional presentado arriba.

2.3 - UBICACION DEL MODULO COMPILADOR EN EL SISTEMA TESEO

El siguiente diagrama ilustra la ubicación del módulo **Compilador** en el sistema TESEO.

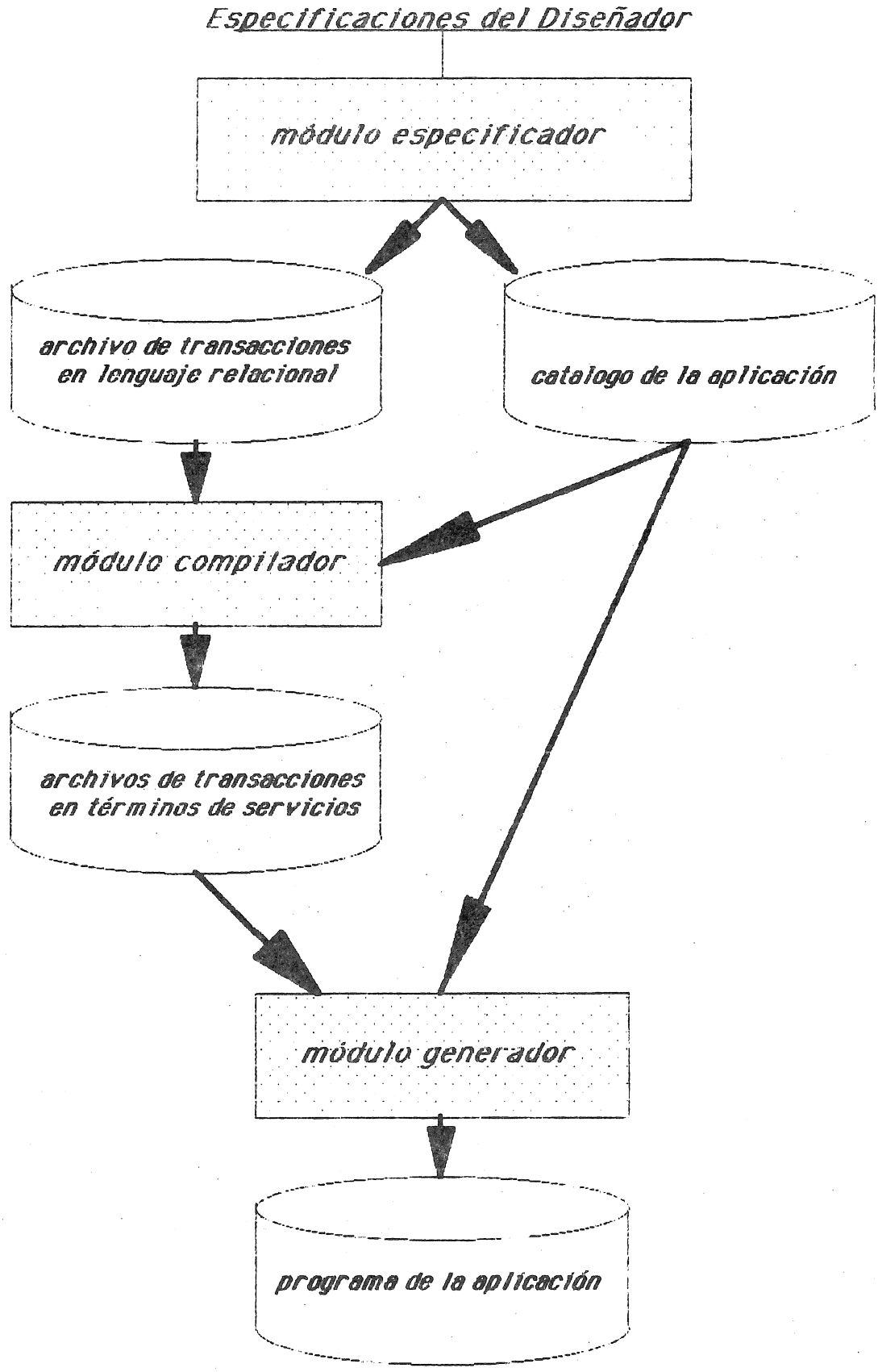


Figura 1: UBICACION DEL MODULO COMPILADOR EN EL PROGRAMA TESEO

Explicación:

Una vez que el diseñador especifica transacciones en lenguaje relacional, el módulo **Compilador** debe transformar el texto de estas transacciones en términos de servicios del sistema y de operaciones entre tablas locales.

El archivo resultante de esta transformación es el que debe tomar como entrada el módulo Generador.

2.4 - AMPLIACIONES AL MODULO GENERADOR

El procesamiento del lenguaje relacional de las transacciones, genera invocaciones a los servicios existentes (Insertar, Eliminar, Modificar, Seleccion-Proyeccion) y a nuevas operaciones locales que van a trabajar sobre tablas locales, resultado de servicios. Por lo tanto, es necesario adicionar al módulo Generador el código de las nuevas operaciones locales, que son las siguientes:

a.- UNION (TABLA1,TABLA2,TAB-RESP)

su función es unir las dos primeras tablas y dejar el resultado en la tabla dada como tercer parametro (TAB-RESP).

b.- JOIN (TABLA1,TABLA2,FORMULA,PROYECCION,TABLA3)

realiza el join de TABLA1 con TABLA2, selecciona aquellas tuplas que cumplan con la FORMULA, las proyecta sobre los atributos de PROYECCION, y las devuelve en TABLA3.

c.- SEL-PROY (TABLA1,FORMULA,PROYECCION,TABLA2)

toma las tuplas de TABLA1 que cumplan con la FORMULA, las proyecta sobre los atributos de PROYECCION y las devuelve en TABLA2.

3. DISEÑO DEL MODULO COMPILADOR

El mayor trabajo del proyecto se concentra en la nueva etapa de compilación, la cual debe solucionar el problema de transformar las clausulas SELECT, INSERT, DELETE, UPDATE (presentadas en la sección 2), a instrucciones en términos de los servicios del sistema: SELECCION-PROYECCION, INSERTAR, ELIMINAR, MODIFICAR y de las operaciones locales UNION, JOIN, SEL-PROY.

El módulo Compilador examina el archivo que contiene el texto de las transacciones y produce un archivo de salida con el texto de las transacciones ya transformado. Es este archivo de salida del Compilador el que debe tomar como entrada el módulo Generador.

3.1 - COMPILACION DEL SELECT

La compilación del SELECT implica una serie de pasos que serán explicados a continuación. En este tratamiento se aplicó una combinación de las transformaciones y optimizaciones expuestas en [Ceri caps 5 y 6, 84]

a. - PRIMERA INTERPRETACION GENERAL DE UN SELECT

La instrucción de consulta

```
SELECT a1,a2,.....,aN INTO tabla
FROM   R1,R2,.....,Rn
WHERE  atomo1 Y atomo Y ... Y atomoN
```

donde <a1> ::= <relacion> <.> <atributo>

<atomoI> ::= <atributo><operador relacional><atributo> /
<atributo><operador relacional><constante> /
<atributo><operador relacional><parametro>.

se interpreta inicialmente como:

π a1,..,aN (σ atomo1 Y..Y atomoN (R1 X R2 X ..... X Rn))

donde los atributos respuesta del SELECT (a1,..,aN) son los atributos de la proyección externa; luego sigue una selección que se aplica sobre los átomos del WHERE (atomo1 Y....Y atomoN), y por último se tienen joins entre las relaciones especificadas en el FROM (R1,R2,...,Rn).

El árbol de consulta global es el resultado de ésta primera interpretación y tiene la siguiente forma:

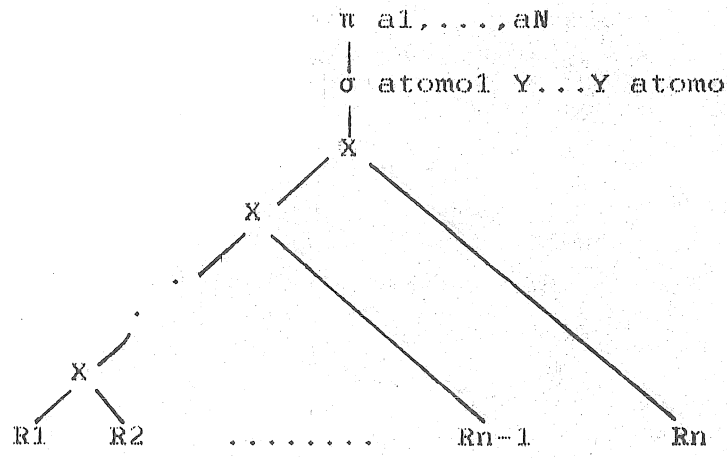


FIGURA 2: FORMA GENERAL DEL ARBOL INICIAL DE CONSULTA

B. - **BUEN ORDEN PARA HACER LOS JOINS**

Con el fin de evitar que los joins se conviertan en producto cartesiano hay que escoger un orden adecuado para hacerlos.

El metodo escogido para hacerlo es el siguiente:

- Se escoge como primera relación aquella que tenga en la fórmula de selección algún atributo relacionado con una constante. Esto se hace porque permite añadir en el árbol de consulta una selección directa sobre la relación, con lo cual se reduce el número de tuplas participantes en el join.

  En caso de que ninguna relación cumpla con esta condición, se escoge cualquiera como primera.

- Como segunda relación se escoge cualquiera que en la fórmula de selección tenga algún atributo relacionado con otro de la relación escogida como primera.

- Las demas relaciones se van escogiendo con el mismo criterio: que tengan en la fórmula algún atributo que las relacione con otro atributo de cualquiera de las relaciones anteriores.

Si quedan relaciones que no tengan nexo alguno con las anteriores, éstas se colocan al final de la secuencia e implicarán inevitablemente productos cartesianos.

Para ilustrar este método se da el siguiente ejemplo:

Se tienen las siguientes relaciones

```
EST     (CARNET,NOMBRE,PROM,....)
VIENDO  (CARNET-E,COD-C,........)
CURSO   (COD,CRED,..............)
```

y se desea hacer la siguiente consulta

```
SELECT est.carnet, est.nombre
FROM   est,viendo,curso
WHERE  est.carnet=viendo.carnet-e  Y viendo.cod-c=curso.cod
         Y curso.cred > 3
```

Siguiendo los pasos especificados anteriormente, se escogería como primera relación CURSO por existir en la fórmula de selección (parte WHERE) el átomo curso.cred>3; luego se escogería VIENDO porque es la única relacionada con CURSO por medio del átomo viendo.cod-c=curso.cod, y por último se escoge EST por estar relacionada con VIENDO por el átomo est.carnet=viendo.carnet-e.

C. - GENERACION DEL ARBOL FINAL

Basándose en el árbol de consulta global y luego de haber escogido el orden de los joins, se procede a la generación del árbol de consulta final.

El árbol que se va a generar equivale a realizar una serie de transformaciones sobre el árbol inicial, a fin de incluir algunas de las simplificaciones y optimizaciones presentadas en [Ceri caps 5 y 6, 84].

El árbol final debe presentar las siguientes características, en relación al árbol inicial:

- debe tener las relaciones (nodos hojas) en un buen orden de acuerdo a los criterios ya expuestos.

- debe generar el máximo número de operadores de proyección y selección a fin de reducir el tamaño de los operandos antes de las operaciones binarias de join.

- debe bajar lo máximo posible esos operadores reductores (π y σ) a fin de aplicarlos lo antes posible.

- debe colocar los átomos paramétricos en el nivel mas superior a fin de evaluar las variables de la consulta una sola vez, en compilación.

Para ilustrar la forma final del árbol para el caso de 3 relaciones usemos el siguiente ejemplo:

```
SELECT  est.carnet, est.nombre
FROM    est, viendo, curso
WHERE   est.carnet=viendo.carnet-e Y viendo.cod_c=curso.cod
        Y curso.cred > 3  Y est.prom > xxx
```

El arbol final correspondiente es:

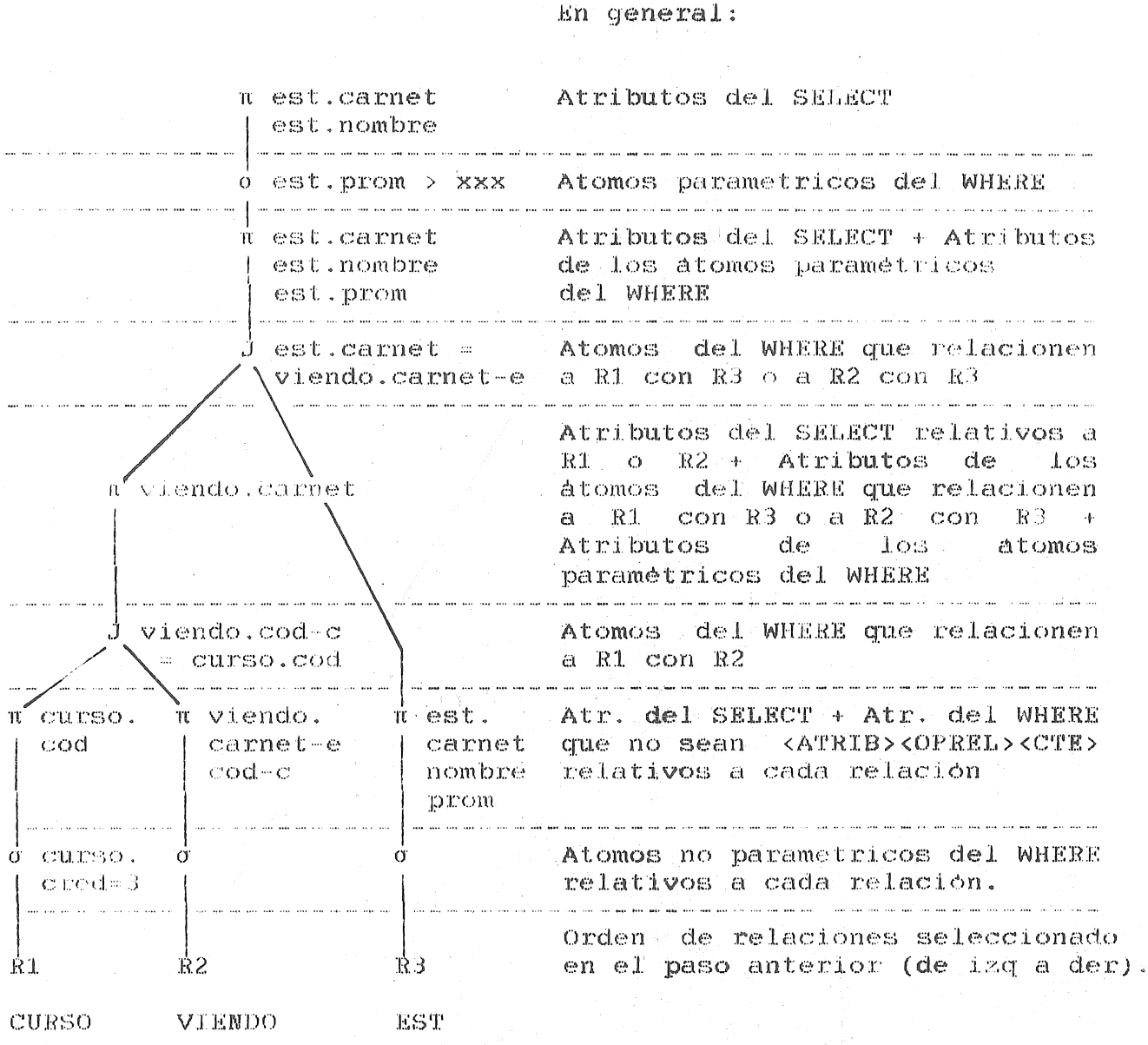


FIGURA 3: EJEMPLO DE ARBOL FINAL DE CONSULTA

Al lado derecho de la FIGURA 3 están especificados los atributos que debe tener cada una de las proyecciones, selecciones y joins en referencia a la forma general del SELECT.

Generalizando a n relaciones, el árbol final es el siguiente (Figura 4):

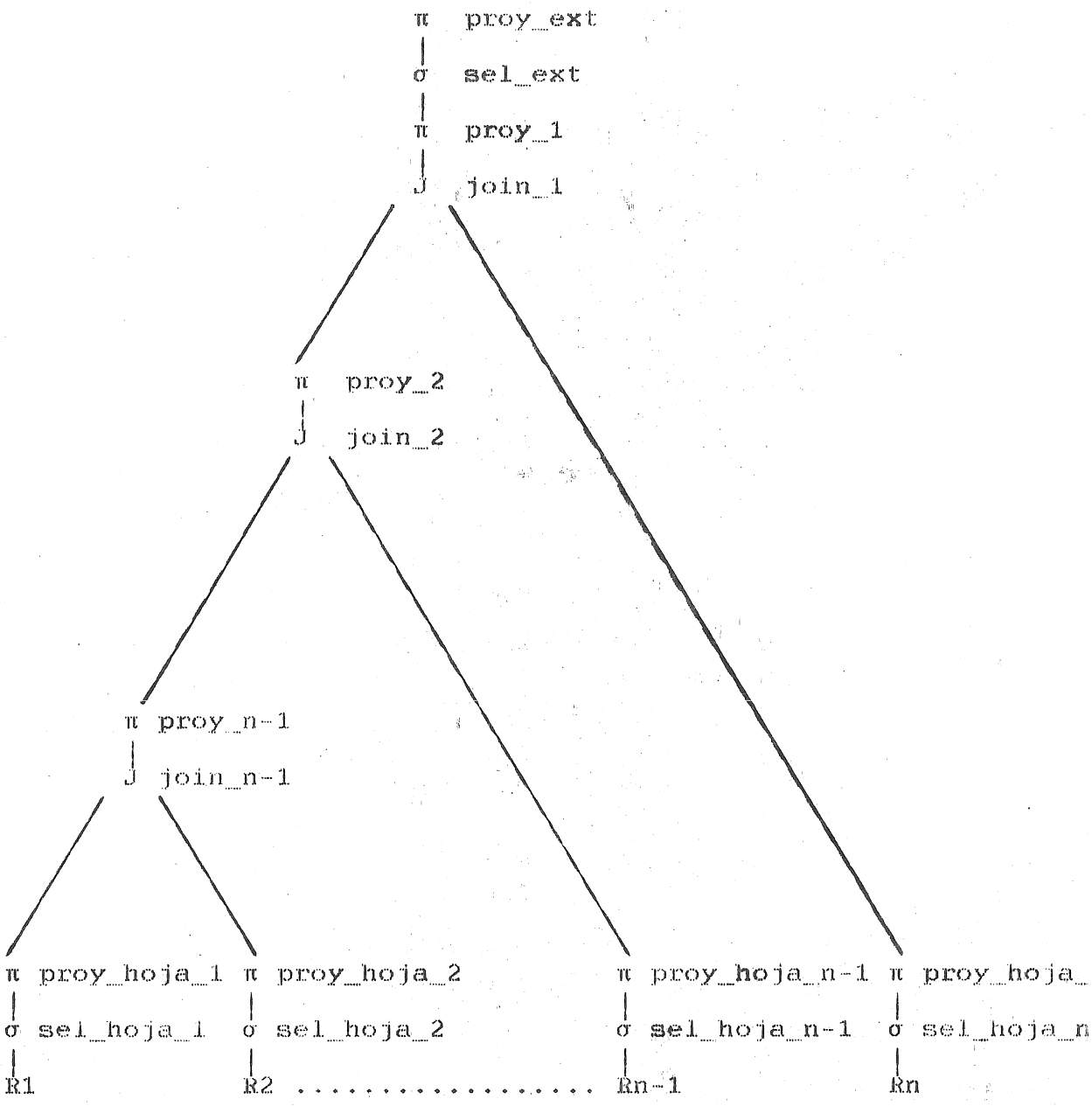


FIGURA 4: FORMA GENERAL DEL ARBOL FINAL DE CONSULTA

La explicación de los atributos correspondientes a cada nodo y la forma como se va generando el árbol de consulta, se explica en el siguiente macroalgoritmo correspondiente al programa que genera el árbol final directamente a partir del árbol inicial.

Sea n el número de relaciones que intervienen en la consulta.

```
MACROALGORITMO ( N )

  SI  hay átomos paramétricos en el where ENTONCES
  |
  |  PROY_EXT   <---   Atributos del select
  |  SEL_EXT    <---   Atomos paramétricos del where
  |
  FIN_SI
  K  <---  N
  PARA  I:=  1, N-1  HAGA
  |
  |  PROY_I     <---   Atributos del select + Atributos de los átomos
  |                    paramétricos del where relativos a las primeras
  |                    K relaciones + Atributos de los átomos
  |                    del where que sean <ATRIB><OPREL><ATRIB> relati-
  |                    vos a las primeras K relaciones, que
  |                    relacionen a cualquiera de las primeras K
  |                    relaciones con cualquiera de las restantes.
  |
  |  JOIN_I     <---   Atomos no paramétricos del where que relacionen
  |                    a REL[K] con cualquiera de las anteriores.
  |
  |  PROY_HOJA_K  <---  Atributos del select + Atributos del where
  |                     que no sean <ATRIB><OPREL><CTE> relativos a
  |                     REL[K].
  |
  |  SEL_HOJA_K   <---  Atomos del where que sean <ATRIB><OPREL><CTE
  |                     relativos a REL[K].
  |
  |  K  <---  K - 1
  FIN_PARA
  PROY_HOJA_K  <---  Atributos del select + Atributos del where
                     que no sean <ATRIB><OPREL><CTE> relativos a
                     REL[K].

  SEL_HOJA_K   <---  Atomos del where que sean <ATRIB><OPREL><CTE>
                     relativos a REL[K].

FIN_MACROALGORITMO
```

El funcionamiento del macroalgoritmo es el siguiente:

Primero se deducen los nodos superiores del árbol PROY_EXT y SEL_EXT (en caso que existan) y luego se deducen los demás nodos en ciclos repetitivos así:

Habiendo deducido los nodos PROY_I, JOIN_I, se deducen los nodos derechos PROY_HOJA_K, SEL_HOJA_K; luego se decrementa K para que en el siguiente ciclo se tomen los nodos hojas correspondientes a la rama derecha de PROY_I, JOIN_I, y por último se incrementa I para desplazarse hacia los nodos izquierdos PROY_I, JOIN_I que serán deducidos en el siguiente ciclo.

d. - TRANSFORMACION DEL ARBOL DE CONSULTA DEL SELECT A PEDIDOS DE SERVICIOS Y A OPERACIONES SOBRE TABLAS LOCALES

Una vez que se obtiene el árbol final de consulta, el compilador debe traducirlo a instrucciones en términos de invocaciones de servicios, de manipulación de relaciones y de operaciones sobre tablas locales. Para ello, el Compilador recorre el árbol en postorden para que cada vez que se vaya a analizar una operación ya estén evaluados sus operandos, y a medida que se vaya avanzando se vayan haciendo las traducciones correspondientes, dependiendo del nodo a evaluar.

Hay 3 clases de nodos: el nodo raiz, los nodos intermedios y los nodos hojas y la traducción para cada uno de ellos es la siguiente:

- Para el nodo raiz (proyección - selección superior del árbol) se debe generar la operación local

  SELECCION-PROYECCION (TABLA, FORMULA, PROYECCION, TABLA-RTA)

  donde TABLA debe contener las tuplas resultantes de todas las operaciones inferiores del árbol y TABLA-RTA va a contener el resultado total de la consulta.

- Para cada uno de los nodos intermedios (proyección - join intermedias) se debe generar la operación local

  JOIN (TABLA1,TABLA2,FORMULA,PROYECCION,TABLA3)

  donde TABLA1 debe contener las tuplas resultantes del subárbol inferior izquierdo del join, y TABLA2 las del subárbol inferior derecho.

- Para cada uno de los nodos hojas (proyección - selección del nivel más bajo del árbol, las que se aplican sobre las relaciones) se genera un llamado al servicio

  SELECCION-PROYECCION (RELACION,FORMULA,PROYECCION,
  TABLA_RESULTADO)

3.2 - COMPILACION DE INSTRUCCIONES DIFERENTES AL SELECT

La compilación de esas instrucciones se realiza así:

a. - INSERT

Forma General: INSERT INTO relación
FROM tabla

El INSERT permite insertar a **relación** varias tuplas que se encuentran en **tabla**.

La tabla puede ser el resultado de otra consulta, pero debe tener como campos todos los atributos de **relación**, en el mismo orden en que fueron especificados al definir la relación.

Al compilar el INSERT se genera un ciclo repetitivo que llama al servicio

INSERTAR (RELACION,LLAVE INFO)

por cada registro de la tabla.

Para poder hacer ésto, por cada INSERT existente en la transaccion, se genera un registro **ZREGIn** de tipo **ZREGINSn** donde n >= 1 es un contador que se incrementa con cada INSERT que aparezca en el texto de la transacción, el cual va a contener todos los atributos de la relación involucrada, y se le asocia a la tabla un descriptor **ZARCHIn** que será de tipo **FILE OF ZREGINSn**.

Con estas estructuras de datos el ciclo repetitivo generado para un INSERT es:

```
REPITA
 Lea un registro de ZARCHIn;
 Coloque a INFO la información correspondiente;
 Coloque a LLAVE la información correspondiente;
 INSERTAR (RELACION,LLAVE,INFO);
HASTA QUE ( fin(ZARCHIn) = verdadero )
```

b. - **DELETE**

Forma General: DELETE Relación
WHERE fórmula

La compilacióon del DELETE implica en primer lugar, la generación de un SELECT de la siguiente forma:

```
SELECT  llaves  INTO  ZZi                    i >= 1
FROM    Relación
WHERE   Fórmula
```

donde: ZZi es un nombre que se le asigna a la tabla resultado. y <llaves> incluye todos los atributos que sean llave en relación.

Este SELECT se genera internamente y se interpreta como si fuera una instrucción más de la transacción; por lo tanto en el archivo de salida se verá como un llamado al servicio

SELECCION-PROYECCION (RELACION,FORMULA,PROYECCION,TABLA_RESULTADO)

La razón por la cual se hace el SELECT, es poder tener en TABLA_RESULTADO las llaves de todas las tuplas que deben ser eliminadas, y así poder llamar dentro de un ciclo repetitivo al servicio

ELIMINAR (RELACION,LLAVE)

por cada tupla (registro) leída en el archivo ZZi (TABLA).

Para poder leer del archivo, se genera un registro ZREGDn de tipo ZREGDELn donde n >= 1, el cual tiene como campos todos los atributos que sean llave en la relación implicada, y se le asocia a la tabla un descriptor ZARCHDn de tipo **FILE OF ZREGDELn**.

Con estas estructuras de datos para un DELETE se genera lo siguiente:

```
SELECCION-PROYECCION(RELACION,FORMULA,PROYECCION,TABLA);
ZARCHDn <--- TABLA
REPITA
 Lea un registro de ZARCHDn;
 Coloque a LLAVE la información correspondiente;
 ELIMINAR(RELACION,LLAVE);
HASTA QUE ( fin(ZARCHDn) = verdadero )
```

c. - UPDATE

Forma General:

```
UPDATE   Relación
SET      atr1=exp1.......,atrn=expn
WHERE    Fórmula
```

Al igual que en el DELETE, al compilar el UPDATE se genera inicialmente un SELECT para poder tener en una tabla las tuplas que deben ser modificadas.

El SELECT sería de la forma:

```
SELECT llaves INTO ZZi  i >= 1
       FROM   Relación
       WHERE  Fórmula
```

donde: ZZi es el nombre que se le asigna a la tabla resultado, y <llaves> incluye todos los atributos que sean llave en relación.

El SELECT se genera y se interpreta internamente tal como si fuera una instrucción más de la transacción y en el archivo de salida del módulo **Compilador** se encuentra como un llamado al servicio

SELECCION-PROYECCION (RELACION,FORMULA,PROYECCION,TABLA)

Luego de haber sido generado el SELECCIONE, se leen las tuplas (registros) del archivo ZZi y por cada una de ellas, se hace un llamado al servicio

MODIFICAR (RELACION,LLAVE,INFO)

Para poder leer del archivo, se genera un registro **ZREDUn** de tipo **ZREGUPDn** donde n >= 1, el cual contiene como campos todos los atributos que sean llave en la relación implicada, y se le asocia a la tabla un descriptor **ZARCHUn** de tipo **FILE OF ZREGUPDn.**

Con estas estructuras de datos para un UPDATE se genera lo siguiente:

```
SELECCION-PROYECCION (RELACION,FORMULA,PROYECCION,TABLA);
ZARCHUn <--- TABLA_RESULTADO
REPITA
 Lea un registro de ZARCHUn;
 Coloque a INFO la información correspondiente;
 Coloque a LLAVE la información correspondiente;
 MODIFIQUE(RELACION,LLAVE,INFO);
HASTA QUE ( fin(ZARCHUn) = verdadero )
```

4. CONCLUSIONES

La realización del módulo COMPILADOR del sistema TESEO, permite ofrecer al diseñador de aplicaciones distribuidas un lenguaje relacional con total transparencia a la fragmentación y distribución de la Base de Datos Distribuida, y que aprovecha todas las facilidades ofrecidas por TESEO (facilidades expuestas en [Franky 87]).

La implementación de este módulo se encuentra actualmente a nivel de prototipo (en microcomputador), pero se espera una implementación más ambiciosa en la siguiente fase del sistema TESEO que será realizada sobre una red local Ethernet de equipos Vax.

BIBLIOGRAFIA

[Franky 87]: Maria Consuelo Franky,
"TESEO: Ambiente de programación de Sistemas Transaccionales Distribuidos",
XIII Conferencia Latinoamericana de Informática (CLEI),
Bogotá- Colombia, Nov. 1987

[Ceri 84]: S.Ceri, G. Pelagatti,
"Distributed Databases: principles and systems",
McGraw-Hill, 1984

[Alvarado y Muñoz 86]: Armando Alvarado, Adriana Muñoz,
"Sistema Generalizado de Transacciones Distribuidas sobre Bases de Datos Homogéneas", tesis de pregrado en Ingenieria de Sistemas ISC-86-I-9,
Universidad de los Andes, Sept. 1986

[Chamberlin 76]: D.D. Chamberlin et al.,
"SEQUEL 2: A unified aproach to Data Definition, Manipulation and Control", IBM Journal of Research and Development,
Vol. 20 No. 6, Nov. 1976

Determinación y Restauración Automática de los Estados Intermedios de una aplicación

Marco Mercinelli

Laboratorio de Electrónica Digital
Universidad Católica Nuestra Señora de la Asunción
Asunción - Paraguay

Los sistemas operativos modernos permiten gestionar aplicaciones muy complejas. Si es necesario una aplicación puede ser desarrollada para un conjunto de varios procesos (programas en ejecución) que se comunican entre ellos y que pueden ser ejecutados en distintas computadoras de una red. En este ambiente aparece importante la disponibilidad de un mecanismo para "congelar" y despues "restaurar" el estado de ejecución de un proceso y sus relaciones con el ambiente. En efecto un mecanismo de este tipo puede ser usado para la realización de muchas funcionalidades de alto nivel como mecanismos para aumentar la fiabilidad de un sistema, para la "migración" de procesos, para el *debugging* de aplicaciones distribuidas y para nuevos algoritmos de *backtracking*.

Este artículo describe los problemas técnicos y prácticos enfrentados en el desarrollo de un mecanismo de este tipo para un sistema operativo standard largamente utilizado, como es el UNIX™†. Este mecanismo, llamado de *snapshot* y *restart*, permite a los usuarios, bajo algunas hipótesis, de "congelar" y escribir en archivos el estado de un proceso en ejecución y de retomar la ejecución de aquel proceso, desde el punto del "congelamiento", en cualquier momento posterior, también despues de una parada de la computadora.

† UNIX es un nombre registrado por los Laboratorios Bell

1. Introducción

El principal objetivo de este proyecto era el de explorar la factibilidad de un mecanismo automático para extraer informaciones sobre el estado y el ambiente de ejecución de un proceso, en una forma bastante completa para poder restaurar el proceso en el mismo estado en un ambiente y/o en un tiempo distinto.

Un mecanismo de este tipo, fiable y eficiente, puede ser usado como "bloque de construcción" para facilitar y acelerar el desarrollo de nuevas aplicaciones y funcionalidades. Algunos ejemplos de los posibles campos de aplicación son los siguientes:

a - Es posible aumentar la fiabilidad de un sistema, guardando periodicamente en memoria estable el estado de ejecución de eventuales procesos "delicados". En caso de necesidad será posible restaurar la ejecución de los procesos a partir del último estado guardado.

b - Los sistemas operativos más avanzados afrontan a menudo el problema de la subdivisión y del balance de la carga de trabajo entre computadoras conectadas en red. Una de las técnicas propuestas es la utilización de mecanismos para hacer "migrar" los procesos de una computadora a otra de la red que tenga menor carga de trabajo. Estos mecanismos se pueden realizar "congelando" un proceso, transfiriendo las informaciones "congeladas", con todos los datos necesarios, a otra computadora de la red menos cargada y restaurando entonces su ejecución en la otra computadora.

c - Algunos algoritmos preveen la posibilidad del *backtracking*, es decir un programa puede retornar a un estado de ejecución alcanzado anteriormente, modificar el valor de algunas variables y retomar la ejecución, pero siguiendo un camino diferente. La disponibilidad de un mecanismo automático para reconstruir un estado precedente de la aplicación puede ayudar en la implementación de estos algoritmos, sobre todo cuando es muy complejo volver a un estado precedente del algoritmo.

d - Algunas técnicas nuevas para el *debugging*, particularmente en ambientes distribuidos, están basadas en la posibilidad de extraer y examinar el estado de los diferentes procesos envueltos en una aplicación sin interferir con su ejecución.

Se optó por desarrollar la investigación sobre UNIX™ porque es un sistema operativo bien conocido, cuyas fuentes están disponibles y son modificables y que se está afirmando como standard sobre todo para las *workstations* y los mini computadores. Además UNIX™ tiene muchas caracteristicas similares con otros sistemas operativos de la misma generación permitiendo la generalización de los resultados de la investigación.

Hemos llamado en general *snapshot* al conjunto de informaciones que es necesario guardar para definir de modo unívoco la ejecución de un proceso. El *snapshot* es una copia en disco del ambiente de ejecución del proceso y comprende una imágen de la memoria, el valor de los registros, el estado de los archivos en uso, el *directory* de trabajo corriente y otras informaciones sobre las relacciones con el sistema y con otros processos. La creación de un *snapshot* no debe modificar en modo alguno el comportamiento del proceso, pudiendo él continuar su ejecución como si nada hubiese pasado. Las informaciones guardadas deben ser suficientes para restaurar la ejecución del proceso, de manera que pueda continuar en manera totalmente transparente desde el punto en que fué efectuado el *snapshot*. En la fase de *restart* todos los recursos utilizados por el proceso tienen que estar disponibles de nuevo y sus estados deben ser adecuadamente restaurados.

El trabajo de investigación mostró que, por algunas razones que serán ilustradas seguidamente, un mecanismo de *snapshot* y *restart* aplicable en forma totalmente general y automática no es realizable en un sistema operativo como el UNIX™. Es necesario entonces definir unas hipótesis para caracterizar el tipo de los procesos previstos y, en algunos casos, direccionar el mecanismo de *restart* según informaciones adjuntas por el usuario. Además surgen múltiples dificultades cuando se quiere tratar grupos de procesos cooperantes, sobre todo a causa de los problemas asociados con la definición del estado de una aplicación

distribuida.

A pesar de estas dificultades se realizó un mecanismo experimental que puede "congelar" y "restaurar", en forma completa y transparente, el estado de cualquier proceso que no tenga comunicaciones abiertas con otros procesos (por lo menos en el momento del *snapshot*), no tenga relacciones con el ambiente codificadas en forma implicita en el algoritmo ejecutado y no tenga dependencias del tiempo de ejecución. Este mecanismo es suficientemente general para tratar ya muchas aplicaciones interesantes.

La extensión del sistema UNIX™ para realizar el mecanismo de *snapshot* y *restart* fué desarrollado sobre una *workstation* SUN 2/120 en el ambito de una colaboración con el centro de investigación italiano CSELT y el Departamento de *Computer Science* de la Universidad de California en Berkeley.

2. Determinabilidad del estado de ejecución de una aplicación

El conjunto de informaciones necesario para definir en forma completa un proceso en ejecución está constituida por su código, que normalmente no es modificable y está definido por el contenido del archivo ejecutable, por el contenido de la parte de datos y del *stack* que, en cambio, son modificadas durante la ejecución, por el contenido y estado de los archivos usados, por el estado de los recursos del sistema correlacionadas en alguna forma con la ejecución del proceso, por las informaciones relativas a las relaciones del proceso con otros procesos residentes en la misma computadora o además de ellos ejecutados en otras computadoras conectadas en red, por el estado y la configuración actual de la computadora y eventualmente de toda la red donde se inserta.

Las informaciones necesarias para componer un *snapshot* "completo" pueden, entonces, ser muchas y no facilmente encontrables. Hay también relaciones con otros procesos y con el ambiente de ejecución que en general no pueden ser deducidas en forma autómatica de los datos del sistema porque están implicadas y codificadas en el particular algoritmo ejecutado por el proceso. Como ejemplos las relaciones entre el interprete de comandos de UNIX™ (*shell*) y los procesos que genera son codificadas en el programa mismo; otros programas, como los dos instrumentos de comunicación de UNIX™, *uucp* y *tip*, sincronizan sus accesos a las lineas de comunicación creando unos *lock files*, cuya existencia es obviamente parte del ambiente de ejecución de los programas, aunque la relación con ellos no sea ni explícita ni deducible autómaticamente con un mecanismo general.

Por consiguiente parece no posible, en forma automática, realizar un mecanismo totalmente general de *snapshot* y *restart* en grado de gestionar todas las posibles aplicaciones que los procesos pueden ejecutar en un sistema complejo como el UNIX™. Entonces es necesario hacer algunas hipótesis para caracterizar la clase de procesos que deben ser considerados, de manera a poder definir exactamente y limitar las informaciones necesarias para formar un *snapshot* consistente de estos procesos.

3. Hipótesis para el mecanismo de *snapshot* y *restart*

Las hipótesis de trabajo utilizadas para realizar un primer mecanismo autómatico de *snapshot* y *restart* son las siguientes:

1 - el ambiente de ejecución y las relaciones de un proceso debe ser determinable totalmente, con un método de tipo general, a partir del contenido de las estructuras de datos del sistema operativo;

2 - el algoritmo ejecutado por el proceso no debe ser dependiente del tiempo, es decir puede ser interrumpido por un período de tiempo de cualquier longitud sin provocar problemas de temporización.

Hemos entonces clasificado las aplicaciones que satisfacen a las hipótesis precedentes en cuatro "clases". Para cada una de estas clases las informaciones necesarias para formar un *snapshot* de los procesos envueltos en la aplicación son limitadas y bien determinadas. Cada clase fué definida como un subconjunto de la precedente de manera que el mecanismo de *snapshot* y *restart* para una clase pudiese ser realizado como una extensión del mecanismo para la precedente:

A) aplicaciones ejecutadas por un solo proceso que no utiliza los mecanismos de comunicación entre procesos, y no utiliza archivos o particulares dispositivos de E/S sino solo los canales standard de input, output y error de UNIX™ conectados a la terminal del usuario;

B) aplicaciones ejecutadas por un solo proceso que utiliza archivos y/o dispositivos de E/S y puede tener necesidad de setear los dispositivos de E/S en una forma particular (por ejemplo la supresión del eco de caracteres en una línea de transmisión);

C) aplicaciones ejecutadas por una o más jerarquías de procesos (un proceso "padre" y sus "descendientes") cada uno del tipo descrito en la clase B, que residen en la misma computadora y se comunican entre ellos, pero usando solo los mecanismos standard de comunicación entre procesos previstos por sistema operativo (IPC);

D) aplicaciones ejecutadas por muchas jerarquías de procesos, del tipo de las descritas en la clase C, que pueden ser ejecutadas en paralelo con varias computadoras de un sistema distribuido o varios procesadores de un sistema con multiprocesador, y que se pueden comunicar a través de los mecanismos standard de comunicación entre procesos (IPC) o por medio de memoria compartida.

Las aplicaciones de clase A no tienen interés desde el punto de vista práctico pero representan un punto de partida conveniente para el desarrollo del mecanismo de *snapshot* y *restart*.

El mecanismo implementado actualmente gestiona las aplicaciones de la clase B y ya puede tratar muchos casos de interés práctico. En el resto del artículo se describe la implementación de este mecanismo.

Las aplicaciones de clase C necesitan de un mecanismo más complejo en cuanto requieren guardar y restaurar informaciones que se refieren a las relaciones entre los procesos. Además es necesario poner particular atención en evitar que alguno de los procesos continúe su ejecución antes que el *snapshot* completo haya sido escrito en memoria estable, pudiendo esto hacer inconsistentes partes de las informaciones guardadas.

Las aplicaciones de clase D son las más difíciles de gestionar principalmente por dos razones: es muy complicado reconstruir el estado de los protocolos de comunicación, y además no es posible aplicar el mecanismo de *snapshot* "exactamente" al mismo tiempo a los procesos en ejecución en máquinas diferentes, a causa de la casualidad de los retardos de transmisión a través de la red. Entonces, según como sea efectuado el *snapshot*, puede no ser posible guardar un estado global consistente de los procesos envueltos en la aplicación. Una metáfora creada por los investigadores Chandy y Lamport ilustra claramente este problema:

> *El algoritmo para guardar el estado de los procesos en un ambiente distribuido puede ser comparado con un grupo de fotógrafos que observan una escena muy grande y dinámica como un cielo cubierto por una bandada de pájaros migradores. La escena es tan grande que no puede ser completamente captada por una sola fotografía. Los fotógrafos deberán tomar muchas fotografías (snapshots) al mismo tiempo y luego juntarlas para poder tener una imagen de toda la escena. Con todo eso, las fotografías no pueden ser captadas "exactamente" en el mismo instante, a causa de la dificultad de sincronizar la acción de los fotógrafos. Además los fotógrafos no pueden "perturbar" el proceso que están fotografiando, por ejemplo no pueden hacer de manera que todos los pájaros permanezcan inmóviles en el cielo mientras son captadas las fotografías. Sin embargo si se quisiera que la imagen final fuese "significativa", es decir describa el fenómeno en modo completo y adecuado. El problema es definir que quiere decir "significativo" en un cierto contexto y entonces determinar como las fotografías deben ser realizadas.*

Chandy y Lamport describieron un algoritmo que permite obtener un estado global consistente. Además Dave Presotto, investigador de los Laboratorios Bell identifica los vinculos, que respetandolos, permiten verificar la consistencia de un estado global. Estos y otros trabajos de investigación razonablemente hacen suponer poder realizar en el futuro un mecanismo suficientemente general también para las aplicaciones de clase D.

4. La gestión de los procesos en UNIX™

El usuario de un sistema UNIX™ considera los procesos como entidades "activas" que ejecutan cada uno una secuencia propia de instrucciones. Cada proceso lleva adelante la tarea asignádale en forma concurrente con todos los otros procesos activos del sistema. A cada proceso está asociado un número, llamado identificador de proceso (*pid*), que es único entre todos los identificadores de los procesos presentes en el sistema en un cierto instante.

En cambio, desde el punto de vista del sistema operativo un proceso es una entidad "inactiva", formada por varias áreas de memoria y estructuras de datos (que forman la "imagen" del proceso para el sistema operativo) que son manipuladas por las rutinas de su *kernel* y ejecutadas por el procesador. El sistema operativo controla al procesador, haciendolo pasar muy frecuentemente por la ejecución de una "imagen" a otra, de manera a dar a cada proceso la posibilidad de llevar adelante la tarea asignádale.

La "imagen" de un proceso está compuesta por un conjunto de informaciones que comprenden las estructuras de datos del sistema operativo que se refieren directamente al proceso, más una parte llamada "parte de usuario" formada por código, datos, *stack* y contenido de los registros del programa en ejecución. El sistema operativo utiliza unas cuantas estructuras de datos para identificar un proceso y los recursos que utiliza. Algunas de estas estructuras contienen datos globales, que deben ser siempre accesibles por el sistema aún cuando el proceso está temporalmente suspendido (por ejemplo tabla de procesos activos, tabla de segmentos de códigos compartidos, etc.). Cada proceso tiene además una estructura de datos asociada llamada impropiamente *User Area*, que contiene los datos y las referencias necesarias cuando el proceso está ejecutando su código (por ejemplo informaciones sobre los archivos abiertos, acciones asociadas a las señales que puede recibir, definición de *timers*, etc.).

El sistema operativo asocia un conjunto de estados a las diferentes situaciones en que un proceso puede encontrarse en el curso de su existencia. La ejecución de las rutinas del *kernel* del sistema puede causar transiciones de un estado a otro. En el curso de este articulo haremos referencia solo a tres estados principales: *running*, *ready* y *sleeping*. El proceso *running* (llamado también proceso corriente) es el único proceso ejecutado por el procesador en un cierto instante; los procesos *ready* son los que esperan en una cola para poder ser ejecutados por el procesador; los procesos *sleeping* son los que están suspendidos en espera de que ocurra algún "evento" significativo para la continuación de su ejecución (por ejemplo la terminación de una operación de E/S).

El *kernel* utiliza una politica de *scheduling* de la ejecución de los procesos basada en una "prioridad" asociada a cada proceso. La prioridad es modificada dinámicamente de manera a asegurar que todos los procesos *ready* puedan obtener atención del procesador dentro de una cantidad de tiempo finita. La politica de variación de las prioridades privilegia los procesos que tienen frecuentes periodos de inactividad (por ejemplo los programas interactivos). El mecanismo de *scheduling* de UNIX™ fué realizado de manera que una substitución del proceso corriente no pueda ocurrir dentro del *kernel* en el curso de la ejecución de una llamada al sistema operativo. Un proceso puede ser forzado a ceder la utilización del procesador solo cuando está por ejecutar el código usuario y si hay en cola otro proceso *ready* que tiene una prioridad más alta que la suya. Un proceso que está ejecutando una rutina del *kernel* del sistema operativo puede dejar espontaneamente la utilización del procesador, pasando al estado *sleeping*, si ha alcanzado un punto más allá del cual su ejecución no puede continuar inmediatamente. Los mecanismos de substitución del proceso en ejecución adoptados por UNIX™ simplifican notablemente los problemas de sincronización dentro de su *kernel*.

El procesador, cuando ejecuta un código por cuenta del proceso corriente, utiliza diferentes "privilegios" y métodos de acceso a la memoria según que se ejecute el código de un programa usuario (modo usuario) o el código de alguna rutina del sistema operativo (modo sistema). El "espacio de direcciones" al cual un proceso puede acceder cuando está en modo usuario es totalmente disjunto del accesible en modo sistema. En el modo usuario un proceso puede acceder solo a las partes usuario de su imagen, es decir a su código, datos y *stack*, mientras que en el modo sistema puede acceder directamente solo al código de las rutinas del sistema operativo y a sus estructuras de datos. Sin embargo, en el modo sistema, un proceso puede también transferir datos en forma indirecta de y hacia el espacio de direcciones del usuario, utilizando procedimientos del *kernel* del sistema operativo a muy bajo nivel.

El *stack* utilizado en el modo sistema es diferente del *stack* usuario y es accesible en una misma dirección por todos los procesos, entonces cada proceso, cuando está suspendido, debe mantener una copia de su *stack kernel*, para utilizar durante su ejecución. Una copia de los registros del procesador utilizados en el modo usuario es guardado en el *stack kernel* cada vez que un proceso comienza a ejecutarse en modo sistema y es restaurada cuando el proceso retorna al modo usuario.

El código y las estructuras de datos globales del sistema operativo residen siempre en la memoria central y son compartidos por todos los procesos cuando ejecutan rutinas del sistema operativo en el modo sistema. En cambio el código, los datos, los *stack* usuario y *kernel* y los datos de sistema (*User Area*) propios de cada proceso pueden ser mantenidos completamente o parcialmente en disco cuando el proceso no está en ejecución. En efecto, el sistema puede decidir copiar en disco parte de la imagen de un proceso suspendido a fin de liberar espacio en la memoria central (mecanismos de *swapping* y memoria virtual). De todas maneras el proceso que está en ejecución puede acceder a cada ubicación dentro de su espacio de direcciones sin preocuparse si esta está actualmente residente en memoria central o si fué copiada en disco: si el proceso hace referencia a una parte de código o a datos que se encuentran en disco, el sistema operativo se preocupa de hacer accesibles nuevamente en la memoria central las partes de la imagen que interesan. El proceso no se da cuenta que parte de su imagen es copiada de y hacia el disco, porque las transferencias son gestionadas en forma "transparente" por los mecanismos de gestión de memoria del sistema operativo.

5. Implementación del mecanismo de *snapshot* y *restart*

Todas las funciones del mecanismo de *snapshot* y *restart* implementado en el curso del proyecto de investigación fueron realizadas dentro del *kernel* de UNIX™.

A primera vista parecería más simple el uso de un mecanismo a nivel usuario, que podría ser realizado en UNIX™ mediante un programa que acceda a las estructuras de datos del *kernel*, leyendolas directamente de la memoria central de la computadora (que en UNIX™ se puede usar como cualquier archivo de sistema). El problema principal es que los procesos a nivel usuario son entidades concurrentes, así que la imagen del proceso del que se quiere tomar el *snapshot* podría modificarse en el curso de la operación de escritura de las informaciones, por el hecho que esto continua su ejecución en paralelo con el que efectua el *snapshot*, y hacer inconsistente las informaciones guardadas. Aunque el proceso fuese suspendido el mecanismo de gestión de la memoria virtual podría todavía mover partes de su imagen de y hacia el disco mientras el programa de *snapshot* está tratando de leerlas. También se podría pensar en insertar un procedimiento para efectuar el *snapshot* directamente dentro del programa usuario, pero esta solución haría al mecanismo mucho menos flexible requiriendo la modificación y la recompilación de todos las aplicaciones que se quisiera poder tratar con el mecanismo de *snapshot*.

6. Informaciones que forman un *snapshot*

Las informaciones que se necesita guardar para realizar un *snapshot* de un proceso de clase B deben comprender: una imagen de la memoria usada, el valor de los registros, el estado de los archivos abiertos, el directorio de trabajo corriente y, en general, todos los datos del sistema operativo relacionados con la ejecución del proceso y los recursos utilizados.

Hemos decidido guardar estas informaciones en dos archivos, uno en el formato del sistema llamado *core* y un otro en un nuevo formato que hemos llamado *snap*.

Los archivos en formato *core* son normalmente generados por el sistema, con fines de *debugging* cuando se verifican particulares errores en un proceso y comprenden copias de: datos de sistema asociados directamente al proceso(*User Area*), contenido de los registros, *stack kernel* y usuario, área de memoria conteniendo los datos del programa usuario. Se eligió mantener este formato standard, poniendo las demás informaciones necesarias en otro archivo, de manera que su contenido pueda ser examinado usando los instrumentos de *debugging* ya disponibles para UNIX™ (adb, sdb, dbx).

El archivo *snap* contiene todas las demás informaciones, necesarias para la definición del ambiente de ejecución del proceso, que no son previstas para el archivo *core:* informaciones extraidas de la tabla de procesos, datos del sistema operativo que describen los archivos y dispositivos de E/S utilizados por el proceso, informaciones sobre el estado de los dispositivos de E/S y en particular el *setting* de los terminales, fecha del *snapshot*, etc.

Las aplicaciones de clase B requerirían guardar copias de todos los archivos abiertos por el proceso y también del código del programa, puesto que podrían ser modificados o removidos luego del *snapshot* y antes del *restart*. Estas copias de los archivos no son siempre necesarias y podrían ser muy "costosas" en terminos de recursos elaborativos y de memoria de masa requerida. Entonces hemos decidido dar al usuario la posibilidad de elegir si efectuarlas o no.

El mecanismo que realiza el *snapshot*, implementado a nivel del *kernel* de sistema operativo, puede acceder solo a la representación "interna" de los diferentes objetos de un sistema y por esta razón hace referencia a los archivos por medio del número del *inode* (la estructura de datos del *file system* de UNIX™ asociada a cada archivo) y no a través de su nombre (*pathname*), como, en cambio, lo hacen los programas normales de los usuarios. Cuando un archivo es copiado, el número de su *inode* cambia aunque las informaciones contenidas sean siempre las mismas. Entonces se tuvo que proveer algunos programas de utilidad para permitir al usuario controlar y modificar las informaciones contenidas en los archivos *snap* y *core*, efectuando eventualmente las transformaciones necesarias entre la representación de los archivos a nivel usuario (*pathname*) y la interna del sistema (*inode*).

7. Realización del *snapshot*

El *snapshot* de un proceso es producido por una rutina del sistema operativo cuando a este proceso se envia una "señal" particular. Las señales en UNIX™ son un mecanismo para interrumpir temporalmente el flujo de ejecución normal de un programa y pedir la ejecución de particulares procedimientos asociados a la señal que, según los casos, pueden ser funciones del sistema operativo o del programa del usuario. Hemos implementado dos nuevas señales en el *kernel* de UNIX™, para dar al usuario una posibilidad mínima de control sobre el mecanismo de *snapshot:* PSNAP y FSNAP. Mientras la señal PSNAP provoca la escritura de la imágen del proceso (en los archivos *core* y *snap*), la señal FSNAP provoca también la realización de las copias de los archivos abiertos.

El uso del mecanismo de las señales tiene muchos inconvenientes. Por ejemplo las señales no son gestionados inmediatamente siendo tratados por el mismo proceso receptor, que tiene que esperar su turno de ejecución. Además si un proceso está suspendido en espera de un "evento" con alta prioridad (o si fué suspendido por el usuario) no es reclamado inmediatamente por el envio de la señal y la gestión de este último será muy retardada. Otro problema no resolvible utilizando el mecanismo de las señales es que estos no proveen una manera de informar el proceso que requiere el *snapshot* (enviando la señal) si este fué efectuado correctamente o si se verifica algún error.

El motivo por el cual se eligió utilizar el mecanismo de señales para gestionar el *snapshot* es que esto permite conocer exactamente "dónde" y "cuándo" puede ser efectuado la gravación de la imagen de un proceso. Sin esta posibilidad de ubicar el *snapshot* el

mecanismo para efectuar el *restart* del proceso no sería realizable, como se verá en seguida. El mecanismo de señales permite localizar el *snapshot* porque hay un único y particular código del *kernel* que controla la llegada de una señal y ejecuta las acciones asociadas. En efecto las señales son recibidas por el proceso solo mientras ejecuta en el modo sistema en algunos puntos muy bien identificados del código del *kernel*. En particular en UNIX™ 4.2BSD la llegada de las señales es controlada siempre y solo antes que un proceso retome su ejecución en modo usuario, luego de un llamado al sistema operativo o de cualquier interrupción síncrona (debida por ejemplo a la rotación en el uso del procesador impuesta por la política de *scheduling)*.

El uso del mecanismo de señales además permite simplificar notablemente la implementación, puesto que la función de *snapshot* es ejecutada mientras que está en ejecución el mismo proceso implicado. En efecto, por los motivos descritos en el capítulo anterior, el proceso puede acceder en forma directa a todas las ubicaciones de la propia imágen sin importar si parte de esta pudieron ser copiadas en disco a causa de los mecanismos de memoria virtual.

En realidad no habría grandes dificultades en tomar un *snapshot* en cualquier momento, sin embargo una solución de este tipo llevaría a encontrarse con dos grandes problemas en la fase de *restart*, haciendo practicamente imposible su realización. El primer problema está ligado a la seguridad del sistema mientras que el segundo es el problema de restaurar los punteros a los elementos de las estructuras de datos del sistema operativo.

El problema de seguridad surge porque, pudiendo también efectuar el *snapshot* en cualquier punto dentro del código del sistema operativo, se obligaría al mecanismo de *restart* a restaurar el estado de toda la cadena de las llamadas de rutinas del sistema operativo que el proceso tenía eventualmente activas en el momento de su *snapshot*. Para efectuar esta restauración sería necesario utilizar en forma directa las informaciones contenidas en la copia del *stack kernel* guardado en el archivo *core*. Desgraciadamente no hay ningún método practicamente realizable que permita asegurar que el contenido de los archivos que contienen las informaciones del *snapshot* no haya sido modificado en forma peligrosa por un usuario experto. Por ejemplo la copia del *stack* podría ser modificada memorizando direcciones de rutinas o datos del sistema operativo que podrían romper la seguridad del sistema o causar un *crash* del mismo.

El problema de la restauración de los punteros es originado por el hecho de que el *kernel* mantiene muchas tablas cuyos *items* son cargados en forma dinámica cuando se presenta la necesidad (tabla de procesos, de los archivos abiertos, de los *inodes*, ...). Estos *items* son desechados cuando las informaciones que contienen ya no son inmediatamente necesarias. Entonces en general las tablas de datos del sistema operativo pueden contener informaciones muy diferentes en distintos momentos de la vida del sistema: pueden estar ausentes o presentes determinadas informaciones, las informaciones relativas a una misma entidad o recurso pueden ser puestas en posiciones de tabla diferentes, además las posiciones libres en que se pueden insertar nuevos elementos no son siempre las mismas. Frecuentemente las rutinas del sistema operativo acceden a recursos o a datos utilizando punteros a los correspondientes *items* en las tablas de datos de sistema, y suponen que la colocación de estos elementos no cambie hasta que sean utilizados. Durante la fase de restauración de un proceso sería muy poco eficiente y a menudo prácticamente no factible la ubicación de todas las informaciones o de todos los datos en las mismas posiciones originales dentro de las tablas del *kernel*. Entonces todas las rutinas del *kernel* que estaban activas en el momento del *snapshot*, luego de la restauración, encontraran todos los elementos de las tablas de datos que estaban utilizando ubicados en diferentes posiciones, y sus punteros serán inconsistentes.

La mejor solución es la de dejar al *kernel* la tarea de ubicar nuevamente los objetos necesarios para la ejecución del proceso que se quiere restaurar, utilizando sus normales mecanismos, y luego recargar los punteros usados en todas las rutinas de sistema activas en el momento del *snapshot*. Pero si el *snapshot* pudiese ser guardado en cualquier punto del *kernel*, esta solución sería difícilmente alcanzable en la práctica, porque requeriera la modificación de casi todo el código del sistema operativo con la introducción de un número absurdo y pesante de controles para recargar los punteros en el caso de que un proceso estuviera en la fase de restauración.

En la implementación del mecanismo de *restart* la posibilidad de localizar el *snapshot* ha permitido afrontar fácilmente el problema de la recarga de los punteros, introduciendo un único control inmediatamente luego del retorno de la rutina de sistema que gestiona las señales. Este control se preocupa simplemente de recargar el único puntero activo en este punto del código del *kernel* (un puntero a la tabla de los procesos) cuando el proceso que sale de la rutina de gestión de las señales es uno que se está restaurando con *restart*.

8. Reconstrucción de un proceso a partir de un *snapshot*

Hemos introducido una nueva llamada de sistema (*restart*) que efectua la restauración de la ejecución de un proceso a partir de las informaciones guardadas en el *snapshot*. Esta llamada de sistema prevee como argumentos los nombres (*pathnames*) de los archivos *core* y *snap* y del archivo que contiene el código del programa ejecutable, algunos parámetros de control, y la dirección de un *buffer* en que pueden ser puestas las informaciones detalladas sobre las eventuales condiciones de error.

La llamada de sistema *restart* provee, al comienzo de su ejecución, para construir un "esqueleto" del proceso que debe ser restaurado: inicializa oportunamente un elemento de la tabla de procesos, construye una *User Area*, reserva la cantidad de memoria virtual necesaria para el proceso y reserva también espacio en un área de disco para utilizar en caso de *swapping*. El proceso "esqueleto" es inicializado de manera a reflejar el ambiente de ejecución del proceso originario en el momento del *snapshot*. Puede surgir una dificultad cuando se trata de restaurar el identificador del proceso (*pid*), puesto que, en el momento del *restart* dicho número podría haber sido asignado a otro proceso. Sería muy ineficiente, a más de complejo, tratar de impedir la ubicación de todos los identificadores de los procesos congelados. La utilización de mecanismos complicados no sería además justificada puesto que la posibilidad de encontrar ya ubicado el mismo identificador de proceso es muy remota, aunque no excluible a priori. Entonces se adoptó una estrategia muy simple: si no hay requerimiento explícito de parte del usuario, cuando el procedimiento de *restart* se da cuenta de que el número identificador del proceso ya está utilizado, ubica otro entre los identificadores disponibles. Si el usuario sabe que el valor del identificador del proceso está utilizado explícitamente en el algoritmo efectuado por el programa (por ejemplo para generar los nombres de archivos temporales) puede forzar, usando un parámetro de control, el retorno de la llamada de sistema en el caso que este no esté disponible.

El nuevo proceso hereda por el proceso que requiere el *restart* la identidad del propietario y del grupo al que pertenece, los derechos de acceso a los archivos y los eventuales límites sobre la utilización de los recursos. Privilegios particulares del proceso originario no son restaurados por motivos de seguridad del sistema. Por este motivo los programas que utilizan explícitamente derechos particulares de acceso o privilegios, ligados a la identidad o al grupo del propietario pueden encontrar problemas si son restaurados por un usuario diferente del originario.

Luego de haber preparado el proceso "esqueleto" la llamada de sistema restaura el directorio de trabajo corriente del proceso, reabre los archivos y los dispositivos de E/S utilizados y reposiciona adecuadamente los punteros de lectura y escritura dentro de cada uno de estos. Entonces restaura el estado y el *setting* de todas las líneas de terminales utilizadas.

La terminal de control del proceso no debe ser necesariamente el original si no el del usuario que pide el *restart*, en esta forma toda la interacción entre el proceso y el usuario puede ser efectuada correctamente aunque el proceso sea restaurado por una terminal diferente de la originaria. Obviamente un programa que utiliza en modo explícito las características particulares de un cierto tipo de terminal no podrá operar correctamente si la terminal de control, a más de estar en una línea diferente, es de tipo diferente respecto a la originaria.

El nuevo proceso asume el *stack* y los registros *kernel* del proceso que requiere el *restart* y solamente los registros utilizados por el programa usuario son restaurados por la

copia guardada dentro del archivo *core*. Entre los registros del *kernel* hay también el *program counter*, entonces el proceso restaurado comenzará su ejecución desde un punto dentro del código de la rutina *restart* y no desde aquel en que fué guardado el *snapshot*. Este hecho, como veremos, no comporta problema alguno puesto que el *snapshot* está perfectamente localizado por los motivos expuestos anteriormente. Los registros originales del *kernel* no son restaurados por los problemas de seguridad explicados antes. La utilización de los registros usuario originales no conllevan peligro alguno para la seguridad del sistema puesto que, aunque la copia de estos fuese manipulada dentro del archivo *core*, al máximo se podría modificar el comportamiento del programa usuario.

En este punto el *restart* puede insertar el proceso en la cola de procesos *ready* a la espera de ser ejecutados por el procesador. Desde este momento en adelante la llamada de sistema *restart* ya no está en posibilidad de avisar al usuario los eventuales errores que pueden todavía ocurrir durante la última fase de la restauración del proceso.

El proceso restaurado, apenas obtiene el control del procesador comienza a ejecutar el mismo código del proceso que ha llamado el *restart*, puesto que heredó su *stack* y sus registros *kernel*, pero está en grado de autoidentificarse utilizando un mecanismo similar al de la llamada de sistema *fork* (que genera un nuevo proceso, creando una copia exacta de un proceso padre con la única diferencia de un distinto código de retorno sobre el *stack*). Entonces enseguida el proceso restaurado puede comenzar a ejecutar un bloque de instrucciones específico.

El código del *restart* ejecutado en el contexto del proceso restaurado se encarga de cargar su área de datos y su *stack* usuario originales leyendolos del archivo *core*. En el caso de que no hubiese suficiente espacio disponible en la memoria central el mecanismo de memoria virtual se encarga automáticamente de desplazar partes de las áreas de memoria en el área de disco reservada para el *swapping* del proceso. El código del programa usuario es cargado del archivo ejecutable, en el caso de que no exista ya una copia utilizada por otro proceso en ejecución (si se trata de código compartido). El código no es leido del todo inmediatamente, como ocurre con los datos y el *stack* usuario, si no que sus partes son cargadas cuando se necesitan, utilizando un mecanismo llamado de "carga por demanda" (*load on demand*). Este mecanismo es obviamente mucho más eficiente del usado por los datos y el *stack*, puesto que permite no leer partes de código que pueden no ser usadas nunca. Desgraciadamente la versión actual de UNIX™ 4.2BSD prevee la carga por demanda solamente del segmento de código del archivo ejecutable y no de los datos y del *stack*.

El proceso en fase de restauración gestiona también el *timer* en tiempo real, decrementandolo del tiempo pasado del momento del *snapshot* y ejecutando las acciones eventuales previstas por el proceso en el caso (probable) que este *timer* haya expirado. Los otros *timers* previstos por el sistema UNIX™ 4.2BSD no necesitan operaciones particulares de restauración puesto que miden tiempos "virtuales" medidos solo mientras el proceso está en ejecución.

La ejecución del programa usuario originario es retomada al termino de esta fase cuando el proceso retorna al modo usuario, luego de terminar la ejecución de la llamada de sistema *restart*, y sus registros usuario son restaurados desde la copia guardada en el archivo *core*.

El programa usuario debe retomar su ejecución sin que su comportamiento sea modificado en modo alguno. Esta total transparencia es posible a condición de que los diferentes contenidos en los registros y en el *stack* usuario sean adecuadamente preservados de ser alterados por el código de retorno de la llamada de sistema *restart*, que obviamente está privado de significado por el programa restaurado. En efecto las llamadas de sistema modifican los registros y el *stack* del programa antes de retornar al modo usuario, de manera a preparar un adecuado código de retorno. Pero el *restart* ejecutado en el contexto del proceso restaurado, no debe generar valor alguno de retorno, que alteraría los registros y el *stack* arruinando un eventual valor de retorno precedente. Pero, como se explicó antes, el *snapshot* del proceso fué guardado durante la fase de gestión de señales que siempre es ejecutada al termino de cualquier llamada de sistema o de la gestión de una interrupción síncrona: en el primer caso los registros y el *stack* usuario ya fueron modificados de manera a contener el valor de retorno de la última llamada de sistema ejecutada antes del *snapshot*, mientras que

en el segundo caso no están modificados puesto que las interrupciones síncronas son gestionados de manera que no puedan alterar el *stack* o los registros. Entonces en nuestra implementación fué necesario modificar el código original de UNIX™, en el punto que gestiona el retorno de las llamadas de sistema, de forma que se efectue un control sobre una condición particular, que es verdadera solo en la última fase del *restart* y en el contexto del proceso reconstruido. El código asociado a este test se encarga de recargar el único puntero (a la tabla de procesos), utilizado en esta parte del código y en evitar la fase de generación del valor de retorno del *restart*.

El *restart* ejecuta numerosos controles durante todas sus fases a fin de evitar las situaciones inconsistentes. Si un control tiene éxito negativo, el procedimiento deshace todo el trabajo hecho, dejando los recursos eventualmente adquiridos, y termina con un código de error. Es importante poner en evidencia que todos estos controles no tienen el objetivo de asegurar la seguridad del sistema, puesto que están basados en las informaciones contenidas en los archivos que componen el *snapshot*, que en todo caso pueden ser modificados por una persona experta, manteniendo un formato consistente en la aparencia.

Un primer grupo de controles tiene la tarea de asegurar la consistencia de las informaciones contenida en los archivos que componen el *snapshot* (*core*, *snap*, programa ejecutable, archivos en uso por el programa) y comprende los controles cruzados sobre el contenido de las copias de las estructuras de datos de sistema guardadas y sobre las fechas y las dimensiones de los archivos. El usuario tiene la posibilidad de pedir evitar los controles sobre las fechas o las dimensiones de los archivos cuando sabe que el programa que quiere restaurar no está influenciado por modificaciones de los archivos usados (por ejemplo en un programa que manipula archivos actualizados en forma secuencial) o cuando sabe que el contenido de los archivos no es diferente aunque sus fechas cambiaron por alguna razón (por ejemplo porque son copias de los originales). El único caso que nunca puede ser aceptado es el de un archivo que tenga una dimensión menor respecto al original y cuyo puntero de lectura y escritura se refiera a una posición sucesiva a la actual longitud del archivo. Otro grupo de controles es efectuado sobre el contenido de las estructuras de datos del sistema (*inodes)* que se refieren a los archivos, al directorio de trabajo corriente y a los dispositivos de E/S utilizados por el proceso restaurado. Cada *inode* debe corresponder a un objeto efectivamente presente en el *file system*, del mismo tipo del original (archivo,directorio o dispositivo de E/S), el cual debe ser todavía accesible usando los nuevos derechos de acceso que hereda desde el programa que pide el *restart*.

La seguridad está todavía basada sobre los mecanismos standard de UNIX™: los procesos restaurados no mantienen los derechos de acceso y los privilegios originarios pero asumen la identidad del usuario y del grupo del proceso que pide el *restart*. El acceso a los archivos es efectuado usando estos nuevos derechos de acceso, que son asignados y controlados en el modo usual por el sistema operativo. Además el *restart* nunca utiliza directamente las informaciones sobre las estructuras de datos del *kernel*, contenidas en los archivos del *snapshot*, pero las reinstala y las inicializa por medio de los normales procedimientos del sistema operativo. En esta forma la seguridad del sistema no es alterada, y es posible evitar que alguna modificación "maligna" a los archivos que contienen el *snapshot* pueda introducir inconsistencias en los datos del *kernel* o crear "agujeros" en los mecanismos de control del sistema.

9. Conclusiones

En un sistema operativo moderno es muy útil dar la posibilidad de identificar y guardar en disco el estado de un proceso, para poder restaurarlo en cualquier momento posterior (eventualmente luego de una parada de máquina), puesto que permite realizar muchas funciones importantes de alto nivel. Un mecanismo de este género fué realizado para el sistema UNIX™ 4.2 BSD.

Hemos confrontado la imposibilidad de realizar un mecanismo completamente general, que fuese capaz de gestionar todas las aplicaciones posibles que se pueden efectuar en un ambiente UNIX™. Entonces fué necesario hacer algunas hipótesis restrictivas y definir entonces las clases de procesos que pueden ser gestionados en forma eficiente en situaciones

reales. De todas maneras estas hipótesis son todavía suficientemente generales para permitir el tratamiento de muchos casos de interés práctico. El mecanismo para efectuar el *snapshot* y el *restart* de un proceso fué realizado dentro del *kernel* del sistema operativo UNIX™ previendo dos nuevas señales y una nueva llamada de sistema.

La implementación fué realizada en una forma de lo más posible "ortogonal" respecto al sistema originario, de manera a permitir una rápida instalación y previendo el poder transportar el mecanismo de *snapshot* y *restart* lo más facilmente posible sobre las nuevas versiones del sistema operativo. Esta filosofía de proyecto hizo modificar el sistema agregando nuevos módulos y modificando solamente una decena de lineas en el código original.

Quedan todavía muchos problemas abiertos para poder extender el mecanismo actual de manera que pueda gestionar clases de procesos más generales: operaciones conducidas por muchos procesos que se comunican, residentes en una misma computadora y operaciones distibuidas conducidas por procesos ejecutados concurrentemente en diferentes computadoras de una red. El mecanismo actual fué además realizado bajo la forma de prototipo en un ambiente de investigación y entonces debe ser optimizado y hecho más razonable desde el punto de vista de la interface con el usuario antes de poder ser insertado en un producto comercial. De todas maneras el trabajo de investigación desarrollado permitió recoger muchas informaciones importantes y experimentar en la práctica nuevas ideas, que pueden ser usadas para continuar en la investigación y para desarrollar ya unas cuantas aplicaciones interesantes.

Bibliografía

[1] K.M. Chandy, L. Lamport, *"Distributed Snapshots: Determining Global States of Distributed Systems"*, ACM Transactions on Computer Systems, 3.1, 1985, pp. 63-75

[2] D. Presotto, M. Powell, *"Publishing: a reliable broadcast communication mechanism"*, Proc. 9th ACM Symposium on Operating Systems Principles, 1983, pp. 100-109

[3] M. Spezialetti, P. Kearns, *"Efficient Distributed Snapshots"*, IEEE 6th International Conference on Distributed Computation Systems, 1986, pp. 382-388

[4] K. Thompson, *"UNIX™ Implementation"*, The Bell System Technical Journal, Vol. 57, No. 6, Julio 1978, pp. 1930-1946

[5] W. Joy, E.Cooper, R. Fabry, S. Leiffer, K. McKusick, D. Mosher, *"4.2BSD System Manual"*, UNIX™ Programming Manual, Vol. 2c, Sect. 68, University of California 94720

Título: OPTIMIZACION EN TELECOMUNICACIONES

Autor: Dr. Eduardo M. López Soriano
D I R E C T O R

Direccion: Centro de Sistemas de Manufactura
Instituto Tecnológico y de Estudios Superiores
de Monterrey
Campus Monterrey
Centro de Sistemas de Manufactura
División de Graduados e Investigación
Sucursal de Correos J
Monterrey, N.L. 64849
M E X I C O

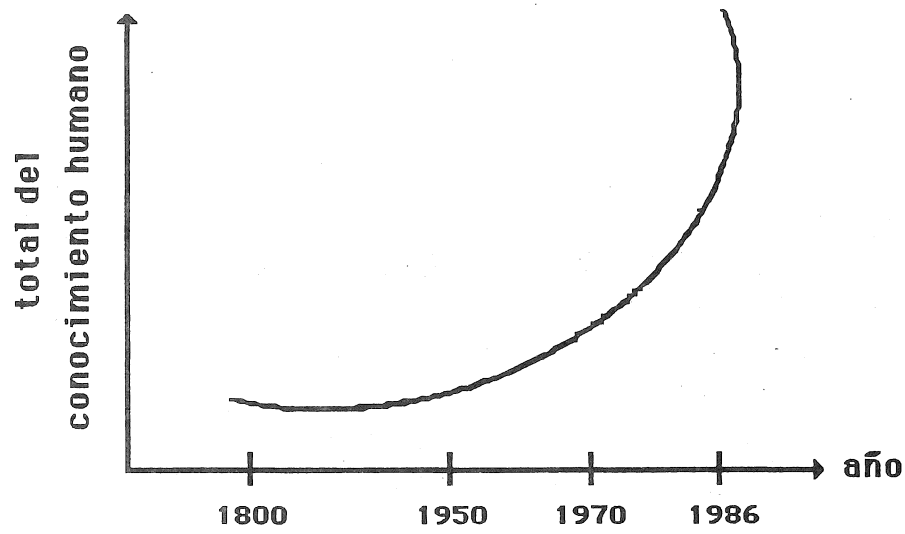
R E S U M E N

Este trabajo presenta dos problemas típicos que forzan a la toma de decisiones adecuada en la práctica de las Telecomunicaciones Industriales. En ambos casos se utilizan herramientas de Optimización Matemática e Investigación de Operaciones para obtener una solución. La primera parte ejemplifica el diseño de una red de Telecomunicaciones bajo un criterio de costo mínimo con lo que se llega a ahorrar hasta un 70% del costo total de diseños más convencionales, sin sacrificar ninguno de los objetivos de la interconexión. La última parte ilustra una forma de enfrentarse a la introducción de ruido en la línea telefónica, que impide la correcta codificación de señales digitales, y se intenta mantener la comunicación de dos computadoras a su máxima velocidad. En cada caso se muestra el problema real que motivó la investigación básica.

I.- INTRODUCCION.

Así como la mezcla de conocimientos de los adelantos textiles y la máquina de vapor durante el siglo XIX revolucionaron el ambiente industrial, la fusión de la ingeniería de telecomunicaciones y la computación nos presenta una gama de nuevos horizontes qué explorar lo que motiva a la investigación superior. Algunas tendencias marcadas se comienzan a visualizar; se está gestionando un cambio en los hábitos de trabajo ya que gracias al teleproceso, mucha gente se queda en casa utilizando una microcomputadora conectada vía satélite a una base de datos remota. Aunque el problema de la "transparencia" en los datos no se ha resuelto en su totalidad, este cambio ha incorporado más gente a la élite productiva.

Sin menospreciar el hecho de los logros alcanzados hasta el momento, se cree firmemente que con la tecnología láser alcanzaremos niveles muy superiores a los presentes en términos de capacidad. Esto es muy importante ya que a medida que avanzamos en el tiempo, mayor es la cantidad de información que debemos procesar con respecto a un tema específico. La siguiente gráfica nos ilustra este hecho:



(fuente: Reporte de AT&T en 1965)

En esta proyección vemos que en el año 1800 se tardaba aproximadamente 50 años en duplicarse el conocimiento humano; en 1950

tomaba 10 años, en 1970 eran 5 años, mientras que para 1986 era 1 año. La cantidad de información generada en el gobierno, el comercio, la industria, las universidades, etc., crece en forma exponencial. No es posible económicamente tener supercomputadoras y bases de datos repetidas por todos lados. De ahí el impacto tan fuerte del teleproceso. Vamos en la dirección de la era Simbiótica: "El limitado cerebro humano será complementado con grandes bases de datos y poder de cómputo de máquinas electrónicas distantes". (Ver [3]).

¿Qué conclusiones podemos sacar al respecto?. Tomando en cuenta que la calidad de las decisiones está en proporción directa a lo actualizado que esté la información que involucramos en el proceso, cualquier organización esparcida deberá considerar el teleproceso como una seria alternativa en el mecanismo del sistema de información para apoyo a la dirección gerencial. De otra manera, es posible que la dinámica y las presiones exteriores sitúen a dicha organización en una posición desventajosa y díficil de mejorar.

Este artículo se presenta con la intención de ilustrar dos tipos de problemas que se pueden considerar como clásicos dentro del ámbito de proceso computacional vía Telecomunicaciones. Ambos ejemplos son producto de la investigación y servicio a la industria que presta el Centro de Sistemas de Manufactura de la división de Graduados e Investigación del ITESM Campus Monterrey. Con esto se pretende mostrar una forma de franquear obstáculos que se presentan en el diseño y la operación de redes de Teleproceso Industrial que deben de funcionar constantemente y con alta confiabilidad.

II.- DISEÑO DE UNA RED DE TELECOMUNICACIONES

A continuación presentamos un ejercicio donde se aplica la toma de decisiones en la etapa de diseño. Es importante hacer notar algunas observaciones y asumciones hechas en el desarrollo: la red de telecomunicaciones se diseña bajo un criterio de costo mínimo y la confiabilidad de la red resultante se dá por un hecho; se asume que existen instalaciones telefónicas adecuadas y suficientes para intercomunicar dos ciudades cualquiera; y se presume que el "software" requerido para la red también está disponible. Comentarios adicionales y alternativas de manejo de estos factores se proporcionarán al final de esta sección.

II.1.- Definición del problema

El Sistema del Instituto Tecnológico y de Estudios Superiores de Monterrey es un conjunto de 26 instituciones mexicanas independientes académicamente pero unificadas bajo una misma administración. Cada uno de los campus pretende ofrecer educación universitaria y apoyar la industria regional para el fortalecimiento y el buen desarrollo económico. El hecho de pertenecer a la misma organización le obliga a cada campus a seguir ciertas normas y niveles de excelencia mínimos indispensables para mantener el prestigio internacional que en más de 40 años se ha forjado el I.T.E.S.M.

Administrar una organización tan esparcida es también una función compleja. La figura No. 1 ilustra la distribución geográfica del sistema ITESM en un mapa de la República Mexicana. 18 de los 31 estados federales tienen representación institucional. La Rectoría del Sistema está localizada en la Cd. de Monterrey. Desde ahí, información fluye en una y otra dirección en forma continua por lo que es importante disponer de los medios adecuados para sortearla, clasificarla, y procesarla a fin de que se presente correctamente a las autoridades que toman las decisiones.

Desde hace un tiempo a la fecha se estuvo manejando la idea de estudiar la posibilidad de comunicar todo el Sistema ITESM desarrollando una red computacional que conectara a todos con el Campus Central. Múltiples beneficios académicos y administrativos se visualizan si se contara con una red de esta naturaleza. El solo hecho de estar en línea con una supercomputadora dá un poder de cómputo adicional además de acceso remoto a fichas bibliográficas, bases de datos, correo electrónico, RJE's,

etc. De la misma manera, la recopilación de información administrativa sería también muy inmediata y se evita la duplicidad de funciones.

Desde el año de 1985, México se convirtió en el primer país Latinoamericano que tiene su propio sátelite para Telecomunicaciones. Esto abrió una nueva dimensión y alternativas interesantes. Aún así, la tecnología es cara y no necesariamente está al alcance de todos. Por lo tanto, nos avocaremos a diseñar una red en forma más convencional que utilize la línea privada telefónica como medio de interconexión entre dos computadoras.

Tradicionalmente, el uso de una línea telefónica se cobra en función de la distancia. El hecho de seleccionar líneas privadas no las exenta de tal característica en el cobro, ya que se hizo por motivos de disponibilidad. En la actualidad se ha alcanzado cierto grado de sofisticación en el software de telecomunicaciones; hubo una época en que la única comunicación entre dos computadoras era a través de una línea física sin puntos intermedios. Tal configuración ha cambiado ya que ahora se permiten "saltos" o "bypasses" de puntos intermedios. Esto permite más flexibilidad de diseño, al incorporarse alternativas adicionales.

El objetivo global será pues diseñar una red de telecomunicaciones que conecte los campus del Sistema I.T.E.S.M. teniendo como Computadora Central o "Host Computer" a la máquina localizada en el Campus Monterrey. Aprovechando ventajas de software que ignora puntos intermedios, nos limitamos a optimizar sobre el conjunto de redes cuya característica es que existe una ruta única entre dos nodos arbitrarios de la red.

El criterio de optimización será encontrar esa red que tenga una cuota mínima con respecto a las otras posibles redes. Como se vió anteriormente, debido a que una línea telefónica se cobra en proporción directa al número de kilómetros aéreos que utiliza, la configuración que buscamos es una red con la mínima distancia total en las líneas de conexión.

II.2.- Modelo Matemático

En forma abstracta podemos representar las localizaciones o ciudades que se quieren incorporar a la red como nodos en una red de teoría gráfica. Dicha red tendría n+1 nodos identificados como $i_0, i_1, i_2, \ldots, i_n$ donde i_0 sería el nodo que representa a la computadora central. A fin de configurarla,

colocamos arcos que comienzan de i_0 y terminan en todos los demás nodos; a su vez, definimos arcos cuyo nodo inicial es i_1 y los nodos terminales son $i_2, i_3, .., i_n$; y así sucesivamente. Con esto generamos una red con $n(n+1)/2$ arcos enlazando a los n+1 nodos. (Ver figura 2).

La numeración o asociación de los nodos es irrelevante excepto por lo que se refiere a la Computadora Central. Por conveniencia se escogió i_0 y nos sirve posteriormente como punto de referencia.

Sea j_{rs} el arco cuyo nodo inicial es i_r y el correspondiente nodo terminal es i_s. Entonces $x(j_{rs})$ es un "flujo" asignado a j_{rs} restringido al intervalo [0,1] y d_{rs} es la distancia total de i_r a i_s. Dentro de la teoría de Optimización de Redes, una red con la característica de tener un camino único entre dos nodos arbitrarios se le denomina un "árbol" (ver [4]). Por lo tanto, la representación matemática de la búsqueda de aquél árbol cuya distancia total sea mínima la podemos ilustrar tomando ventaja del flujo que impusimos en la red y expresando el modelo como un problema de Programación Líneal:

$$\text{Minimizar} \quad \langle d, x \rangle$$

$$\text{sujeto a:} \quad Ex = y$$
$$ex = n$$
$$|E|\, x \geq 1$$
$$x \in [0,1]$$
$$x : \text{entero}$$

donde E es la matriz de incidencias de la red, e es un vector de 1's, y |E| es una matriz que contiene el valor absoluto de los elementos de cada vector de incidencias de todos los posibles cortes de la red que no sean el conjunto vacío.

II.3.- Alternativas de Algoritmos

Tratando de buscar solución a nuestro problema, pudimos encontrar respuesta aplicando el Método Simplex al modelo de Programación Lineal. Sin embargo, una vez dejada la parte experimental y tratar de utilizar

dicha metodología al problema real, uno se enfrenta a la problemática de la inspección de cortes en una red. Si n computadoras se quieren conectar a una computadora central, con la configuración de red abstracta que formamos para imponerle un flujo se generan 2^n-1 ecuaciones, debido a los cortes que se deben de considerar, haciendo el problema de Programación Lineal totalmente impráctico para redes medianas o grandes.

Aún así, continuamos la busqueda y se detectó un algoritmo diseñado por Kruskal que en forma ágil va configurando la red de comunicaciones. El procedimiento en sí es sencillo: dada una red "optima", se identifica a aquel nodo no conectado cuya distancia sea mínima a la red ya configurada y se incorpora apropiadamente. En exactamente n iteraciones el procedimiento termina sin problemas de inestabilidad numérica o errores de truncación y redondeo. Para mayores detalles en la mecánica, el lector es referido a [4] y [5].

II.4.- Resultados

Una vez implementado el algoritmo en la IBM PC, la etapa de recolección de datos se llevó a buen fin y varias alternativas de diseño se llevaron a cabo. La tabla 1 muestra la asignación de nodos en la red. Es importante remarcar el hecho de que dos Campus fueron eliminados por estar en la misma ciudad donde una supercomputadora iba a estar instalada, haciendo su enlace trivial.

El primer diseño asume que el Campus Central de la ciudad de Monterrey tiene la capacidad de dar servicio a todo el Sistema Nacional. La figura 3 ilustra la forma clásica de conectar en forma directa cada computadora a la máquina central dando una distancia total de 15,529.5 Kms.

La figura 4 muestra el diseño final una vez aplicado el algoritmo. En esta ocasión la distancia total es de 4,714.4 Kms. lo que representa un ahorro de 70% en el pago de la cuota mensual. A largo plazo esto es muy significativo ya que el costo de operar la red se vuelve altamente contributivo y difícil de controlar. Además, cambios en la red ya operando son demasiado complicados lo que hace que una vez diseñada la estructura de comunicación, se mantenga fija. Es por eso que el incorporar consideraciones de operación en la etapa de diseño es imperativo para augurar el éxito de la implementación y operación.

Una vez presentado este diseño, una alternativa posterior surgió al colocarse una IBM 4381 en el Campus Querétaro y otra igual en el Campus

Estado de México. Esos dos Campus fueron conectados directamente y una línea privada se instaló del Campus Querétaro al Campus Central mientras que una línea adicional sale de Monterrey a San Antonio, Texas conectando el Sistema ITESM a la red norteamericana universitaria BITNET. Con esta nueva configuración, se pensó entonces que el diseño se haría en función de recibir servicio de una computadora conectada al BITNET, no necesariamente la del Campus Monterrey. La figura 5 ilustra el diseño que interconecta cada computadora en forma directa a la supercomputadora más cercana. La distancia total es de 10,135.3 Kms. incluyendo la conexión Monterrey-Querétaro-Edo de México.

En la figura 6 se muestra la configuración final diseñada a través del algoritmo y cuya distancia total es de 4,965.9 Kms. lo que totaliza un ahorro de 51% en la cuota mensual.

II.5.- Extensiones y Otras Aplicaciones

Un problema característico de las redes de Telecomunicaciones es la confiabilidad de la configuración. Imagínese el lector que se adopta el primer diseño propuesto por el algoritmo y una vez implementado, se dá uno cuenta de que constantemente se cae la línea de comunicación que une el Campus Saltillo con el Campus Central. De inmediato notamos que el resto del Sistema queda sin servicio haciendo inútil todo el esfuerzo. Con ligeras modificaciones, el algoritmo utilizado nos ayuda a identificar la red más confiable, aunque no necesariamente será la más barata. Un balance entre ambos criterios, por ejemplo: encontrar la red más confiable que no exceda un presupuesto, sería tal vez lo más apropiado. De ahí el valor de modelar el Problema como uno de Programación Lineal, ya que puede ayudar a encontrar soluciones en forma estratificada.

Una aplicación adicional del algoritmo se ha encontrado en el diseño de redes de abastecimiento como suministro de agua, de gas natural, electricidad, etc. En ese tipo de actividades también el costo de instalación está en proporción directa con la distancia que se cubre, haciendo atractivo el objetivo de minimizar la distancia total. De hecho, ya en la operación, los costos de inspección y mantenimiento también son función directa de la distancia. Conceptualmente, todos estos sistemas se pueden modelar bajo las mismas consideraciones matemáticas colocándolos en la misma clase de problemas.

III.- DETERMINACION DEL TAMAÑO OPTIMO DEL BLOCK DE TRANSMISION.

En Septiembre de 1985 la ciudad de México sufrió una de las tragedias más amargas de toda su historia. Un terremoto de intensidad 8.2 en la escala de Richter devastó la ciudad matando a miles de personas y destruyendo la zona centro. Uno de los edificios de Teléfonos de México, donde residían un buen porcentaje de las instalaciones dirigidas a telecomunicaciones, quedó reducido a ruinas dejando prácticamente incomunicada la capital de México con el resto del mundo. Casi de inmediato, la Secretaría de la Defensa Nacional asumió el control de todas las líneas privadas de transmisión de datos, cuyas instalaciones de apoyo no sufrieron avería, ya que el estado de emergencia así lo exigía.

Todas las Compañías y organismos que fueron afectados por la pérdida de su línea privada, se avocaron a transmitir en forma conmutada y se enfrentaron a una problemática relativamente diferente por las características más vulnerables de este tipo de telecomunicaciones.

III. 1.- Identificación del problema.

La mayoría de las líneas de telecomunicaciones no fueron diseñados para transmitir datos computacionales, conocidos como señales digitales. Fueron hechas para enviar y recibir señales de voz, denominadas señales analógicas. Por lo tanto, equipo especial es requerido para convertir unas señales a otras dependiendo de quien envía y quien recibe los datos. Una configuración ilustrativa se muestra en la figura 7.

Las líneas conmutadas (o públicas), a diferencia de las líneas privadas, están sujetas a mayor cantidad de "ruido" o distorsión que afecta el tiempo total de transmisión. Ruido es una incertidumbre que puede ser originada por causas externas (estática atmosférica, cables tocando la rama de un árbol, etc.) o por causas internas (malas conexiones, registros subterráneos inundados, etc.) y genera errores al decodificar la señal recibida. Cuando un error es detectado, la estación receptora envía un mensaje negativo a la estación transmisora, lo que forza a que ésta envíe los datos otra vez. A mayor longitud de mensaje, mayor probabilidad de error; por eso, un mensaje largo puede que necesite de varios intentos hasta ser recibido correctamente.

Una forma de mejorar la eficiencia en la transmisión es dividiendo el mensaje en blocks de datos de tal manera que si un error se detecta, solamente un block se repite y no todo el texto. Sin embargo, el fraccionar un texto en blocks no debe de ser hecho en forma arbitraria ya que cada block es verificado y la estación receptora debe de contestar si hubo discrepancia o no; por lo tanto, subdividir un texto en muchos blocks retardaria el tiempo total de transmisión considerablemente. Es de esperarse que de alguna manera exista un balance, o sea, que dada una longitud de texto y después de estimar la cantidad de ruido en la línea conmutada podamos determinar en cuántos blocks se debe fraccionar el mensaje a fin de que el tiempo total de transmisión sea mínimo. Basados en la experiencia de apoyar a una empresa manufacturera cuya dependencia en telecomunicaciones para facturación, control de pedidos e inventarios, y decisiones sobre crédito y cobranza en un depósito localizado en la Cd. de México era crucial, presentamos el análisis desarrollado y la metodología propuesta para solucionar este problema. Aunque las líneas privadas no están exentas de ruido, la intensidad del mismo es mucho más remarcable en líneas conmutadas lo que orilla a buscar alternativas como la que se propone.

III. 2.- La Mecánica de transmisión.

La interferencia y ruido en las líneas de comunicaciones pueden cambiar bits de datos de tal manera que no serían reconocidos por la estación receptora. Además, la estación receptora puede también aceptar ruido como bits de datos. Si un bit en un caracter cambia, ese caracter y posiblemente los caracteres subsiguientes no serán válidos. Por lo tanto, los datos son generalmente validados contra cambios, pérdidas, o adiciones de bits.

Existen tres formas comunes de validar datos:

a) VRC (Vertical Redundancy Checking) que inspecciona cada caracter del block y determina la paridad.

b) LRC (Longitudinal Redundancy Checking) el cual utiliza todos los bits de cada caracter en el block para formar un caracter de validación.

c) CRC (Cyclic Redundancy Checking) que apoyándose en una formulación matemática, genera un par de caracteres de validación. Este es actualmente el procedimiento más acertado de los tres.

Todos los métodos son efectuados en forma automática por el programa de telecomunicaciones. Para mayor certeza en los datos, los tres métodos son generalmente adoptados ya que se complementan unos a otros.

III. 3.- Representación Matemática.

Supongamos que q es la probabilidad de que un byte transmitido tenga error en el transcurso del proceso. Asumiendo que el error de transmisión sucede en forma independiente de byte a byte y considerando m bytes por block, tenemos que la probabilidad de error en la transmisión de un block completo está dada por:

$$p(n) = 1 - (1 - q)^{L/n}$$

donde

L = longitud del texto que se quiere transmitir (en número de bytes).

n = número de blocks en que se fracciona el texto.

$$m = \frac{L}{n}$$

Nuestra incógnita o variable de decisión será n, la cual es un número comprendido entre 1 y L, que deberá ser entera. La figura 8 nos ilustra el comportamiento de p(n) en donde se puede observar que a mayor número de blocks en que subdividimos el texto, menor será la probabilidad de que se incurra en error al transmitir un block; por otro lado, a medida que crece la longitud de texto aumenta la probabilidad de error en un block si n se mantiene constante.

Con esta probabilidad p(n) podemos calcular entonces el número de veces promedio que un block de información se deberá de transmitir hasta que sea recibido en forma exitosa:

$$\frac{1}{1 - p(n)} = 1 + \frac{p(n)}{1 - p(n)} = 1 + h(n)$$

Dado un texto de longitud L, se debe determinar el tiempo total de transmisión en función del número de blocks n, para luego identificar el valor de n que minimiza la función tiempo total. Al enviar un block de m bytes existe un tiempo de verificación c al inicio y al término del mismo, además de un tiempo z de cambio de transmisor a receptor y viceversa cuando se detecta error en la telecomunicación. El modem, trabajando a una velocidad de transmisión v = bps/8, tarda T=1/v en enviar un byte de computadora a computadora. En paralelo se van efectuando los procedimientos de verificación de los cuales tanto VRC como LRC tardan en forma directamente proporcional al número m [o(m)]. Por otro lado, CRC es una formulación matemática más compleja que al investigar el número de operaciones que desarrolla, se relaciona directamente proporcional al cuadrado del número de bytes que componen el block [$o(m^2)$]. Al conjuntar todos estos componentes del tiempo de transmisión por block y multiplicar por n (el número de blocks en que fraccionamos L) obtenemos la siguiente expresión del tiempo total de transmisión:

$$\text{tempo total de transmisión} = n\left(1 + h(n)\right)\left\{\frac{LT}{n} + c + K_1\frac{L}{n} + K_2\frac{L}{n} + K_3\left(\frac{L}{n}\right)^2\right\} + 2n \cdot h(n) \cdot z$$

lo que es lo mismo que:

$$\text{tiempo total de transmisión} = f_1(n) + f_2(n) + \text{constante}$$

donde f_1 (n) es una función lineal que incrementa monotónicamente, f_2 (n) es una función no lineal convexa, y se suma una constante adicional de tiempos no dependientes del número de blocks. La figura 10 nos muestra el comportamiento de las funciones y del tiempo total de telecomunicación.

Una vez desarrollada la expresión matemática, existen varias alternativas para encontrar n^* (el número de blocks que minimiza el tiempo total). Sin embargo, es muy probable que ese valor obtenido no sea entero. Se aconseja calcular el tiempo total de transmisión para n_i y n_s, donde n_i y n_s son números enteros tales que

$$n_i \leq n^* \leq n_s$$

$$n_i + 1 = n_s$$

$$n_i, n_s : \text{enteros}$$

y entonces tomar una decisión.

III. 4.- Comentarios, Extensiones, y Otras Aplicaciones.

Procedimientos que involucran la identificación de todos los elementos que contribuyen a un sistema son muy importantes dentro del ámbito industrial. En particular, sistemas de costeo e ingeniería económica manufacturera tienen un valor estratégico en las empresas que pueden significar ese diferencial competitivo que se está buscando. Curiosamente este ejercicio tiene una similitud en la mecánica con otros relacionados hacia consumos industriales y costos de producción que inclina a pensar en la unificación de criterios y formulaciones en ambientes bajo incertidumbre.

La estimación del valor de la probabilidad de error en un byte (q) puede calcularse de diferentes maneras. Sin embargo, no es una estimación única la que se debe hacer sobre todo si el texto es largo ya que debido a la dinámica de las telecomunicaciones es muy factible que la magnitud de q varíe durante el proceso. Esto también conduce a la elaboración de un software de control capaz de alterar la forma de bloquear dinámicamente como procedimiento retroalimentador.

IV.- CONCLUSIONES.

Ambos ejemplos ilustrados en este artículo presentan una metodología en común: se detecta el sistema físico en el estado en que se encuentra conceptualizándose en forma abstracta a través de un modelo matemático, para después llevar al sistema a un nivel "optimo" de operación utilizando un procedimiento que manipula el modelo. A esto se le conoce dentro del área de Investigación de Operaciones como un programa y nos ayuda a determinar las magnitudes de los parámetros que intervienen para funcionar cómodamente bajo un criterio establecido.

La parte más difícil del proyecto es posterior a estas etapas. Llevar a la práctica los resultados obtenidos no es necesariamente inmediato y generalmente salen a relucir todos aquellos factores no considerados en el modelo matemático o en el algoritmo de optimización. La mayoría de los casos exige una negociación entre resultados teóricos y resistencia al cambio. Sin embargo, cualquier modificación en la dirección óptima es superior a las condiciones actuales de operación. Los dos ejercicios sirven también de aliciente para la adopción de herramientas que motiven el mejoramiento continuo dentro del ámbito computacional.

V.- BIBLIOGRAFIA

1.- Dantzig, George B.
Linear Programming and Extensions
Princeton University Press
Princeton, New Jersey
1963

2.- Martin, James T.
Telecomunications and the Computer
Prentice-Hall, Inc.
Englewood Cliffs, N.J.
1969

3.- Martin, James T.
Future Developments in Telecomunications
Prentice-Hall, Inc.
Englewood Cliffs, N.J.
1977

4.- Rockafellar, R. Tyrrel
Network Flows and Monotropic Optimization
Wiley, 1984

5.- Whitehouse, Gary E.
Systems Analysis and Design Using Network Techniques
Prentice-Hall, Inc.
Englewood Cliffs, N.J.
1973

RELACION DE LOS CAMPUS DEL SISTEMA TECNOLOGICO DE MONTERREY

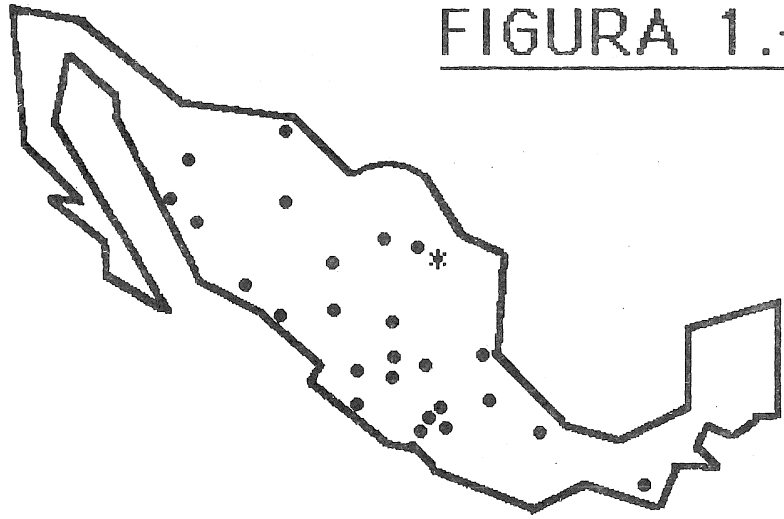
CONSIDERADOS EN EL ESTUDIO

Nodo	Campus
0.-	Monterrey
1.-	Querétaro
2.-	Edo. de México
3.-	Guaymas
4.-	Chihuahua
5.-	Cd. Obregón
6.-	Laguna (Torreón)
7.-	Sonora Norte (Hermosillo)
8.-	Saltillo
9.-	Mazatlán
10.-	Cd. Juárez
11.-	Sinaloa (Culiacán)
12.-	San Luis Potosí
13.-	León
14.-	Colima
15.-	Irapuato
16.-	Tampico
17.-	Toluca
18.-	Zacatecas
19.-	Guadalajara
20.-	Morelos (Cuernavaca)
21.-	Hidalgo (Pachuca)
22.-	Central de Veracruz (Córdoba)
23.-	Chiapas (Tuxtla Gutiérrez)

NOTA: No se tomaron en cuenta los Campus Eugenio Garza Sada (Monterrey) y Cd. de México por considerarse obvia su vía de comunicación y enlace.

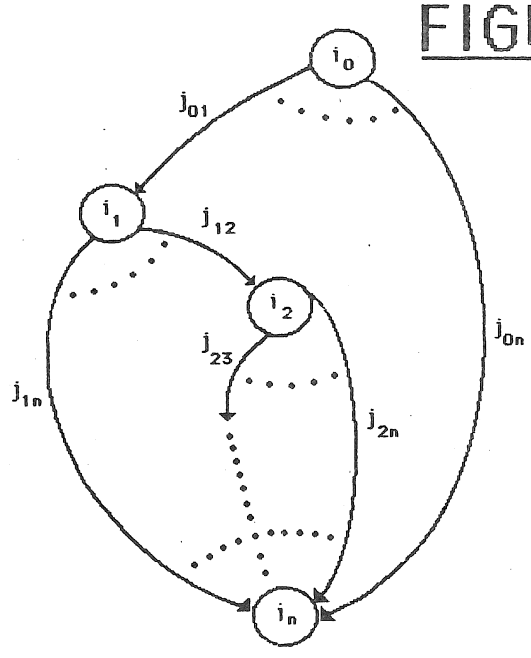
TABLA 1

FIGURA 1.-



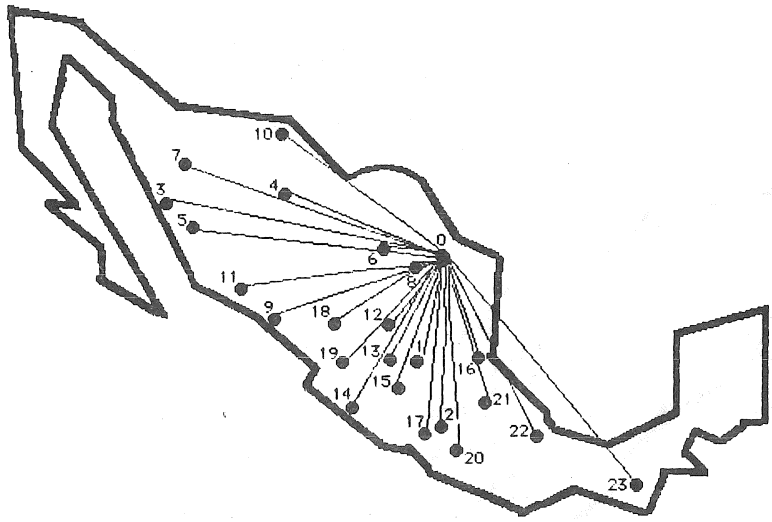
LOCALIZACION DE
LOS CAMPUS

FIGURA 2.-



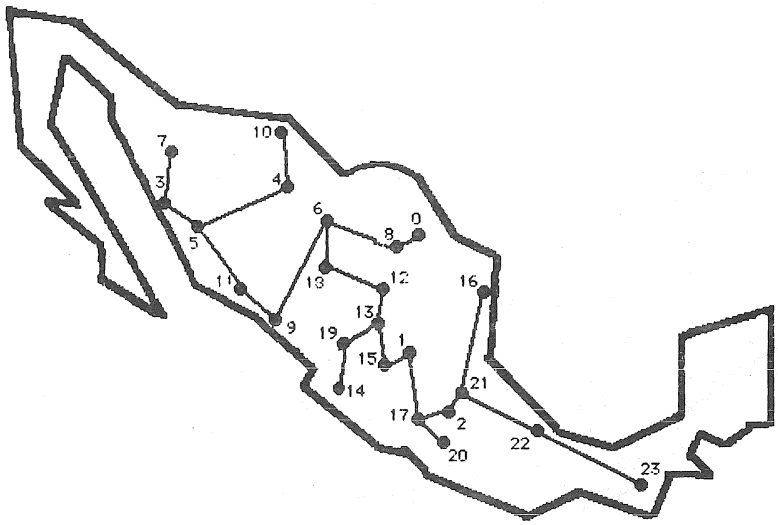
CONFIGURACION DE LA RED

FIGURA 3.-



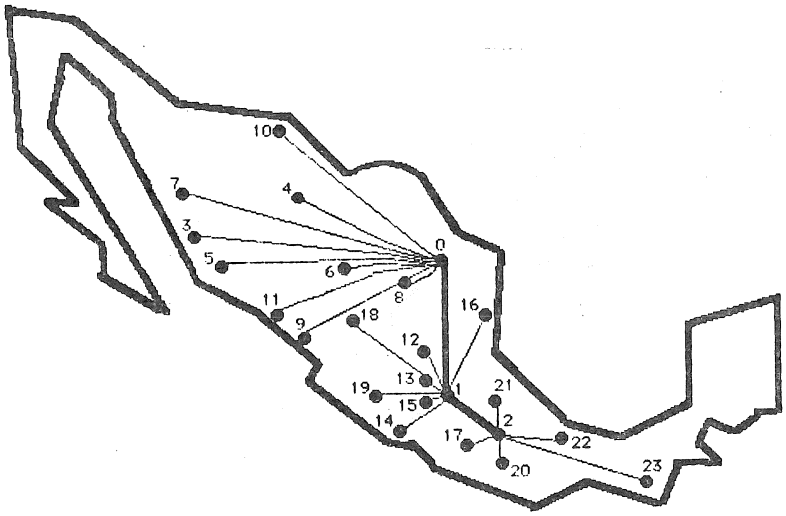
DISEÑO TRADICIONAL CON SERVICIO CENTRALIZADO

FIGURA 4.-



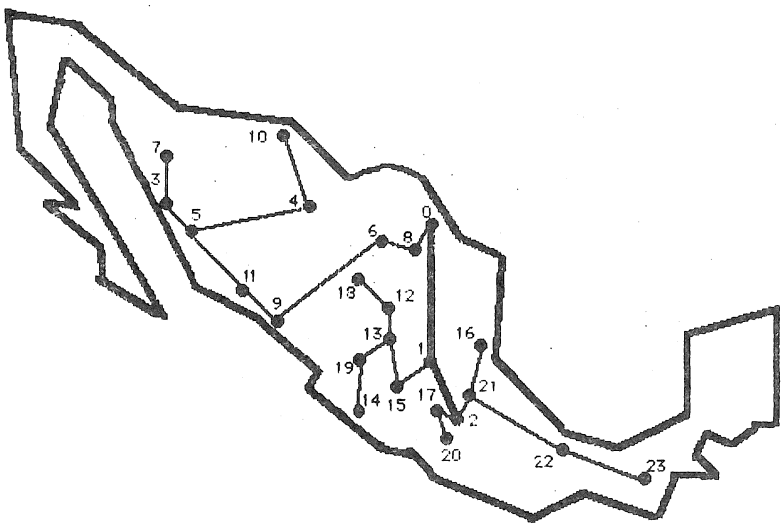
RED "OPTIMA" CON SERVICIO CENTRALIZADO

FIGURA 5.-



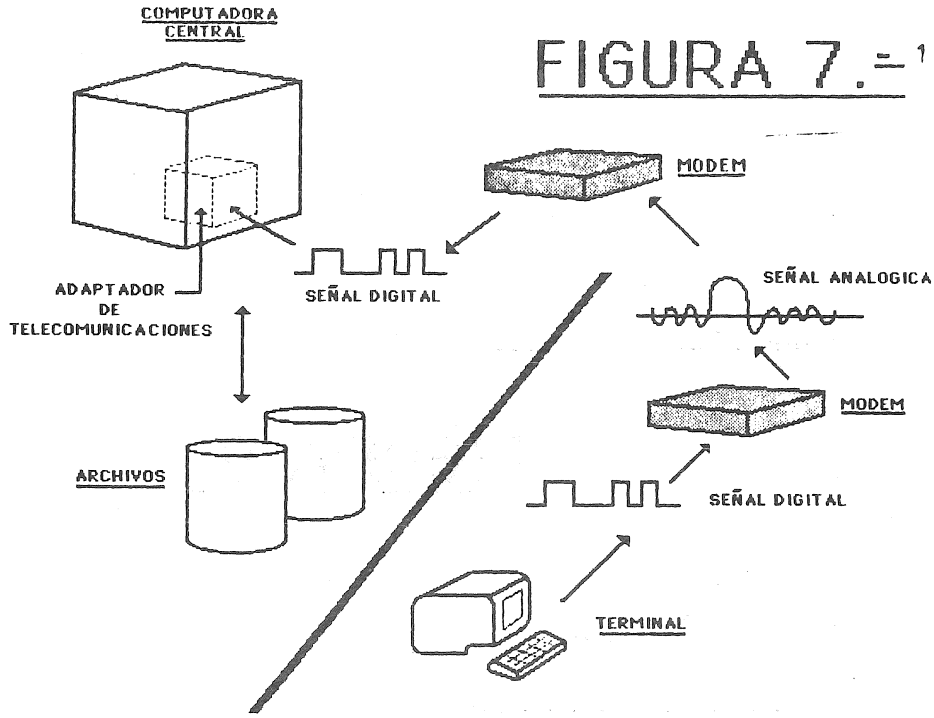
CONFIGURACION TRADICIONAL CON SERVICIO DISTRIBUIDO

FIGURA 6.-



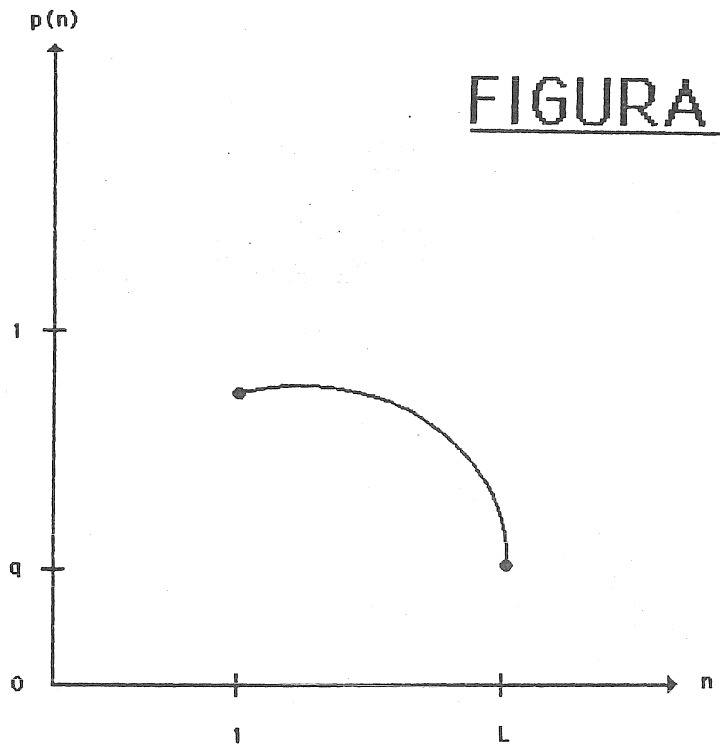
DISEÑO "OPTIMO" CON SERVICIO DISTRIBUIDO.

FIGURA 7.-

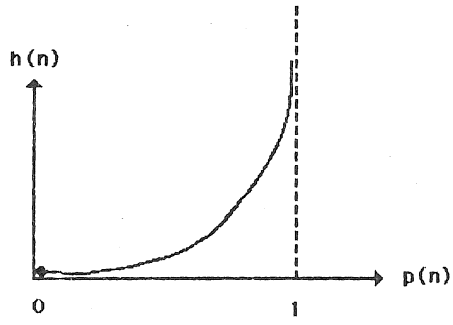


EJEMPLO DE CONFIGURACION EN TELECOMUNICACIONES

FIGURA 8.-



COMPORTAMIENTO DE LA PROBABILIDAD DE QUE EXISTA ERROR EN LA TRANSMISION DE UN BLOCK.



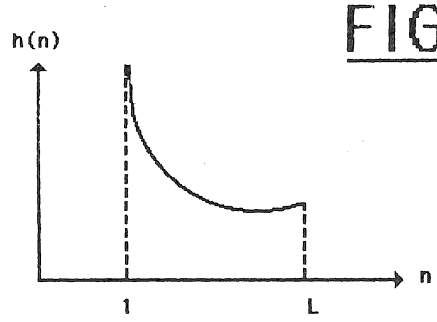


FIGURA 9.-

COMPORTAMIENTO DEL NUMERO DE VECES QUE SE REPITE
LA TRANSMISION DE UN BLOCK.

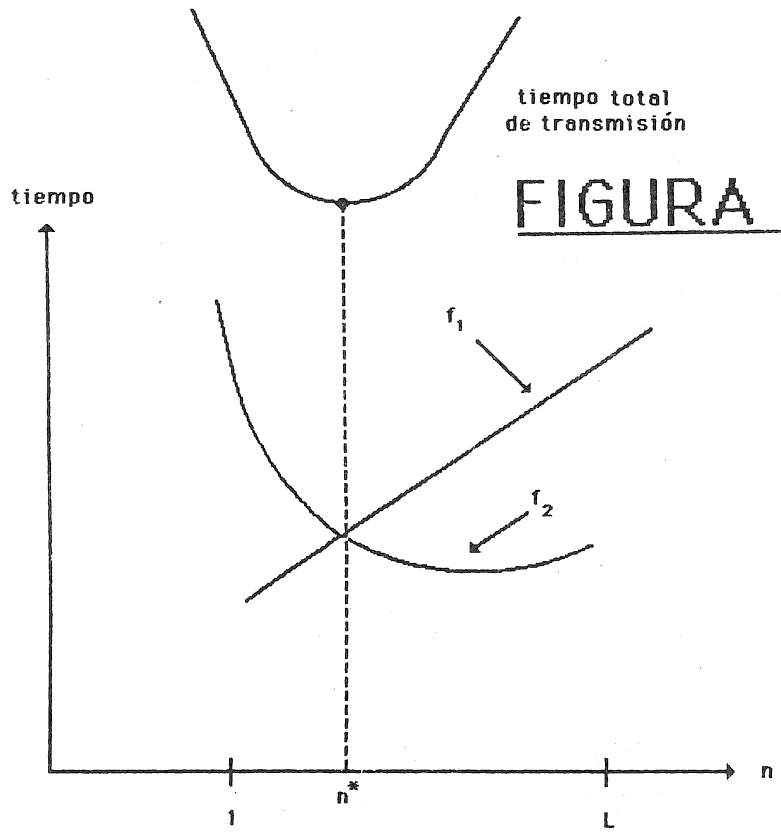


FIGURA 10.-

LOCALIZACION DEL VALOR OPTIMO DE n

Análisis de la utilización máxima de un canal de comunicaciones en un enlace vía radio

Bria Oscar Norberto *
De Giusti Marisa Raquel**

Trabajo realizado en el marco del Proyecto Procesadores enlazados vía radio.
Dirección del Proyecto: Lorente Hugo Enrique ***
Subdirección del Proyecto: De Giusti, Armando Eduardo ****

El presente Proyecto se está desarrollando en el Centro de Técnicas Analógico- Digitales (CeTAD)- Facultad de Ingeniería - Universidad de La Plata- Buenos Aires - Argentina.
Director del CeTAD: Quijano, Antonio Adrian *****.

* Becario de Perfeccionamiento Comisión de Investigaciones Científicas de la Pcia. de Buenos Aires- Argentina.

** Becario de Formación Superior del Consejo Nacional de Investigaciones Científicas y Técnicas. Argentina.

*** Profesor Titular de la Universidad Nacional de La Plata. Buenos Aires - Argentina.

**** Investigador Independiente Consejo Nacional de Investigaciones Científicas y Técnicas. Argentina y Profesor titular de la Universidad Nacional de La PLata.

***** Profesor Titular de la Universidad Nacional de La Plata y Presidente de la Comisión de Investigaciones de la Universidad Nacional de La Plata.

Abstract:

Packet Radio Networks exhibit certain special characteristics, attending its natural "half duplex" link scheme,and high bit error rate.

This paper demostrates the convenience of partitioning the transmission burst in a set of isolated sequencial frames, and gives some rules for selecting the partitioning number.

1) Resumen:

Las redes de conmutaciòn de paquetes enlazadas vìa radio (Packet-Radio networks), pueden ser organizadas en forma de una estructura jerårquica tal como la definida por la Organizaciòn Internacional de Standards (ISO), para facilitar la intercomunicaciòn de dispositivos de procesamiento de datos.

El presente artículo està relacionado con el Nivel 2 o Nivel de Trama (modelo OSI de interconexiòn de sistemas abiertos de la ISO) y provee modelos y gràficos ùtiles como guìa para la selecciòn del tamaño òptimo de la trama (*) y el nùmero de tramas involucradas en cada transmisiòn simple en el medio de radio.

En este anàlisis se considera como fuente de error la presencia de ruido completamente aleatorio en el canal que afecta a las tramas intercambiadas entre procesos pares, considerando el consumo de bits por sincronismo, secuenciaciòn de tramas enviadas y tècnica de reconocimiento "a caballito" (piggybacked ack).

* El tèrmino trama (frame) no es el tèrmino oficial de la ISO para la unidad intercambiada por los procesos pares del nivel 2. El tèrmino oficial es data-link-service-data-unit, sin embargo es aquèl el usado màs frecuentemente.

2) Introducciòn:

El nivel de trama debe realizar la funciòn de tomar una facilidad de comunicaciòn y transformarla en una lìnea que aparece libre de errores de transmisiòn para el nivel superior (Nivel de red). Para cumplir esta tarea particiona los datos de entrada en tramas, transfiriendolas secuencialmente y procesando las estructuras de reconocimiento enviadas por el receptor.[Ref.1,2,3,4,5].

Una trama o frame es una estructura particular compuesta fundamentalmente (y con diferencias de protocolo a protocolo) de cuatro campos bàsicos:

Clase Secuencia Reconocimiento Informaciòn

Los tres primeros contienen informaciòn de control y el ùltimo contiene los datos a ser transferidos; los tres campos iniciales constituyen la "cabeza" (header) de la trama.

El campo de clase nos dice si hay datos en la trama o si lleva sòlo informaciòn de control.

Los campos de secuencia y reconocimiento asignan nùmeros consecutivos (que reciclan al llegar a un màximo) a cada porciòn de informaciòn y reconocimiento respectivamente.

En la mayoría de los protocolos se asume un canal de comunicaciones que pierde tramas completas en determinadas ocasiones. Para solucionar estos problemas el transmisor inicializa por ejemplo, un temporizador interno cada vez que envía un frame; si en un apropiado tiempo no recibe confirmación de su trama, el reloj genera una alarma y la computadora encargada de las comunicaciones reenvía la unidad.

Dado que el receptor no puede distinguir entre un frame y otro, existiría la posibilidad de duplicaciones de información: por ello el transmisor debe adicionar a la información enviada un número de secuencia,de modo que el receptor pueda chequear este número para ver si la trama está duplicada o no.

Dado que no puede obviarse el hecho que el receptor no puede procesar datos a una velocidad infinita, deben implementarse determinados mecanismos de prevención. [Ref.1,6].

Uno de estos mecanismos se denomina "pare y espere" (stop and wait) y obliga al transmisor a enviar cada trama y esperar una confirmación de la llegada correcta de su trama al receptor.

Si la trama tiene una estructura como la descripta anteriormente el receptor puede enviar nueva información y "montada" en ella el número de secuencia del último "frame" recibido.

La otra categoría de protocolos tendientes a controlar el flujo de información se denomina de "ventana deslizante" (sliding window); su utilidad aparece cuando hay retardos considerables entre envío de trama y retorno de confirmación, dado que en tales situaciones dejar al transmisor desocupado disminuye la utilización del ancho de banda.

En este caso el retardo se "ocupa" con transmisiones de tramas que quedan en una lista de "frames" pendientes de confirmar, similarmente el receptor mantiene una memoria de frames que puede aceptar.

Las características del enlace de radiofrecuencia (canal "half duplex" y tasas de errores elevadas), adicionadas a aquéllas del acceso múltiple, han llevado a algunos autores a analizar las ventajas y desventajas de usar, en este ambiente, técnicas inicialmente propuestas para cable y transmisión full-dúplex con bajas tasas de error y secuenciación asegurada por el cable.[Ref.7,8,9,10,11].

En el presente artículo se obtienen expresiones para evaluar la eficiencia de un enlace punto a punto de radiofrecuencia, con tasas de error aleatorio variables a voluntad, considerando el efecto del sincronismo necesario para establecer la comunicación.

Se muestran las expresiones que maximizan la utilización del ancho de banda, en función del número de tramas enviadas por transmisión simple y la longitud de datos útiles de cada trama.

El conjunto de gráficos que acompañan al artículo ayudan a visualizar los resultados obtenidos.

3) Anàlisis preliminar:

Cuando se analiza un protocolo de comunicaciones deben tenerse en cuenta tres factores de pèrdida que alteran la performance, medida en tèrminos de su eficiencia:

1- Pèrdida debida a los errores en la transmisiòn de la trama. (Factor A).

2- Pèrdida debida a los errores en la transmisiòn del ACk. (Factor C).

3- Pèrdida debida al overhead que introducen los siguientes tres elementos: Header del paquete + Sincronismo + ACK. (Factor B).

De este modo la eficiencia puede ser considerada asì:

$$Uc = F(A,B,C) = A.B.C$$

El factor C de la fòrmula anterior varìa de acuerdo a la tècnica de envìo del ACK, verificàndose lo siguiente:

* C = 1 si la probabilidad de pèrdida del ACK vale 0 y

* C = A si la probabilidad de pèrdida del ACK = probabilidad de pèrdida del frame de datos.

Esta ùltima opciòn resulta lògica si se analiza precisamente un canal de acceso ùnico de radiofrecuencia half duplex. Como tanto los "frames" de datos como el ACK transitan el mismo medio resulta sensato igualar las probabilidades de pèrdida de ambos tipos de tramas, suponiendo adicionalmente un esquema del tipo "piggyback".

Sin embargo, por el momento (y dada la ductilidad de este factor), se considerarà C = 1 para simplificar el anàlisis.

4) Anàlisis suponiendo probabilidad de pèrdida de Ack nula:

La suposiciòn (ideal) de que la probabilidad de pèrdida del ACK es cero lleva a

$$Us = A.B$$

Debe notarse que la suposiciòn anterior significa sòlo que el ACK no es afectado por el ruido, aunque sin embargo se sigue considerando el efecto de su presencia en el factor B. El no considerarlo allì supondrìa una situaciòn màs irreal: que el ACK no "consume" bits.

Si se analiza el factor B de la fòrmula anterior y su sensibilidad a variaciones de N (nùmero de tramas por transmisiòn ùnica) los dos casos de interès son : N = 1 y N distinto de 1 .

Para este anàlisis se definen los siguientes tèrminos:

D: Nùmero de bits de datos en la trama.

H: Nùmero de bits en el encabezamiento de la trama + bits para el CRC + bits para el ACK.

En este caso la relación de bits útiles a bits enviados (factor B) vale:

$$B = \frac{D}{D + H}$$

Este factor permanece constante para transmisiones simples o múltiples, ya que ninguno de estos términos depende de N.

Se considera ahora el retardo de conmutación del transceptor necesario para establecer la comunicación (impuesto por limitaciones técnicas insalvables): este retardo es constante tanto para transmisión simple como para múltiple y si se llama sincronismo (S) a este tiempo de conmutación y se analiza su efecto de overhead por paquete se obtiene:

En transmisión simple (N = 1):

$$Ss = S/1$$

En transmisión múltiple (N distinto de 1), donde en un único período de transmisión se envían N tramas sucesivas:

$$Sm = S/N$$

De manera que en la expresión de B agregando el sincronismo:

$$B = \frac{D}{D + H + S/N}$$

La función B depende ahora de N, y tiene un máximo para N = OO que vale D/D+H, y un mínimo para N = 1.

Con relación al caso de "transmisión múltiple" es necesario realizar ciertas aclaraciones: en ese caso suele usarse (con cierta libertad) la terminología ventana deslizante. Sin embargo, es necesario recordar que en un esquema de radio el método de control de flujo responde a una técnica "pare y espere", ya que el transceptor (independientemente del número de bloques que envíe), conmuta a recepción luego de haber realizado su transmisión, por lo que no hay ventana deslizante, tal como ocurre cuando se trata de transmisión full duplex.

Para analizar el término A, si se supone que los errores del canal ocurren independientemente a velocidad p, la probabilidad de que un frame de largo D+H llegue sin error en una transmisión simple vale:

$$Q = (1 - p)^{D + H}$$

Esto significa que cuanto mayor es la longitud del frame, más probable es que la información se altere por efecto del ruido.

Para estudiar correctamente el factor A en una tranmisión múltiple debe por lo tanto seleccionarse previamente una estrategia de transmisión.

Si en una transmisiòn N-mûltiple (con N > 2) el ruido afecta al paquete 2 (y este hecho es detectado por el receptor), los sucesivos paquetes de la transmisiòn serân descartados independientemente del efecto del ruido sobre ellos.**

Si Q es la probabilidad de èxito, (1-Q) es la probabilidad de fracaso considerando todos los posibles casos en una transmisiòn y la estratègia mencionada se llega a que el factor A vale [Ref.10]:

$$Q.\ (1-Q) + Q.Q.\ (1-Q) + Q.Q.Q.\ (1-Q)... \quad Q^{N}.N = \quad Q.(1-Q) + Q^{N}.N$$

Resolviendo se llega a:

$$A = \frac{Q.\ (1-Q^{N})}{N\ (1-Q)}$$

Si ahora se considera el producto de los factores A y B, se tiene:

$$Us = \frac{Q.(1-Q^{N})}{N.(1-Q)} \cdot \frac{D}{D+H+S/N}$$

Donde el anàlisis matemàtico de la funciòn Us(N) revela :

- Tiene un cero en cero.
- Tiene un cero en infinito.
- Tiene un màximo para un N intermedio (N > = 1 y entero).

Para la bûsqueda de dicho valor de N es necesario considerar que, suponiendo formato HDLC y un equipo dado (S determinado) la optimizaciòn de Us debe buscarse en dos dimensiones, N y D, y no sòlo en funciòn de D. Por esta razòn, en este artículo se ha optado por buscar Us (màximo) en funciòn de ambas variables simultàneamente.

Considerando ruido completamente aleatorio:

Entonces p = BER y q = (1 - BER) y Q = q

Por lo tanto la expresiòn de Us, en tèrminos de D y N, vale:

$$U(D,N) = \frac{q^{D+H} \cdot (1-q)^{N(D+H)}}{1-q^{D+H}} \cdot \frac{D}{D+H+S/N}$$

La expresiòn anterior ha sido validada por simulaciòn en GPSS V e implementada a travès de un programa en Pascal, que calcula la funciòn Us, para determinados valores de error en el canal, tamaño de la cabeza y del sincronismo.

La funciòn resultante ha sido graficada y se muestra en la Figura 1 para valores tìpicos.

Continuando el estudio analítico, si se realiza en la fórmula anterior un apropiado cambio de variables se tiene

$$T = (D+H) \cdot N$$

$$Us(T,N) = \frac{q^{T/N} \cdot (1 - q^{T})}{(1 - q^{T/N})} \cdot \frac{T/N - H}{T + S}$$

Para encontrar el máximo se evalúan las derivadas parciales, se igualan a cero y utilizando la aproximación ln q = -p se llega a:

$$1) \qquad p\,(To + S) = e^{pTo} - 1$$

$$2) \qquad a\,No + b = \{ c\,No^{2} + d\,No + e \}\,(1 - q^{f/No})$$

Con

$$a = H\,To$$

$$b = p\,(To + s)\,To^{2}$$

$$c = (To + S)\,H\,R$$

$$d = To\,\{ H + (To + S)\,(1 + R) \}$$

$$e = To^{2}$$

$$f = To$$

$$R = p\,To\,\frac{q^{To}}{1 - q^{To}}$$

De 1) se obtiene el valor de T óptimo (To) y por lo tanto de D óptimo (Do).

De 2) se obtiene el valor de N óptimo (No)

En las Figuras 2 y 3 pueden observarse Do y No en función de p con parámetros S y H.

El parámetro a analizar en este caso es la relación entre la eficiencia máxima obtenida para N óptimo y la eficiencia máxima para N = 1. Esta relación se observa en la Figura 4.

5) Probabilidad de pérdida de Ack igual a la de la trama de datos:

El caso C = A supone respetar el formato HDLC en el sentido de que cada trama lleva univocamente reconocida un único frame pendiente, aún en el caso de utilizar tres bits para el número de secuencia de unidades confirmadas.

$$U = A^{2} . B$$

Con un análisis similar al anterior se obtienen las Figuras 5, 6 y 7.

** Este método cuyo nombre habitual es "go back-N- recovery" no se distingue por su eficiencia pero si por su característica de no imponer requerimientos de memoria, por su fácil implementación y su facilidad intrínseca de entregar los paquetes en secuencia.

6) Conclusiones:

De lo expuesto analítica y gráficamente se concluye que con tasas de error elevadas, se obtiene una mayor eficiencia enviando más de una trama por transmisión simple, dado que el retardo por sincronismo se divide por N.

Asimismo la inmunidad al ruido aumenta al disminuir el número de bits de cada trama.

7) Bibliografía:

1- Tanenbaum Andrew S.- Computer Networks - Prentice Hall - 1981.

2- Karn Phil - Link Level Protocol Revisited - 1986.

3- Field J.A.- Efficient Computer Computer Communication - Proc. IEE, Vol. 123.- Agosto 1976.

4- Pouzin L., Zimmerman H. : "A Tutorial on Protocols", Proc. IEEE, Vol. 66, pp 1346-1370, Nov. 1978.

5- Chien R.T., Clayton V.E., Boudreau P.E. and Locke R.R. : " Error Correction in a radio based data communications system". IEEE Trans. on Comm. Abril 1975.

6 - Davies Watts Donald: " The Control of congestion in packet switching networks".

7 - Tobagi F. A. and Kleinrock L.: "The effect of Ack traffic on the capacity of packet- switched radio channels". IEEE Trans.on Comm. Vol Com. 26. N* 6 Junio 1978.

8 - Kleinrock L. and Tobagi F. : " Packet switching in radio channels: Part I Carrier Sense multiple access modes and their throughput delay characteristics". IEEE Trans. on comm. Vol. Com 23. Dec. 1975.

9 - Tobagi F and Kleinrock L.: "Packet switching in radio channels: Part II- The hidden terminal problem in Csma and the busy tone solution ". IEEE trans. on comm. Vol.Com 23. Dic. 1975.

10 -Kahn R.E., Gronemeyer S.A., Burchfield J. and Kunzelman R.C.: "Advances in packet radio technology". Proc. of the IEEE. Special issue in packet communication networks. Nov. 1978.

11- Field J. A. : "Efficient Computer-computer communication". Proc. IEE Vol 123 Nro 8, Agosto de 1976.

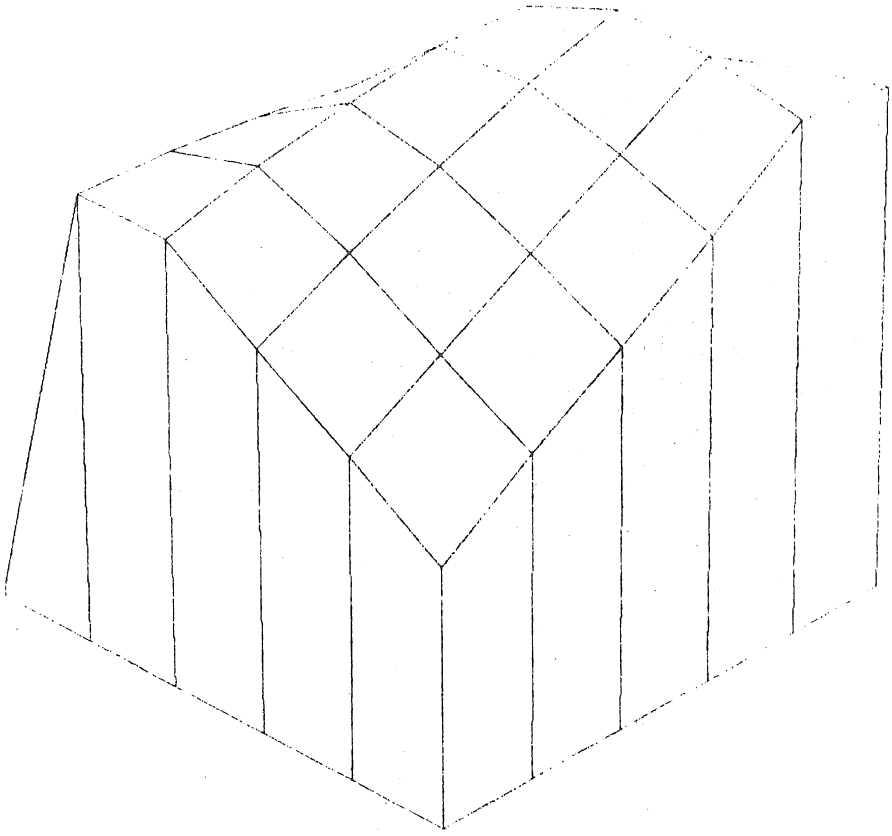


Figura 1 U(N,D) H=48 S=600 Lg(p)=-3

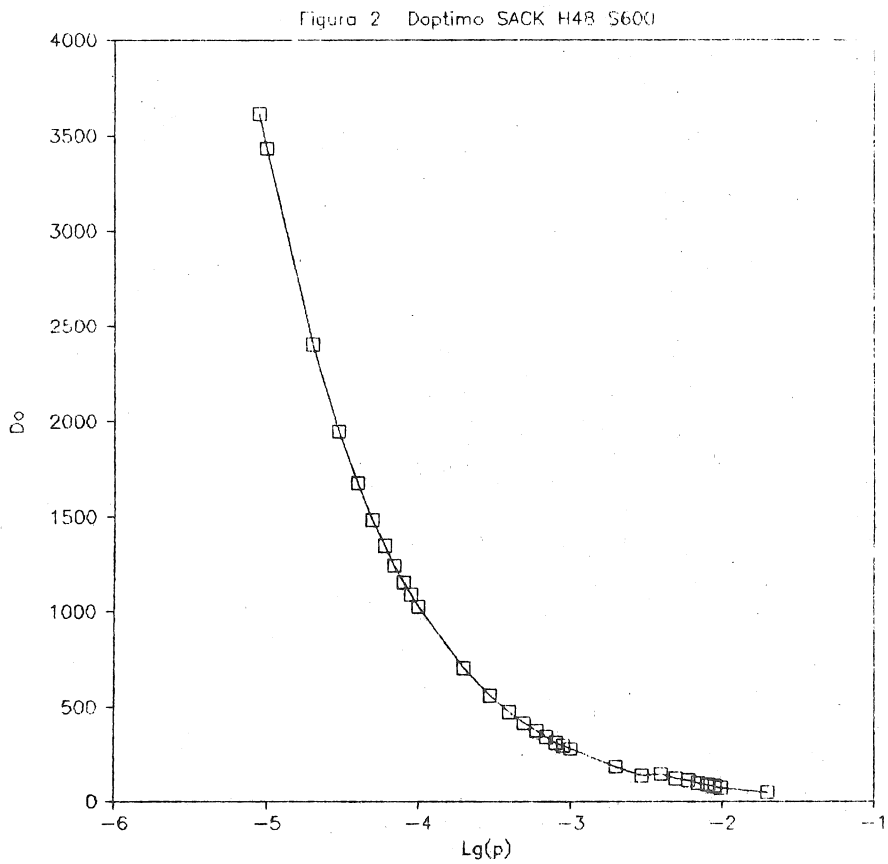


Figura 3 Noptimo SACK H48 S600

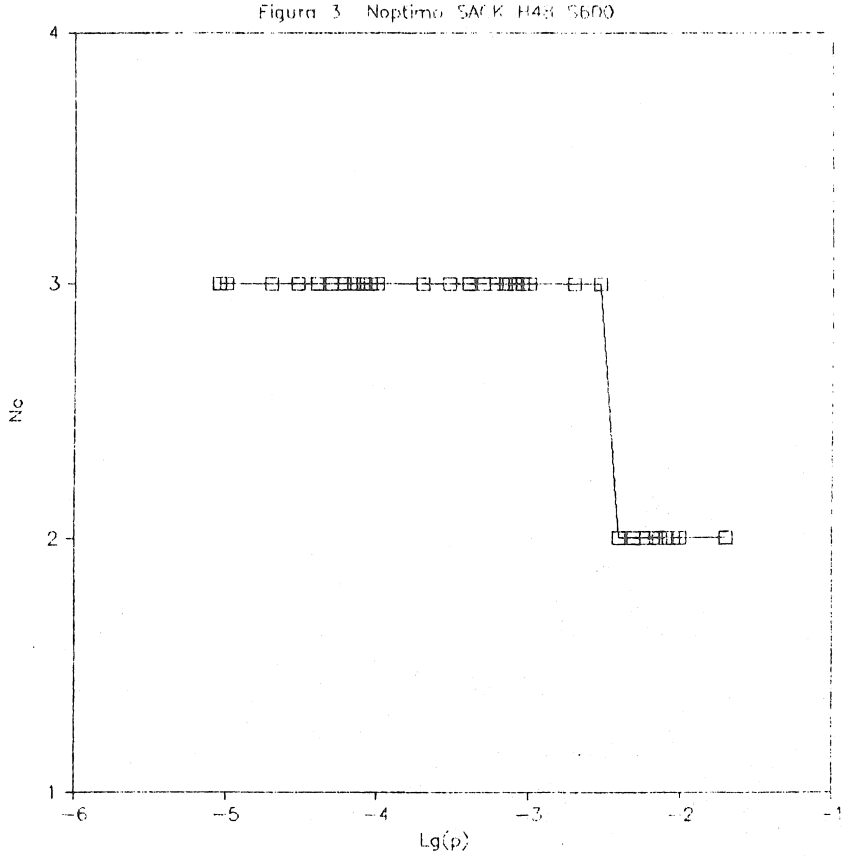


Figura 4 U/U1 SACK H48 S600

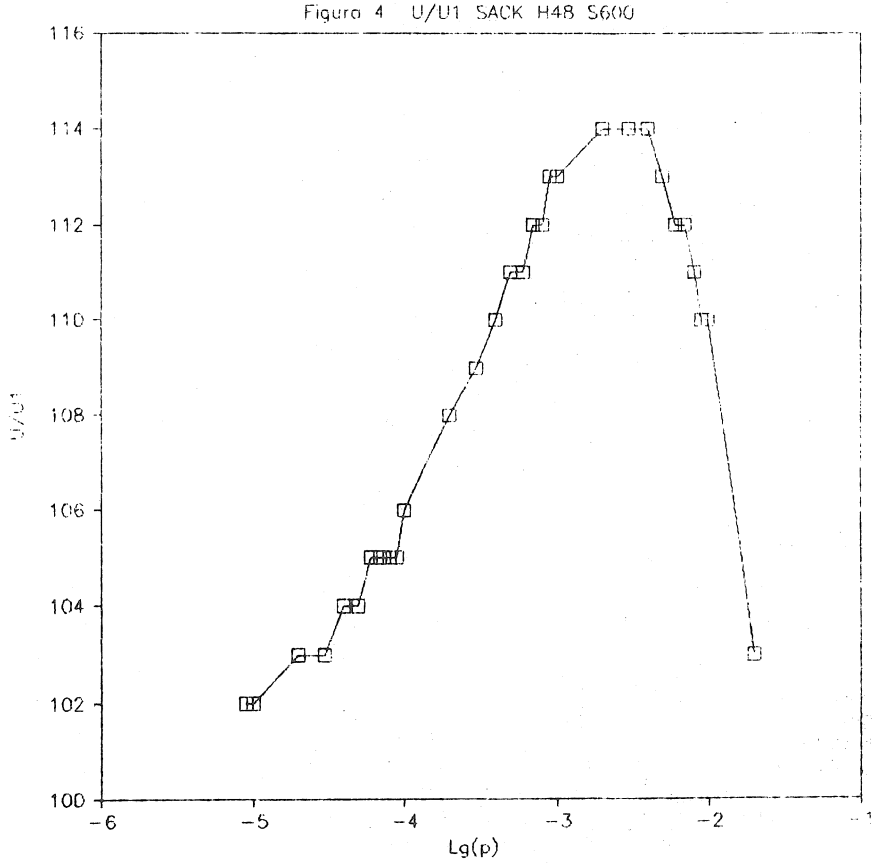


Figura 5 Doptimo CACK H48 S600

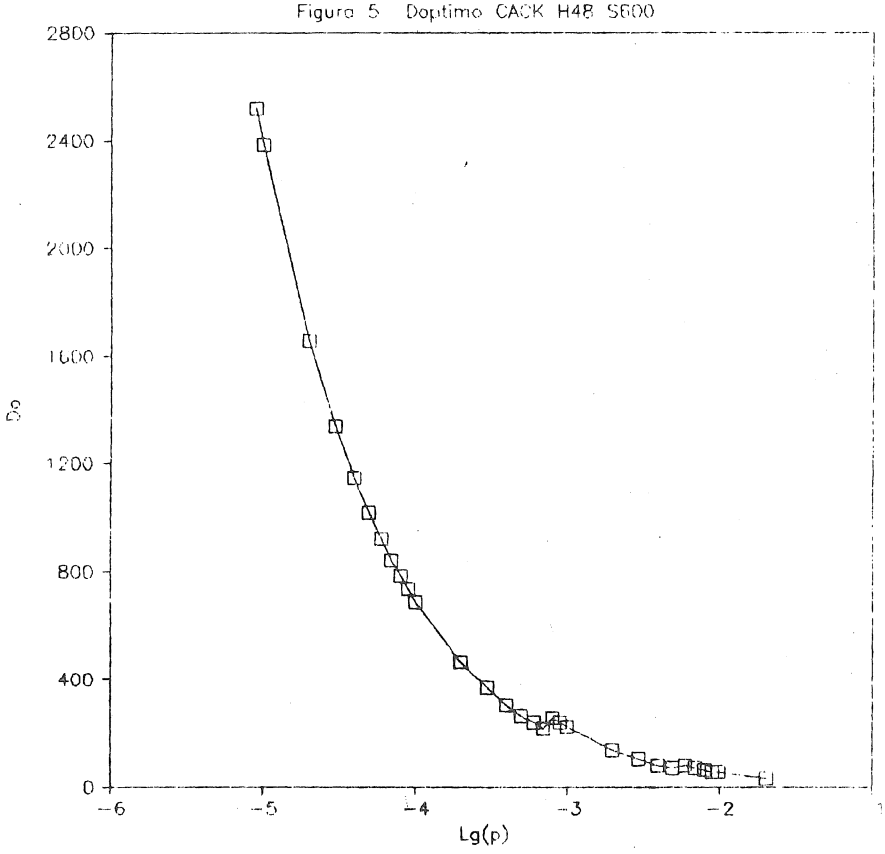


Figura 6 Noptimo CACK H48 S600

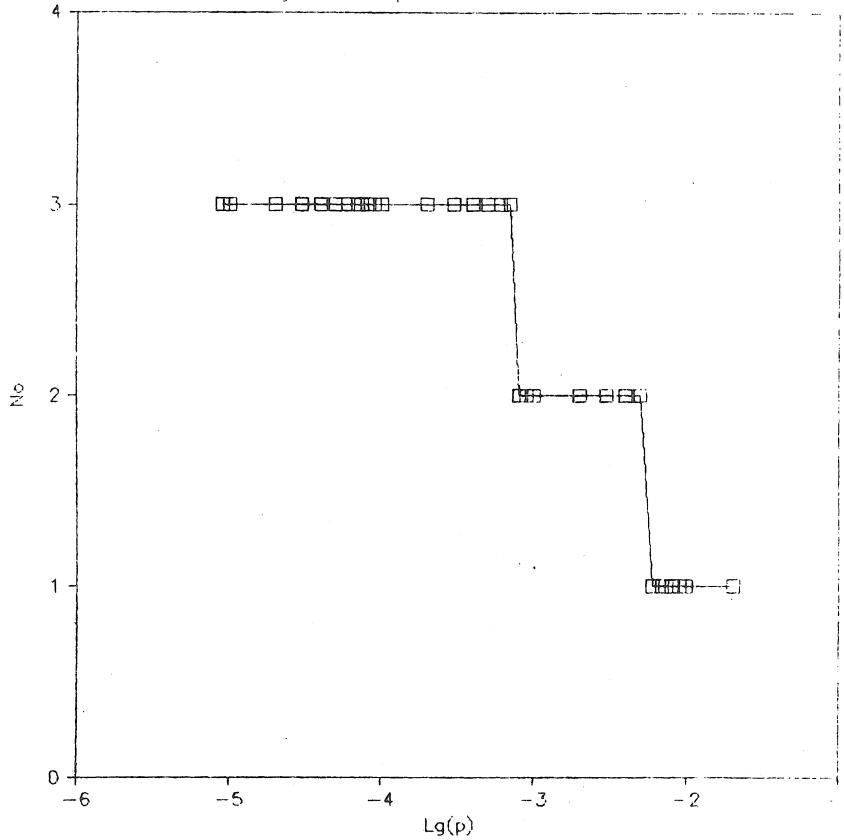
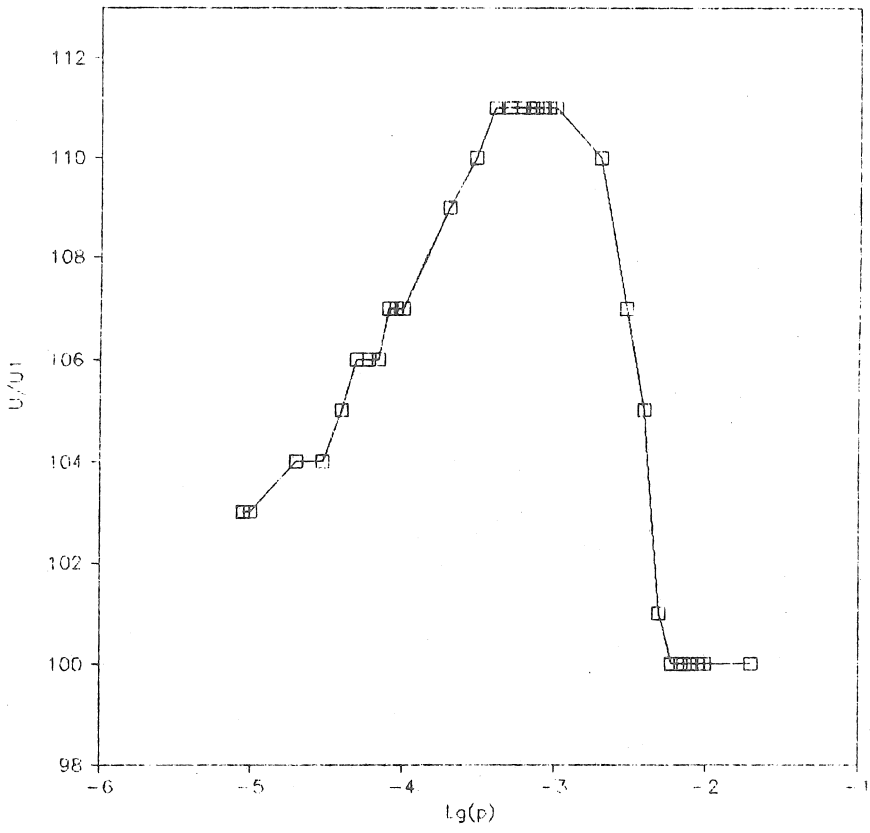


Figura 7 U/U1 CACK H48 5600



TÍTULO: "ESPECIFICAÇÃO DA CAMADA DE SESSÃO PARA A REDE-RIO"

RESUMO: Este artigo descreve os serviços da camada de sessão derivados da proposta de padronização apresentada pela ISO. é apresentado como o provedor de serviço de sessão e utiliza os serviços de transporte, o modelo funcional, as tabelas de estados e seus procedimentos e, finalmente, as estruturas de dados utilizadas.

PALAVRA-CHAVE:

Redes;
Interconexão de Sistemas Abertos;
Sessão;
Arquitetura OSI.

AUTORA:

LUCI PIRMEZ
Mestrado em Ciência da Computação
Coordenadora do Grupo de Projetos de Rede da Área de Desenvolvimento e Analista Consultor do Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro.

Núcleo de Computação Eletrônica
Universidade Federal do Rio de Janeiro
Caixa Postal 2324 - CEP 20.001
Telefone: 290 - 3212, ramal 294
Rio de Janeiro, RJ

"UMA ESPECIFICAÇÃO DA CAMADA DE SESSÃO PARA À REDE-RIO"

I - INTRODUÇÃO

As normas ISO/DIS 8326 e ISO/DIS 8327 são responsáveis, respectivamente, pela definição dos serviços de sessão e especificação do protocolo de sessão.

A especificação da camada de sessão e futuramente sua implementação faz parte de um projeto denominado REDE-RIO. Este projeto tem como objetivo principal, possibilitar a interconexão dos computadores de grande porte das universidades do RIO DE JANEIRO.

A implementação dos serviços a serem oferecidos pela REDE-RIO seguem a tendência internacional de basear os desenvolvimentos de software/hardware segundo o modelo OSI/ISO. As sete camadas especificadas por este modelo são: físico, enlace, redes, transporte, sessão, apresentação e aplicação.

As três camadas inferiores já estão definidas pelo CCITT e constitui o protocolo X.25. O padrão X.25 é oferecido pela RENPAC (Rede Nacional de Pacotes) é o que será utilizado como meio de interconexão entre os vários centros participantes da REDE-RIO. O hardware e software necessários para permitir a interconexão serão adquiridos diretamente dos fabricantes.

O projeto REDE-RIO tem como tarefas o estudo, a especificação e a implementação das seguintes camadas:

- a camada de transporte;
- a camada de sessão;
- a camada de aplicação:

  - o serviço de Terminal Virtual;
  - o serviço de Correio Eletrônico;
  - o serviço de Manipulação e Transferência de Job;
  - o serviço de Manipulação, Acesso e Transferência

de Arquivo.

Na Universidade Federal do Rio de Janeiro, o computador que se interligará a RENPAC é o VAX 780.

O objetivo desse artigo é apresentar uma proposta de especificação da camada de sessão para o projeto Rede-Rio. Ele está organizado da seguinte forma:

(1) Uma visão geral dos serviços de sessão;
(2) Utilização dos serviços de transporte;
(3) O funcionamento da sessão e suas interfaces;
(4) Tabela de estados de uma conexão de sessão;
(5) Estrutura de dados utilizada.

II - UMA VISÃO GERAL DOS SERVIÇOS DE SESSÃO

A camada de sessão é o meio do usuário ter acesso a rede.É com essa camada que o usuário deve negociar para estabelecer uma conexão com um processo em outra máquina. Essa conexão é normalmente chamada de sessão.

O objetivo da camada de sessão é organizar e sincronizar o diálogo e gerenciar as trocas de dados entre usuários cooperantes do serviço de sessão.

O serviço de sessão é prestado pelo protocolo de sessão fazendo uso dos serviços disponíveis na camada de transporte.

O serviço de sessão oferece as seguintes funções:

(1) Estabelecer uma conexão com o outro usuário do serviço de sessão, trocar os dados com este usuário de uma forma sincronizada e liberar a conexão de uma forma ordenada;

(2) Negociar o uso de fichas para troca de dados, sincronização e liberação da conexão, e combinar se a troca de dados será half-duplex ou duplex;

(3) Estabelecer pontos de sincronziação dentro do diálogo e nos eventos de erro restabelecer o diálogo em um ponto de sincronismo combinado;

(4) Interromper o diálogo e retornar mais tarde num ponto já pré-estabelecido.

Uma descrição mais detalhada dos serviços da camada de sessão é encontrada no relatório técnico NCE00787.

Fazendo-se uma análise dos serviços de sessão, podemos afirmar que, para o projéto Rede-Rio, a escolha de utilizar a classe básica, é justificada pelo fato de já ser suficiente os mecanismos de erros existentes nas camadas inferiores. Utilizar o mecanismo de sincronização e/ou mecanismo de gerenciamento de atividade, seria afetar o desempenho do sistema. Isto é notório pois, excesso de proteção acaba degradando o funcionamento do sistema.

III - A UTILIZAÇÃO DOS SERVIÇOS DE TRANSPORTE

Este ítem define como as primitivas do serviço de transporte (tabela 1) são utilizados pelo provedor de serviço.

(1) Atribuição da conexão de sessão a uma conexão de transporte

a) Descrição

Uma conexão de sessão é atribuída ou a uma conexão de transporte já existente e que permite sua reutilização, ou uma nova conexão de transporte é criada para este propósito.

Quando uma conexão de sessão é terminada, a conexão de transporte em questão também é finalizada, a menos, é claro, que esta conexão de transporte permita a sua reutilização.

Vale ressaltar que somente o inicializador de uma conexão de transporte é que pode emitir um pedido de conexão de sessão.

b) Primitivas do serviço de transporte

São utilizadas pelo provedor de serviço as seguintes primitivas:

T-CONNECT	request
	indication
	response
	confirmation
T-DISCONNECT	request
	indication

(2) Reutilização da conexão de transporte

a) Descrição

Quando uma conexão de sessão é recusada, ou tem sido sucessivamente conectada e subsequentemente desconectada, por aborto ou liberação ordenada, a conexão de transporte em questão pode ser ou desconectada ou reutilizada.

A conexão de transporte pode ser reutilizada desde que:

(a) Não seja disponível o serviço de transferência de dados urgentes;

(b) O provedor de serviço que estabeleceu a conexão de transporte permite sua reutilização através do "ABOUT SPDU" ou do "FINISH SPDU";

(c) O provedor de serviço que estabeleceu a conexão de transporte receba um "REFUSE SPDU" que possibilite a reutilização da conexão.

Somente o inicializador da conexão de transporte pode reutilizar esta mesma conexão enviando um "CONNECT SPDU" para estabelecer uma nova conexão de sessão.

b) Primitivas do serviço de transporte

São utilizadas pelo provedor de serviço primitivas do serviço de transporte.

T-DATA	{	request
		indication

(3) Uso dos serviços de transferência de dados normais da camada de transporte

a) Descrição:

Todas as SPDUs são inseridas no campos de dados do usuário da primitiva de serviço de transferência de dados normais.

b) Primitivas do serviço de transporte

São utilizadas pelo provedor de serviço primitivas do serviço de transporte:

T-DATA { request / indication }

c) Tamanho máximo da TSDU

Quando o tamanho máximo da TSDU for selecionado, o tamanho da SPDU não pode exceder o tamanho máximo da TSDU selecionada para esta direção.

(4) Disconexão de transporte

a) Descrição

Depois que a conexão de sessão for liberada ou abortada e a conexão de transporte não permite sua reutilização, esta conexão de transporte é disconectada.

Quando um "T-DISCONNECT indication" for recebido, como resultado de uma deteção de erro pelo provedor do serviço, o provedor de serviço emite um "S-P-ABORT indication" para o usuário local.

Quando é transmitido um "T-DISCONNECT, o provedor de serviço pode opcionalmente usar o campo de dados do usuário do "T-DISCONNECT" para indicar a razão da disconexão da conexão de transporte para o provedor de serviço remoto. O código da razão da disconexão consiste de um octeto com os seguintes valores:

(0) Erro do protocolo de sessão que impede a transmissão de um "ABORT SPDU";

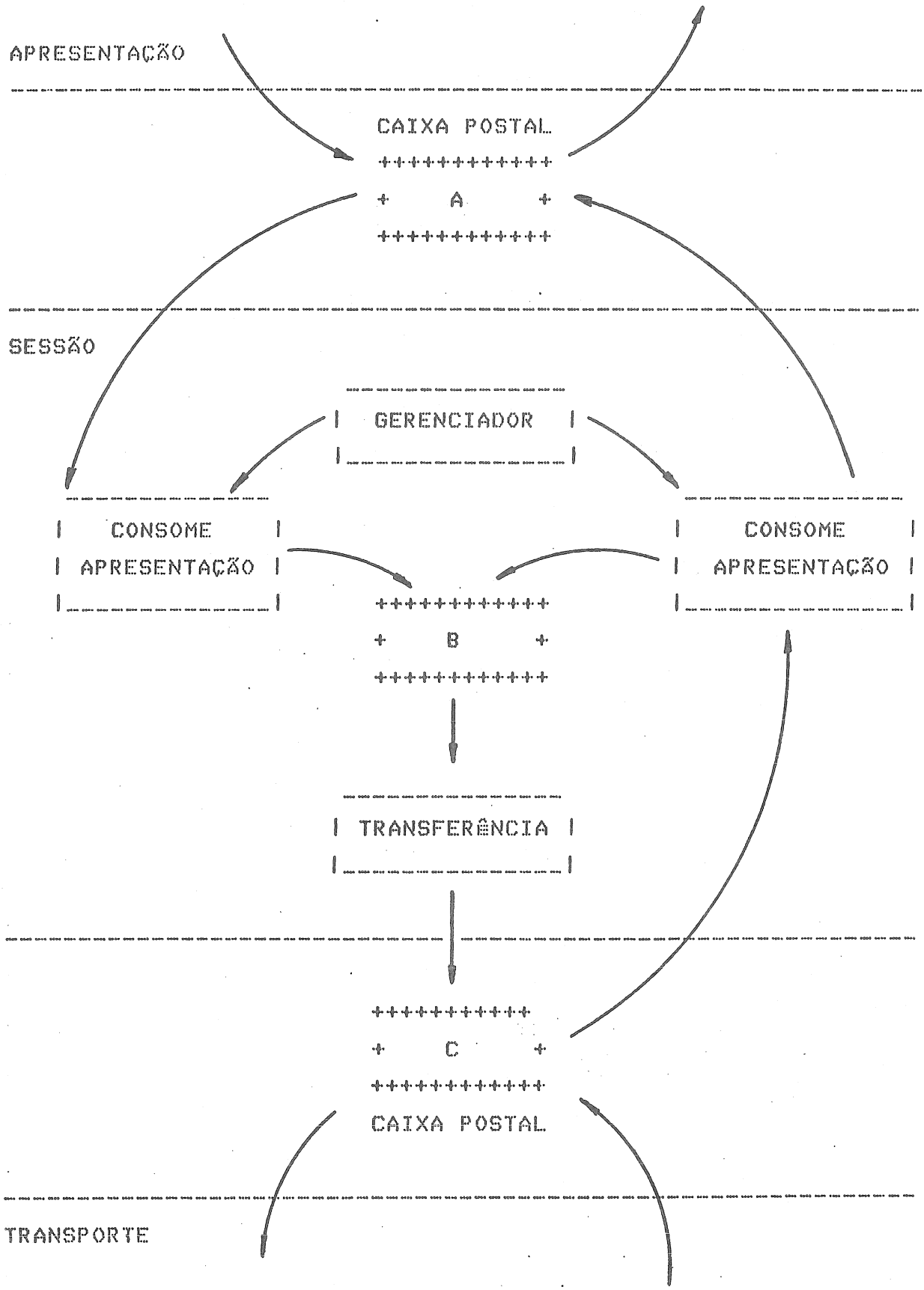
(1) Disconexão normal de uma conexão de transporte quando esta não permite a sua reutilização;

(2) Disconexão normal de uma conexão de transporte quando esta permite a sua reutilização, mas sua reutilização não é possível por razões locais.

PRIMITIVA		PARÂMETRO
T-CONNECT	request	endereço chamado, endereço chamador, opção de dados urgentes, qualidade, dado do usuário.
	indication	
T-CONNECT	response	qualidade, endereço chamador, opção de dados urgentes, dado do usuário.
	confirm.	
T-DATA	request	dado do usuário
	indication	
T-DISCONNECT	request	dado do usuário
T-DISCONNECT	indication	razão

TABELA 1 - PRIMITIVAS DA CAMADA DE TRANSPORTE

IV - FUNCIONAMENTO DA CAMADA DE SESSÃO



MODELO FUNCIONAL

O sistema que implementa a camada de sessão é controlada por um gerenciador que é responsável pela sincronização de todas as tarefas necessárias ao seu funcionamento. Dessa forma, o gerenciador é o responsável pela execução de todas as operações deste sistema.

O procedimento básico do funcionamento do gerenciador, do ponto de vista interno, consiste em um ciclo de interrogações. Este ciclo analisa a comunicação com a camada de apresentação, analisa a comunicação com a camada de transporte, e ativa, quando necessário, os procedimentos de transferência de mensagens para a camada de transporte ou de apresentação. Cada análise realizada se refere a uma possível recepção de uma mensagem proveniente da camada de apresentação ou de transporte.

Para o tratamento de entrada e saída de mensagens neste sistema, adotou-se o emprego do dispositivo virtual, a caixa postal (Mailbox). Dessa forma, o sistema apenas consulta uma determinada caixa postal quando deseja verificar a existência de mensagens.

Este sistema pode ser dividido em quatro módulos, (processos) que são:

a) O gerenciador

Módulo responsável em controlar a execução das operações de todo o sistema. O procedimento básico consiste de:

- Inicializar as áreas de dados, as tabelas, as caixas postais, etc.;

- Criar subprocessos;

- Ativar subprocessos;

- Detetar a ocorrência do "TIMER" e tomar as devidas providências chamando as rotinas encarregadas do tratamento do evento.

b) Módulo Consome$Apresentação

Módulo responsável em detetar a recepção de mensagens (primitivas) da camada de apresentação, processar a mensagem recebida e pedir ao módulo TRANSFERÊNCIA para transmitir uma determinada unidade de dados. Este módulo será implementado como sendo um único processo que ficará bloqueado a espera de mensagem do Mailbox A da figura do modelo funcional.

c) Módulo Transferência

Este módulo é responsável no processamento de mensagens (primitivas) a serem transmitidas para a camada de transporte. Nesta mensagem, serão inseridas as unidades de dados. Este módulo (processo) ficará bloqueado a espera de mensagens do Mailbox B da figura do modelo funcional.

d) Módulo Consome$Apresentação

Este módulo é responsável em detetar a recepção de mensagens (primitivas) da camada de transporte, processar a mensagem recebida e/ou pedir ao módulo TRANSFERÊNCIA para transmitir uma determinada unidade de dados ou encapsular a unidade de dados na mensagem (primitiva) que será armazenada no Mailbox A da figura do modelo funcional.

V - TABELA DE ESTADOS DE UMA CONEXÃO DE SESSÃO

Neste tópico é descrito o protocolo de sessão em termos de tabela de estados. A tabela de estados mostra o estado

da conexão de sessão, os eventos que ocorrem no protocolo, as ações a serem tomadas e o estado resultante.

Notação para a tabela de estados:

--> ESTADOS:

NÚMERO	NOME	DESCRIÇÃO
1	INOTRANSP	Repouso, não existe conexão de transporte
2	W$AA	Espera por ABORT ACCEPT SPDU
3	W$TCONCNF	Espera por T-CONNECT CONFIRMATION
4	I$TRANSP	Repouso, existe conexão de transporte
5	W$AC	Espera por ACCEPT SPDU
6	W$DN	Espera por DISCONNECT SPDU
7	W$SCONRSP	Espera por S-CONNECT RESPONSE
8	W$SRELRSP	Espera por S-RELEASE RESPONSE
9	W$TDISIND	Espera por S-RELEASE RESPONSE
10	TRANSFER	Transferência de dados

--> EVENTOS:

NOME	DESCRIÇÃO
SCONREQ	Primitiva S-CONNECT REQUEST
SCONIND	Primitiva S-CONNECT INDICATION
SCONRSP	Primitiva S-CONNECT RESPONSE
SCONCNF	Primitiva S-CONNECT CONFIRMATION
SDTREQ	Primitiva S-DATA REQUEST

SDTIND	Primitiva S-DATA INDICATION
SRELREQ	Primitiva S-RELEASE REQUEST
SRELIND	Primitiva S-RELEASE INDICATION
SRELRSP	Primitiva S-RELEASE RESPONSE
SRELCNF	Primitiva S-RELEASE CONFIRMATION
SUABREQ	Primitiva S-U-ABORT REQUEST
SUABIND	Primitiva S-U-ABORT INDICATION
TCONREQ	Primitiva T-CONNECT REQUEST
TCONIND	Primitiva T-CONNECT INDICATION
TCONRSP	Primitiva T-CONNECT RESPONSE
TCONCNF	Primitiva T-CONNECT CONFIRMATION
TDISREQ	Primitiva T-DISCONNECT REQUEST
TDISIND	Primitiva T-DISCONNECT INDICATION
TIM	Estouro do tempo
AA	SPDU de ABORT ACCEPT
AB-nf	SPDU de ABORT com utilização
AB-nr	SPDU de ABORT sem utilização
AC	SPDU de ACCEPT
CN	SPDU de CONNECT
DN	SPDU de DISCONNECT
DT	SPDU de DATA TRANSFER
FN	SPDU de FINISH
RF-r	SPDU de REFUSE com utilização
RF-nr	SPDU de REFUSE sem utilização

--> OUTROS:

VTCA	Indica se este usuário é quem inicializou a conexão de transporte
VCOLL	Informa se ocorreu colisão na conexão
VTRR	Informa se a conexão de transporte pode ser reutilizada
LOCAL	Informa se o procedimento é local

TABELA DE ESTADO DO ESTABELECIMENTO DE CONEXÃO

EVENTO	INO TRANSP	W$ AA	W$ TCONCNF	IDLE$ TRANSP	W$ AC	W$ SCON RSP	W$ TDIS IND
AC		W$ AC		(7) INOT	(9) TRANSFER		W$TDIS IND
CN		(7) W$AA		(8)			(12) INOT
RF- nr		W$ AA		(7) INOT	(10) INOT		W$TDIS IND
RF- r		W$ AA		(7) INOT	(11)		W$TDIS IND
SCON REQ	(1) W$ TCONCNF			(4)W$ SCONREQ			
SCON RSP+						(5) TRANSFER	
SCON RSP						(6)	
TCON CNF			(2) W$AA				
TCON IND	(3)I$ TRANSP						

PROCEDIMENTOS:

(1) -

Envia TCONREQ
VTCA = true

(2) -

Envia SPDU de CN

(3) -

Envia TCONRSP
VTCA = false

(4) -

Se VTCA = true
    Envia SPDU de CN

(5) -

Envia SPDU de AC
VCOLL = false

(6) -

Se LOCAL = true
    Envia SPDU de RF
    ESTADO = I$TRANSP
Se LOCAL = false
    Envia SPDU de RF-nr
    Inicializa o timer
    ESTADO = TDISIND

(7) -

Envia TDISREQ

(8) -

Se VTCA = true
    Envia SCONIND
    ESTADO = SCONRSP
Caso contrário
    Envia TDISREQ
    ESTADO = I$TRANSP

(9) -

Envia SCONCNF
VCOLL = FALSE

(10) -

Envia SCONCNF
Envia TDISREQ

(11) -

Se LOCAL = true
    Envia SCONCNF (-)
    Envia TDISREQ
    ESTADO = I$TRANSP
Caso contrário
    Envia SCONCNF (-)
    Envia TDISREQ
    ESTADO = INOTRANSP

(12) -

Envia TDISREQ
Interrompa o timer

TABELA DE ESTADO DE TRANSFERÊNCIA DE DADOS

EVENTO	W$AA	I$TRANSP	W$DN	W$SREL RSP	W$TDIS IND	TRANSFER
DT	W$AA	(1) I$NO $TRANSP	(2) W$DN		W$TDIS IND	(4) TRANSFER
SDTREQ				(3) W$SREL RSP		(3) TRANSFER

PROCEDIMENTOS:

(1) -
Envia TDISREQ

(2) -
Se VCOLL = false
Envia SDTIND

(3) -
Se VCOLL = false
Envia SPDU de DT

(4) -
Se fim de SSDU (segmentação)

TABELA DE ESTADO DE LIBERAÇÃO DE CONEXÃO

EVENTO	W$AA	I$TRANSP	W$DN	W$SREL RSP	W$TDIS IND	TRANSFER
DN	W$AA	(1) I$NO TRANSP	(2)	(5)	W$TDIS IND	
FN nr	W$AA	(1) I$NO TRANSP	(3)		W$TDIS IND	(8) W$SRELRSP
FN r	W$AA	(1) I$NO TRANSP	(4)		W$TDIS IND	(9) W$SRELRSP
SRELREQ				(6)		(10)
SRELRSP				(7)		

PROCEDIMENTO:

(1) -

Envia TDISREQ

(2) -

Se REUSE = true
    Envia SRELCNF (+)
    ESTADO = I$TRANSP
Caso contrário
    Envia SRELCNF (+)
    Envia DISREQ
    ESTADO = INOTRANSP

(3) -

Envia SRELIND
VTRR = false
VCOLL = true
ESTADO = W$SRELRSP

(4) -

Se VTCA = true
    Envia SRELIND
    VTRR = false
    VCOLL = true
    ESTADO = SRELRSP

(5) -

Se VCOLL = true
    Envia SRELCNF
    ESTADO = W$SRELRSP

(6) -

Envia SPDU FN-nr
VTRR = false
VCOLL = true
ESTADO = W$SRELRSP

(7) -

Se REUSE = false
    Envia SPDU de DN
    Iniciar o relógio
    ESTADO = W$TDISIND
Caso contrário
Se VCOLL = false
    Enviar SPDU de DN
    ESTADO = W$TRANSP
Se não
    Enviar SPDU de DN
    VTRR = false
    ESTADO = W$SN

(8) -

Envia SRELIND
VTRR = false

(9) -

Se VTCA = true
    Envia SRELIND
    Se LOCAL = true
    VTRR = false
Se não
    VTRR <----- Valor do
campo PV do SPDU de DN

(10) -

Se LOCAL = false
    Envia SPDU de FN-nr
    VTRR = false
    ESTADO = W$DN
Se não
    Envia SPDU de Fn-r
    VTRR = true
    ESTADO = W$DN

TABELA DE ESTADOS DE ABORTO

EVENTO	INOTRANSP	W$AA	W$TCONCNF	I$TRANSP	W$AC
AA		(2) I$TRANSP		(3) I$NO TRANSP	
AB-nr	(1) I$NO TRANSP	(1) I$NO TRANSP		(3) I$NO TRANSP	(7) I$NO TRANSP
AB-r		(2) I$ TRANSP		(5)	(6)
SUABREQ			(3) I$NO		(8)
TDISIND		(2) I$NO TRANSP	(4) I$NO TRANSP	I$NO TRANSP	(4) I$NO TRANSP
TIMER		(3) I$NO TRANSP			

TABELA DE ESTADO DE ABORTO (Continuação)

EVENTO	W$DN	W$SCONRSP	W$SRELRSP	W$TDISIND	TRANSFER
AA				(1) I$NO TRANSP	
AB-nr	(7) I$NO TRANSP	(7) I$NO TRANSP	(7) I$NO TRANSP	(1) I$NO TRANSP	(7) I$NO TRANSP
AB-r	(6)	(6)	(6)	(1) I$NO TRANSP	(6)
SUABREQ	(8)	(8)	(8)		(8)
TDISIND	(4) I$NO TRANSP	(4) I$NO TRANSP	(4) I$NO TRANSP	(2) I$NO TRANSP	(4) I$NO TRANSP
TIMER				(3) I$NO TRANSP	

PROCEDIMENTOS:

(1) -

Stop timer
Envia TDISREQ

(2) -

Stop timer

(3) -

Envia TDISREQ

(4) -

Envia SPABIND

(6) -

Se LOCAL = false
Envia SPSU de SUABIND
Envia TDISREQ
ESTADO = INOTRANSP
Se não
Envia SUABIND
Envia SPDU de AA
ESTADO = I$TRANSP

(7) -

Envia SUABIND
ENVIA TDISREQ

```
(5) -
     Se LOCAL = false
          Envia TDISREQ
          ESTADO = I$NO$TRANSP
     Se não
          Envia SPDU de AA
          ESTADO = I$TRANSP

(8) -
     Se LOCAL = false
          Envia SPDU de AB-nr
          Interrompa o timer
          ESTADO = W$DISIND
     Se não
          Envia SPDU de AB-r
          ESTADO = W$AA
```

VI - ESTRUTURA DE DADOS UTILIZADA

1 - Estrutura de Dados Utilizada Internamente

Do ponto de vista interno, a camada de sessão opera sobre a área de dados e tabela de informações, relativas a uma determinada conexão.

A cada conexão de sessão estão associadas duas áreas: uma área de dados, onde são armazenados as mensagens propriamente ditas, e uma tabela de informações relativas a esta conexão.

Uma área de dados é alocada/desalocada dinamicamente de uma região pré-definida de memória. As áreas disponíveis nesta região de memória estão indicadas através de uma lista encadeada cujos ponteiros são formadas por posições das próprias áreas livres. O início da lista será indicado pelo ponteiro disponível e o algoritmo de alocação usado é o first fit.

Devido à necessidade de frequentes operações de inclusão/exclusão ordenada na tabela de áreas livres, é aconselhável que ela seja estruturada na forma de lista a fim de diminuir o "overhead" de rearranjo. Ao liberar uma área da tabela de áreas livres é verificado se ela é contígua a uma (ou 2) área(s) livre(s).

A tabela de informação denominada de TAB$CONEXÃO consta de 12 campos inteiros por conexão de sessão. As diversas áreas de informações são alocadas, consecutivamente, em endereços crescentes de memória, e são constituídas pelos seguintes campos:

. USER$FONTE

Endereço do usuário do serviço de sessão (S.S.) chamador.

. USER$DEST

Endereço do usuário de serviço de sessão (S.S.) chamado.

. SSAP$FONTE

Identificador do ponto de acesso do S.S. (SSAP) chamador.

. SSAP$DEST

Identificador do ponto de acesso do S.S. (SSAP) chamado.

. TSAP$FONTE

Identificador do ponto de acesso do serviço de transporte (TSAP) chamador.

. TSAP$DEST

Identificador do ponto de acesso do serviço de transporte (TSAP) chamado.

. ESTADO

Indica o estado em que se encontra uma determinada conexão de sessão. Eles podem ser: I$TRANSP, I$NO$TRANSP, W$AA, WAC, WDN, W$TCONCNF, W$CONRSP, W$SRELRSP, W$TDISND, TRANSFER.

. VTCA

Indica se este usuário é quem inicializou a conexão de transporte.

. VCOLL

Informa se ocorreu colisão na conexão.

. VTRR

Informa se a conexão de transporte pode ser reutilizada.

. LOCAL

Informa se o procedimento é local.

. REUSE

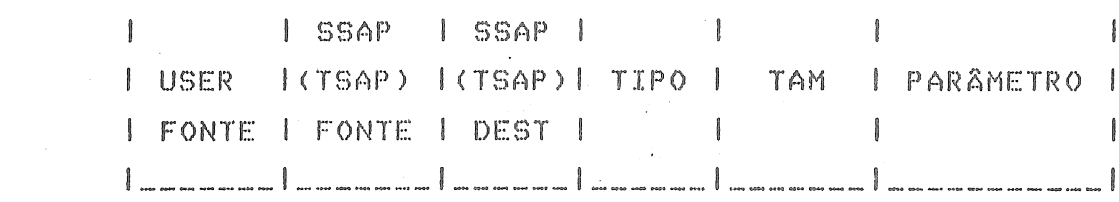
Informa se a conexão de transporte pode ser reutilizada.

Existem mais duas tabelas associadas a cada linha da tabela anterior: a TAB$TX (transmissão) e TAB$RX (recepação). Estas tabelas contém informações sobre a mensagem que está sendo tratada em um determinado instante de tempo, como o tamanho da mensagem e o endereço da mensagem que está sendo tratada.

2 - Interfaces:

As informações trocadas entre duas camadas de um mesmo sistema pode ser implementado através de dispositivos virtuais de I/O denominado caixa postal (mailbox).

A mensagem que contém as informações consumida ou produzida do/para o mailbox apresenta o seguinte formato:



onde:

USER - Endereço do usuário do serviço de sessão (S.S.) chamador;

SSAP$FONTE - Identificador do ponto de acesso do S.S. (SSAP) chamador;

TSAP$FONTE - Identificador do ponto de acesso do serviço de transporte (S.T.) chamador;

SSAP$DEST - Identificador do SSAP chamado;

TSAP$DEST - Identificador do ponto de acesso do serviço de transporte chamado;

TIPO - Identifica a primitiva da camada de apresentação ou transporte [7];

TAMANHO - Informa o tamanho em octetos ocupados pelos parâmetros;

PARÂMETRO - Este campo contém os parâmetros da primitiva de apresentação ou transporte [7].

Para esta camada, será necessário dois (2) mailboxes para a troca de informação com o nível superior, apresentação, e com o nível inferior, transporte. Esses mailboxes são denominados de:

(1) TXSA - é o mailbox responsável pela troca de informações entre a camada de sessão e a camada de apresentação;

(2) TXST - é o mailbox responsável pela troca de informações entre a camada de sessão e a camada de transporte.

CONCLUSÃO:

A especificação encontrada neste artigo será utilizada na implementação da camada de sessão do projeto Rede-Rio. Esta especificação embora haja algumas simplificações (NCE 00787) mostrou ser satisfatória.

Esta mesma camada deverá ser especificada seguindo uma metodologia formal para, além de adquirir conhecimento da linguagem ESTELLE, evitar interpretações errôneas.

BIBLIOGRAFIA:

1. ISO 7498
Information Processing Systems - Open Systems Interconection - Basic Reference Model;

2. ISO 8326
Information Processing Systems - Open Systems Interconection - Basic Connection Oriented Session Service Definition;

3. ISO 8327
Information Processing Systems - Open Systems Interconnection - Session Protocol;

4. GIOZZA, E. et all
"Redes Locais de Computadores
Protocolos de Alto Nível e Avaliação de Desempenho"
Mac-Graw-Hill - 1986;

5. KNUTH, DONALD E.,
"The Art of Computer Programming - Fundamental Algorithms",
vol. 1, Addison Wesley, 634p., 1969;

6. PIRMEZ, LUCI,
"Estudo dos Serviços de Sessão"
Relatório Técnico
NCE 00787 - Maio 1987;

7. PIRMEZ, LUCI,
"Especificação da Camada de Sessão"
Relatório Técnico a ser publicado

SISTEMA DE EMULACION DE TERMINAL Y TRANSFERENCIA DE ARCHIVOS.

Autores :

Hugo Sin Triana
Mauricio Arango Gómez

Dirección de los autores :

Centro de Cómputo
Universidad de Los Andes
Apartado Aéreo 4976
Bogotá, D.E. Colombia

La tecnología ha permitido desarrollar microcomputadores a un costo relativamente bajo, lo cual los hace accequibles a diferentes personas o entidades. Estos equipos se convierten en una herramienta de uso general y de gran importacia dadas las características computacionales. Sin embargo, se ha visto la necesidad de hacer uso de equipos computacionales medianos y grandes con el fin de cubrir todos los requerimientos de algunas organizaciones. La forma óptima de cubrir dichos requerimientos consiste en la integración de los diferentes recursos computacionales existentes en las organizaciones. Para lograr la integración, es necesario realizar la comunicación entre los equipos, lo cual se hace mediante el uso de componentes de hardware y software.

En este articulo, se muestra el desarrollo realizado por el Centro de Cómputo de la Universidad de Los Andes para lograr la integración de los recursos computacionales existentes. Dicho desarrollo establece la integración del computador central de la universidad, computador Burroughs 6800, y microcomputadores IBM PC o compatibles, para lo cual se elaboraron dos aplicaciones : emulación de terminal y transferencia de archivos.

Los tópicos aquí tratados describen los diferentes elementos que están involucrados en el desarrollo de las aplicaciones, así como la forma general en que fueron tratados.

INTRODUCCION

Una terminal es un dispositivo que permite la interacción entre un usuario y un computador. En general, las terminales son controladas por uno o más microprocesadores que pueden tener programada una gama muy amplia de funciones, lo que resulta en terminales " brutas " o muy simples, hasta terminales " inteligentes ", que son prácticamente microcomputadores con funciones especiales de comunicaciones y de manejo de pantalla y teclado.

En los últimos años se ha hecho cada vez más necesaria la comunicación entre los microcomputadores y los sistemas multiusuario (minis, mainframes). Las dos formas principales de comunicación entre micros y sistemas multiusuario son el modo terminal, es decir utilizar el micro como una terminal del sistema multiusuario y el modo de transferencia de archivos en el que se puede enviar archivos del micro al multiusuario y viceversa.

El primer modo se denomina "Emulación", es decir, se programa el micro para que emule las funciones de una terminal. Esto es posible dado que los micros son máquinas programables, lo cual hace factible el desarrollo de software para realizar dicha función, y además tienen dispositivos para realizar la comunicación.

El segundo modo de comunicación entre microcomputadores y sistemas multiusuario (transferencia de archivos) permite transferir archivos que se encuentran localizados en alguno de los dos tipos de equipos mencionados hacia el otro, obteniendo en éste último una copia del archivo transferido. En cada una de las máquinas involucradas, los archivos tienen diferentes características que deben ser tenidas en cuenta al realizar las transferencias. Los archivos corresponden a archivos de tipo texto y consisten básicamente de resultados de diferentes programas, códigos fuente de programas, documentos y datos.

El objetivo de éste proyecto ha sido el diseño e implementación de software para emular terminales Burroughs tipo TD 800 y ET 1100 en microcomputadores IBM PC o compatibles y software para permitir la transferencia de archivos entre los micros mencionados y mainframes Burroughs.

1 TERMINALES BURROUGHS.

El equipo computacional de base para el desarrollo de éste proyecto es un computador Burroughs 6800. Sin embargo, el software desarrollado para operar sobre dicho computador también funciona si el equipo de base es algún computador Burroughs que opere bajo el sistema operacional MCP.

1.1 COMUNICACIONES COMPUTADOR-TERMINAL EN LOS SISTEMAS BURROUGHS

A nivel físico, el computador Burroughs 6800 tiene dos esquemas de conexión con terminales. El primero es el esquema punto a punto, a través de un enlace RS 232 C entre la terminal y el computador. El segundo es un esquema multipunto, es decir que varias terminales comparten una sola línea física de comunicaciones y que en la nomenclatura de los computadores Burroughs se le da el nombre de TDI. Para ambos esquemas se utiliza un sistema de acceso de tipo polling, denominado **poll/select**, en el cual el computador continuamente monitorea la(s) terminal(es) en la línea controlando el orden en que las terminales envían o reciben datos del/al computador. Las líneas de conexión son independientes unas de otras, pero el tratamiento que el computador da a cada una es similar. Más adelante se da una explicación detallada del protocolo poll/select. En la figura1.1 se presentan los diagramas de los diferentes tipos de conexión que pueden establecerse entre el computador Burroughs, las terminales y los microcomputadores. Cuando se realiza una conexión entre un microcomputador y el computador Burroughs utilizando el esquema multipunto, es necesario utilizar un dispositivo que convierta la señal RS 232 C del microcomputador a señales TDI y viceversa.

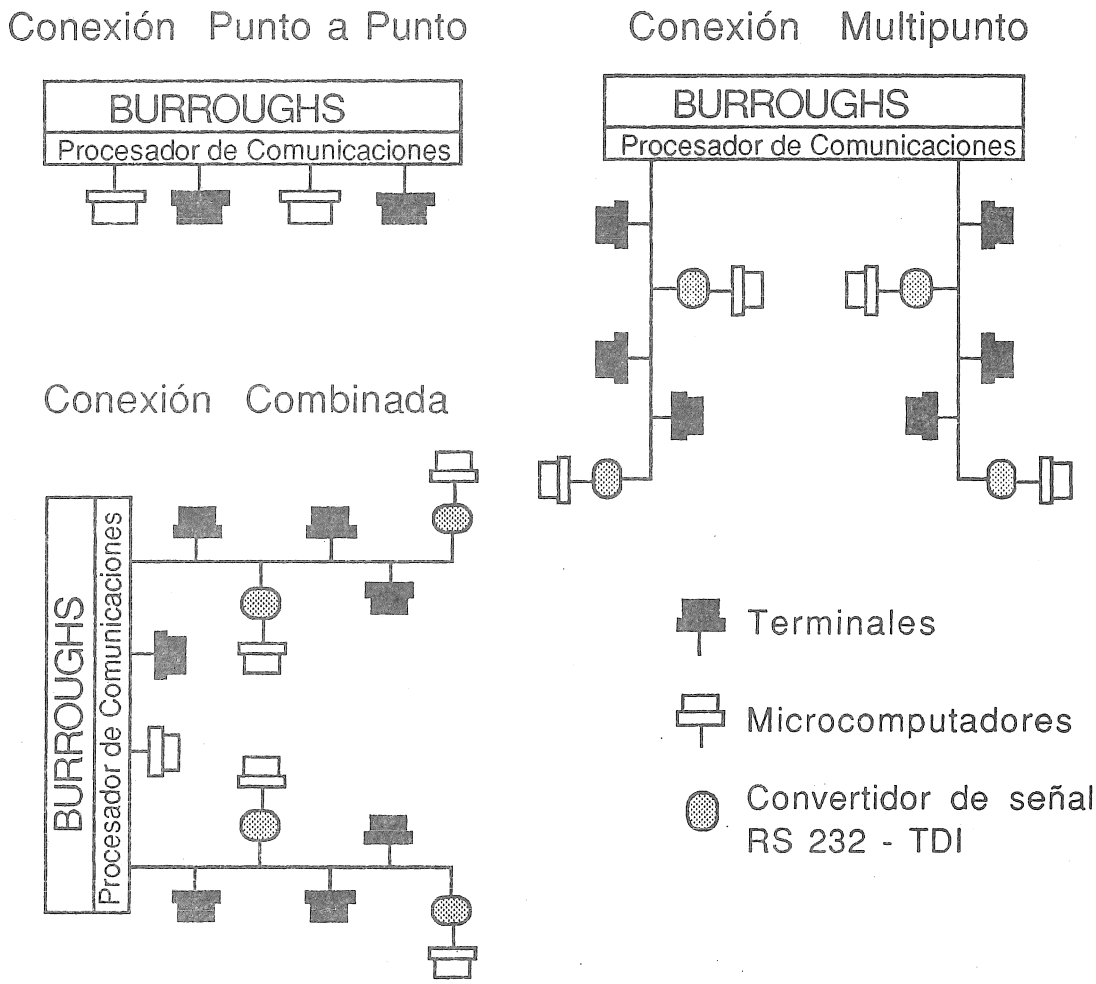


Figura 1.1

1.2 DESCRIPCION DE LAS TERMINALES BURROUGHS.

Existen diferentes características que permiten clasificar las terminales. Estas características pueden ser, por ejemplo, la representación interna utilizada para los caracteres (ASCII, EBCDIC,BCL etc.), la forma como está organizada la información que fluye entre estas y el computador (caracter por caracter, bloques de caracteres) y la existencia o no de un sincronismo en el envío de caracteres (sincrónicas, asincrónicas), entre otras. Las terminales TD 800 y ET 1100 son terminales ASCII, asincrónicas y organizan la información por bloques de caracteres.

Los bloques de caracteres que fluyen en la comunicación terminal-computador son denominados mensajes, y pueden corresponder a mensajes con información, mensajes con elementos de control o una combinación de éstos. Los mensajes con información son aquellos que deberán ser desplegados en la pantalla mientras que los mensajes de control representan órdenes que el computador envía a la terminal y que deben ser interpretados por esta. Los mensajes que la terminal envía al computador solo contienen información y no elementos de control.

1.2.1 MODOS DE OPERACION.

Los componentes de la terminal utilizados por el usuario son la pantalla y el teclado. A través del teclado, el usuario suministra información al computador, esta información, en forma de caracteres, aparece en la pantalla, la cual, a su vez, despliega la información que envía el computador. Las terminales de referencia tienen dos puertos para comunicarse con el computador central, permitiendo los dos tipos posibles de conexión (punto a punto con enlace RS 232 C y multipunto sin utilizar convertidor de señal).

Durante su operación, la terminal funciona en alguno de los siguientes tres estados básicos :

Local - Es el estado en el cual la terminal recibe información del teclado y la refleja en la pantalla. Mientras se esta en estado local, la terminal no está en capacidad de enviar ni recibir información al/del computador, y por ende no hay flujo de mesajes entre estos. Todo aquello que el usuario de la terminal realiza se procesa localmente, reflejándose solo en la terminal. Esto último implica que no existe intervención por parte del computador central.

Recibir - En este estado la terminal está en capacidad de recibir mensajes del computador central.

Transmitir - En el estado transmitir, la terminal envía al computador los caracteres de información que el usuario ha tecleado mientras se encuentra en modo local.

Las terminales solo podrán estar en uno de los tres estados mencionados anteriormente.

La pantalla corresponde a una matriz de 25 filas por 80 columnas, que puede utilizarse de dos modos diferentes. En el primer modo se utiliza como una página completa, en la cual el usuario tiene acceso a todos los puntos excepto a la fila 25. En el segundo modo, modo formateado, se tienen definidos unos campos a los cuales tiene acceso el usuario. El elemento que indica la posición donde el usuario se encuentra actualmente en la pantalla es el cursor. El cursor se puede localizar en cualquier parte de la pantalla cuando esta se usa en modo página, y solo se localiza en los campos definidos cuando la pantalla se usa en modo formateado. La pantalla despliega la información que proviene del computador central así como la información que teclea el usuario para enviar al computador.

La terminal permite al usuario trabajar localmente y decidir cuando comunicarse con el computador central. El usuario manipula la terminal haciendo uso del teclado y los efectos son desplegados en la pantalla. Esta manipulación consiste en la realización de operaciones, que se dividen en tres grupos : aquellas que corresponden a caracteres que se despliegan en la pantalla, aquellas que permiten el cambio de estado de la terminal y aquellas que alteran lo que hay en la pantalla sin reflejar ningún caracter (ejemplo: borrar pantalla, mover cursor, rotar pantalla). De acuerdo a lo anterior, el teclado se encuentra dividido en tres grupos : teclas de caracteres, teclas de control de estados y teclas de control de pantalla.

El primer grupo contiene las teclas correspondientes a los caracteres que tienen una representación gráfica (alfanuméricos, caracteres de puntuación, etc.). Todas estas teclas al ser presionadas por el usuario, causan el despliegue del caracter correspondiente en la pantalla, en la posición en la cual se encuentre el cursor.

El segundo grupo de teclas le permite al usuario modificar el estado de la terminal, de tal manera que pueda controlar el envío y/o recepción de información al/del computador central. Cuando se oprime la tecla para cambiar al estado de transmisión, el bloque de caracteres que se envía al computador central, si la pantalla se está utilizando en modo página, corresponde a aquellos caracteres que se encuentran localizados entre el extremo superior izquierdo de la pantalla y el

cursor. Si la pantalla se está manejando en modo formateado, el bloque corresponde a aquellos caracteres que se encuentran en los campos del formato.

El tercer grupo de teclas le permite al usuario manejar la pantalla de una manera global incluyendo las siguientes operaciones :

Rotación hacia arriba o hacia abajo de la página.
Borrar hasta el fin de línea o hasta el fin de la pantalla o toda la pantalla.
Insertar un caracter o una línea.
Borrar un caracter o una línea.
Movimiento del cursor en cuatro direcciones ortogonales.
Movimiento del cursor a la primera posición de la pantalla.

1.2.2 PROTOCOLO DE COMUNICACION.

El protocolo de comunicación entre la terminal y el computador central es un protocolo de línea cuyas funciones son controlar el acceso a la línea tanto para las terminales como para el computador (poll/select) y garantizar la transmisión sin errores de caracteres a través de la línea.

El computador está en capacidad de establecer diálogo con varias terminales a la vez, pero solamente recibe información o envía información de/a una terminal por línea en un mismo tiempo. Para ello se basa en el protocolo de línea poll/select que permite identificar plenamente la terminal con la cual intercambia información. Toda terminal tiene una identificación única en la línea a la cual se encuentra conectada. Esta identificación se denomina dirección de la terminal y está compuesta por dos caracteres alfanuméricos.

Todos los mensajes que fluyen desde una terminal al computador central y viceversa, son "escuchados" por las otras terminales que se encuentran en la misma línea. En cada terminal se procesan únicamente los mensajes que están dirigidos a ella, el resto de los mensajes son ignorados.

Para garantizar una comunicación sin errores, en cada mensaje se incluye un caracter de control de error, el BCC, que corresponde al OR exclusivo de todos los caracteres del mensaje. En el receptor se vuelve a calcular dicho valor y si no es igual al caracter BCC, se envía una confirmación negativa al transmisor (computador o terminal) con el fin de que se retransmita el mensaje.

De acuerdo con lo anterior, cada mensaje está compuesto por un caracter que delimita su comienzo, dos caracteres que indican la dirección de la terminal que lo envía o lo debe recibir, un caracter opcional que indica el número de transmisiones hechas del mismo mensaje, dos caracteres que delimitan el texto del mensaje, un caracter de control y los caracteres que componen el texto del mensaje. Es por estas razones que los mensajes tienen una estructura determinada, la cual se muestra en la figura 1.2.

Estructura de los mensajes

Figura 1.2

El significado de los elementos que conforman la estructura de los mensajes es el siguiente :

SOH : Caracter que indica comienzo del mensaje.
DIR1 : Primer caracter de la dirección de la terminal.
DIR2 : Segundo caracter de la dirección de la terminal.

XMN : Byte que representa el número de transmisión del mensaje. Este es opcional.
STX : Caracter que indica final de encabezado y comienzo del texto del mensaje.
TEXTO Corresponde al texto del mensaje que se envia (o recibe).
ETX : Caracter que indica finalización del texto del mensaje.
BCC : Es el caracter que permite realizar la verificación del mensaje.

Protocolo POLL/SELECT

Las figuras 1.3 y 1.4 describen el funcionamiento general del protocolo. Estos diagramas corresponden a la interacción computador-terminal para aquella terminal cuya dirección está conformada por los dos caracteres DIR1 y DIR2.

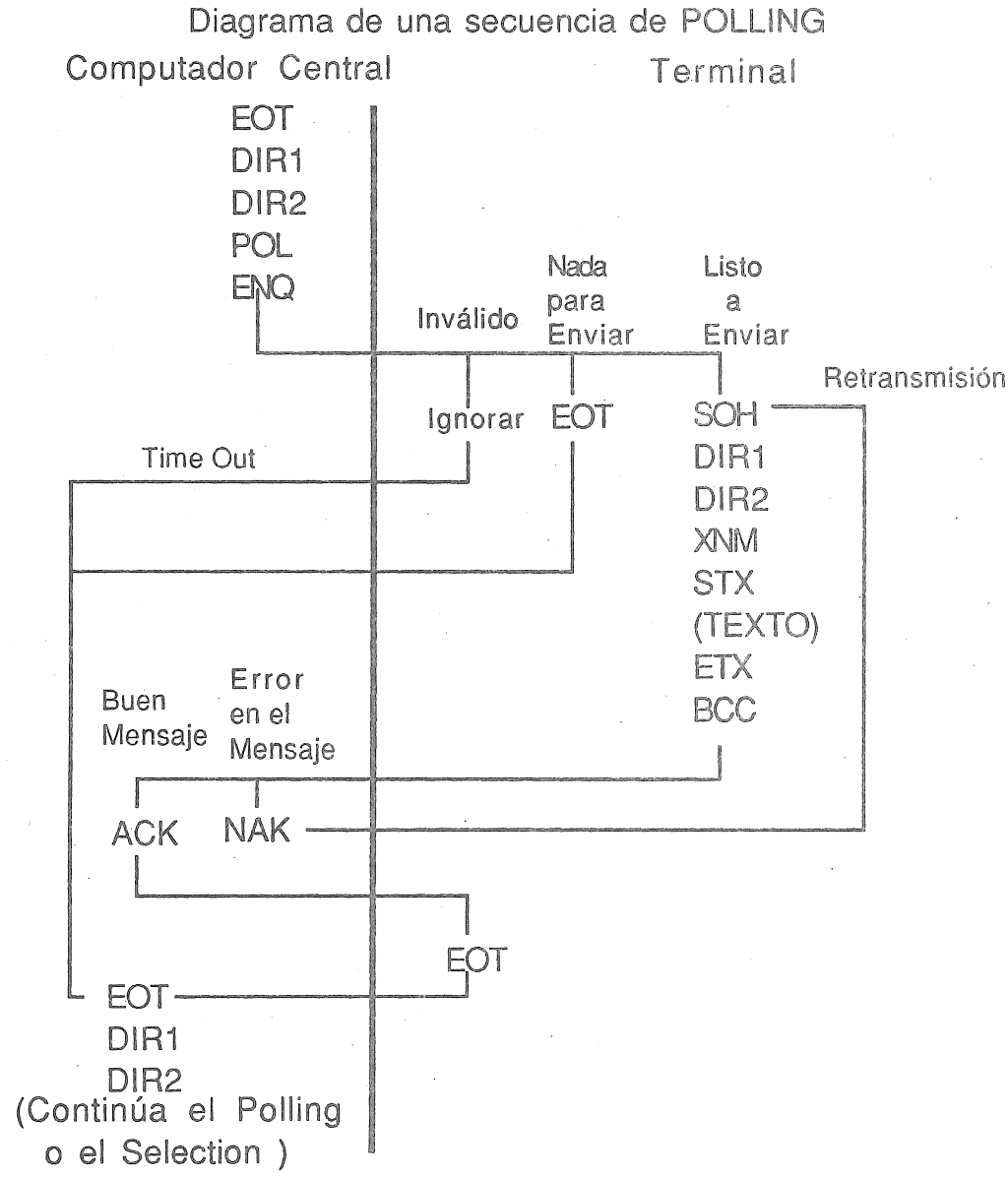


Figura 1.3

En este protocolo se cumplen dos procesos básicos : El primero, es aquel en el cual el computador le pregunta a cada terminal si tiene algo que enviarle y la terminal le responde con un mensaje cuando

tiene algo que enviar o con un caracter especial (EOT) que indica que no hay nada para enviar. Este proceso se conoce como **polling**. El segundo es cuando el computador tiene algo para enviarle a la terminal, caso en el cual le pregunta si está en capacidad de recibir; si la terminal se encuentra en un estado que le permite recibir información, envia un caracter que indicará una respuesta afirmativa - ACK -, o una respuesta negativa - NAK - en caso contrario. Este proceso se conoce como **selection**.

El polling se efectúa continuamente; la frecuencia con que se hace es una función del procesador de comunicaciones del computador central y puede variar durante su operación sin que la terminal pierda en ningún momento su conexión con el computador.

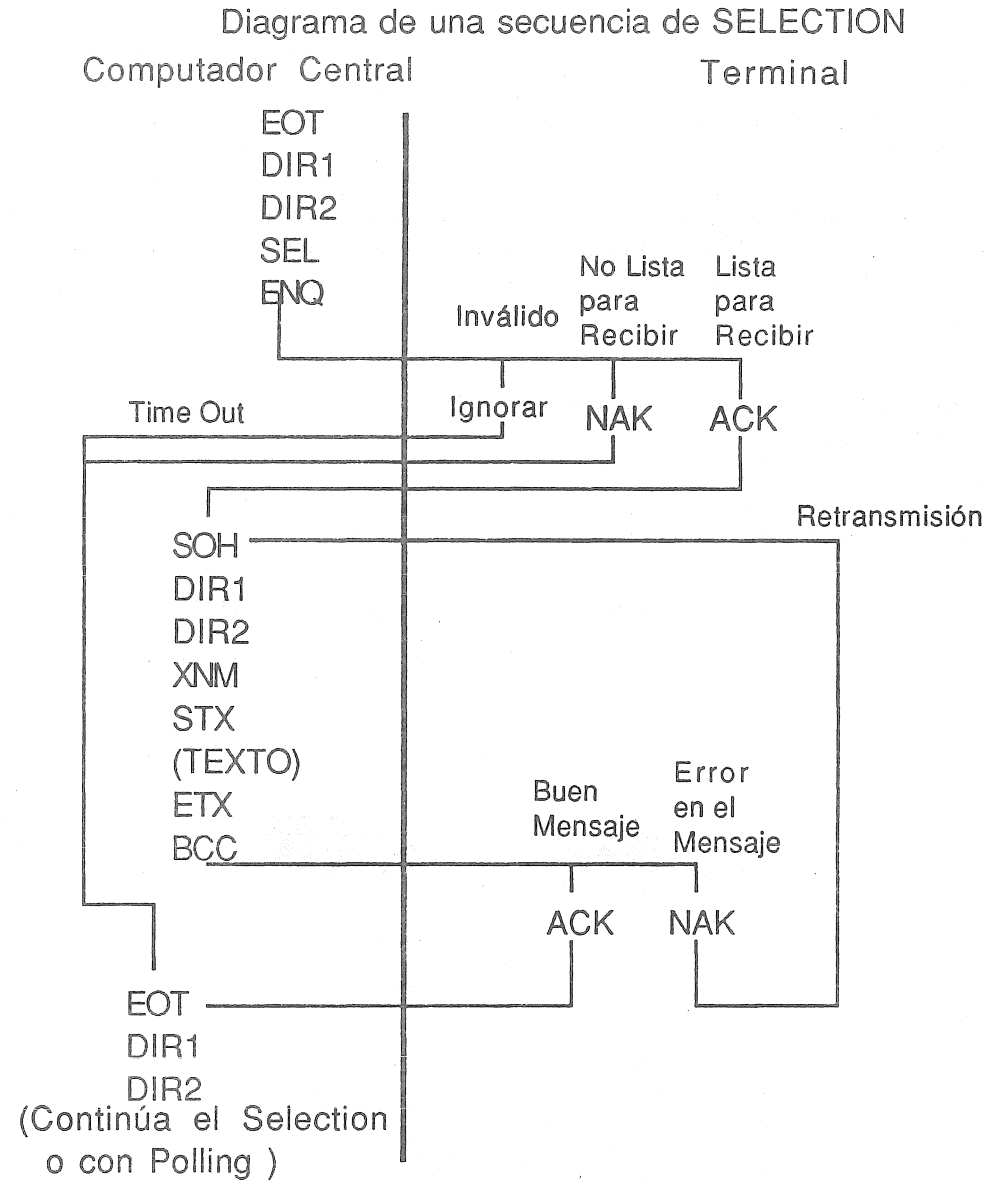


Figura 1.4

2 DESCRIPCION DEL SISTEMA.

2.1 ASPECTOS GENERALES.

El sistema se compone de dos módulos principales, cada uno de los cuales cumple con alguno de los objetivos antes mencionados. El primer módulo al cual se hace referencia tiene como objetivo realizar la tarea de emulación de terminal. El segundo cumple la función de transferencia de archivos.

Para el módulo de emulación no fué necesario efectuar ningún tipo de desarrollo en el computador central, puesto que para dicho equipo, el que haya un microcomputador emulando una terminal es transparente. Sin embargo, para el módulo de transferencia de archivos se desarrolló software tanto para el microcomputador como para el computador central. El software desarrollado para el computador central tiene como objetivo el manejo del texto de los mensajes que fluyen entre el computador y el microcomputador. La estructura de los mensajes no varía cuando se emula terminal o cuando se transfieren archivos y por ende, el envío y recepción de estos es igual en los dos casos. La diferencia entre los módulos se encuentra en la interpretación y manejo del texto de los mensajes.

2.1.1 MANEJO DE LOS DISPOSITIVOS DE ENTRADA Y SALIDA EN EL MICROCOMPUTADOR.

En la figura 2.1 se indican los elementos utilizados por el microcomputador para realizar las operaciones de entrada y salida de información.

ELEMENTOS UTILIZADOS PARA
ENTRADA Y SALIDA DE INFORMACION

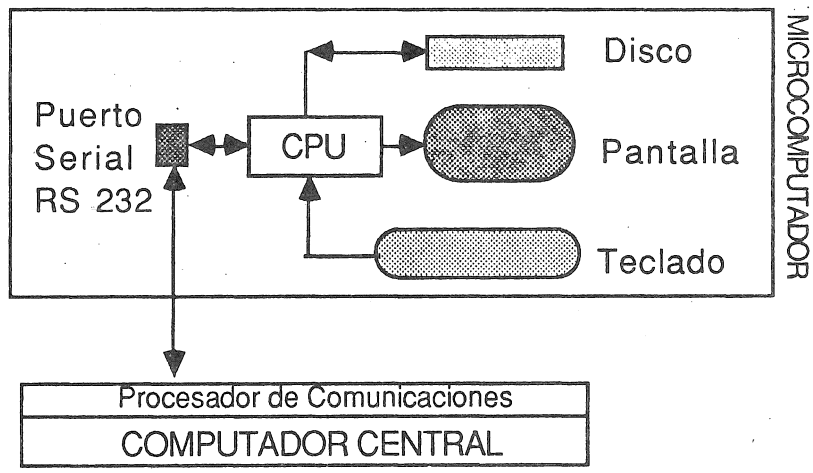


Figura 2.1

El puerto serial permite entrada y salida de información; la unidad de almacenamiento en disco permite las mismas operaciones; la pantalla permite solo salida de información y el teclado permite solo entrada de información.

El ingreso de la información al microcomputador puede producirse en cualquier momento y puede provenir tanto del usuario como del computador central. La información que ingresa por alguno de los dispositivos de entrada requiere que se efectúe alguna operación con ella, generalmente utilizando alguno de los dispositivos de salida. Esto significa que al existir un elemento ingresando al microcomputador por alguno de los periféricos que permiten entrada, se crea un requerimiento de servicio para dicho dispositivo. Cada vez que alguno de estos dispositivos tiene un requerimiento, lo indica a la unidad central de procesamiento (CPU) mediante una interrupción. Cuando se produce dicha interrupción, la unidad central atiende ese llamado dando control a una rutina donde se encuentran todas las operaciones que deben ser ejecutadas con el fin de dar el servicio al periférico que produjo la interrupción. Las rutinas a las cuales se hace mención son conocidas como rutinas de servicio de interrupción (RSI) o rutinas para manejo de la interrupción. Para cada posible interrupción debe existir una rutina que le de servicio. Estas rutinas pueden pertenecer al sistema operacional o ser escritas por el programador.

2.1.2 ESTRUCTURA DE NIVELES DE SOFTWARE.

El manejo de todos los elementos involucrados tanto en emulación de terminal como en transferencia de archivos se encuentra clasificado en niveles, los cuales representan diferentes grados de abstracción. Estos niveles tienen funciones plenamente definidas y presentan utilidades a los niveles superiores. Entre dos niveles inmediatos se crea una interface que permite al nivel superior utilizar las funciones del nivel inferior, creando una transparencia de la forma como los diferentes niveles ejecutan las funciones. La estructura de niveles se utilizó tanto en el software desarrollado para el microcomputador como en el software desarrollado para el computador central.

El esquema de dichos niveles se muestra en la figura 2.2, y el significado y función de los niveles es el siguiente:

<u>En el microcomputador</u> :

NIVEL DE APLICACION (ET, TA) : Emulación de Terminal y Transferencia de Archivos. En éste nivel se encuentran todas las rutinas que interpretan el texto de los mensajes provenientes del computador central y/o forman el texto de los mensajes que deben ser enviados al computador. Adicionalmente, se encuentran los elementos necesarios para para interpretar los requerimientos del usuario. Aún cuando las rutinas son diferentes para las aplicaciónes de emulación y transferencia, el nivel de abstracción de estas es el mismo.

NIVEL DE SERVICIO (RSI, MV) : Rutinas de Servicio de Interrupción y Manejo de Video. Las funciones encomendadas a este nivel corresponden a prestar servicio a las interrupciones y el manejo del video. Entre estas funciones se encuentra el manejo del protocolo utilizado para la comunicación entre los equipos, el manejo o interpretación del teclado y el manejo de algunos tipos de errores en la operación de entrada y salida a la unidad de almacenamiento en disco. Adicionalmente, se manejan las características de los caracteres en el video o pantalla.

NIVEL DE SISTEMA OPERACIONAL (DOS). En este nivel se encuentra el sistema operacional con el cual se operó en el microcomputador (MS DOS).

NIVEL DE HARDWARE (HW). Nivel más bajo que corresponde a la máquina física.

<u>En el computador central</u> :

NIVEL DE APLICACION (MCS) : Message Control System. En este nivel se tiene el software encargado de interpretar los mesajes provenientes de las diferentes terminales y de armar los mensajes que deben ser enviados a ellas. Los mensajes en el computador central provienen y/o son recibidos por programas que requieren interacción con terminales.

NIVEL DE SERVICIO (NDL) : Network Definition Language. En este nivel se encuentran definidas las características de la red de comunicaciones : qué estaciones pueden ser manejadas por el procesador de comunicaciones, cómo estarán organizadas en las diferentes líneas, las características físicas de las terminales, el programa con el cual deberán interactuar, el protocolo que se utilizará para las líneas y el manejo del mismo.

NIVEL DE SISTEMA OPERACIONAL (MCP) : Master Control Program. Es el sistema operacional del computador central.

NIVEL DE HARDWARE (HW) : Hardware. Nivel más bajo que corresponde a la máquina física.

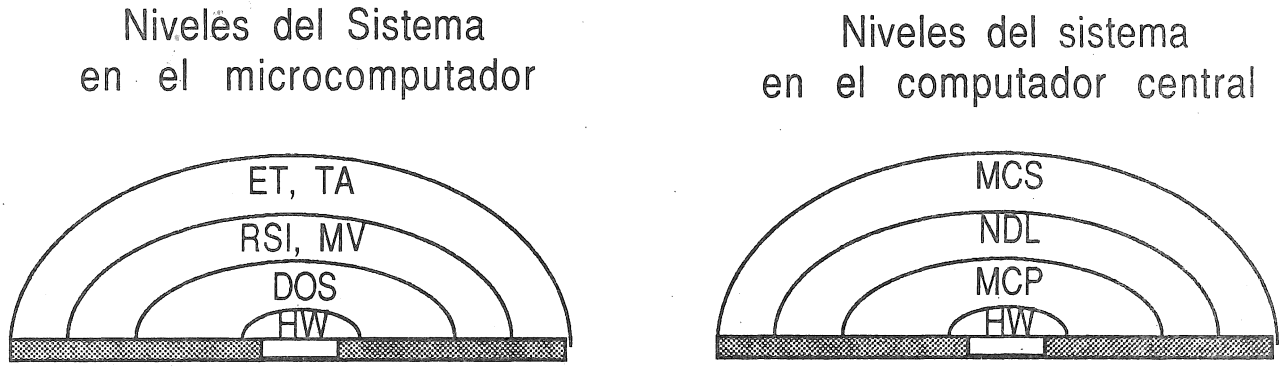


Figura 2.2

Se puede realizar un paralelo entre los niveles para cada una de las máquinas involucradas, obteniendo niveles de abstracción similares en ambas.

El desarrollo de software que se realizó en el microcomputador corresponde a los niveles de aplicación y de servicio, utilizando las facilidades o características de los niveles de sistema operacional y hardware. En el computador central, solo se desarrolló software en el nivel de aplicación utilizando lo ya existente en los niveles inferiores; esta parte corresponde a la aplicación de transferencia de archivos.

2.2 DESCRIPCION DEL SOFTWARE.

2.2.1 APLICACIONES.

2.2.1.1 EMULACION DE TERMINAL.

Las características de la terminal que se desea emular deben ser manejadas por la aplicación de emulación de terminal, de tal forma que un usuario tenga todas las posibilidades de realizar las operaciones que haría en una terminal; el sistema de emulación debe permitirle al usuario toda la gama de posibilidades que presenta una terminal y posiblemente más. Adicionalmente, para el computador central debe ser transparente la existencia de un equipo que emula, y debe ser tratado como cualquier terminal.

El software correspondiente a la emulación de terminal se encuentra localizado en el nivel de aplicación lo que le permite hacer uso de las utilidades contenidas en los niveles de servicio, sistema operacional y hardware. Sus funciones principales son la interpretación del texto de los mensajes provenientes del computador central, y la ejecución de las órdenes dadas por el usuario.

La operación básica del emulador consiste en verificar continuamente la existencia de información proveniente del computador central o del usuario. Una vez sea ha determinado la existencia de dicha información, esta debe ser tomada y manipulada según las especificaciones de la terminal que se emula. El mecanismo de ingreso de esa información es transparente a este nivel, siendo el nivel de servicio el encargado de ello. La interface entre los niveles está compuesta de una serie de buffers donde se acumula la información de entrada o salida, un conjunto de variables que permiten el manejo de los buffers y una tabla en la cual se codifica la información de entrada del teclado. Toda información que provenga del computador o del usuario se encontrará localizada en buffers independientes y es de allí que deberá ser extraída. La información que se envía al computador central debe ser colocada en un buffer de donde será tomada para transmitirla. El despliegue de información sobre la pantalla no requiere el uso de buffers, pues su tratamiento se realiza al ser interpretada la información de entrada.

2.2.1.2 TRANSFERENCIA DE ARCHIVOS.

En esta aplicación, el mecanismo de interacción entre el microcomputador y el usuario cumple una serie de pasos predeterminados, los cuales se llevan a cabo mediante el uso menus de opciones y preguntas del sistema al usuario. Adicionalmente, en esta aplicación interviene la unidad de almacenamiento en disco, de donde serán extraídos los archivos a tranferir del microcomputador al computador central o donde serán colocados los archivos transferidos del computador central al microcomputador.

Los archivos que son manipulados en el computador central tienen características que los clasifican en diferentes grupos o tipos. Las diferentes aplicaciones o programas que utilizan esos archivos deben considerar dichas características. Dentro de estas características se encuentran, por ejemplo, la longitud de los registros, la existencia de una numeración de estos, el tipo de protección, el usuario a quien pertenece, la localización lógica dentro del sistema de almacenamiento en disco, entre otros. Los archivos manejados en el microcomputador tienen también ciertas características aún cuando no son tantas como en el computador central. Todas las características mencionadas deben ser claramente manejadas para lograr la transferencia de archivos entre los dos equipos. Esto implica que existe un mecanismo que "transforma" las características de los archivos de un computador a otro. Esta función es realizada por la aplicación de transferencia de archivos, la cual incrementa información a los registros o elimina de los mismos según las características de los archivos.

El funcionamiento de esta aplicación requiere tener una descripción de las características que el usuario logra ver en los archivos, características correspondientes a los atributos en el computador central. Las características que son relevantes para el usuario son el nombre del archivo, el código de usuario al que pertenece dicho archivo, la clasificación (tipo de archivo) y posiblemente la longitud de cada registro. Las demás características deben ser inferidas por el sistema a partir de las ya enumeradas.

La aplicación de transferencia de archivos organiza los diferentes registros del archivo a transferir en bloques de caracteres que constituyen el texto de los mensajes que fluyen entre el computador central y el microcomputador. Para realizar el envío y recepción de los mensajes, se utilizan las diferentes herramientas encontradas en el nivel de servicio.

Existe, para esta aplicación, un protocolo para el envió, recepción y verificación de los archivos. Este protocolo tiene cuatro pasos principales : identificación del usuario, identificación del sentido de transferencia, identificación del archivo a transferir y transferencia del archivo. El primer paso tiene como función identificar al usuario tal como "aparece" en el computador central. El segundo paso corresponde a identificar si el archivo debe ser transferido del computador central al micro o si la transferencia debe realizarse en sentido contrario. El tercer paso consiste en determinar la información del archivo a transferir : nombre actual, tipo de archivo, nombre como

se desea que quede en el receptor y, algunas veces, longitud de cada registro. El cuarto y último paso consiste en la transferencia del archivo, en el cual debe ser verificada la información adquirida en el tercer paso y si es válida realizar la transferencia o, en caso contrario, indicar la invalidez. Esta transferencia se hace por bloques de registros, a los que se le adiciona información de control, y cada vez que se realiza el envío de un bloque se espera la confirmación por parte del receptor con el fin de repetir el envío del bloque o enviar el siguiente bloque. Un posible elemento que se recibe en alguno de los dos equipos es aquel que indica que no es posible realizar el siguiente envío de registros por presentarse algún problema no solucionable, como sería, por ejemplo, el daño físico del dispositivo de almacenamiento o la no existencia de espacio suficiente para almacenar el bloque de registros recibido.

El envío y recepción de los bloques de registros se efectua de una manera similar al envío y recepción de la información en la aplicación de emulación, teniendo en cuenta que ello no se encuentra sujeto a las acciones realizadas por el usuario, sino que está sujeta únicamente al protocolo de transferencia. Para el computador central, en el nivel de servicios, el microcomputador se comporta como una terminal más. La aplicación de transferencia de archivos tiene la posibilidad de establecer interacción para transferencia de archivos con diferentes estaciones en un mismo tiempo.

El manejo de la pantalla no requiere la implementación de las diferentes operaciones presentadas en la aplicación anterior, pues no se encuentra sujeta a las órdenes del computador ni del usuario.

2.2.2 SERVICIOS.

El segundo nivel de abstracción del sistema en el microcomputador es el nivel de servicio, el cual provee herramientas para lograr la comunicación tanto con el usuario como con el computador. Estas herramientas consisten en rutinas que dan transparencia al nivel superior o de aplicación en cuanto a características físicas de la máquina, específicamente en lo referente a las características de los siguientes componentes : el monitor, el puerto de comunicaciones, el teclado, y por último el dispositivo de almacenamiento en disco.

El sistema operacional se encarga de dar parte de esta transparencia, la que puede no ser útil en todo momento. Los problemas de sincronización de tiempos exigen algunas veces el desarrollo de herramientas aún cuando el sistema operacional ya las tenga. Adicionalmente, existe la posibilidad de que las herramientas que presenta el sistema operacional no puedan ser utilizadas debido a la incompatibilidad de estas con las herramientas desarrolladas. Un ejemplo de esto último es el caso en que se tengan desarrolladas todas las herramientas para el manejo del teclado (control sobre el teclado) y algún elemento del sistema operacional requiera información ingresada por el teclado, para lo cual supone que se está utilizando las herramientas de manejo del teclado contenidas en el mismo sistema operacional.

A continuación se describen las herramientas de software desarrolladas dentro del nivel de servicio en el microcomputador.

2.2.2.1 SOFTWARE DE MANEJO DE INTERRUPCIONES.

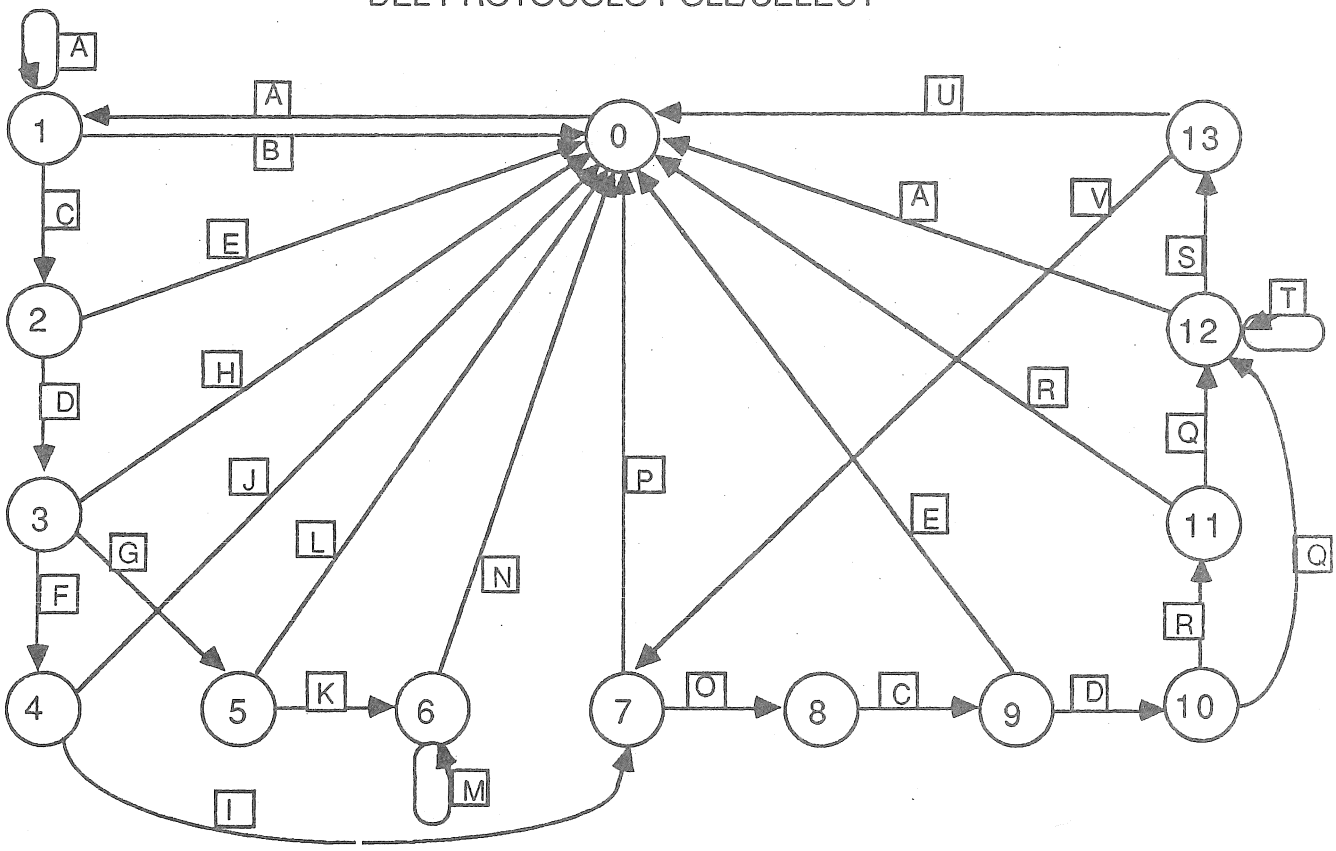
Puerto Serial

Todos los elementos necesarios para el manejo del puerto serial fueron desarrollados como parte del software del sistema. Los elementos se dividen en dos grupos principales. El primero corresponde a la programación del puerto, que consiste en darle las características con las cuales debe operar. El segundo grupo consiste en el manejo de la llegada o envío de los mensajes, en el cual está incluida la implementación del protocolo de comunicación entre el computador central y el microcomputador.

La operación de programación del puerto es una de las herramientas que se ofrecen en este nivel al nivel de aplicación. Los parámetros que requieren ser programados en el puerto son:

- Velocidad de Transmisión
- Número de Bits por caracter
- Bits de parada
- Paridad

DIAGRAMA DE ESTADOS PARA EL MANEJO
DEL PROTOCOLO POLL/SELECT



A : EOT
B : no EOT, ASCII < 31
C : DIR1
D : DIR2 y si terminal
E : DIR2 y no terminal
F : Select
G : Poll

H : no Select, no Poll
I : ENQ y si recibir
J : no(ENQ y si recibir)
K : ENQ y si enviar
L : no(ENQ y si enviar)
M: NAK
N : no NAK

O : SOH
P : no SOH
Q : STX
R : no STX
S : ETX
T : no EOT, no ETX
U : BCC correcto
V : BCC incorrecto

Figura 2.3

El otro grupo de operaciones o funciones consiste en la recepción y envío de los mensajes del o hacia el computador central. Como se anotó anteriormente, al nivel de aplicación le corresponde la interpretación y manejo del texto de los mensajes. Por esta razón es el nivel de servicio donde se

maneja el protocolo utilizado y la estructura del mensaje entregándole al (o recibiendo del) nivel superior solo la parte correspondiente al texto del mensaje.

La rutina encargada de manejar el protocolo interpreta "la pregunta" que realiza el computador central de si hay algo para enviar. En ese momento, verifica la existencia o no de algún texto en el buffer de salida y si éste existe entonces lo envía esperando la respectiva confirmación del computador central. Si la confirmación corresponde a una respuesta negativa, realiza de nuevo el envío del mensaje, sin que los elementos del nivel de aplicación intervengan. Este grupo de operaciones se encuentra localizado en la rutina de servicio de interrupción del puerto serial, puesto que cada vez que llega un caracter al puerto se produce una interrupción. Para el manejo de esta interrupción se utiliza una máquina de estados cuya descripción aparece en la figura 2.3. Esta máquina de estados se concentra en el manejo del protocolo poll/select, teniendo en cuenta la interface con el nivel de aplicación y el formato de los mensajes. Cada vez que se produce una interrupción por la llegada de un caracter, éste puede producir el cambio de estado o la permanencia en el mismo. Los estados que realizan una función adicional a la de esperar un caracter, interpretarlo y cambiar a otro estado son :

- Estado número 4 : Envía un caracter ACK o NAK dependiendo de si se encuentra en estado hábil para recibir el mensaje o no respectivamente.

- Estado número 5 : Si tiene un mensaje para enviar, lo envía. Sino tiene mensaje envía un caracter EOT.

- Estado número 6 : Si el caracter que recibe corresponde al caracter NAK, repite el envío del mensaje sin pasar a ningún otro estado.

- Estado número 12 : Es el estado en el cual se recibe el texto del mensaje.

- Estado número 13 : Al recibir el caracter, éste es comparado con el BCC calculado y si es correcto envía un ACK. Si es incorrecto, envía un NAK.

Teclado.

Cada elemento (tecla) del teclado tiene una identificación numérica única. Al presionar o al liberar una tecla se produce una interrupción, generándose en el puerto del teclado el código correspondiente a la tecla involucrada. Este código equivale a la identificación de la tecla cuando lo que se hizo fué oprimirla. Cuando se libera la tecla, el código que queda en el puerto del teclado corresponde a la identificación de la tecla mas un valor numérico igual a 128. Las rutinas que se encargan de manejar las interrupciones de teclado, interpretan este hecho realizando la conversión al caracter correspondiente, teniendo en cuenta las características de los caracteres en cuanto a su estado de mayúsculas o minúsculas, así como el uso simultaneo de varias teclas que tengan un significado especial, como las teclas CTRL y ALT combinadas con otras.

Las características del teclado exigen el tener en cuenta dos estados : mayúsculas y minúsculas. La identificación de la tecla presionada o liberada es independiente del estado del teclado, lo cual genera la necesidad de controlar dicho estado por software. Para manejar esto, se tienen elementos en memoria que indican el estado del teclado y la tabla, en la cual se encuentra la codificación de las teclas y que pertenece a la interface con el nivel de aplicación, tiene dos posiciones para cada tecla. La primera posición corresponde al valor que debe colocarse en el buffer si el estado del teclado es minúsculas y la segunda posición corresponde al valor a colocar si el estado es el contrario. La forma de accesar esta tabla consiste en indexarla por la identificación de la tecla oprimida. Los valores de la tabla se organizan de la siguiente manera: Si la tecla corresponde a un caracter alfanumérico o a un signo, el ASCII correspondiente se coloca en la tabla en las posiciones respectivas, reflejando lo que el usuario ve en el teclado. Si la tecla corresponde a alguna de las que

tienen función determinada, se coloca un valor que corresponda a un caracter de control en la tabla ASCII . Estos valores son interpretados por el nivel de aplicación.

Dispositivo de almacenamiento en disco.

Cuando se desea leer o escribir del/al disco, no siempre se cumplen las condiciones para que la operación sea efectiva. Estas operaciones se le encargan al sistema operacional. Si ocurre algún error, se genera una interrupción para la cual el sistema operacional tiene una rutina de manejo. Sin embargo, éste manejador de interrupción requiere el ingreso de información a través del teclado lo cual entra en conflicto con la aplicación de transferencia de archivos. Por esta razón, se desarrolló una rutina propia para el manejo de dicha interrupción. La interface con el nivel superior consiste en un elemento que indica si hubo o no error y es el nivel superior quien se encarga de decidir que hacer al respecto.

2.2.2.2 SOFTWARE DE MANEJO DEL VIDEO.

Para manejar el video no se utiliza ningún tipo de interrupción y tampoco todas las funciones provistas por el sistema operacional. La forma como el sistema lo maneja es colocando directamente en el espacio de memoria correspondiente al video la información y las características de la misma. De esta forma se incrementa velocidad en el manejo del video. Sin embargo, para el manejo de algunos efectos que deben ser realizados sobre el video se utilizan operaciones del sistema operacional. Estos efectos corresponden a eliminar una línea de la pantalla e insertar una línea en la pantalla, borrar la pantalla y borrar hasta el fin de una línea, operaciones que realiza el sistema operacional de una manera más eficiente de lo que se lograría al realizarlas sobre la memoria del video directamente.

La memoria de video consiste en una secuencia de 4000 bytes agrupados de dos en dos. El primer byte de cada grupo corresponde al caracter que debe mostrarse en pantalla y el segundo byte corresponde a las características con las cuales debe aparecer dicho caracter (resaltado, subrayado, inverso e invisible). Puesto que la pantalla corresponde a una matriz de 25 filas por 80 columnas, cada posición de dicha matriz tiene su información en los elementos de la secuencia de bytes indicada anteriormante.

3. CONCLUSIONES.

Al llevar a cabo este proyecto existe un objetivo de fondo que consiste en aprovechar los diferentes recursos computacionales de la universidad, integrándolos para ampliar el radio de acción del Centro de Cómputo. Las aplicaciones de emulación de terminal y transferencia de archivos han permitido facilitar el uso del computador central de la universidad a aquellos usuarios de las diferentes dependencias que cuentan con un microcomputador con las caraterísticas mencionadas.

La aplicación de emulación de terminal se puso en funcionamiento de prueba en Julio de 1986. Desde entonces se ha logrado corregir los diferentes errores que, como es natural, se presentan en el software. Adicionalmente, los usuarios del sistema realizan, además de las pruebas, una serie de recomendaciones que permiten adaptar el sistema para lograr mayores facilidades y utilidades para ellos.

En Agosto de 1987 se puso a prueba la aplicación de transferencia de archivos. Esta aplicación ha tenido gran acogida por aquellos usuarios que manipulan información localmente pero que requieren, en algún momento, el poder computacional del mainframe. Adicionalmente, el fácil uso de la aplicación ha permitido que las operaciones de transferencia de archivos sean efectuadas por personas que no tengan conocimiento de la forma como se interactúa con el computador central, dando lugar a que se amplíe el rango de usuarios que pueden utilizar estas funciones.

Todo el sistema fué diseñado e implementado en función de las necesidades de la universidad, generando para esta no solo una herramienta de uso cotidiano, sino también conocimiento que ha de ser compartido con otras personas interesadas en desarrollos que involucran cualquiera de los aspecto descritos.

REFERENCIAS

P. Norton, **The Peter Norton programers guide to the IBM PC**, Microsoft Press, 1985.

DOS Technical reference version 3.10, IBM Corporation, 1985.

IBM Personal Computer. Technical reference, IBM Corporation, 1983.

NDL Reference manual, Burroughs Corporation, 1977.

MCS User´s guide, Burroughs Corporation, 1971.

TD 700/ TD 800 Equipment. Reference manual, Burroughs Corporation, 1973.

ET 1100 Ergonomic Workstation. Installation, operation and maintenance guide, Burroughs Corporation, 1983.

ANALISE E ESPECIFICAÇÃO DOS SERVIÇOS DE TRANSFERENCIA, ACESSO E MANIPULAÇÃO DE ARQUIVOS ("FTAM")

RESUMO:

Este artigo analisa os serviços de Transferência, Acesso e Manipulação de Arquivos (FTAM) derivados da proposta de padronização apresentada pela ISO. De acordo com a proposta oferecida, é feita uma especificação dos elementos necessários ao funcionamento do FTAM no contexto da REDE-RIO.

PALAVRA-CHAVE: REDES;
INTERCONEXÃO DE SISTEMAS ABERTOS;
TRANSFERENCIA, ACESSO E MANIPULAÇÃO DE ARQUIVOS;
FTAM;
ARQUITETURA ISO.

AUTORA: Esther de Castro Pacitti
Analista do Projeto de Redes do
Núcleo de Computação Eletrônica da
Universidade Federal do Rio de Janeiro.

ENDEREÇO: NÚCLEO DE COMPUTAÇÃO ELETRONICA
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
CAIXA POSTAL 2324 - CEP 20.001
TELEFONE: 290-3212, RAMAL 294
RIO DE JANEIRO
BRASIL

1. INTRODUÇÃO

Nos dias atuais, as redes públicas são uma realidade. O compartilhamento de recursos entre diversos centros computacionais torna-se um investimento de grandes atrativos. Os serviços de transferência, acesso e manipulação de arquivos vêm ocupar este espaço, fornecendo ao usuário todo um universo de possibilidades para o compartilhamento de arquivos.

O ambiente de Transferência, Acesso e Manipulação de arquivos para arquitetura OSI, é coletivamente chamado de File Transfer Acess and Manipulation (FTAM). Este ambiente permite que arquivos de entidades computacionais distintas sejam compartilhadas. Neste cenário, um usuário conectado ao microcomputador, a fim de implementar uma certa aplicação, deseja executar um programa, no hospedeiro "mainframe", que dispõe de ferramentas que a aplicação exige. O programa encontra-se armazenado em disco, num servidor de arquivos. Os resultados da execução devem ser impressos na instalação do usuário. Operações possíveis neste caso seriam acesso (pelo usuário) ao arquivo no servidor de arquivos, a transferência desses arquivos para compilação e execução no mainframe, e finalmente a transferência do arquivo, com os resultados para instalação do microcomputador. O compartilhamento de bases de dados num banco de dados distribuídos ilustra a operação de manipulação de dados. Para que as operações requisitadas pela aplicação sejam realizadas, é necessário resolver as incompatibilidades de cada equipamento. Em sequida é preciso viabilizar a comunicação entre os sistemas (remotos) cooperantes para que eles sejam informados das operações a serem efetuadas e ajam de modo adequado.

Este artigo analisa os pontos mais importantes dos serviços de FTAM num contexto de interconexão de sistemas abertos proposto pela ISO baseado no modelo de referência OSI. Na sequência é feito uma análise dos elementos envolvidos na especificação dos serviços de FTAM de forma conclusiva.

A especificação dos serviços de FTAM e futuramente sua implementação estão incluídos num projeto denominado REDE-RIO. Este projeto possui como objetivo principal interconectar os ambientes computacionais das universidades do RIO DE JANEIRO. Deste modo as propostas encontradas neste artigo dependem dos serviços oferecidos pelas camadas inferiores:, apresentação, sessão e transporte definidas pelo projeto como um todo.

2. VISÃO GERAL DOS SERVIÇOS DE FTAM

Os servicos de FTAM permitem que sistemas incompatíveis cooperem eficazmente para realizarem operações em arquivos . Esses serviços definem como devem ser a transferência, o acesso e a gerência de arquivos armazenados ou trocados pelos sistemas de uma rede de computadores, permitem adicionar ou remover dados de uma facilidade de arquivamento e descrever os dados armazenados sem a necessidade de informações sobre detalhes de implementação/manutemção do sistema de arquivos.

Para resolver estas incompatibilidades é definido o arquivo virtual que tem como função criar um modelo de arquivo que independa da maquina hospedeira. Ao conjunto de arquivos virtuais armazenados num sistema aberto denomina-se Depósito de Arquivos Virtuais. Deste modo ao transferir um arquivo para um hospedeiro remoto transforma-se o arquivo no formato virtual para em sequida transferí-lo.

Os serviços oferecidos pelo FTAM são definidos sob unidades funcionais. Uma classe de serviço consiste na combinação de um grupo de unidades funcionais para um determinado propósito. O nível de serviço define se o protocolo é confiável ou não. Deste modo em cada inicialização da associação de aplicação é necessário negociar a classe de serviço, as unidades funcionais opcionais e o nível de serviço.

A operação do protocolo de FTAM cria um contexto para

realização das atividades desejadas pela entidade iniciadora. A criação desse contexto se dá em estágios. O periodo de tempo em que alguma informação contextual permanece válida é chamado de regime. São definidos quatro regimes : FTAM, seleção, arquivo aberto e transferência de dados. Os regimes são aninhados e a conclusão de um regime encerra todos os outros que o sucederam. Em cada associação de aplicação só pode ocorrer uma instância de cada regime. Um regime concluido pode ser reiniciado quando necessário ou desejado.

Os serviços de FTAM são realizados através das primitivas de serviço que estão descritas nas normas ISO 8571/3.

O SASE (Elemento Específico da camada de aplicação) FTAM utiliza obrigatoriamente os serviço comum de aplicação (CASE) KERNEL e opcionalmente o CCR (Commitment Concurrency and Recovery), dependendo das unidades funcionais negociadas na inicializacão. A utilização do CASE CCR torna algumas primitivas de serviço atômicas, proporcionando maior confiabilidade no protocolo.

Existem dois tipos de protocolos especificados pelas normas 8571/4: o protocolo básico e o de recuperação. O protocolo básico suporta o nível de serviço corrigivel pelo usuário (UCFS). O protocolo de recuperação suporta o nível confiável utilizando o UCFS, ou seja, ele é uma estensão do protocolo básico. A utilização destes protocolos também dependem das unidades funcionais negociadas na inicialização.

Na transferência, acesso e manipilação de um arquivo é necessário especificar o formato do arquivo virtual que será utilizado na opereção. Os modelos virtuais utilizam FADU´s (File Acess Data Unit´s) para descrever a posição de cada registro dentro da estrutura hierárquica, denotando seus irmãos, filhos, pai, se possui unidade de dados associado e revela sua identidade dentro da estrutura. Deste modo caso a operação a ser realizada no arquivo for só de transferência as informações e espaço ocupado pelas FADU´s tornam-se redundantes. Por outro lado se a operação a ser realizado envolve acesso e manipulação em registros do arquivo,

torna-se necessário a utilização de modelos virtuais mais complexos. A estes modelos virtuais adicionado as informações de operações válidas no arquivo dá-se o nome de conjunto restrito. Os conjuntos restritos definido pelas normas são desetruturado (possui apenas uma FADU), sequencial, e hierarquico.

A implementação do depósito de arquivos virtuais é um problema local de cada sistema aberto não possuindo nenhuma padronização referente ao assunto.

Em sequida serão apresentados os serviços das camadas inferiores necessários a implementação integral dos serviços de FTAM.

3. SERVIÇOS NECESSARIOS DAS CAMADAS INFERIORES

Na transferência de dados através dos serviços de apresentação, é necessário definir uma representação para os dados envolvidos. Para isso, é necessário definir uma descrição genérica das estruturas de dados (sintaxe abstrata) e negociar as regras de codificação através dos serviços de apresentação.

O conjunto sintaxe abstrata e regras de codificação é definido como sintaxe de transferência. A sintaxe de transferência específica utilizada numa transferência de dados utilizando os serviços de apresentação, é dado o nome de contexto de apresentação.

O SASE FTAM, o CASE Kernel e o CASE CCR utilizam para especificação da sintaxe abstrata de suas unidades de informação a ASN-1 (notação para sintaxe abstrata-1) definida pela norma 8224/2. E também especificado que pelo menos as regras de codificação propostas pela norma 8825 (regras básicas de codificação para ASN-1) devem estar presentes na implementação. Porém não é descartado a hipótese de utilização de outras regras de codificação.

Deste modo as unidades funcionais necessárias da

camada de apresentação são: kernel, resincronização(opcionalmente) e de manipulação (opcionalmente).

O nível de controle de sessão é o meio do usuário ter acesso a rede. E neste nivel que se realiza a negociação para o estabelecimento de uma conexão com um processo em uma maquina hospedeira remota. Outro serviço necessário desta camada é sincronização o diálogo e o gerenciamento as trocas de dados entre os usuários cooperantes do serviço de sessão. Neste escopo as unidades funcionais requisitadas pelos serviços de FTAM são: Kernel, Duplex, resincronização, sincronização secundária e Dado Transparente.

As primitivas de serviço do SASE FTAM são mapeadas nas primitivas de serviço do CASE Kernel e em alguns casos são tratadas como ações atômica embutidas nas primitivas de serviço do CASE CCR ou então são mapeadas diretamente nas primitivas de serviço da camada de apresentação. Não há nenhuma restrição sobre o nível de transporte e rede.

4. SERVIÇOS PROPOSTOS PARA A IMPLEMENTAÇÃO DO FTAM

Em função das restrições impostas pelos serviços oferecidos pelas camadas inferiores (apresentação e sessão) selecionou-se para especificação um subconjunto dos serviços de FTAM. Deste modo é possível definir o modelo funcional do sistema, a classe e nível de serviços a serem implementados, a estrutura do depósito de arquivos virtuais e consequentemente o protocolo a ser utilizado.

A classe de serviços escolhida foi a de ACESSO, que possui como obrigatório as seguintes Unidades Funcionais:

UNIDADES FUNCIONAIS	SERVIÇOS OFERECIDOS
KERNEL U1	.Estabelecimento de Regime FTAM .Término de Regime FTAM ordenadamente .Ter mino de Regime FTAM desordenamente .Seleção de Arquivo .Fim de Seleção de Arquivo
LEITURA U2	.Leitura de Massa de Dados .Trasferência de Unidade de Dados .Fim de Transferência de dados .Fim de Transferência .Cancelamento de Transferência .Abertura de Arquivo .Fechamento de Arquivo
ESCRITA U3	.Escrita de Massa de Dados .Transfrência de Unidade de Dados .Fim de Transferência de Dados .Fim de Transferência .Cancelemento de Transferência .Abertura de Arquivo .Fechamento de Arquivo
ACESSO A ARQUIVO U4	.Localização de FADU .Apagamento de FADU

TABELA 1 - UNIDADES FUNCIONAIS DE ACESSO

As sequintes unidades funcionais opcionais foram selecionadas para enriquecer a classe de ACESSO:

UNIDADE FUNCIONAL	SERVIÇOS OFERECIDOS
MANIPULAÇÃO DE ARQUIVO LIMITADA U5	.Criação de Arquivo .Deleção de Arquivo .Leitura de Atributos
MANIPULAÇÃO DE ARQUIVO AUMENTADA U6	.Requer U5 .Mudança de Atributos

TABELA 2 - UNIDADES FUNCIONAIS OPCIONAIS DE ACESSO

A não inclusão das unidades funcionais de Recuperação e de Recomeço de Transferência de Dados se deve ao fato que a implementação destas unidades requerem alguns serviços do nível de sessão que não serão ofereciadas (Sincronismo secundário resincronização e Dado transperente).

O nível de serviço UCFS será adotado, uma vez que as

unidades funcionais apresentadas para este nível se adequam com as utilizadas na classe de serviço escolhida.

A implementação do protocolo de Recuperação e do CASE CCR exigem que as unidades funcionais de recomeço e recuperação tenham sido negociadas com sucesso. Como tais unidades não são suportdas pelo nível de sessão, optou-se pela implementação do protocolo Básico, que não inclui obrigatoriamente uso de tais unidades funcioanis, e a não utillização do CASE CCR .

O conjunto restrito que predominará em todo o DAV será desestruturado por apresentar o modo de representação simples, se enquadrando na estrutura de arquivo sequencial simples e possuindo apenas uma FADU apontando para o primeiro registro do arquivo. Neste conjunto restrito são permitidas operações de leitura, escrita, apagamento e estensão no arquivo como um todo.

Os serviços apresentados acima proporcionam bastantes facilidades e recursos para um usuário dos serviços de FTAM, de modo que as restrições encontradas no ambiente de implementação e os serviços oferecidos pelas camadas inferiores não afetarem o resultado final do projeto.

5. MODELO FUNCIONAL FTAM

De acordo com o modelo apresentado na figura 1 é descrito o funcionamento da entidade de aplicação FTAM.

O Sistema implementa os serviços de FTAM é controlado pelo SASE FTAM que é responsável por gerenciar e sincronizar todas as tarefas necessárias ao funcionamento da aplicação.

O procedimento básico do SASE FTAM, do ponto de vista interno, consiste em um ciclo de interrogações . Cada ciclo analisa os requerimentos do elemento usuário, analisa as solicitações do CASE KERNEL e a comunicação com a camada de

apresentação e ativa quando necessário o módulo de mapeamento, acesso e manipulação de arquivos. Cada análise efetuada se refere a uma possível recepção de uma mensagem proveniente do elemento usuário, do CASE KERNEL, do módulo de mapeamento, acesso e manipulação de arquivos ou da camada de apresentação.

Para o tratamento de entrada e saída entre os diversos módulos e a camada de apresentação, optou-se pelo emprego do dispositivo virtual (mailbox). O SASE FTAM apenas consulta as caixas postais quando deseja verificar a existência de mensagens.

O sistema FTAM pode ser dividido em cinco módulos (processos) que são:

5.1. Sitema de Arquivo Local (SAL):

Sistema de arquivos utilizado pelo sitema aberto localmente.

5.2. Mapeamento, Acesso e Manipulação de arquivos:

Responsável mapeamento dos arquivos reais para o formato virtual e vice-versa. Além disso são realizadas as operações de manipulação e acesso nos arquivos virtuais solicitadas pelo usuário requisitante através do SASE FTAM. A este processo está associado o depósito de arquivos virtuais (DAV). Deste modo todas as operações relativas aos arquivos são realizadas neste módulo.

5.3. SASE FTAM

O SASE FTAM é o provedor se serviços, agindo do seguinte modo:

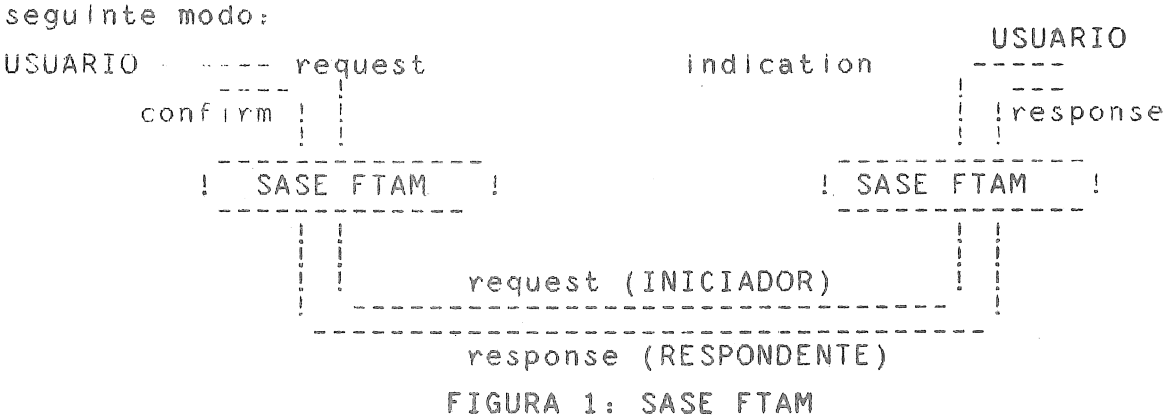


FIGURA 1: SASE FTAM

Este módulo é responsável por operar a maquina de estados de estabelecimento de associação de regimes FTAM, SELEÇAO, ARQUIVO ABERTO e de TRANSFERENCIA DE DADOS. O SASE FTAM é o processo central, ele possui acesso e informações a todos os outros processos. Qaulquer operação a ser realizada sobre o DAV é controlado pelo SASE FTAM.

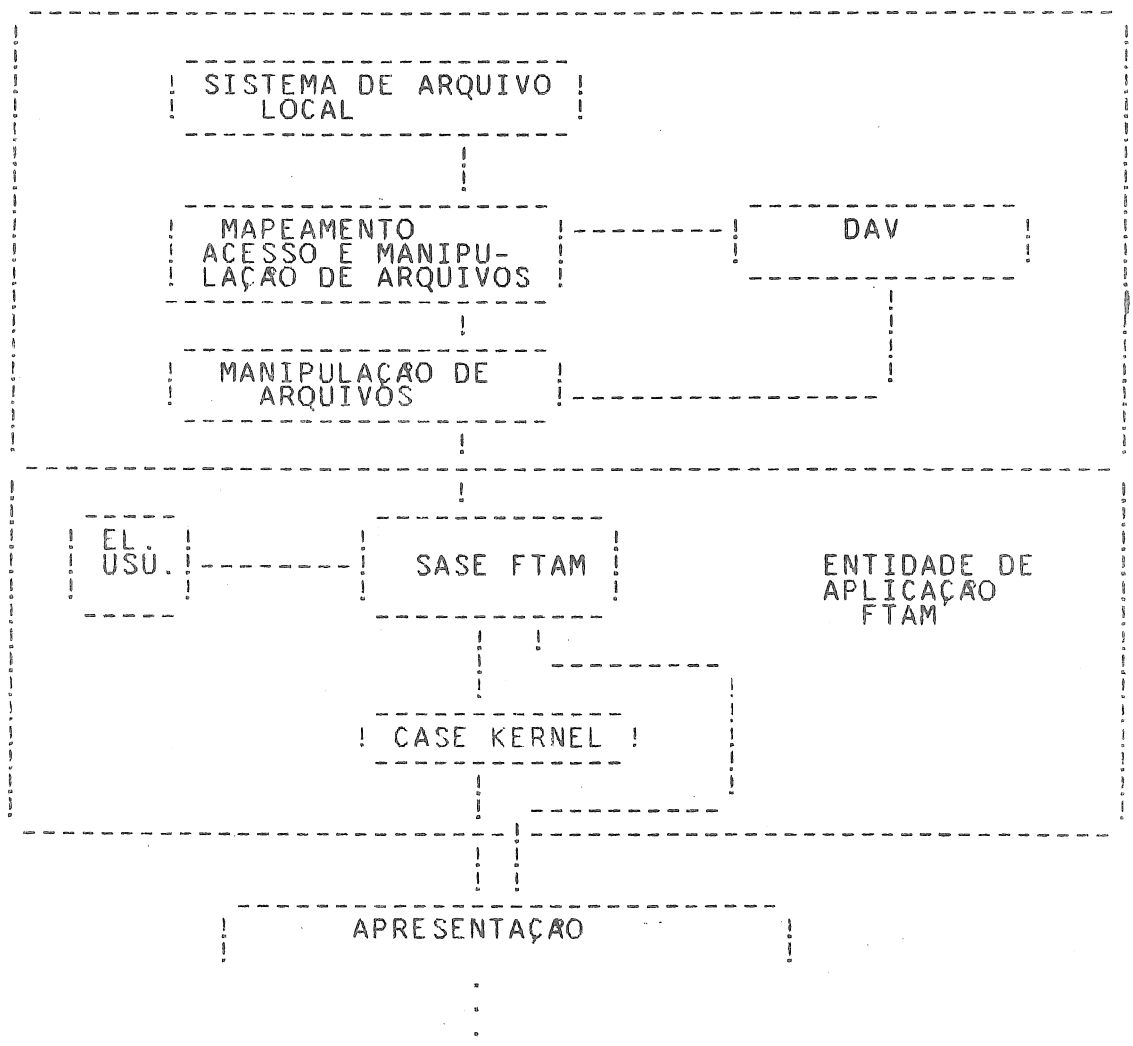


FIGURA 2: MODELO FUNCIONAL DO FTAM

5.4. CASE KERNEL

Responsável por efetuar e encerrar as associações de aplicação. Algumas primitivas de SASE FTAM são mapeadas como PDU's (Protocol Data Unit) nas primitivas do CASE KERNEL, outras são mapeadas diretamente como PDU's nas primitivas de apresentação.

5.5. Elemento Usuário

Consiste no usuário do FTAM, pode ser um usuário humano ou um programa de aplicação. Neste módulo são obtidos os requisitos do usuário, ou seja, verificar quais as funções que são desejadas mapeando-as nas primitivas de serviço do SASE FTAM.

6. INTERFACES ENTRES OS PROCESSOS

Como mencionado, as informações trocadas entre os processos internos ao sistema e com a camada de apresentação pode ser implementado através de dispositivos virtuais de entrada e saída denominado caixa postal (mailbox).

A utilização deste dispositivo como interface permite que as estruturas de dados sejam locais a cada processo da aplicação, proporcionando maior grau de modularidade no sistema.

As mensagens que contém as informações consumidas ou produzidas do/para o mailbox possuem o sequinte formato:

6.1 Processo Usuário X Processo SASE FTAM

USUARIO FONTE - Identidade da Aplicação chamadora

USUARIO DESTINO - Identidade da Aplicação chamada

TIPO DE FUNÇÃO - Identifica a primitiva de serviço a ser acionada

PARAMETROS - Contém os parametros da primitiva de serviço

6.2 Processo SASE FTAM X Processo de Mapeamento, Acesso e Manipulação de arquivos

TIPO DE OPERAÇÃO : Indica a opreração a ser executada

NOME DO ARQUIVO: Arquivo a ser manipulado

PARAMETROS : Parametros necessários a realização da operação

PTR$DADO: Caso a operação seja de leitura ou de escrita, este campo aponta para área de dados

6.3 Processo SASE FTAM X Processo CASE KERNEL

USUARIO	PSAP FONTE	PSAP DESTINO	TIPO	PARAMETROS

USUARIO - Identidade da aplicação chamadora

PSAP FONTE - Indentificador do ponto de acesso do serviço de apresentação chamador

PSAP DEST - Identificador do ponto de acesso do serviço de apresentação chamado

TIPO - Identifica a primitiva do CASE KERNEL

PARAMETROS - Contém os parametros da primitiva de serviço

6.4 Processo SASE FTAM X Apresentação

Possui os mesmos campos apresentados acima acrescido de:

PTR - Caso a primitiva de serviço envolva a transferência de dados, este campo aponta para a área de armazenamento.

6.5 Processo CASE KERNEL X Apresentação

Possui os mesmos campos apresentados no item 6.3.

7. CONCLUSÃO:

Devido a grande abrangência dos serviços de FTAM, optou-se como primeira versão de especificação escolher um subconjunto dos serviços que são fundamentais para a transferência de arquivos. A modularidade do sistema permite que novos serviços possam ser acrescidos sem maiores alterações funcionais do projeto.

Cabe mencionar que a aplicação será implementada no VAX/VMS utilizando os serviços de um rede pública.

O direcionamento deste trabalho visa especificar os procedimentos e as estruturas de dados internos de cada prcesso apresentado. Estes procedimentos deverão ser

especificados seguindo uma metodologia formal para, além de adquirir conhecimento da linguagem ESTTELE, evitar interpretações errôneas.

8. REFERENCIAS:

1. ISO 7498
   Information Processing Systems Open Systems Interconection -
   Basic Reference Model;

2. ISO 8571/1
   File Transfer, Acess and Manipulation -
   Part 1 : General Introduction;

3. ISO 8571/2
   File Transfer, Acess and Manipulation -
   Part 2: The Virtual File Definition;

4. ISO 8571/3
   File Transfer, Acess and Manipulation -
   Part 3: The Virtual File Definition;

5. ISO 8571/4
   File Transfer, Acess and Manipulation -
   Part 4: The File Protocol Specification;

6. GIOZZA,E. et all
   "Redes Locais de Computadores
   Protocolos de Alto Nível e Avaliação de
   Desempenho"
   Mac-Graw-Hill - 1986;

7. PIRMEZ,LUCI
   "Estudo dos Serviços de Sessão"
   Relatório Técnico - Maio 1987;

8. RUST, Luis Fermendo e PACITTI,Esther
"Estudo dos Serviços de Transferência e
Manipulação de Tarefas (JTM)"
Relatório Técnico
NCE - ØØ887 - Maio 1987;

9. PACITTI,Esther
"Estudo dos Serviços de Transferência Acesso e
Manipulação de Arquivos em uma Rede de Sistemas
Abertos"
Relatório Técnico
NCE Ø1187 - Julho de 1987

ANALISIS DEL VIDEOTEX COMO SERVICIO TELEMATICO PARA COLOMBIA

Héctor Alirio Rojas
Departamento de Sistemas
Universidad Nacional Bogotá Colombia

Se pretende con este trahajo realizar un estudio preliminar del Videotex para su posible implantación en Colombia. Por Videotex se entiende la difusión de información textual y gráfica radicada en bases de datos utilizando terminales de bajo costo para hacerla accesible a usuarios no entrenados en computación. Se cubre por tanto el servicio unidireccional, llamado teletext y el interactivo (bidireccional) frecuentemente conocido como videotex.

Inicialmente se hace una descripción de las principales características de este servicio telemático. Como actua, protocolos de presentación existentes, configuraciones posibles, protagonistas del servicio, terminales, redes de comunicación factibles, etc. Esta descripción se cree necesaria por la poca bibliografía existente en Colombia sobre el tema.

Se realiza un estudio socio-económico que busca determinar la preferencia de tipos de información, tarificación del servicio, aceptación del servicio cruzandola con datos como uso y propiedad de otros medios de informacion e ingreso familiar. Este estudio se hizo a través de una encuesta a cerca de 200 profesionales de diversas ramas del conocimiento.

Posteriormente se buscaron identifar factores de politica que jugaran un papel crucial en el desarrollo de esta tecnologiá en Colombia: organismos y reglamentaciones sobre telecomunicaciones e informática, politicas de servicios telemáticos. Esto da origen a plantear una serie de alternativas frente a problemas como estandares, estructura del mercado videotex, acceso del público al servicio, privacia y seguridad, etc.

Finalmente, se presentan unas recomendaciones para la implantación de esta tecnologia: propuesta de configuración básica del servicio y algunos factores reglamentarios y administrativos que deberan ser tenidos en cuenta.

HECTOR ALIRIO ROJAS BORBON

ESTRUCTURAS DE INFORMACION

Y

BASES DE DATOS

COMPORTAMIENTO DEL ARBOL-BD EN LAS FASES CRECIENTE, DECRECIENTE Y ESTACIONARIA

O. SANTANA, O. MAYOR, M. DIAZ, O. LOPEZ
E.U. Informática, Universidad Politécnica de Canarias
Aptdo. 550, Las Palmas de Gran Canaria, España

El presente artículo lleva a cabo un estudio experimental de la estructura denominada árbol-BD. Esta estructura es mejorada en las extracciones mediante una operación de recombinación que no permite punteros nulos y optimiza la ocupación. Dicho estudio aparece dividido en tres fases, cada una de ellas caracterizada por el tipo de operaciones a realizar. Previamente, se han realizado una serie de pruebas destinadas a comparar las características de dos posibles redistribuciones locales alternativas, a fin de escoger la más adecuada.

La primera fase, de estado creciente, corresponde a la etapa de crecimiento del árbol. En ella, se procede a realizar diez mil inserciones en cada árbol, midiendo durante este proceso una serie de parámetros que han de mostrar el comportamiento de la estructura.

La segunda fase, es la de estado estacionario. En ella se mantiene prácticamente fijo el número de elementos mediante una mezcla aleatoria de inserciones y extracciones, a partes iguales. En esta fase, a parte de la medición de parámetros de la estructura frente a operaciones alternadas, se llevan a cabo pruebas de las diferentes peticiones de información.

Por último, en la tercera fase se realiza la prueba del estado decreciente: extrayendo diez mil elementos hasta vaciar el árbol. Así, se pueden medir parámetros de la estructura ante situaciones críticas, simétricas, en cierto modo, de las de la primera fase.

COMPORTAMIENTO DEL ARBOL-BD EN LAS FASES CRECIENTE, DECRECIENTE Y ESTACIONARIA

O. SANTANA, O.MAYOR, M. DIAZ, G. LOPEZ
E.U. Informática, Universidad Politécnica de Canarias
Aptdo. 550, Las Palmas de Gran Canaria, España

0) INTRODUCCION

A menudo, es necesario recuperar información de una base de datos que está localizada en un espacio n-dimensional. Por ejemplo, en una base de datos puede ser necesario ver todos los datos con los atributos localizados en una zona rectangular dada. Una recuperación de este tipo se denomina búsqueda multidimensional.

El campo de aplicación de estas estructuras de ficheros no se queda sólo en las bases de datos. La capacidad de recuperación multiclave es especialmente requerida por aplicaciones que abarcan áreas como la inteligencia artificial, procesamiento de imágenes, robótica, cartografía, reconocimiento de formas, estadística, y recuperación de información, entre otras.

La estructura de árbol-BD, propuesta por Ohsawa y Sakauchi [4], realiza una partición dinámica del espacio n-dimensional, adaptándose al flujo de datos de entrada, y se lleva a cabo eficientemente usando la idea de zonas discriminadoras y sus expresiones derivadas. El origen del nombre es: división Binaria n-dimensional con expresión de zona Discriminadora.

En el crecimiento de la estructura de árbol-BD se utiliza el método de partición descrito en [1,2,4], para resolver las situaciones de sobrecarga de una celda. La información de los nodos internos es una expresión de zona discriminadora, EZD, [1,2,4], que representa a la zona correspondiente.

La estructura de árbol-BD puede no conducir a un árbol bien equilibrado. Se utilizan tres mejoras para resolver este problema: intercambio de EZD [4], redistribuciones locales [1,2,4] y arreglo local [4].

Intercambio de EZD

El árbol-BD tiene la propiedad de que para cada nodo el subárbol izquierdo representa un volumen menor o igual que el representado por el subárbol derecho. Por tanto pueden ocurrir situaciones no deseadas, tal como tener un nodo con un hijo derecho cuya EZD tiene menos longitud que la de su padre, lo cual significa que el hijo representa un volumen mayor que el de su padre. En este caso el árbol se puede desequilibrar y por tanto ser ineficaz para alguna secuencia de datos de entrada.

Se distinguen dos casos. En el primero, la EZD del hijo, prefija a la de su padre. En el segundo, las dos EZD son diferentes.

Redistribución local

Esta técnica propone posponer la división de una celda si el nuevo punto se puede acomodar entre las dos celdas hermanas. Esto es, cuando una celda está sobrecargada, antes de dividirla, se redistribuyen los puntos de la misma y de la celda hermana. Usando este esquema, se puede mejorar la utilización del fichero. Existen dos tipos alternativos de redistribución local uno propuesto por Dandamudi y Soreson [1,2], que se denota por RL-DS, y otro propuesto por Ohsawa y Sakauchi [4], que se denota por RL-OS.

Arreglo local

El árbol-BD tiende a partir más frecuentemente el subárbol derecho que el izquierdo. Si se generan zonas discriminadoras con el mismo volumen configuran el árbol de forma similar a una escalera. Esta situación no es deseable, y para mantener un árbol equilibrado se introduce un esquema de arreglo local.

En el proceso de decrecimiento del árbol se presenta en este trabajo una reestructuración denominada recombinación. La recombinación geométrica de la celda correspondiente al interior de una zona con la celda de su exterior, se produce cuando la suma del número de puntos de las dos celdas no exceda de un cierto umbral del tamaño máximo de celda, tmc. Para llevar a cabo dicha recombinación se elimina la celda correspondiente al interior de la zona y esto provoca una extracción de un nodo interno.

BUSQUEDA Y RECUPERACION

Una petición de búsqueda asociativa consiste en especificar ciertas condiciones que deben de satisfacer las componentes de los puntos del espacio que se desean recuperar.

En el árbol-BD se llevan a cabo los siguientes tipos de recuperaciones de información [2]:

I) Petición Exacta
II) Petición Parcial
III) Petición en Rango
IV) Petición en Rango Parcial

La petición exacta especifica un valor para cada clave.

Una búsqueda en rango en un espacio n-dimensional consiste en recuperar todos los puntos, r(j) j=1..n, tales que

$$rmin(j) \leq r(j) \leq rmax(j) \quad j=1..n$$

donde las componentes de los vectores rmin y rmax son los límites inferiores y superiores, respectivamente, del rango. Una petición en rango consistirá, por tanto, en especificar rmin(j) y rmax(j) para j=1..n.

La petición parcial especifica un valor para nce claves, con nce < n, y deja las restantes sin especificar.

La petición en rango parcial es similar a una petición en rango en la que algunas claves tienen el rango sin especificar.

La sección 1 de este artículo sirve como introducción al estudio experimental que se lleva a cabo sobre la estructura de árbol-BD. Dicho estudio se va desgranando en las sucesivas secciones de una forma acorde con el carácter dinámico de la estructura [3]. Así, en la sección 2 se desarrolla la fase del experimento que estudia la estructura en crecimiento continuo. Previamente, en esta misma sección se presentan las pruebas necesarias para dilucidar entre los dos tipos de redistribuciones locales. La sección 3 contiene el estudio de la fase estacionaria y las pruebas realizadas para medir diferentes parámetros de la estructura ante las peticiones de información solicitadas. Por último, la sección 4 estudia la fase decreciente del experimento. En la sección 5 se presentan las conclusiones de este trabajo.

1) DESCRIPCION DEL ESTUDIO EXPERIMENTAL

El rendimiento de una estructura de datos está determinado por dos criterios: el tiempo de procesamiento y la utilización de memoria.

Se han realizado varias simulaciones para estudiar el rendimiento del árbol-BD, midiendo estos criterios en base a los siguientes parámetros:

a) número de nodos internos totales, NIT
b) número de celdas totales, CT
c) número de nodos internos accedidos, NIA
d) ocupación, PO
e) altura promedio, HP
f) reestructuraciones, R

De acuerdo con el caracter dinámico de la estructura el estudio se ha realizado en las siguientes fases:

1) Estado creciente. Inserciones repetidas.
2) Estado estacionario. Inserciones y extracciones manteniendo aproximadamente constante el número de registros.
3) Estado decreciente. Extracciones repetidas.

Todos los números aleatorios utilizados en los experimentos se han obtenido a partir de un generador de números pseudoaleatorios de distribución continua uniforme en [0,1]. Para cada registro se generaron nd, siendo nd la dimensionalidad del espacio, valores aleatorios pertenecientes a una distribución uniforme discreta entre 0,01 y 1,00.

2) ESTADO CRECIENTE

El estado creciente corresponde a la fase de ampliación del árbol a través de sucesivas inserciones.

Se construyeron árboles de 10000 registros. Los párametros anteriormente descritos, desde el a hasta el e, se midieron en intervalos regulares de 100 inserciones, excepto la altura promedio que se midió en intervalos de 1000. El número de nodos internos difiere del número de celdas en uno, ya que no existen punteros nulos. Las reestructuraciones se midieron a nivel de promedios y los tipos estudiados fueron: particiones, intercambio de EZD, redistribución local y arreglo local.

Una cuestión previa es el estudio comparativo de la estructura para los dos tipos de redistribuciones locales: la RL-OS y la RL-DS. Para ello se construyeron dos árboles de dimensionalidad 6 y tmc 5, obteniéndose que:

- La diferencia del número de nodos internos de ambos se fue incrementando a lo largo del proceso de inserción, alcanzándose al final un árbol de 2762 con la primera y de 2564 con la segunda.

- El número de nodos internos accedidos por inserción coincidió para ambos árboles a lo largo de todo el proceso.

- La ocupación fue próxima a un 78% con la RL-DS y a un 73% con la RL-OS, manteniendo una diferencia constante a lo largo de todo el proceso, como se observa en la figura 1.

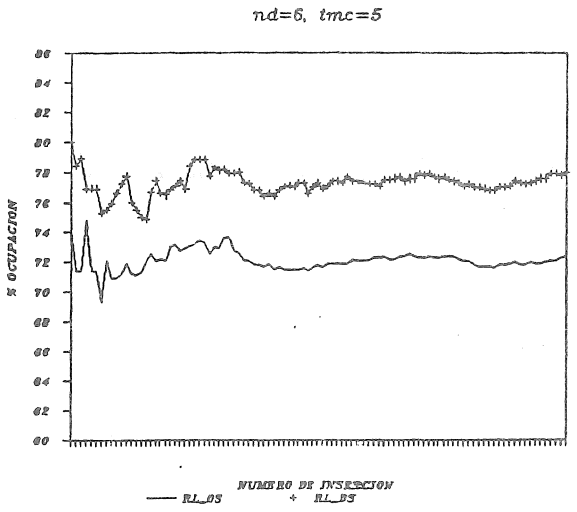


Figura 1

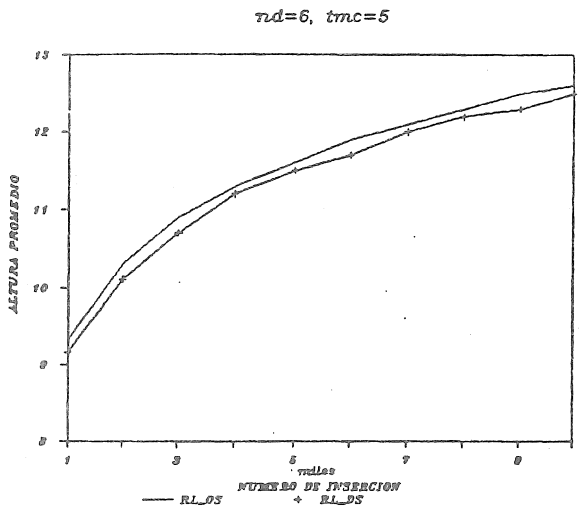


Figura 2

- La altura promedio es creciente y mantiene una diferencia mínima constante entre ambos árboles, a favor de la RL-DS, como se observa en figura 2.

- La estructura requiere un gasto adicional para mantener el árbol equilibrado que viene dado por los promedios de intercambios de EZD, arreglos locales y redistribuciones. Cuando se utiliza la RL-DS no se realiza ningún arreglo local, mientras que si se realizan para la RL-OS. En la tabla 1 se muestran estos resultados, además del número de particiones y del tiempo por inserción obtenidos en la comparación:

PROMEDIO DE	BD con RL-OS	BD con RL-DS
RL-DS		0.1244
RL-OS	0.0332	
INTERCAMBIO DE EZD I	0.0445	0.0324
INTERCAMBIO DE EZD II	0.0447	0.0353
ARREGLO LOCAL	0.0037	
PARTICION	0.2768	0.2562
TIEMPO/INSERCION (sg)	0.0423	0.0398

Tabla 1

Estos resultados indican un mejor rendimiento de la estructura con la RL-DS. Por tanto, todos los experimentos que se presentan a continuación se realizaron utilizando dicha redistribución local.

Se han construido 20 árboles, teniendo en cuenta la variación de la dimensionalidad y tmc, para los siguientes valores:

nd : 4, 6, 8, 10
tmc: 5, 10, 15, 20, 25

obteniéndose que:

- El número de nodos internos se incrementó en todos los casos, a lo largo del proceso, obteniéndose finalmente árboles cuyo NIT disminuye al crecer tmc. Por otro lado NIT no aparece afectado por la dimensionalidad.

- El número de nodos internos accedidos para cada valor de tmc, se comprobó que prácticamente no variaba con la dimensionalidad. Para una dimensionalidad dada al aumentar tmc tal y como se observa en la figura 3 el número de accesos disminuye.

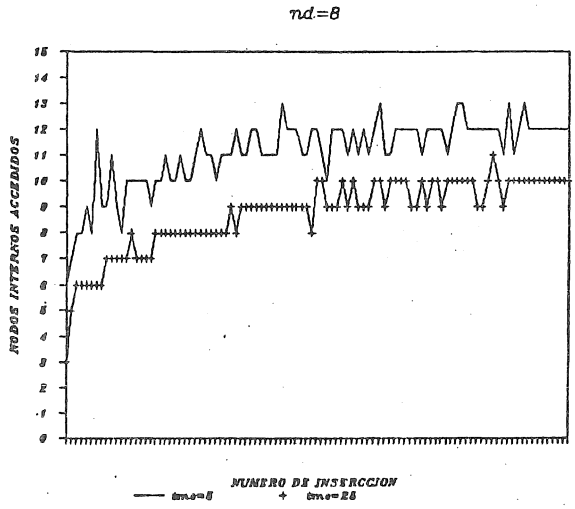


Figura 3

- En la ocupación tampoco influye la dimensionalidad al fijar tmc, como se observa en las figuras 4 y 5, donde cada línea representa una dimensionalidad distinta. Así mismo la ocupación crece al disminuir tmc, observándose que en cada caso están presentes oscilaciones de tipo sinusoidal, asintóticas a un valor constante, siendo estas oscilaciones mayores cuanto mayor es tmc.

- La altura promedio para cada valor de tmc coincidió exactamente al variar la dimensionalidad. Es mayor para valores de tmc menores, manteniéndose una diferencia constante durante el proceso de inserción, como se observa en la figura 6.

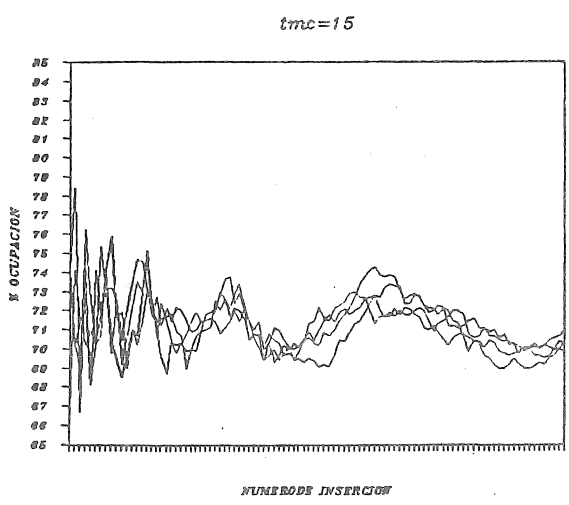


Figura 4

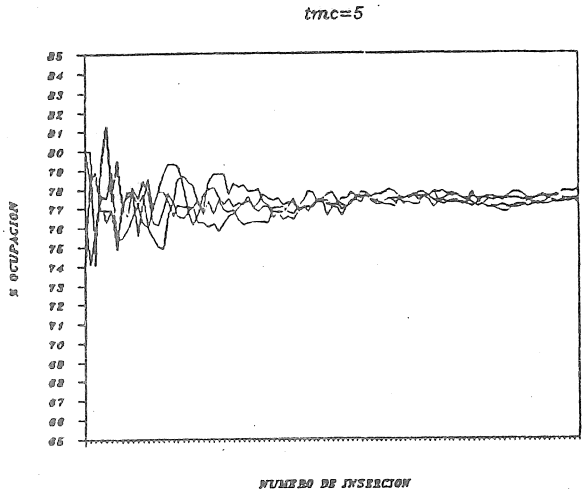


Figura 5

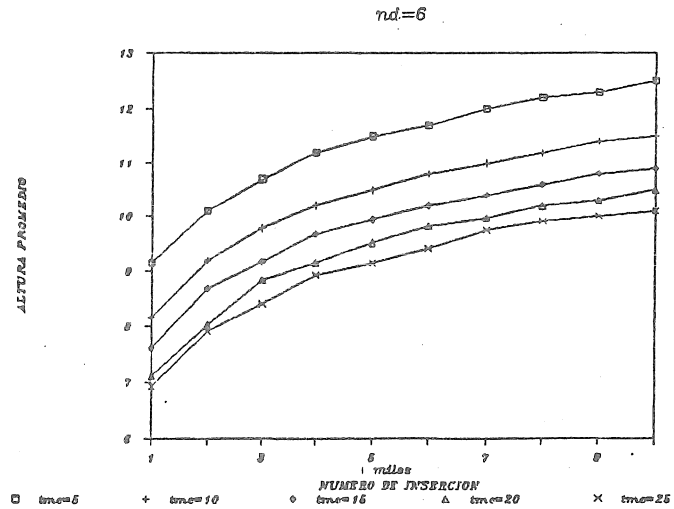


Figura 6

3) ESTADO ESTACIONARIO

El estado estacionario corresponde a la fase de actualización del árbol, a través de la selección de inserciones y extracciones de forma aleatoria.

A partir de un árbol de 10000 registros construidos en el estado creciente, se insertaron 1000 nuevos registros y se extrajeron 1000 de los ya existentes.

Debido a que la variación de la dimensionalidad prácticamente no influye en los parámetros medidos, para los valores estudiados, y que la variación de tmc

presenta resultados análogos, el estudio experimental en este estado se realizó con dimensionalidad 6 y tmc 10.

Los parámetros anteriormente descritos, desde el a hasta el e, se midieron en intervalos regulares de 100 inserciones+extracciones, excepto la altura promedio que se midio en intervalos de 400. Las reestructuraciones utilizadas fueron, por un lado, en las inserciones las mismas que en el estado creciente, y por otro, en las extracciones: la extracción de un nodo interno, la recombinación en función de los distintos umbrales. Se midió el promedio de las reestructuraciones de forma conjunta.

- Se observó durante el proceso un incremento del número de nodos internos, indicando un aumento de entropía de la estructura, debido a la diferencia de eficacia en la aplicación de los procesos de mantenimiento en las fases de inserción y extracción.

- El número de nodos internos accedidos fue prácticamente constante durante el proceso.

- En correspondencia con el hecho de que el número de nodos internos aumenta, se obtuvo que la ocupación disminuye a medida que aumenta el número de inserciones+extracciones.

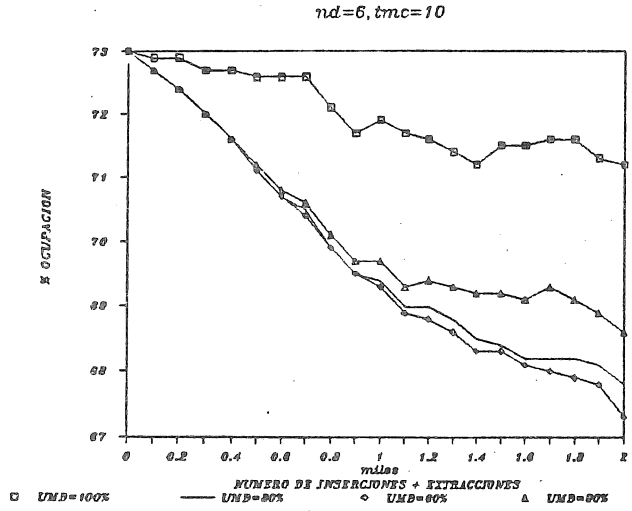


Figura 7

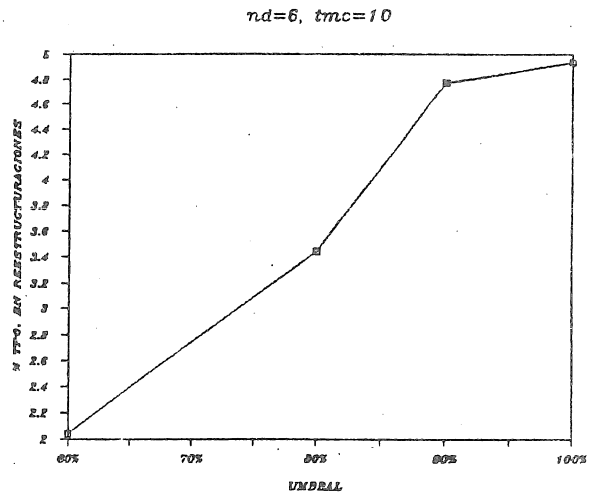


Figura 8

Como se observa en la figura 7, la disminución del umbral de recombinación produce un efecto negativo en la ocupación de la estructura. Este hecho, aparece agravado por la tendencia del proceso a aumentar el número de nodos, tendencia que aparece más fuerte cuanto menor sea el umbral de recombinación.

En la figura 8 se muestra la porción del tiempo total que se emplea en las operaciones de reestructuración. Como era de esperar se comprueba que el aumento del umbral de recombinación hace crecer el número de operaciones de mantenimiento precisas.

3.1 Estudio experimental de las peticiones exacta, parcial, en rango y en rango parcial

Otro criterio para determinar el rendimiento de una estructura es el tiempo de respuesta a los diversos tipos de peticiones.

El tiempo de respuesta se mide en función de los siguientes parámetros:

c) número de nodos internos accedidos, NIA.
g) número de celdas visitadas, CV.
h) número de registros recuperados, RR.
i) eficiencia de la petición = (RR / RT) / (CV / CT), donde RT es el número de registros totales.

Durante el estado estacionario se midieron estos parámetros como promedio, con umbral de recombinación 100%, para los distintos tipos de peticiones, en cinco instantes (inserción+extracción) del mismo: 0, 500, 1000, 1500 y 2000.

Las peticiones exactas se eligieron de los ficheros utilizados para la construcción de cada árbol. Las parciales se generaron a partir de las exactas seleccionando aleatoriamente las posiciones de las claves sin especificar, variando el número de claves sin especificar de 1 a 4. Las peticiones en rango se generaron a partir de las exactas, considerándolas como límite inferior del rango, sumándole a cada componente una longitud fija para obtener el límite superior; se tomaron longitudes de 0.1, 0.2, 0.3 y 0.4. Sólo se tuvieron en cuenta aquellos hipercubos que estuviesen contenidos en el espacio considerado. Las peticiones en rango parcial se generaron a partir de las de rango seleccionando las posiciones y el número de claves sin especificar como en las peticiones parciales.

El número de pruebas realizadas en cada instante y para cada petición fue de: 500 en la exacta, 2000 en la parcial (correspondiendo 500 a cada valor del número de claves sin especificar), 2000 en rango (correspondiendo 500 a cada tamaño del hipercubo), y 8000 en rango parcial (teniendo en cuenta el número de claves sin especificar y el tamaño del hipercubo).

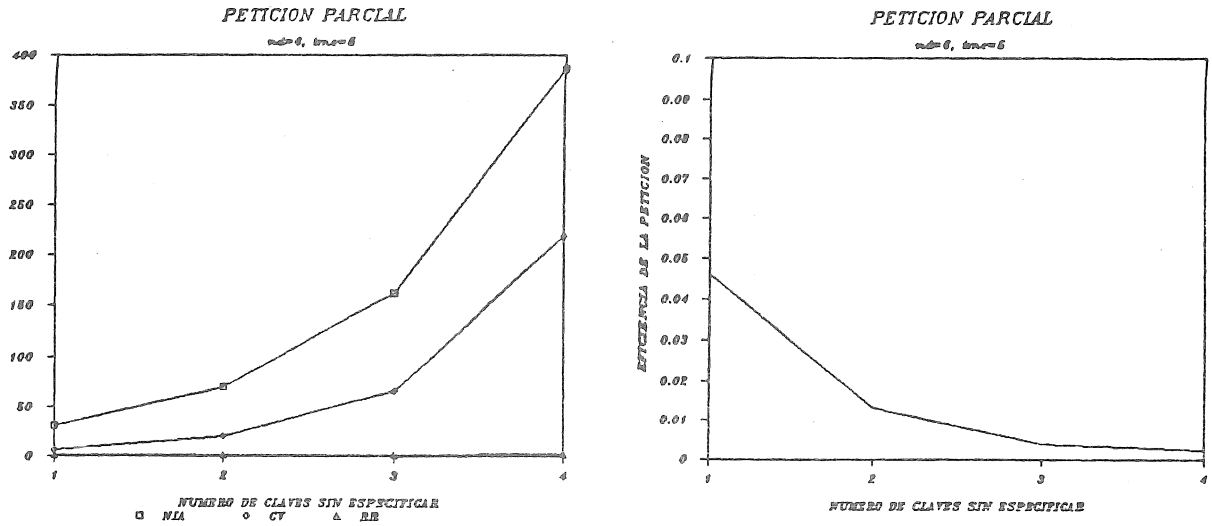


Figura 9

Los resultados obtenidos mostraron la independencia de los parámetros c, g, h e i, del instante en que se realice cualquier petición.

En una petición exacta se visita una sola celda y el número de puntos recuperados puede ser uno o ninguno, dependiendo de si la búsqueda es exitosa o infructuosa. El número promedio de accesos obtenido fue 12.4128.

El comportamiento de la petición parcial en función del número de claves sin especificar se presenta en la figura 9. El crecimiento del número de registros recuperados no es tan significativo como el de celdas debido al gran volumen del espacio en el que se está trabajando. La eficiencia de la petición es baja por la misma razón.

Los resultados de la petición en rango en función del tamaño del hipercubo, th, se muestran en la figura 10. Nuevamente se observa que el crecimiento del número de registros recuperados es inferior al número de celdas debido a la baja densidad de puntos. La eficiencia de la petición es baja por la misma razón.

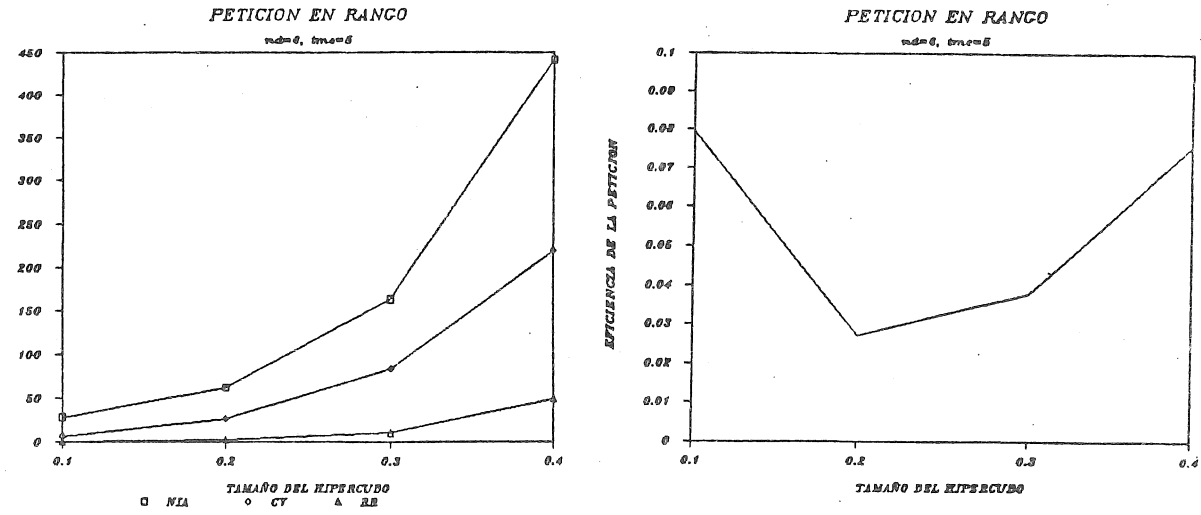


Figura 10

En la petición en rango parcial se presentan, en la figura 11, los resultados en función del número de claves sin especificar y del tamaño del hipercubo. Como era de esperar, los cuatro valores medidos aumentan tanto con el número de claves sin especificar como con el tamaño del hipercubo. Con estos niveles de vaguedad en la petición y a pesar de la baja densidad de puntos del árbol, se presenta por primera vez en el número de registros recuperados un crecimiento similar al del número de celdas visitadas.

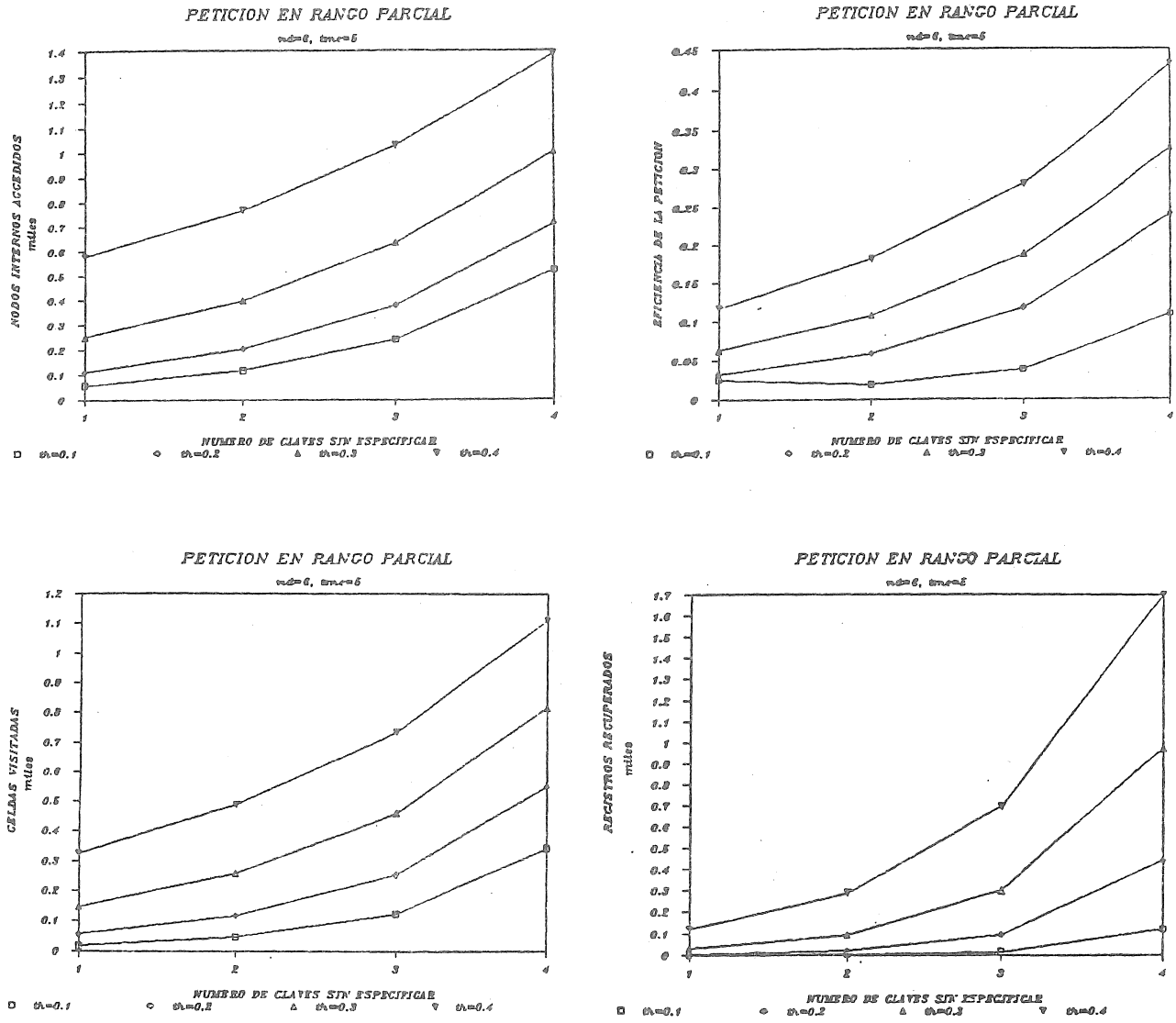


Figura 11

4) ESTADO DECRECIENTE

El estado decreciente corresponde a la fase de disminución del árbol a través de sucesivas extracciones.

A partir del árbol final del estado estacionario se realizaron las extracciones de los 10000 registros, midiendo los mismos parámetros y en iguales intervalos que en el estado creciente. Cada uno de los parámetros fue estudiado para diferentes umbrales de recombinación. Obteniéndose que:

- El número de nodos internos accedidos, presenta una ligera disminución con el número de extracciones hasta un punto muy cerca del final del proceso en el cual desciende drásticamente. La variación del umbral no presenta diferencias significativas.

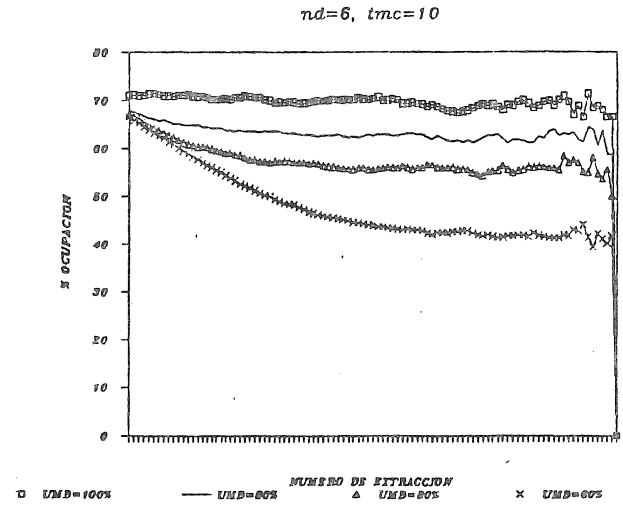


Figura 12

- En cuanto a la ocupación se observa, figura 12, de forma general, una ligera disminución a lo largo del proceso. Tan solo en la extracción de los últimos registros aparece un descenso brusco. Esto hace pensar en lo adecuado del sistema que permite mantener un alto nivel de ocupación hasta alcanzar las últimas extracciones. En cuanto al estudio de los distintos umbrales aparece una caída del nivel de ocupación más precipitada cuanto menor sea el umbral de recombinación.

5) CONCLUSIONES

El presente estudio experimental se ha llevado a cabo utilizando la redistribución local RL-DS, puesto que en las comparaciones previamente efectuadas se manifestó como la más adecuada, tanto en compactación del almacenamiento utilizado, como en tiempo de proceso.

En la fase de crecimiento se observó la independencia de los parámetros medidos con respecto a la dimensionalidad. Sin embargo, y tal como era de esperar, el número de nodos internos totales, el número de nodos accedidos, la ocupación y la altura promedio presentaron crecimientos al disminuir el tamaño máximo de la celda.

La fase estacionaria presentó un ligero, pero continuo, aumento del número de nodos internos totales. Sin embargo, los valores del número de nodos internos accedidos si que se mantuvieron prácticamente constantes. La ocupación presentó también una tendencia a la baja, tendencia que se ve agravada al producirse la más mínima variación del umbral de recombinación óptimo (100%). Como es lógico el aumento del umbral de recombinación trajo consigo un crecimiento del tiempo empleado en operaciones de reestructuración. El estudio de los distintos tipos de peticiones se mostró independiente del punto de la fase estacionaria en que se lleva a cabo.

Por último, la fase decreciente mostró como, a pesar de que la ocupación es menor cuanto menor es el umbral de recombinación, el descenso brusco se produce para todos los umbrales tan solo en la extracción de los últimos registros.

REFERENCIAS:

[1] DANDAMUDI, S.P. AND SORENSON, P.G.
Algorithms for the BD Tree Structure.
Dept. of Computational Science. University of Saskatchewan Saskatoon. Saskatchewan, Canada. 1984.

[2] DANDAMUDI, S.P. AND SORENSON, P.G.
An Empirical Performance Comparison of Some Variations of the K-d Tree and BD Tree.
International Journal of Computer and Information Sciences, Vol. 14, Nº 3, 1985.

[3] NIEVERGELT, J.; HINTERBERGER H. AND SEVCIK K.C.
The Grid File: An Adaptable, Symmetric Multi-key File Structure.
Institut für Informatik.
ETH Zürich. 1981.

[4] OHSAWA, Y. AND SAKAUCHI, M.
The BD-Tree-A New N-Dimensional Data Structure with Highly Efficient Dynamic Characteristics.
Institute of Industrial Science, University of Tokyo 22-1, Roppongi 7, Minato-ku, Tokyo 106, Japan. 1983.

BUSQUEDA DE LOS M-VECINOS MAS PROXIMOS EN EL ARBOL-BD

O. SANTANA, M. DIAZ, O. MAYOR, J. GONZALEZ
E.U. Informática, Universidad Politécnica de Canarias
Aptdo. 550, Las Palmas de Gran Canaria, España

El presente estudio está dedicado a desarrollar un esquema para las recuperaciones de los m-vecinos más próximos en la estructura de Árbol-BD. La primera parte del trabajo lleva a cabo el planteamiento algorítmico para este tipo de recuperaciones. En ella, se han desarrollado una serie de tests específicos para la estructura, sobre todo en cuanto a los tests de solapamiento. El clásico test de solapamiento con la esfera ha sido desdoblado en dos, uno de solapamiento con el interior de una zona y otro con el exterior. Así mismo, ha sido necesario el desarrollo de otro test adicional para detectar los solapamientos con el exterior en los descensos alternativos (exterior de interior).

La segunda parte del artículo está dedicada al estudio experimental de la estructura ante el esquema de búsqueda de los m-vecinos más próximos. Previamente se justifica experimentalmente el uso del test de solapamiento exterior de interior. Otra disyuntiva se plantea en el uso de dos tipos alternativos de distancias. Se opta por la utilización paralela de ambas, a lo largo de todo el proceso de estudio, mostrándose una de ellas superior en todo momento, para este tipo de estructura. Los parámetros fueron medidos frente a variaciones del tamaño de celda, la dimensionalidad y el número de vecinos a localizar.

BUSQUEDA DE LOS M-VECINOS MAS PROXIMOS EN EL ARBOL-BD

O. SANTANA, M. DIAZ, O. MAYOR, J. GONZALEZ
E.U. Informática, Universidad Politécnica de Canarias
Aptdo. 550, Las Palmas de Gran Canaria, España

Resumen:

En este trabajo se presenta la petición de los m_vecinos más próximos en la estructura de árbol_BD. Se introduce un tipo de solapamiento denominado solapamiento exterior en los descensos alternativos (exterior de interior), que mejora el tiempo de respuesta. Además se muestra experimentalmente como influye la dimensionalidad, el tamaño de la celda, y el número de vecinos más próximos en dicho tiempo.

0) INTRODUCCION

A menudo, es necesario recuperar información de una base de datos que está localizada en un espacio nd_dimensional. El campo de aplicación de estas estructuras de ficheros no se queda solo en las bases de datos. La capacidad de recuperación multiclave es especialmente requerida por aplicaciones que abarcan áreas como la inteligencia artificial, procesamiento de imágenes, robótica, cartografía, reconocimiento de formas, estadística, y recuperación de información, entre otras.

Uno de los tipos de recuperación de información que tiene más posibilidades de generar campos de aplicación, es la búsqueda de los m_vecinos más próximos. Este artículo está dedicado al planteamiento algorítmico de este tipo de búsqueda [3] y a la realización del estudio experimental del mismo.

La estructura de árbol_BD utilizada en el estudio procede básicamente de la propuesta por [4] y optimizada por [1,2,5]. Para ella, se ha desarrollado el esquema de búsqueda de los m_vecinos más próximos que se presenta en la sección 1.

La sección 2 presenta el estudio experimental. Plantea previamente unas comparaciones a fin de escoger el esquema de búsqueda adecuado, y a continuación se realizan las medidas paramétricas necesarias para mostrar la respuesta de la estructura a este tipo de requerimientos. Las conclusiones de este artículo aparecen en la sección 3.

1) BUSQUEDA DE LOS M-VECINOS MAS PROXIMOS

Este tipo de búsqueda consiste básicamente en, dados un espacio nd_dimensional, un punto del mismo, y una función distancia, recuperar los m_registros más próximos al punto dado.

Para realizar esta búsqueda, se parte del nodo cabeza y se recorre el árbol de forma recursiva hasta alcanzar una celda, análogamente al proceso de búsqueda exacta. Si el punto es localizado en la celda, finaliza el proceso. En caso contrario, se insertan en una lista los m_elementos de la celda más próximos al punto, siempre que en la celda existan más de m_elementos, si no se insertan todos. A continuación se recorre el árbol retrocediendo recursivamente en el camino de bajada inicial, probando a la vez otras alternativas de descenso, hasta obtener una lista con los m_vecinos más próximos de todo el árbol.

1.1 Lista de los m-vecinos más próximos

En el recorrido inicial por el árbol se llega a una celda (zona donde pudiese estar el punto), y se testea si algún registro de la misma coincide con

la petición.

Una vez que se ha comprobado que el punto no existe, se comienza a crear la lista de forma ordenada con los registros existentes en la celda. Sus elementos están clasificados en el sentido de mayor a menor distancia al punto. La función que evalúa dicha distancia es la siguiente:

```
función DISTANCIA (ec)
{ec elemento de celda}
sum=0
para j = 1 hasta nd hacer
     sum=sum + dist (p(j),ec(j))
fin para
DISTANCIA=F(sum)
retornar
```

Las funciones dist y F dependen de la métrica utilizada.

Los m primeros puntos probados se insertan en la lista de forma ordenada sin ninguna restricción. Posteriormente, cualquier otro punto sondeado pasará a formar parte de la lista si su distancia al punto es menor que la correspondiente del primer elemento, eliminando a este último e insertándose en la posición adecuada según la ordenación.

Por lo tanto, la inclusión o no de un punto en la lista, viene determinada por el radio de la esfera, considerado como la distancia del punto al vecino más alejado que pertenezca a la lista. El acceso a las restantes celdas se realiza retrocediendo recursivamente por el recorrido de bajada inicial, probando otras alternativas de descenso utilizando el criterio de los solapamientos de la esfera.

1.2 Solapamientos de la esfera

Cuando se retrocede recursivamente en el camino de bajada inicial se prueba la contención de la esfera actual con la zona, siempre que se llegue a ella por el interior. Si se verifica la contención, finaliza el proceso, si no, se prueba el solapamiento de la esfera con el exterior de dicha zona y si se verifica, se desciende recursivamente; en caso contrario se sigue retrocediendo. Si se llega a la zona por el exterior, se prueba el solapamiento con el interior. Si se verifica, se desciende recursivamente y si no se sigue retrocediendo.

1.2.1 Solapamientos de la esfera con el interior de una zona

En el estudio del solapamiento de la esfera con el interior de una zona se distinguen dos casos:

a) El punto p, está situado fuera de la zona.

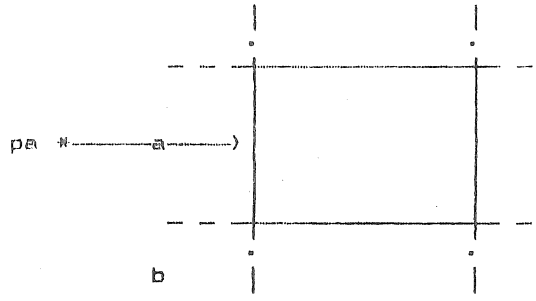


Figura 1

Según se muestra en la figura 1, para posiciones análogas a pa se determina la distancia a y para posiciones análogas a pb se determina la distancia b. En cualquier caso, si el radio de la esfera es superior a dichas distancias hay solapamiento. La función que lo verifica es la siguiente:

```
función SOLESFIN_BD (liz(j),lsz(j);j=1..nd)
{liz,lsz son los límites de la zona sobre la que se quiere sondear el
 solapamiento de la esfera}
sum=0
para k=1 hasta nd hacer
     si p(k) < liz(k) entonces
        sum=sum + dist (p(k),liz(k))
        si F(sum) > radio entonces SOLESFIN_BD=falso; retornar fin si
                   si no
        si p(k) > lsz(k) entonces
           sum=sum + dist (p(k),lsz(k))
           si F(sum) > radio entonces SOLESFIN_BD=falso; retornar fin si
        fin si
     fin si
fin para
SOLESFIN_BD=verdadero; retornar
```

b) El punto es interior a la zona (figura 2).

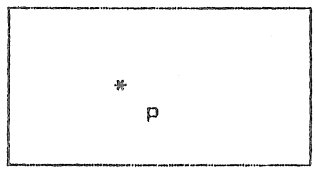


Figura 2

El solapamiento es obvio en este caso. El problema consiste en determinar si la zona contiene a la esfera. Para ello se calculan las distancias del punto a los límites de la zona. Si alguna de ellas es inferior al radio no hay contención y en caso contrario sí la hay. La función que lo verifica es la siguiente:

```
función CONESF_BD (liz(j),lsz(j);j=1..nd)
{liz, lsz son los límites de una zona que incluye a la petición}
para k=1 hasta nd hacer
     si liz(k) () domin(k) entonces
        si dist (p(k),liz(k)) ( radio entonces
           CONESF_BD=falso; retornar
        fin si
     fin si
     si lsz(k) () domax(k) entonces
        si dist (p(k),lsz(k)) ( radio entonces
           CONESF_BD=falso; retornar
        fin si
     fin si
fin para
CONESF_BD=verdadero; retornar
```

1.2.2 Solapamientos de la esfera con el exterior de una zona

Una clasificación de este tipo de solapamiento debe considerar tres zonas: la zona del punto, Zp, aquella sobre la que se determinó el solapamiento interior, Z, y la zona más pequeña que contenga a esta última, CZ. Todas las posibles situaciones que se pueden presentar se estudian a continuación.

a) Zp está incluida en Z, y ésta a su vez, lo está en CZ.

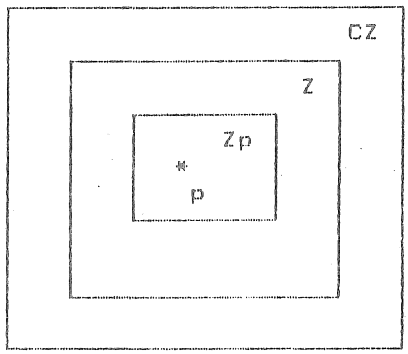


Figura 3a

Respecto a la situación mostrada en la figura 3a, siempre que Z no contenga a la esfera, hay solapamiento con el exterior de Z, si Z y CZ no tienen límites comunes. Si hay algún límite común, como en la figura 3b, es necesario medir las distancias a los límites no comunes. Si alguna de ellas es inferior al radio de la esfera hay solapamiento. La función que realiza el proceso se presenta a continuación.

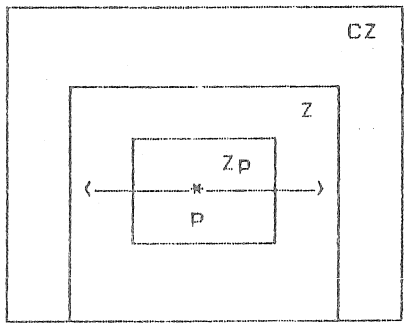


Figura 3b

```
función SOLESFEX_BD (liz(j),lsz(j),cliz(j),clsz(j):j=1..nd)
{liz, lsz son los límites de la zona Z}
{cliz, clsz son los límites de la zona CZ}
para k=1 hasta nd hacer
     si liz(k) <> cliz(k) entonces
        si dist (p(k),liz(k)) < radio entonces
           SOLESFEX_BD=verdadero; retornar
        fin si
     fin si
     si lsz(k) <> clsz(k) entonces
        si dist (p(k),lsz(k)) < radio entonces
           SOLESFEX_BD=verdadero; retornar
        fin si
     fin si
fin para
SOLESFEX_BD=falso; retornar
```

b.1) CZ incluye a Z, y Zp es una zona exterior a CZ.

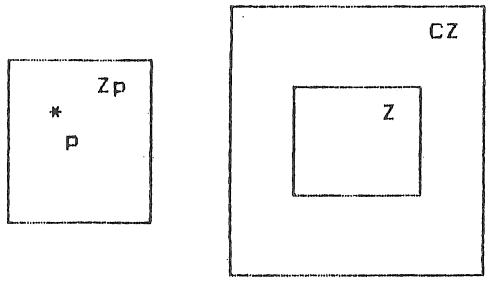


Figura 4a

b.2) CZ incluye a Z, y Zp es mimplemente el exterior de CZ.

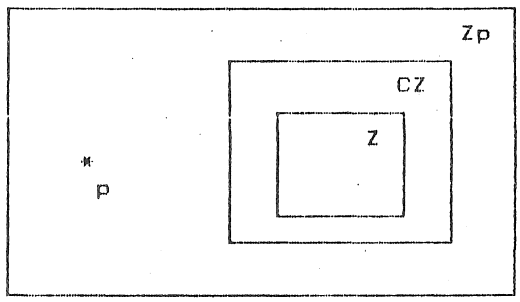


Figura 4b

Para determinar el solapamiento de la esfera con la parte exterior de la zona hay que tener en cuenta la posibilidad de que existan límites comunes entre las zonas Z y CZ.

En el caso de la figura 4a, siempre que Z y CZ no tengan ningún límite común, o en el caso de tenerlo, si éste no está hacia la parte exterior en la que se encuentra el punto, como se muestra en la siguiente figura 4c, el solapamiento está garantizado.

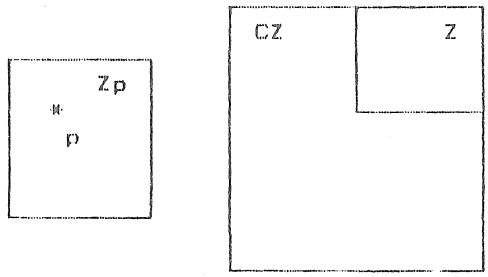


Figura 4c

En cualquier otro caso se presentan situaciones análogas a las de la figura 4d.

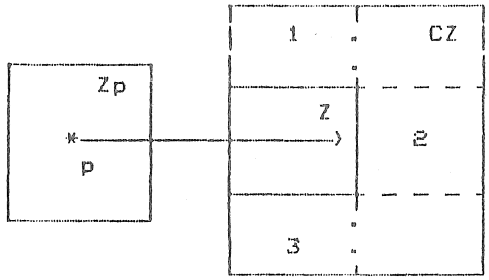


Figura 4d

La existencia de solapamiento se verifica dividiendo la parte exterior de Z en subzonas como se indica en la figura 4d, y estudiando el solapamiento interior a cada una de ellas a través de la función SOLESFIN_BD. Alguna de las zonas puede no existir, debido a que Z y CZ tengan límites comunes.

El estudio del solapamiento en el caso de la figura 4b es idéntico al de la figura 2a. La situación para límites comunes se presenta en la figura 4e.

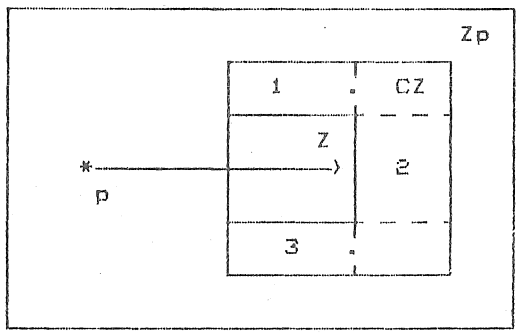


Figura 4e

c) Zp y Z no se contienen y están ambas incluídas en CZ.

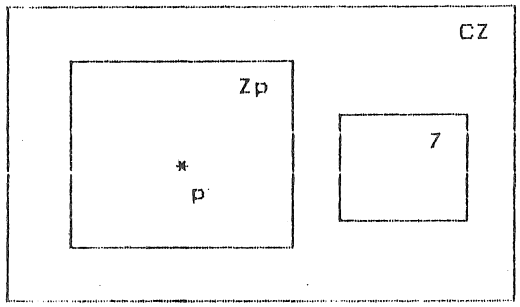


Figura 5a

En el caso de la figura 5a, siempre que Z no tenga límites comunes con Zp, el solapamiento está garantizado; si esto no ocurre, se presentan situaciones análogas a las de la figura 5b.

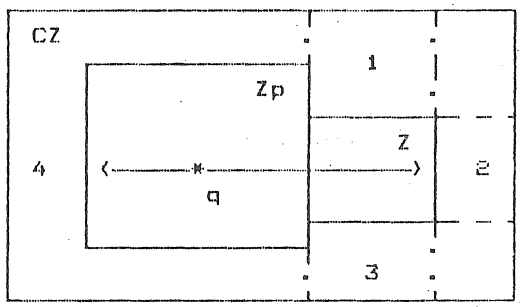


Figura 5b

Una estrategia para verificar la existencia de solapamiento consiste en dividir la parte exterior de Z en subzonas como se indica en la figura 3b, y sondear el solapamiento interior con 1, 2 y 3, a través de la función SOLESFIN_BD, y el solapamiento exterior en 4, a través de la función SOLESFEX_BD.

1.3 Alternativas de descenso

Se realiza el recorrido de los subárboles alternativos al camino de bajada inicial. El descenso en cada subárbol se hace de forma recursiva, probando en cada nivel el solapamiento con el interior. Si se verifica se desciende por la rama interior y cuando retorna se prueba el solapamiento con el exterior de ese interior, para decidir si se desciende por la rama exterior o si se retorna al nivel de recursividad anterior. En caso contrario, el recorrido continúa por el exterior sin necesidad de probar el solapamiento correspondiente, ya que el procedimiento fue invocado, la primera vez, al existir solapamiento con la parte del árbol que se está rastreando.

```
procedimiento DA_BD (nodo,lizI(j),lszI(j),cliz(j),clsz(j):j=1..nd)
{lizI, lszI son los límites de ZI}
{cliz, clsz son los límites de la zona a que se desciende}
si nodo es celda entonces
   LVMP (nodo)    {realiza el mantenimiento de la lista}
                  si no
   liz(j)=cliz(j) j=1..nd
   lsz(j)=clsz(j) j=1..nd
   CALCULO_LIMITES_DE_ZONA (liz(j),lsz(j):j=1..nd)
   si SOLESFIN_BD (liz(j),lsz(j):j=1..nd) entonces
      DA_BD (hijoizdo de nodo,lizI(j),lszI(j),liz(j),lsz(j):j=1..nd)
     si SOLESFEXIN_BD (liz(j),lsz(j),cliz(j),clsz(j):j=1..nd) entonces
        DA_BD (hijodcho de nodo,lizI(j),lszI(j),cliz(j),clsz(j):j=1..nd)
      fin si
                                              si no
      DA_BD (hijodcho de nodo,lizI(j),lszI(j),cliz(j),clsz(j):j=1..nd)
 fin si
fin si
retornar
```

1.4 Procedimiento de búsqueda de los m-vecinos más próximos

Este procedimiento realiza el recorrido inicial descendiendo recursivamente por el árbol hasta alcanzar la celda donde pudiese estar el punto. En cada retorno al nivel de recursividad anterior, se prueba la posibilidad de descenso alternativo a través de los solapamientos con el exterior o el interior, según la rama opuesta al descenso inicial, con el fin de actualizar la lista de los m_vecinos más próximos.

```
procedimiento BUSVMP_BD (nodo,zi(j),zs(j):j=1..nd)
{nodo  nodo interno testeado actualmente}
{zi, zs son los límites de la zona, dentro de la cual está la petición}
nzi(j)=zi(j) j=1..nd
nzs(j)=zs(j) j=1..nd
CALCULO_DE_LOS_LIMITES_DE_ZONA (nzi(j),nzs(j):j=1..nd)
si la petición es interior a la zona de nodo entonces
   si hijoizdo de nodo es celda entonces
      LVMP (hijoizdo de nodo)   {realiza el mantenimiento de la lista}
                                si no
      BUSVMP_BD (hijoizdo de nodo,nzi(j),nzs(j):j=1..nd)
   fin si
   si CONESF_BD (nzi(j),nzs(j):j=1..nd) entonces
      retornar al programa principal
                                          si no
      si SOLESFEX_BD (nzi(j),nzs(j),zi(j),zs(j):j=1..nd) entonces
         DA_BD (hijodcho de nodo,nzi(j),nzs(j),zi(j),zs(j):j=1..nd)
      fin si
   fin si
                                                si no
   si hijodcho de nodo es celda entonces
      LVMP (hijodcho de nodo)
                                si no
      BUSVMP_BD (hijodcho de nodo,zi(j),zs(j):j=1..nd)
   fin si
   si SOLESFIN_BD (nzi(j),nzs(j):j=1..nd) entonces
      DA_BD (hijoizdo de nodo,zi(j),zs(j),nzi(j),nzs(j):j=1..nd)
   fin si
fin si
retornar
```

2) ESTUDIO EXPERIMENTAL DE LA PETICION DE LOS M-VECINOS MAS PROXIMOS

El tiempo de respuesta a una petición de los m_vecinos más próximos se mide en función de los siguientes parámetros:

a) número de nodos internos accedidos, NIA.
g) número de celdas visitadas, CV.
j) número de distancias calculadas, DISTC.
k) número de solapamientos y contenciones estudiados.

Las pruebas se realizaron con distintos árboles de 10.000 registros cada uno, teniendo en cuenta las variaciones de dimensionalidad, nd, y tamaño de la celda, tmc.

Las peticiones se generaron de forma aleatoria, con la condición de que no existiesen en el árbol. Inicialmente, para cada árbol se realizaron búsquedas de 5_vecinos más proximos, con las distancias euclídea e infinito y se estudiaron posteriormente los resultados variando este número de vecinos, según los valores 10, 15, 20 y 25. En cada caso se realizaron 2000 pruebas y todos los resultados se dan como promedio.

Una de las cuestiones planteadas era la utilidad de un test de solapamiento exterior de interior. El test decide si se debe bajar por el exterior y por lo tanto ahorra recorrido por el árbol, pero a costa de cálculos adicionales en el nodo. En la figura 6 se muestran los resultados de los tiempos promedios por petición de 5_vecinos más próximos para la comparación de la búsqueda con, los test de solapamiento interior, exterior y exterior de interior (I+E+EI), y sin el test exterior de interior (I+E), con distancia euclídea, frente a la variación de tmc. Obteniéndose un mejor resultado para la primera, debido al ahorro progresivo de distancias calculadas y a la disminución de la diferencia en tiempos de solapamientos y contenciones, a medida que aumenta tmc, como se observa en las figura 7.

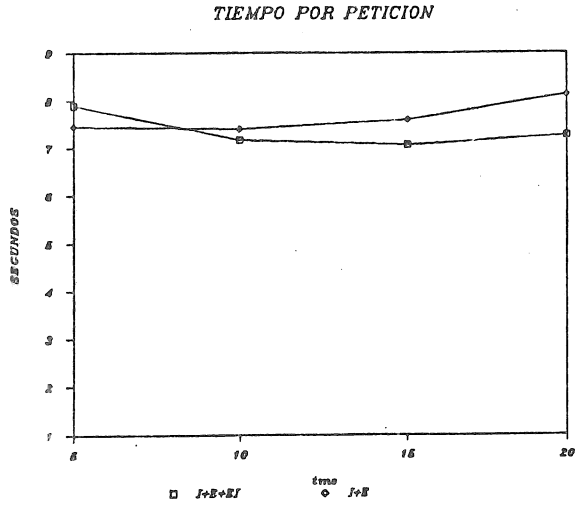


Figura 6

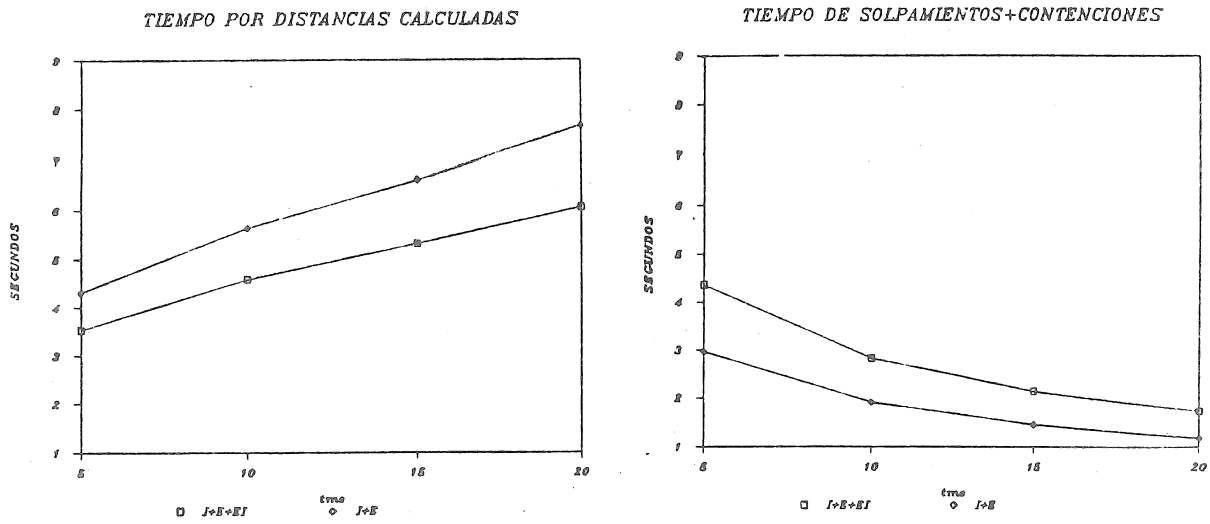


Figura 7

En un espacio nd_dimensional con la distancia euclídea, el conjunto de puntos que distan de uno fijo menos que un radio dado, geométricamente representan una hiperesfera; si la distancia considerada es la infinita se tendría un hipercubo. A igual radio la hiperesfera está inscrita en el hipercubo.

- En la búsqueda de los vecinos más próximos se obtienen obviamente hipercubos que corresponden a un radio menor o igual que el de las hiperesferas. Por ejemplo para dimensión dos se comprueba en la figura 8 que la relación que existe entre los radios es: reuclídea/raiz(2) (= rinfinito (= reuclídea. Los resultados experimentales, figura 9, muestran una mayor proximidad de rinfinito a reuclídea/raiz(2).

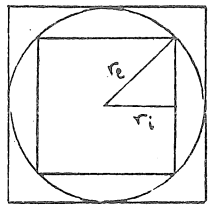


Figura 8

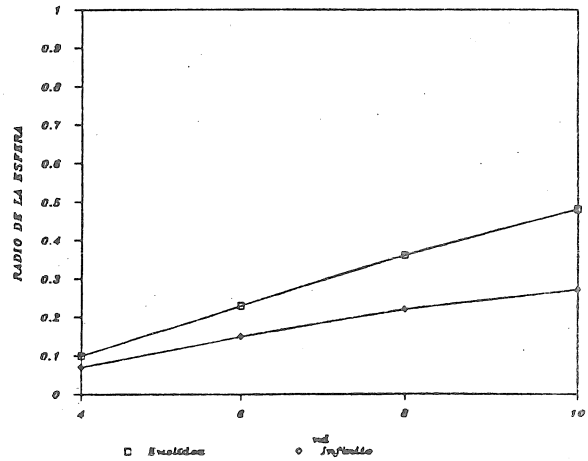


Figura 9

Esta relación entre los radios indica que el volumen del hipercubo es menor que el de la hiperesfera. Por otra parte, el hipercubo es más adaptable a la división del espacio en zonas, que se realiza en el árbol_BD. Esto explica el mejor comportamiento de la distancia infinito frente a la euclídea.

- Dado tmc, para cada métrica, el incremento de la dimensionalidad implica un incremento, figuras 10 y 11, en el número de nodos internos accedidos y en el de celdas visitadas, ya que para cada una de ellas aumenta el número de colindantes.

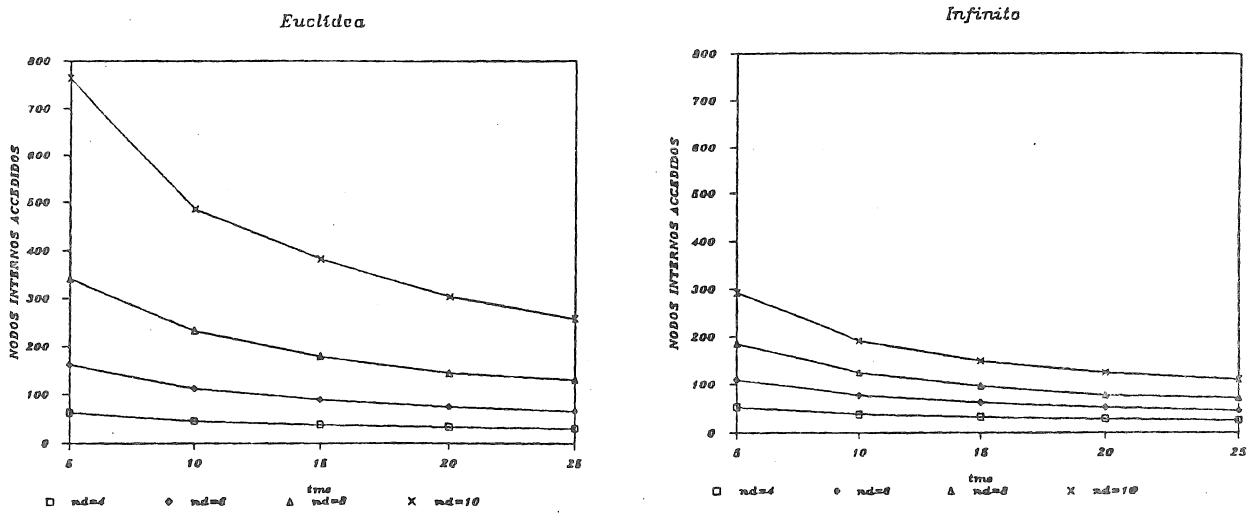


Figura 10

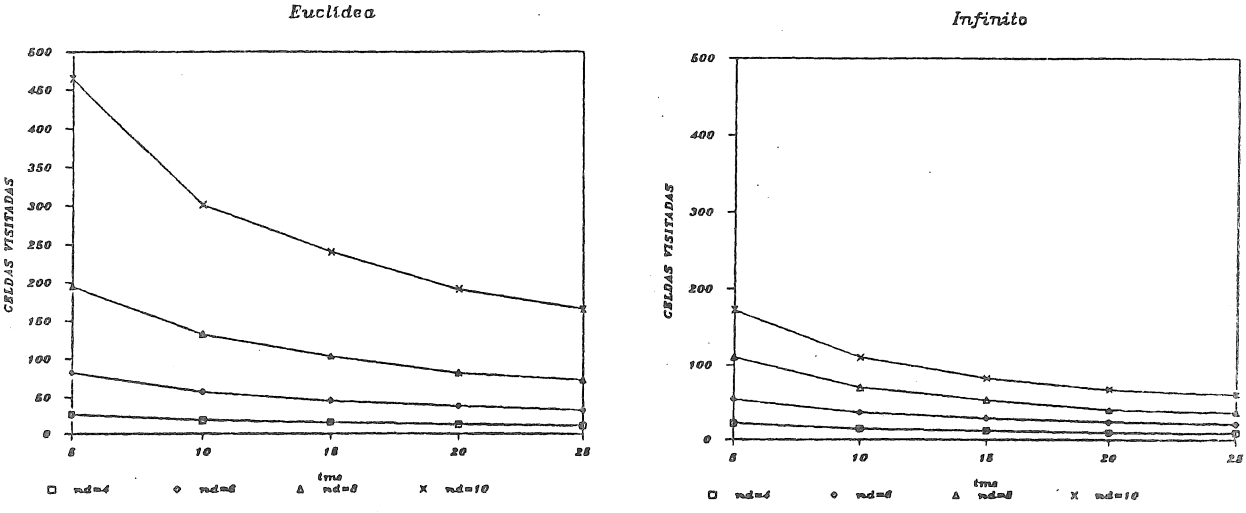


Figura 11

- El número de celdas visitadas decrece al aumentar tmc, debido a que las celdas contienen más puntos, por consiguiente el número de nodos internos accedidos también decrece.

- El número de distancias calculadas está en correlación con el tmc y nd, como se observa en la figura 12.

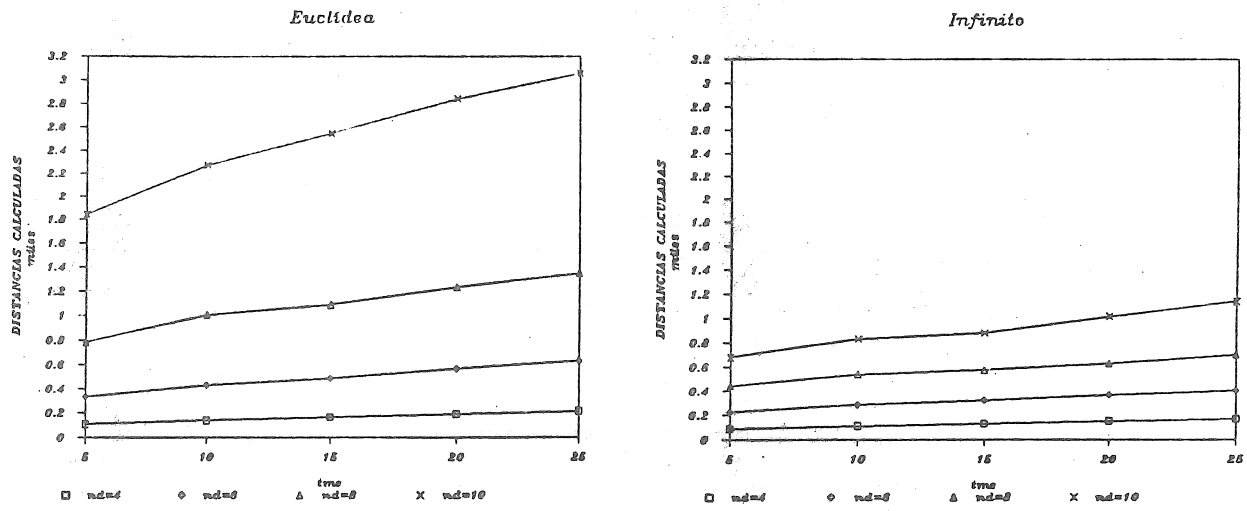


Figura 12

- El número de solapamientos con el interior, de las zonas representadas por los nodos internos, obviamente, tiene el mismo comportamiento que el número de nodos internos accedidos al variar nd y tmc para cada distancia, respectivamente. Las diferencias son mínimas y se deben a los nodos del camino de bajada inicial en los que se comprueba la contención de la esfera y a aquellos otros que están situados en niveles superiores al nodo en el que se verifica la contención.

- El número de solapamientos exteriores de interior tiene un comportamiento análogo al de los solapamientos con el interior, la diferencia está en que a los nodos del camino de bajada inicial no se les sondea este tipo de solapamiento y tampoco a los nodos de las alternativas de descenso para los que el solapamiento con el interior es falso.

- El número de solapamientos exteriores coincide con el numero de contenciones falsas de la esfera y los resultados obtenidos, figura 13, están en correlación con las alturas promedio de los árboles considerados al variar tmc. El aumento de la dimensionalidad retrasa la contención de la esfera y por tanto favorece el incremento ligero de los solapamientos exteriores.

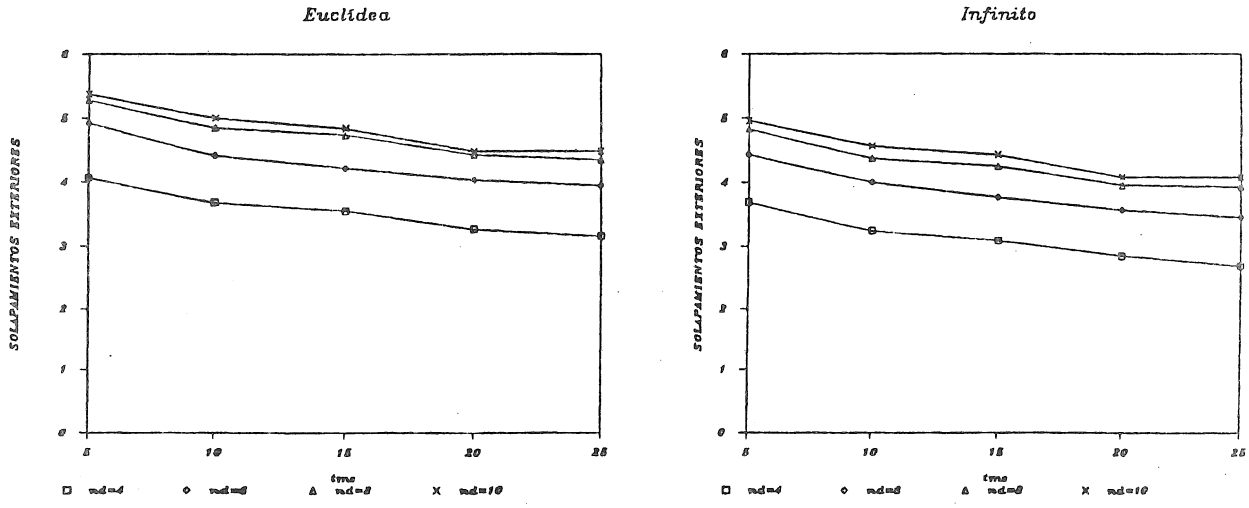


Figura 13

El incremento del número de **vecinos más próximos** provoca un aumento de todos los parámetros medidos, como se observa en la figura 14 con nd=8. La variación de la **dimensionalidad** provoca un incremento de los parámetros de acuerdo con los resultados comentados para cinco **vecinos más próximos.**

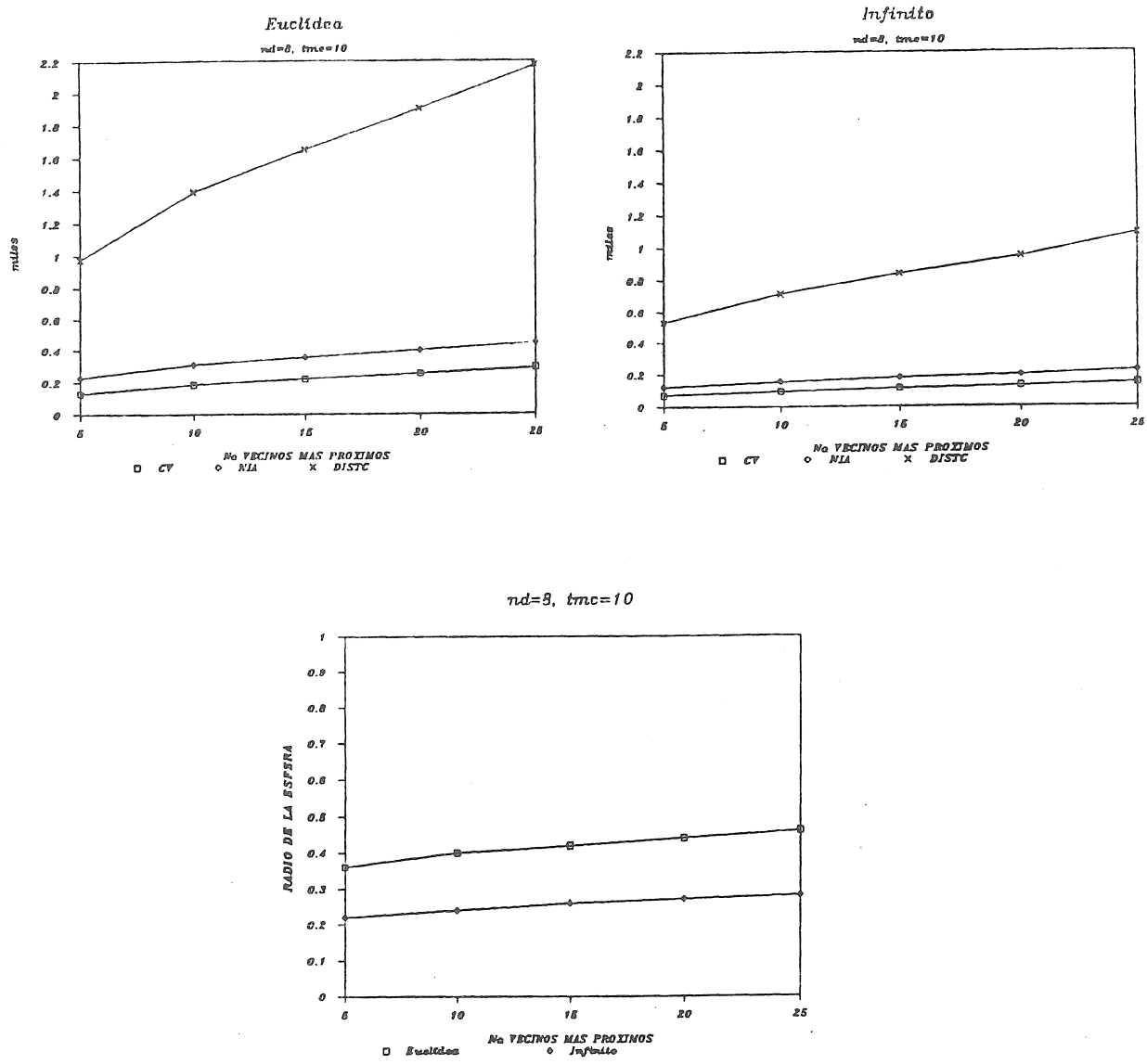


Figura 14

3) CONCLUSIONES

Se ha de citar en primer lugar que se demostró la utilidad de usar el test de solapamiento exterior de interior, al menos para tamaños máximos de celda mayores que 5, dado que el ahorro en el tiempo utilizado en calcular distancias es mayor que el tiempo perdido en realizar los test indicados. Asimismo, la distancia infinito mostró siempre mejor comportamiento que la euclídea. En cuanto a los parámetros medidos, debe indicarse que tanto el número de nodos internos accedidos, como el número de celdas visitadas, disminuyen al aumentar el tamaño máximo de celda y crecen al aumentar la dimensionalidad. El número de distancias calculadas esta en correlación con el tamaño máximo de celda y la

dimensionalidad. Tanto el número de tests de solapamiento con el interior como el de exterior de interior tienen comportamientos muy similares al del número de nodos internos accedidos. El número de tests de solapamiento con el exterior y el de contenciones de la esfera están en correlación con las alturas promedio y con la dimensionalidad. Por último, el aumento del número de vecinos a localizar provoca un aumento de todos los parámetros medidos.

BIBLIOGRAFIA

[1] DANDAMUDI, S.P. AND SORENSON, P.G.
Algorithms for the BD Tree Structure.
Dept. of Computational Science. University of Saskatchewan Saskatoon. Saskatchewan, Canada. 1984.

[2] DANDAMUDI, S.P. AND SORENSON, P.G.
An Empirical Performance Comparison of Some Variations of the K-d Tree and BD Tree.
International Journal of Computer and Information Sciences, Vol. 14, Nº 3, 1985.

[3] FRIEDMAN, J.H.; BENTLEY, J.L. AND FINKEL, R. A.
An Algorithm for Finding Best Matches in Logarithmic Expected Time.
ACM Transactions on Mathematical Software. Vol. 3, Nº 3:209-226, 1977.

[4] OHSAWA, Y. AND SAKAUCHI, M.
The BD-Tree-A New N-Dimensional Data Structure with Highly Efficient Dynamic Characteristics.
Institute of Industrial Science, University of Tokyo 22-1, Roppongi 7, Minato-ku, Tokyo 106, Japan. 1983.

[5] O.SANTANA, O. MAYOR, M. DIAZ, G. LOPEZ
Comportamiento del Árbol_BD en las fases creciente, decreciente y estacionaria.
E.U. Informática, Universidad Politécnica de Canarias. España. 1987.

An Algorithm for Dynamic Storage Allocation and Its Application to B$^+$-trees

Ricardo A. Baeza-Yates
Data Structuring Group
Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1 *

July 10, 1987

Abstract

We derive a practical algorithm for dynamic storage allocation based on a theoretical result. We apply this algorithm to disk storage allocation of B$^+$-trees with partial expansions. Simulation results show that this algorithm works well for this case.

1 Introduction

The dynamic storage allocation problem consists in a storage of consecutive cells (memory) of size S and an allocation algorithm that answers requests for allocating new blocks or freeing blocks to the storage. Also a set of possible block sizes $\{b_1, b_2, \cdots, b_n\}$ is given, such that $1 \leq b_1 < b_2 < \cdots < b_n$.

The problem is to find a good strategy for the allocation algorithm to minimize the wasted space (*fragmentation*) in the storage. Given an upper limit M for the total number of occupied cells at any time, this problem can be stated as: What is the minimum number of cells $S(M, \{b_1, b_2, \cdots, b_n\})$ with which an allocation algorithm can answer any sequence of requests?

In [9] it was proved that the limit

$$\lim_{M \to \infty} \frac{S(M, \{b_1, b_2, \cdots, b_n\})}{M}$$

exists. We will call this limit $S(\{b_1, b_2, \cdots, b_n\})$. The case $b_j = 2^{j-1}$ was studied by Robson [10], who found that

$$S(\{1, 2, \cdots, 2^r\}) = 1 + \frac{r}{2}.$$

*This work was also supported by the University of Chile, Santiago, Chile

Krogdahl [6] extended this result for the case in which each of the ratios b_j/b_{j-1} is an integer, showing that

$$S(\{b_1, b_2, \cdots, b_n\}) = 1 + (1 - \frac{b_1}{b_2}) + \cdots + (1 - \frac{b_{n-1}}{b_n}).$$

In this paper we extend the latter result, showing that this result is also an upper bound for any set $\{b_1, b_2, \cdots, b_n\}$. From this we derive a practical algorithm for this problem. In the last section we apply this algorithm to the disk storage allocation of B$^+$-trees with partial expansions.

2 A General Upper Bound

The next theorem is based on [6].

THEOREM 2.1 *For any set of possible sizes $\{b_1, b_2, \cdots, b_n\}$ such that each b_i is an integer and $1 \leq b_1 < b_2 < \cdots < b_n$ then*

$$S(\{b_1, b_2, \cdots, b_n\}) \leq 1 + \sum_{i=2}^{n}(1 - \frac{b_{i-1}}{b_i}).$$

Proof: Let M be a multiple of $m = MCM(b_1, b_2^2, \cdots, b_n^2)$, where $MCM(a, b)$ means the minimum common multiple of a and b. We shall show that there is an algorithm that can meet any sequence of requests with a storage of at least size $M(1 + \sum_{i=2}^{n}(1 - \frac{b_{i-1}}{b_i}))$. Since we know that the limit exists, this will show that the statement of the theorem is true.

The storage is divided into n pieces, P_i, such that the size of each piece s_i is

$$s_i = \begin{cases} M & i = 1 \\ (1 - \frac{b_{i-1}}{b_i})M & 2 \leq i \leq n \end{cases}$$

Note that each s_i is a multiple of any b_j, for any i and j. A request for a b_j block is always placed in any one of the pieces $P_1, \cdots, P_j$ and always an integer multiple of its own length away from the boundary of the selected piece. This implies that the smallest block ever allocated in piece P_j is of size b_j. Assume that a b_k block cannot be allocated. Then each of the P_i $(i = 1, \cdots, j)$ pieces is occupied by at least b_i blocks. This then means $b_i P_i/b_k$ occupied cells in each P_i. Hence, the total number of occupied cells is

$$\frac{b_1}{b_k}M + \frac{(1 - \frac{b_1}{b_2})b_2}{b_k}M + \cdots + \frac{(1 - \frac{b_{k-1}}{b_k})b_k}{b_k}M = M.$$

This means that the request is illegal. ■

We were not able to prove that this upper bound is the exact result. However, taking in account that the problem is more difficult for a general set $\{b_1, b_2, \cdots, b_n\}$, we conjecture that this upper bound is in fact an equality.

Note that this upper bound in the amount of space gives a lower bound in the storage utilization (that is the reciprocal of S/M). As a special case, we have that if any integer size between 1 and n is allowed, then

$$S(1, 2, \cdots, n) \leq H_n = \ln n + O(1),$$

where $H_n = \sum_{i=1}^{n} 1/i$ are the harmonic numbers. This result proves half of the conjecture raised in [10].

3 The Allocation Algorithm

In practice, it is possible to implement the algorithm as presented in the previous section. However, the implementation could be difficult and expensive. In addition, there are some aspects of the algorithm not completely defined. For example, if there exist more than one place for a new block, which one we must use.

The first observation is that for the purposes of the proof, M is in general, a multiple of a very large number. This is not necessary, since each piece P_i is also a multiple of $B = MCM(b_1, b_2, \cdots, b_n)$. Then, instead of having n pieces of different sizes, we can divide the storage into many pieces of equal size B, and the allocation algorithm will be the same for each piece. The amount of initial storage available must be a multiple of B, and in general M will be much bigger than B. Note that we are not using the condition that a block of size b_i must be allocated in only some part of the storage. The reason for this is that without this condition the algorithm is simpler and on average the behaviour will be similar.

A request for a b_j block is placed in any of these pieces and always an integer multiple of its own length away from the boundary of the selected piece. In other words, the blocks are boundary aligned. A similar idea is used in the *best fit aligned* allocation method [4].

To find free space efficiently and to know if a request to free a block is valid, we need additional information. For this, we associate with each piece some status information that we will call the *state* of the piece. The possible number of states is the number of possible different allocation patterns in one piece. This number is less that 2^B, so we need less than B bits to represent the state of each piece, and therefore we need less than 1 bit for each cell of the storage. For example, if each cell is a 4-byte word, we need 1 bit for each 4 bytes.

There are two obvious ways to describe the states of all the pieces. The first is to have a bit map, storing sequentially the state of each piece. To find a free space we search linearly in the bit map for a piece in a state that has space for the block requested. To improve the worst case, it is possible to maintain n pointers to the first piece that has space for a block of size b_j. The search will be intrinsically linear, and the space needed will be $O(M)$, but with a small constant much less than 1.

The second way is to have a list for each state with the addresses of all the pieces in that state. To find free space we search in the lists of states having space for the requested block. In the worst case this search can be linear in the number of states, but usually will take $O(1)$. For this we need 2 words per piece, that is space $O(M/B)$.

Whatever we use for the description of the piece, the selection of the first adequate piece will be like a first fit algorithm. If we have more than one choice, we can have a best fit algorithm. In this case best fit will mean *best state*. For example we could choose the state more filled, or the state that has more blocks of the same size, etc. After choosing which piece, we need to decide where in that piece we put the new block if we have more than one choice. For example, the best fit aligned position. These *policies* for allocating a new block will depend on the application.

Having selected the location of the new block, we need to compute the new state of the piece. The same is true if we are freeing a block. There are 2 possibilities: we compute the new state or we have a transformation table which, given a state and an action, yields the new state.

The last problem, concerns to what happens if we cannot answer a request. We have two solutions. The first one is compaction. However, for compaction we need to know all the pointers to the pieces. This means that we need an additional data structure for translating the pointers of the application to our pointers. In that way we can move our pointers without change the pointers of the application. In general, compaction is expensive, and the results are often not worthwhile [4]. For this reason we recommend the next solution.

The second solution is to obtain more free space. This is very easy. We need only to request a piece of size B contiguous to our storage of size M. This means that our storage can grow incrementally, starting from one piece to many as we need. This also provides a way to use this algorithm over other storage environment. We have a linked list with all the current used pieces, and each time that we need more space we allocate a new block of size B. This is particularly suitable if we want to have a better allocation algorithm than the one provided by the environment; moreover we only need an storage environment that provides a good allocation strategy for blocks of a unique size. With the same structure, compaction is also possible, because the linked list is the structure that transforms the pointers of the application to our pointers.

This algorithm is a general setting for the problem of dynamic storage allocation with many possible sizes. We think that in many cases it is a good choice, and we demonstrate this in the next section for a particular case.

4 An Application

4.1 B^+-trees with Partial Expansions

A B-tree is a balanced multiway search tree. We define a B-tree of order m as follows:

1. The root has between two and $2m + 1$ descendents.
2. Any other internal node has between $m + 1$ and $2m + 1$ descendents.
3. All the leaves are at the same level.

In a B^+-tree all data records are stored at the lowest level (*buckets*), and the upper levels are a B-tree index to the data buckets [5]. All the buckets are of the same size. File growth

is handled by bucket splitting, that is, when a bucket overflows, an additional bucket is allocated and half of the records from the overflowing bucket are moved to the new bucket. The same method is applied to index nodes. The average storage utilization of a B$^+$-tree is asymptotically 69% [1].

To improve the storage utilization of the B$^+$-tree some overflow techniques have been proposed (see [5] for a summary). One of the best known are B^*-trees. In this case, if we have an overflow, first we check at its adjacent brothers for free space. If there is space, some records are shifted and a split is avoided. If not, it means that at least one other bucket is in overflow. Then, an additional bucket is allocated, and the records of the two overflowing buckets are distributed in three buckets. With this the minimum storage utilization is 67%. However, the insertion time is greater than the simple case.

In a recent paper [8], Lomet proposed the use of *elastic buckets*. The idea is simply to increase the size of an overflowing bucket instead of splitting it. Each expansion step is called a *partial expansion* and a bucket is expanded until it reaches some predetermined maximum size. When a bucket of maximum size overflows, it is split into two buckets (of minimum size) in the same way as for a standard B$^+$-tree. This process of a bucket gradually growing from its minimum size until it splits, is called a *full expansion*. A full expansion increases the number of buckets by one. The idea of partial expansions has previously been applied to Linear Hashing [7].

Let r be the number of expansion steps required to grow a bucket from its minimum size up to and including the final split. For simplicity we assume that records are of fixed length and measure the bucket size (capacity) in number of records. Let the r different bucket sizes be $s_0, s_1, ..., s_{r-1}$, where $s_0 < s_1 < ... < s_{r-1}$. We call this the bucket *growth sequence*. In principle, any strictly increasing sequence is a valid growth sequence, provided that $2s_0 \geq s_{r-1} + 1$. However, due to hardware and software limitations, the following growth sequence seems most practical [7]. Let a *page* be the smallest unit of transfer between disk and main memory, and a *page block* some fixed number of consecutive pages. Assume that the capacity of a page block is b records. For simplicity, we assume that b is integer. Then choose the growth sequence rb, $(r+1)b$, $\cdots$, $(2r-1)b$. In other words, the minimum bucket size is rb records and each partial expansion increases the bucket size by one page block, up to the maximum bucket size of $(2r-1)b$. The minimum possible storage utilization is $r/(r+1)$ (in the first expansion).

Example: For page blocks of size 5, and 3 partial expansions, the growth sequence is 15, 20, 25. When inserting the 26th record into a bucket, it splits into two buckets of size 15, each one containing 13 records. The minimum storage utilization for the various bucket sizes is $13/15 = 0.866...$, $16/20 = 0.8$, $21/25 = 0.84$.

For a complete analytical and empirical analysis of this trees see [2]. All the theoretical results mentioned in this section are from the previous reference. All the results are asymptotical in the number of records inserted in the tree (no deletions). For example, the asymptotic average storage utilization for large buckets for B$^+$-trees with r partial expansions is [2]

$$U = \frac{\ln 2}{H_{2r-1} - H_{r-1}}$$

and one advantage is that the insertion time is only a little more expensive than that of simple B$^+$-trees. However, the disadvantage is that disk space management is more difficult.

4.2 Disk Storage Management

For the disk space management we apply our algorithm. The basic cell unit will be a page block, and if we are using r partial expansions, the set of allowed blocks is $\{r, ..., 2r-1\}$. Hence, the piece size B will be the minimum common multiple of all these numbers. Clearly, $B \leq r(r+1)\cdots(2r-1)$. Now we are interested in the total storage utilization in the used pieces. Applying the lower bound of the first section, and using the minimum storage utilization in a bucket, we obtain a lower bound for the total storage utilization. This lower bound is the worst case of the original algorithm. Similarly, the average total storage utilization is bounded from above by the storage utilization in the B$^+$-tree. Hence, if U_t is the average total storage utilization, we have

$$\frac{r}{(r+1)(1+H_{2r-1}-H_r)} \leq U_t \leq U$$

For large r, this becomes

$$\frac{1}{1+\ln 2} + O(1/r) \leq U_t \leq 1 - O(1/r)$$

The value of the lower bound is 59%.

Which value of r is the best? From the point of view of internal storage utilization, r must be as large as possible. However, we have some drawbacks. First the size of B grows with r and then each new piece allocated significantly changes the storage utilization. In other words, the transient phase will be longer and unstable. Second, the external fragmentation grows as we have more sizes. We have selected $r = 2$ for testing the algorithm, as a compromise between storage utilization and the dynamic storage management.

With two partial expansions, we divide the disk storage in pieces of size $B = 6$ page blocks. When more space is needed for the file, an additional piece is requested from the operating system. A piece stores either one, two or three buckets. Denoting the relative bucket sizes by 2 and 3, the following classification of the states of a piece is sufficient for our purpose:

- empty, 2+0, 0+2+0, 0+2, 3+0, 0+3, 2+2+0, 2+0+2, 0+2+2, 2+3, 3+2, 2+2+2, 3+3

The notation 2+0 means that a bucket of size 2 occupies the two leftmost page blocks and the rest are free. 2+0+2 means that there is free space in the middle of the piece. An in-core table is used for keeping track of the state of each piece of the file. We must be able to distinguish between the states listed above. Hence this scheme requires 4-bit entries in the table (*bit map*). From this table we can find out not only whether a piece has room for a new bucket, but also the exact size and location of the free space within the piece. We can therefore write out a new bucket directly without first having to read in the piece. Even for a large file, the bit table is small enough to be kept in main memory. For example, if we have 50 million records, we will need approximately 335K pieces if a page block has

a capacity of 30 records (small bucket size 60). This means a bit map of approximately 170K. The fact that the search for free space is linear is not important given the difference in access time between primary and secondary memory.

Space may now be wasted because of internal and external fragmentation. Internal fragmentation refers to the space wasted within buckets. External fragmentation refers to the space wasted because pieces are not completely filled.

Now, we must define the allocation policy for this particular case. The algorithm searches for pieces with free space in a certain order. To speed up the search we maintain a pointer to the first piece in each state. The order used was (depending on the size requested)

Size 2: 0+2+0; 2+2+0 or 2+0+2 or 0+2+2; 2+0+0 or 0+0+2; 0+3 or 3+0

Size 3: 0+3 or 3+0; 0+0+2 or 2+0+0

That is, when space is needed for a bucket of size 2, we first try to find a piece in state 0+2+0. If no such pieces exist, we try to find a piece in one of the states 2+2+0, 2+0+2, or 0+2+2, and so on. When we find space we can answer the request immediately, and we can update the pointers in parallel with the system.

4.3 Analysis

In general, the analysis of dynamic storage allocation algorithms is very difficult (see [3]). However, in this case we have more information. We only have two types of events. If a block of size 2 is freed, then a block of size 3 will be requested. If a block of size 3 is freed, then two blocks of size 2 will be requested.

With this, it is possible to obtain a good lower bound for the average total storage utilization. For this, we follow the same allocation schema, with the restriction that all the requests are resolved within the same piece. If there is no more space in a piece, then a new piece is allocated. The result will be a lower bound, because we not use the free space available in other pieces.

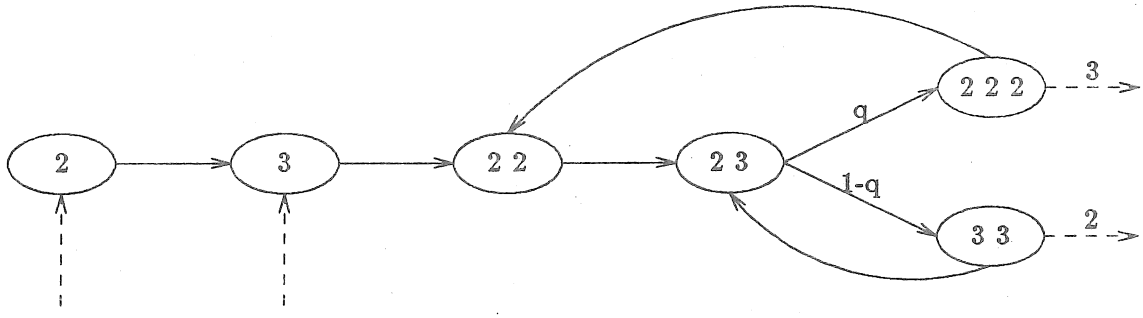


Figure 1: Transition Diagram of the Allocation Process in a piece

Figure 1 shows the transition diagram that represents the possible states of a piece and the allocation process. The dashed arrows represent the creation of new pieces. The probability q represents how frequently a block of size 3 is transformed into two of size 2

(split). Hence, the probability of a partial expansion is $1-q$. We will reach a steady state, asymptotically in the number of records inserted. If p_i is the probability of a piece being in state i we have

$$p_2 = 0, \quad p_3 = p_2, \quad p_{22} = p_3 + p_{222}, \quad p_{23} = p_{22} + p_{33}, \quad p_{33} = (1-q)p_{23}, \quad p_{222} = q\, p_{23}$$

and, of course,

$$p_2 + p_3 + p_{22} + p_{23} + p_{33} + p_{222} = 1.$$

We need another equation to obtain the value of q. For this, we use the ratio between buckets of size 2 and buckets of size 3. Let $f_2(b)$ and $f_3(b)$ be the asymptotic fraction of buckets of size 2 and 3, respectively, when the page block size is b. Then

$$\frac{p_2 + 2p_{22} + p_{23} + 3p_{222}}{p_3 + p_{23} + 2p_{33}} = \frac{f_2(b)}{f_3(b)} = g(b).$$

From [2] we have

$$f_2(b) = \frac{b+1}{2b+1} \quad \text{and} \quad f_3(b) = \frac{b}{2b+1}.$$

So $g(b) = \frac{b}{b+1}$. Solving the system we obtain

$$p_2 = 0, \; p_3 = 0, \; p_{22} = \frac{2b+3}{16b+7}, \; p_{23} = \frac{7b+2}{16b+7}, \; p_{33} = \frac{5b-1}{16b+7}, \; p_{222} = \frac{2b+3}{16b+7}$$

and $q = \frac{2b+3}{7b+2}$. For large bucket sizes (or page block sizes) q converges to $2/7$.

Let $u_2(b)$ and $u_3(b)$ be the asymptotic storage utilization on buckets of size 2 and 3 respectively. Then, the total storage utilization in one piece is bounded by

$$U_t \geq (p_2 + p_{22} + \frac{p_{23}}{3} + p_{222})u_2(b) + (p_3 + \frac{p23}{2} + p_{33})u_3(b)$$

$$= \frac{(38b+40)u_2(b) + 51bu_3(b)}{6(16b+7)}$$

As already mentioned, an upper bound on the total storage utilization is the internal storage utilization of the buckets. For $r = 2$ [2] this value for a page block size b is

$$U_t \leq \frac{(2b+1)(3b+1)}{b(5b+2)}\alpha(b)$$

with

$$\alpha(b) = \begin{cases} H_{3b+1} - H_{(3b+1)/2} & b \text{ odd} \\ H_{3b} - H_{3b/2} & b \text{ even} \end{cases}.$$

From [2] we know that

$$u_2(b) = \begin{cases} \frac{(2b+1)(3b+1)}{2b(b+1)}(H_{2b+1} - H_{(3b+1)/2}) & b \text{ odd} \\ \frac{(2b+1)(3b+1)}{2b(b+1)}(H_{2b+1} - H_{3b/2} - \frac{1}{3b+1}) & b \text{ even} \end{cases}$$

and

$$u_3(b) = \frac{(3b+1)(2b+1)}{2b^2}(H_{3b+1} - H_{2b+1}).$$

Hence, for large bucket sizes we have

$$\frac{21}{16}\ln 2 - \frac{1}{8}\ln 3 + O(1/b) \leq U_t \leq \frac{6}{5}\ln 2 + O(1/b)$$

or

$$.7724 + O(1/b) \leq U_t \leq .8318 + O(1/b).$$

4.4 Simulation Results

In this section, simulation results are presented. Figure 2 shows the development of the average total storage utilization for smallest bucket size 60. The solid line represent averages from 100 simulated file loadings. The top line represents the internal storage utilization (upper bound) of the B$^+$-tree with 2 partial expansions. The bottom line represents the theoretically expected storage utilization for a standard B$^+$-tree with bucket size 60. Initially the file consisted of completely filled smallest buckets. This initial state was deliberately chosen to see how the schemes perform when there is heavy expansion and splitting activity, and when there is a significant excess of buckets of one type.

It is apparent from figure 2 that the fragmentation is very little, and that the total storage utilization out performs the simple B$^+$-tree. For this algorithm, the lowest storage utilization occurs when there has been an excess of bucket of size 3 and they start splitting. When there is an excess of buckets of size 3, there will be many buckets in state 3+3. When these buckets start splitting, a significant fraction of buckets in state 2+3 (or 3+2) will be created, wasting 1/6 of a piece. This is further exacerbated by the low internal storage utilization in the newly created buckets of size 2.

Table 1 shows simulation results and the theoretical upper bounds for large files and two different smallest bucket sizes. Note that the upper bound cannot be achieved without relocation of buckets. In the steady state there is still expansion and splitting activity going on. When a bucket is expanded or split, a "hole" may be created which then will persist for some time until it is eventually filled. At any given point in time, a few such "holes" may exist. To fill a "hole" immediately would require relocation of some existing bucket.

Smallest bucket size	Lower Bound	Simulation results (100 runs)	Upper bound
12 (20000 records)	0.7970	0.82800 ± 0.00071	0.85696
60 (150000 records)	0.7776	0.81020 ± 0.00056	0.83706

Table 1: Total storage utilization for large files (95% confidence interval)

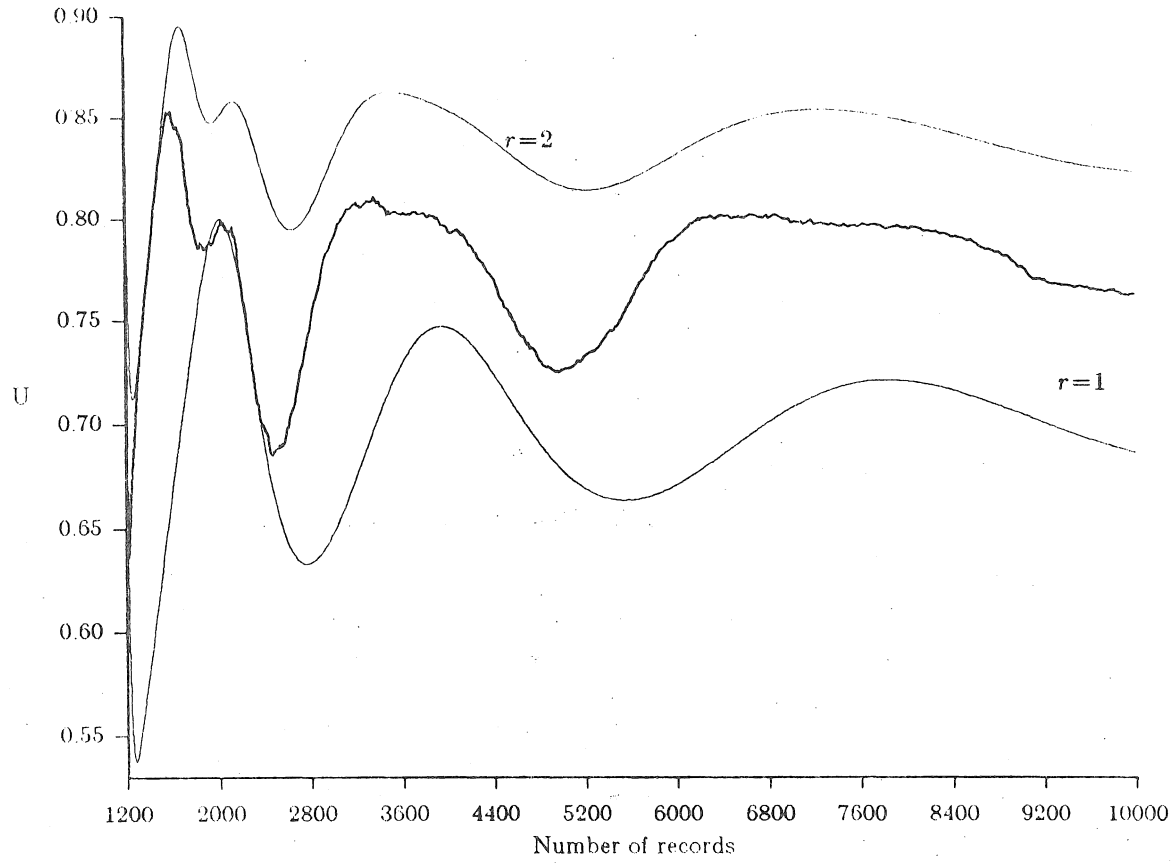


Figure 2: Average total storage utilization for smallest bucket size 60

The total storage utilization is over 80%, which is more than 10 percentage points higher than the storage utilization of a standard B^+-tree. The results are close to the upper bound, which indicates that there is little to be gained by bucket relocation.

5 Conclusions

We have presented a general algorithm for the dynamic storage allocation problem. This algorithm can be adapted for many different problems, and can be used in an environment that provides storage allocation for blocks of a unique size.

We have applied this algorithm to a particular case: B^+-trees with partial expansions. Simulation results confirm the fact that the storage utilization obtained with this algorithm is almost optimal. As a further conclusion, B^+-trees with partial expansions using this allocation algorithm for secondary storage is the better implementation of B^+-trees, to the best of our knowledge. The main reason being that it provides a storage utilization of around 81% without increasing the insertion time [2]. On the other hand, B^*-trees provide

the same storage utilization, but double the insertion time [1].

Acknowledgements

I wish to thank Jorge Olivos for starting my interest in this problem, Per-Åke Larson for suggesting the application, Jose Pino for encouraging me to write this paper, and Tony Gahlinger for his helpful comments.

References

[1] Baeza-Yates, R., The Expected Behaviour of B^+-trees, Department of Computer Science, University of Chile, Santiago, Chile, 1985. Also available as Technical Report CS-86-68, Dept. of Computer Science, University of Waterloo, 1986.

[2] Baeza-Yates, R. and Larson, P-Å. Analysis of B^+-trees with Partial Expansions, Research Report CS-87-04, Dept. of Computer Science, University of Waterloo, Waterloo, Canada.

[3] Coffman, E. "An Introduction to Combinatorial Models of Dynamic Storage Allocation", *SIAM Review* 25, 3 (1983), 311-325.

[4] Coffman, E. and Leighton, F. "A Provably Efficient Algorithm for Dynamic Storage Allocation", Proceedings of the 18^th annual ACM Symposium on Theory of Computing, Berkeley, California (1986), 77-90.

[5] Comer, D., The Ubiquitous B-Tree, *Computing Surveys* 11, 2 (1979), 121-137.

[6] Krogdahl, S. "A Dynamic Storage Allocation Problem", *Information Processing Letters* 2 (1973) 96-99.

[7] Larson, P-Å. Linear Hashing with Partial Expansions, Proc. 6th Int. Conf. on Very Large Data Bases, (Montreal, Canada, 1980), 224-232.

[8] Lomet, D., Partial Expansions for File Organizations with an Index, *ACM Trans. on Database Systems* 12 (1987), 65-84.

[9] Robson, J. "An Estimate of the Store Size Necessary for Dynamic Storage Allocation", *Journal of the ACM* 18 (1971) 416-423.

[10] Robson, J. "Bounds for Some Functions Concerning Dynamic Storage Allocation", *Journal of the ACM* 21 (1974) 491-499.

ESTRUCTURA DE DATOS PLEX

Ardenghi, Jorge
Cicileo, Guillermo
Rueda, Sonia
Zanconi, Marcelo

CRIBABB-PLAPIQUI 12 de Octubre 1865
(8000) Bahía Blanca República Argentina

Resumen

La mayor parte de los problemas encierran en si mismos una estructura. Así, un problema puede ser especificado en término de los objetos que lo caracterizan, sus relaciones y las operaciones que se ejecutan sobre ellos.

Una característica deseable de un programa, es que respete, en la medida de lo posible, la estructura del problema que intenta resolver.

La estructura de datos Plex permite la representación de los datos de un problema consistente de entidades, agrupadas en categorías y caracterizadas por atributos que reciben valores dentro de un dominio.

La aplicación de la estructura Plex provee gran flexibilidad, ya que, aunque las categorías y los dominios son fijos para un problema dado, es posible agregar nuevas entidades con una estructura propia.

Por otra parte la introducción de Tipos de Datos Abstractos permite el oscurecimiento de información a través del encapsulamiento de datos.

Estructura de Datos Plex

1.0 Estructuras de Datos

Una estructura de datos es una colección de items de datos organizados de alguna manera, junto con las operaciones para acceder a estos datos.

Una estructura de datos puede definirse en varios niveles :

- A nivel abstracto o lógico la estructura se define independientemente de una representación particular, sin connotaciones físicas. Las operaciones se definen desde el punto de vista de que es lo que hacen y no de como lo hacen.

- A nivel operativo o de aplicación la estructura se define en términos del tipos de problemas a los cuales se adapta. Es la parte más cercana al usuario de la estructura. En ocasiones se escribe una interfase, de forma tal que el usuario tiene acceso a las operaciones sobre la estructura a través de operaciones más cercanas a su aplicación particular.

- A nivel físico o de implementación la estructura se define en términos de otras estructuras más simples, creadas posiblemente a partir de mecanismos y facilidades provistos por el lenguaje. Esta descripción puede dividirse a su vez en subniveles, de forma tal que los niveles superiores son aún algo abstractos, mientras que los inferiores son mucho más cercanos a la implementación particular y a las primitivas provistas por el lenguaje. Las operaciones se definen como procedimientos que manipulan a la estructura en términos de operaciones primitivas.

1.1 Definicion De Plex A Nivel Logico

Informalmente se puede pensar a la estructura Plex como un conglomerado de celdas, conjunto de datos con cohesión conceptual, agrupadas de alguna manera. Las celdas varían en número y tipo de los datos constituyentes llamados componentes. Cada celda puede estar asociada con cero o más celdas de la misma u otra agrupación.

Más formalmente se dice que a nivel lógico la estructura de datos Plex se define como una estructura dinámica, heterogénea, no lineal y multiasociada.

Dinámica: porque puede crecer y contraerse tanto en el número de elementos como en sus tamaños.

Heterogenea: porque los elementos son potencialmente distintos en número y

tipo de componentes y pueden estar agrupados en subestructuras.

No lineal: porque no existe una relación de orden entre sus elementos.

Multiasociada: porque cada elemento puede estar asociado a cero o más elementos.

Se da ahora una definición más **operativa** de la estructura, apuntando al tipo de problemas a que se adapta.

1.2 Definicion De Plex A Nivel Aplicacion

La mayor parte de los problemas encierran en si mismos una estructura.

Es deseable, aunque no siempre posible, que cuando los datos referidos a un problema se almacenan en una computadora, éstos esten organizados respetando la estructura inherente al problema.

Una estructura de datos plex permite almacenar la descripción de un problema caracterizado por:

1) Objetos distinguibles que llamaremos **entidades**.

2) Las entidades pueden estar agrupadas en **categorías** cuando representan a objetos de un mismo tipo.

3) Una entidad está caracterizada por ciertas propiedades denominadas **atributos**.

4) Cada atributo toma valores dentro de un **dominio**.

5) Un atributo común a todas las entidades es el **nombre** que permite identificarla.

6) Dos o más entidades pueden estar asociadas a un mismo conjunto de atributos, en especial si pertenecen a una misma categoría, pero en general tendrán distintos valores para uno o más atributos.

7) En general no existen dos entidades de distintas categorías asociadas un mismo conjunto de atributos.

8) Las entidades pueden estar **asociadas** a entidades de la misma y/o de otras categorías.

1.3 Definicion De Plex A Nivel Implementacion

La implementación de una estructura de datos compleja puede presentarse en distintos niveles de abstracción. En los niveles más altos se describe a la estructura en términos de otras estructuras más simples. Sin embargo si estas estructuras no son soportadas por el lenguaje de implementación, es posible que sea necesario definirlas a su vez en términos de su representación física, y asi siguiendo hasta alcanzar primitivas del lenguaje.

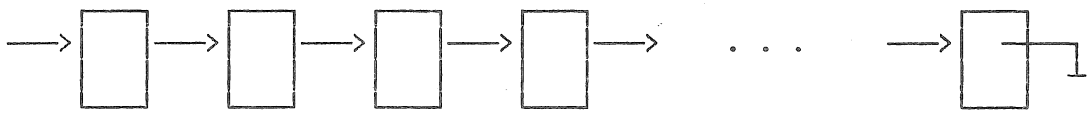
Se verá entonces una implementación posible para la estructura de datos plex, utilizando estructuras de datos más simples como listas y tablas hash, y se describirá luego como estas estructuras pueden representarse en Pascal.

1.3.1 Implementación en términos de otras estructuras

En una estructura de datos plex la información asociada a una entidad es almacenada en una **celda**. La celda es entonces, el elemento primitivo a partir del cual se construye la estructura.

Las celdas asociadas a entidades de una misma categoría están organizadas dentro de la estructura como una **lista enlazada**.

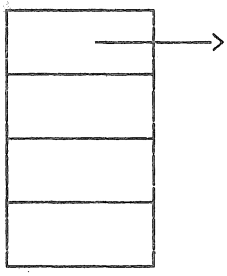
Categoria



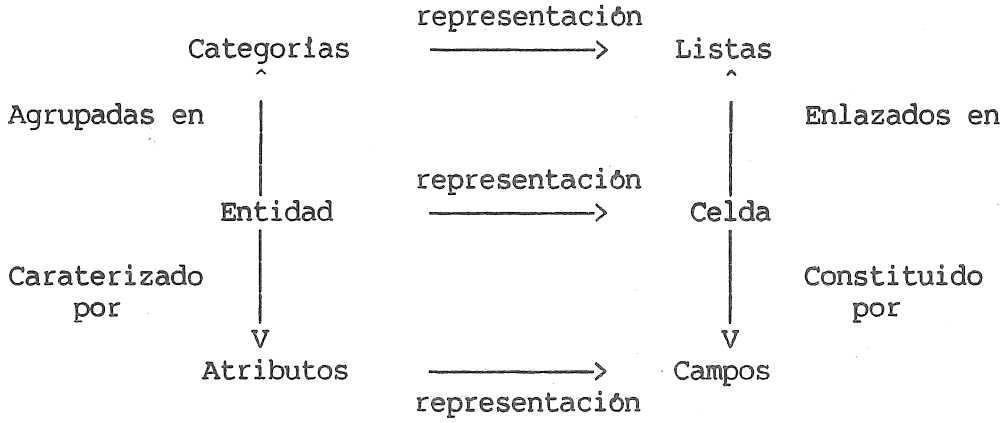
Entidad 1 Entidad 2 Entidad 3 Entidad 4 Entidad n

Una celda esta constituida por campos, cada uno de los cuales corresponde a un atributo de la entidad asociada a la celda, y por lo tanto recibe valores dentro de un dominio.

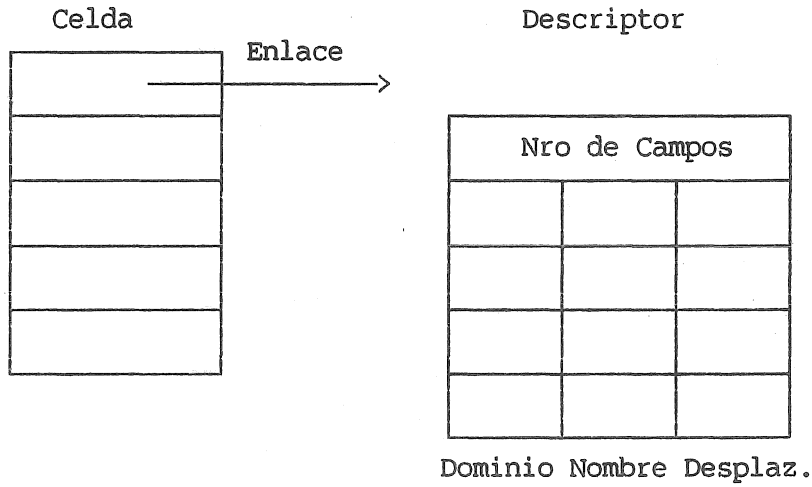
Celda



Gráficamente:

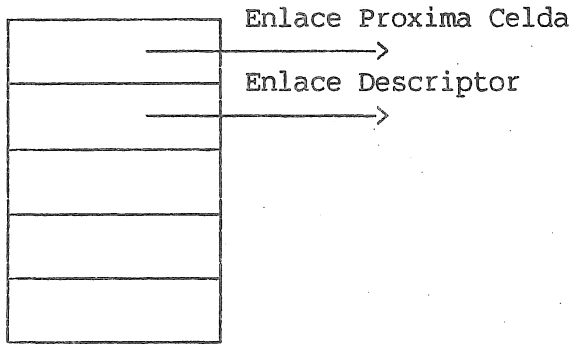


La estructura de una celda está definida a través de un descriptor asociado a la celda. Este permite almacenar el número de campos de la celda, y el dominio, nombre y desplazamiento de cada uno de ellos :



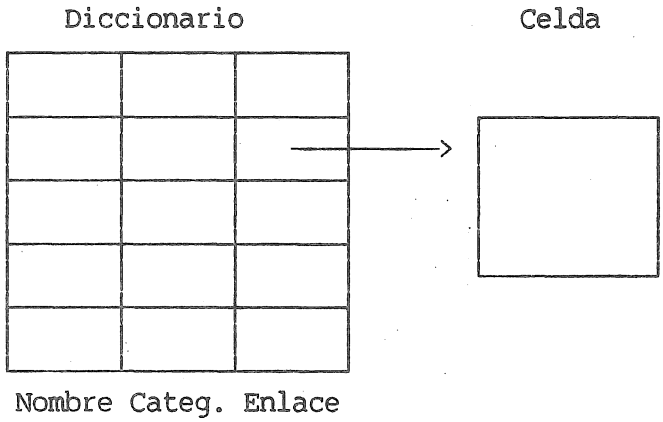
Por estar asociado a una entidad, cada celda tiene un nombre. Una celda tiene asignada además una dirección.

El número de campos de una celda es variable, pero contiene al menos dos campos fijos: la dirección de la próxima celda de la lista - si existe - y la dirección del descriptor. Dado que todas las celdas tienen estos dos campos, el descriptor no los incluye.



Una celda puede ser accedida por su nombre o por su dirección. Cuando se dispone del nombre, el acceso se realiza por medio de una estructura auxiliar : el diccionario.

El diccionario contiene una entrada de tres componentes por cada celda. La primer componente contiene el nombre de la celda y la segunda su dirección. Como cada descriptor también tiene asociado un nombre, el diccionario contiene además una entrada nombre-dirección por cada descriptor. Por cada entrada se almacena entonces un tercer valor, el tipo de entrada, que permite distinguir si el par nombre-dirección corresponde a un descriptor o a una celda.



Una lista enlazada de celdas se distingue entonces, de una lista enlazada común por dos hechos fundamentales. Primero no todas las componentes tienen necesariamente la misma estructura, segundo el acceso a una celda puede hacerse por medio del diccionario y no requiere recorrer la lista desde el principio. Estas dos características provocan que cada celda deba contener un puntero a su descriptor, porque aún en el caso de que todos los elementos de la lista tuvieran la misma estructura, al acceder a elementos interiores de la lista podemos no conocer por anticipado de que tipo de celda se trata.

Aunque la representación completa depende de la aplicación particular, estamos en condiciones de introducir ciertas características de la estructura general.

La estructura se completa con una cabeza conteniendo información general, como por ejemplo, punteros a las listas de celdas, punteros a las primeras locaciones libres en el pool, en la tabla de descriptores, etc.

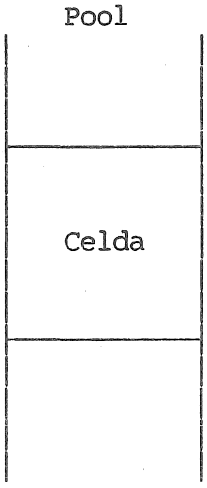
Tanto la cabeza como el diccionario y los descriptores, contienen unicamente información estructural para permitir almacenar y consultar la información relativa al proceso.

La información propiamente dicha reside en las celdas. Una celda contiene siempre parte de la información que describe al problema. Sin embargo también contienen parte de la información estructural. Esto es, las celdas contienen un puntero a un descriptor y al menos un puntero a otra celda - la próxima de la lista -.

1.3.2 Implementación en términos de primitivas de un Lenguaje

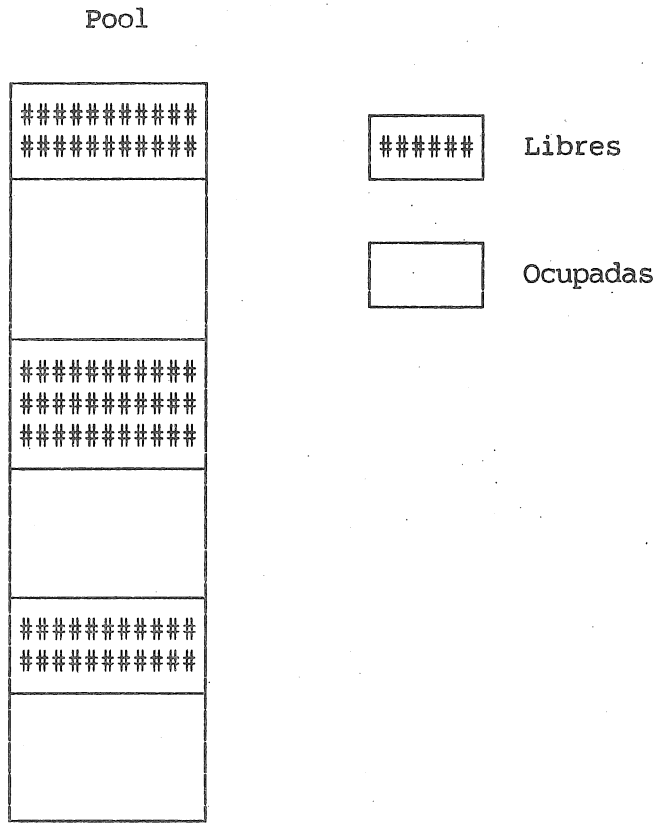
Las operaciones primitivas para manipulación de la estructura plex están implementadas en Pascal. La decisión para adoptar este lenguaje se basó fundamentalmente en la disponibilidad de constructores de tipos de datos muy flexibles y en la posibilidad de combinarlos, formando estructuras más complejas. Por otra parte se evitó el uso del tipo puntero, con el fin de que las rutinas puedan ser invocadas desde programas escritos en otros lenguajes que no dispongan de este tipo de datos.

Fisicamente una celda esta constituída por un bloque de locaciones contiguas de almacenamiento, contenidas en un pool. La dirección de una celda es la posicion de su primera locación dentro del pool.



El tamaño de cada locación es fijo, pero el número de locaciones de cada celda es variable. En nuestra implementación cada campo de la celda ocupa una locación.

Las celdas son creados dinamicamente desde un espacio de almacenamiento libre, a medida que son requeridos. El pool está constituído tanto por las locaciones libres como por las que están asociadas a celdas.



El hecho de que en un mismo espacio residan celdas y locaciones libres y que las locaciones no sean de un tipo de dato fijo, impide que el pool pueda ser accedido secuencialmente.

Cuando se elimina una celda sus locaciones son devueltas al espacio disponible.

- El espacio de almacenamiento de información - pool - tiene que ser capaz de almacenar valores de cualquiera de los dominios inherentes al problema, además de punteros a celdas y a descriptores.

Como en Pascal una variable solo puede ser capaz de almacenar valores de un único tipo, se convino en que la unidad de almacenamiento sea un registro variante con un único campo que puede ser de cualquiera de los dominios indicados en el párrafo anterior.

El pool esta implementado entonces por medio de un arreglo de registros variantes sin etiqueta, consistente cada uno de ellos de un único campo. La variación entre un registro y otro no es entonces el número de campos, sino el tipo del campo.

El control del tipo del contenido de la celda se realiza por programa, es decir, el registro no contiene un campo etiqueta. De hecho cada celda tiene asociado un descriptor que determina univocamente el tipo de cada campo de la celda.

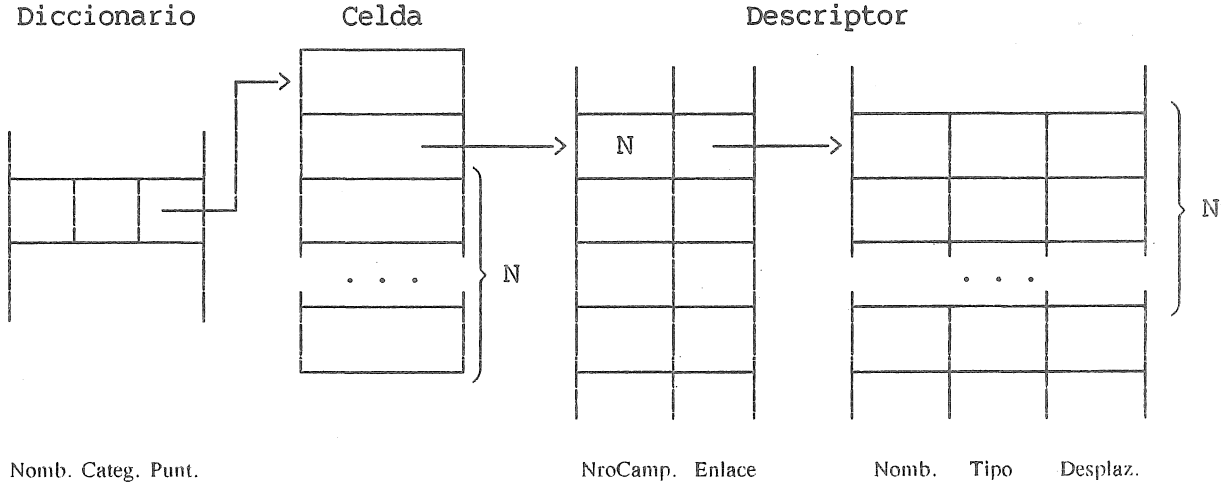
El pool contiene parte de la información estructural. Esto es, las celdas contienen un puntero a un descriptor y al menos un puntero a otra celda - la próximo de la lista.

- El diccionario, es una tabla con organización hash interno con resolución lineal de colisiones, constituída por ternas cuyas componentes contienen un nombre, una dirección y una indicación de si se trata de un nombre de descriptor o de celda.

La implementación de la tabla se realiza a través de un arreglo de registros, cada uno de los cuales consta de tres campos capaces de contener la terna que identifica a una entrada.

- Los descriptores estan distribuídos en dos tablas. Esto se debe a que cada descriptor contiene un número que indica la cantidad de campos que contiene la celda y una terna nombre-tipo-posición por cada campo de la celda.

Cada una de las tablas esta implementada a traves de un arreglo de registros. Una de las tablas contiene registros de dos campos, el primero indica la cantidad de campos de la celda, por ejemplo n, mientras que el segundo es un puntero al primero de los n registros de la segunda tabla, que permiten describir a los n campos de la celda. La segunda tabla contiene registros con un campo para cada componente de la terna nombre-tipo-posición que describe a un campo. La descripción de los n campos de una celda reside en n registros contiguos en la tabla.



- En nuestra implementación todo el área de almacenamiento es estático. La cabeza de la estructura contiene entonces los siguientes campos:

a) Total de Espacio Utilizado en el Pool

b) Total de Espacio Libre en el Pool

c) Puntero a la Primera Locación Libre en el Pool

d) Puntero a la Ultima Locación Libre en el Pool

e) Puntero a la Primera Locación Libre en la Tabla de Descriptores de celdas

f) Puntero a la Primera Locación Libre en la Tabla de Descriptores de Campos

g) Punteros a cada una de las listas de celdas.

2.0 Aplicaciones de la Estructura Plex

Una de las limitaciones de los simuladores de procesos actuales, es el hecho de que los datos que modelan un proceso, residen en estructuras de datos definidas a partir de constructores fijos, por ejemplo arreglos dimensionados. Esto provoca cierta inflexibilidad, ya que requiere que el diseñador del sistema conozca toda la información que será manipulada durante la simulación. Cualquier modificación del proceso provoca alteraciones en el diseño o al menos en las estructuras de datos [Ev 77].

La estructura de datos Plex surge como alternativa al uso de estructuras fijas.

La estructura de datos Plex provee entonces una forma elegante de representar los datos que describen un simulador de "flow-sheet".

El "flow-sheet" es representado por la estructura Plex en sus aspectos espaciales y temporales - topología y variaciones de su contenido en el proceso que se modela -, incrementándose su flexibilidad debido a que es posible modificar la topología sin trabajo extra. El concepto de estructura Plex ha sido incorporado en paquetes de simulación como el ASPEN, ampliamente usados en la industria petroquímica.

Los conceptos que importan a su definición la hacen sumamente útil y versátil como para extender el rango de su uso a otro tipo de simulaciones en donde estén involucradas transacciones de algún tipo entre objetos. Ejemplo clásicos pueden ser los circuitos eléctricos, las redes digitales y redes de computadoras.

La ventaja fundamental de esta estructura respecto a cualquier estructura fija es su flexibilidad, ya que es posible alterar la configuración de los datos sin alterar la estructura abstracta de los mismos. Además el espacio inutilizado disminuye, al tiempo que desaparecen las restricciones impuestas por una implementación utilizando constructores de datos fijos.

La implementación de Plex como tipo de dato abstracto y el encapsulamiento de la información que de ella se deriva, hacen que solamente pueda ser accedida por el usuario a traves de rutinas "ad-hoc" y permita establecer cierto tipo de protección sobre los datos con verificación en los derechos de acceso. La organización de los datos es asi transparente para el programador de aplicación, incrementando la modularidad de sus programas.

La forma de representación de la información, asi como su recuperación puede derivar en una base de datos que tanto sirve para la simulación en si misma como para la documentación o fuente de consulta, con todas sus características de flexibilidad y modularidad.

Este tipo de estructura constituye una buena base para próximos proyectos sobre diseño asistido por computadora. Esta última sería la linea de futuros trabajos en donde la estructura Plex juega un rol importante en cuanto al soporte de todo el sistema pero no es el fin último.

Apenice I

RUTINAS PARA MANEJO DE PLEX

1. Inicializa (EspLibre, Nomb, Plex)

   Inicializa:

   a) La cabeza de la estructura, dandole un valor nulo a los punteros a listas, uno a los punteros a locaciones libres en las tablas e insertando el nombre de la estructura y su espacio requerido.

   b) La tabla de descriptores con valores definidos como nulos.

   c) El diccionario con valores definidos como nulos.

   d) El pool, enlazando cada locacion con la siguiente, excepto la última donde se inserta nulo.

2. RequerirEspacio (Nb, N, Error, Plex)

   Este procedimiento toma N locaciones contiguas del espacio disponible, devuelve en Nb un puntero a la primera de ellas y actualiza el puntero a la primera locación libre en la cabeza.

   Como no todo el espacio disponible es contiguo, el procedimiento recorre la lista de locaciones libres, buscando las primeras N contiguas.

3. DevolverEspacio (Nb, N, Plex)

   No en todas las aplicaciones tiene sentido la eliminación de celdas, pero si esta acción fuera requerida, las N locaciones que ocupaba a partir de Nb son devueltas al final de la lista de locaciones libres en el pool.

4. InsertarDiccionario (Nomb, Pn, CodEnt, Error, Plex)

Inserta en el diccionario una entrada con la terna dada por los tres primeros parámetros, excepto si ya existía una terna con el primer valor igual a Nomb, en cuyo caso devuelve un código de error en Error.

La elección de la locación en el diccionario se realiza aplicando un función cuyo dominio son nombres y el rango son las locaciones del diccionario, si ya estaba ocupada se busca linealmente una locación libre.

5. RecuperarEntradaDiccionario (Nomb, Punt, CodEnt, Plex)

Devuelve en Punt y CodEnt los valores de la entrada en el diccionario asociada al nombre Nomb.

6. EliminarEntradaDiccionario (Nomb, CodEnt, Error, Plex)

Elimina, si existe, la entrada en el diccionario del tipo de CodEnt con nombre Nomb, insertando en su lugar un código de borrado.

7. CrearDescriptor (Nombre, NroCamp, Pd, Error, Plex)

Crea un descriptor llamado Nombre con NroCamp campos, devuelve en Pd un puntero al descriptor creado.

Actualiza los campos de la cabeza.

8. DefinirCampoDescriptor (Nomb, TipoCamp, K, Pd, Plex)

Dado el descriptor apuntado por Pd, define su K-esimo campo llamándolo Nomb y con tipo TipoCamp

9. CrearBead (Nb, NombDesc, NombBead, Error, Plex)

Recupera, si existe, el descriptor deseado a partir de la búsqueda de su nombre en el diccionario.

Reserva el espacio necesario en el pool, de acuerdo a lo indicado por el descriptor.

Inicializa el primer y segundo campo de la celda, con el puntero a la próxima celda y al descriptor, respectivamente.

Inserta el nombre de la celda en el diccionario.

10. RecuperarPunteroBead (NombBead, Nb, Error, Plex)

Dado un nombre, recupera si existe, el puntero a la celda asociada a ese nombre. Esta rutina no se ocupa de la categoria de la celda.

11. RecuperarPunteroDesc (NombDesc, Nd, Error, Plex)

    Devuelve en Nd en puntero al descriptor cuyo nombre es NombDesc, a partir de una búsqueda en el diccionario.

Notas :

1. El paquete se completa con rutinas que sirven como vinculo entre el programador y la estructura Plex, que dependen de la aplicación y que son las que en definitiva van a ser invocadas por los programas desarrollados por el usuario. Por ejemplo los programas de aplicacion no invocaran a CrearBead sino que dispondrá de una rutina por cada categoria, que ademas de crear la celda, la insertará en la lista adecuada.

2. El parámetro Plex incluye la totalidad de la estructura, esto es, cabeza, pool, diccionario y descriptores.

BIBLIOGRAFIA

[Ah 83] Data Structures and Algorithms
Aho A.V., Hopcroft J.E. and Ullman, J.D.
Addison-Wesley 1983

[Ho 76] Fundamentals of Data Structures using Pascal
Horowitz E. and Sahni S.
Computer Science Press 1976

[St 80] Data Structures Techniques
Standish T.A.
Addison-Wesley 1980

[Te 81] Data Structures using Pascal
Tenenbaum A., Augenstein M.
Prentice-Hall 1981

[Ev 77] System Structures for Process Simulation
Evans L.B., Joseph B., Seider W.D.
AIChe Journal (Vol.23,No.5,Septiembre 1977)

ARBOLES QUINTARIOS: ESTUDIO EXPERIMENTAL PARA INTERROGACIONES EXACTAS, PARCIALES, EN RANGO Y EN RANGO PARCIAL

O. SANTANA, O. MAYOR, M. DIAZ, S. REINA
E. U. Informática; Universidad Politécnica de Canarias
Aptdo. 550, Las Palmas de Gran Canaria, España.

Se presenta un estudio experimental del costo de almacenamiento y el tiempo de respuesta en una estructura de datos denominada Arbol Quintario para un sistema de recuperación de información basado en las interrogaciones exacta, parcial, en rango y en rango parcial.

Para valores moderados de la dimensionalidad, cardinalidad e intervalo de variación de los valores de clave se han obtenido árboles de grandes volúmenes, siendo el parámetro que más influye, la dimensionalidad.

Los números de accesos y comparaciones aumentan con la cardinalidad y la dimensionalidad en los cuatro tipos de interrogaciones. En lo relacionado con el intervalo de variación de los valores de clave se obaerva que los resultados se ven afectados por varios factores. En las interrogaciones exacta y parcial aunque se produce algún crecimiento, este es muy ligero. Y en las interrogaciones en rango y en rango parcial se produce un decrecimiento.

En definitiva, el tiempo de respuesta es bueno para interrogaciones exactas y parciales, y aceptable para interrogaciones en rango y en rango parcial, siendo la dimensionalidad el parámetro que influye de forma negativa para este factor.

ARBOLES QUINTARIOS: ESTUDIO EXPERIMENTAL PARA INTERROGACIONES EXACTAS, PARCIALES, EN RANGO Y EN RANGO PARCIAL

O. SANTANA, C. MAYOR, M. DIAZ, S. REINA
E. U. Informática; Universidad Politécnica de Canarias
Aptdo. 550, Las Palmas de Gran Canaria, España.

RESUMEN

En éste artículo se realiza un estudio experimental de una estructura de datos denominada Arbol Quintario, diseñada para un sistema de recuperación de información basado en los siguientes tipos de interrogaciones: exacta, parcial, en rango y en rango parcial.

0) INTRODUCCION

Han aparecido muchos métodos para la construcción de sistemas de recuperación de información, la mayoría de ellos se centran sobre alguno de los cuatro tipos de interrogaciones [2,3,4,5,6,10]. La idea del Arbol Quintario surge a partir de dos estructuras: el Arbol de Rango, desarrollado por Bentley y Friedman [1], especial para interrogaciones en rango; y el Arbol Complejo de Búsqueda Binaria (BST-Complejo), desarrollado por Lien, Taylor y Driscoll [8], adecuado, principalmente para interrogaciones exactas.

El Arbol Quintario fue propuesto por D.T. Lee y C.K. Wong [7], como una estructura de datos con un tiempo de respuesta bajo, para el peor caso, en los cuatro tipos de interrogaciones, a costa de un gasto de almacenamiento extra.

1) ARBOL QUINTARIO

Se considera un fichero F de n registros, donde cada registro es una nd-tupla de valores ordenados. Cada componente de la nd-tupla, (k0,k1,...knd-1), se denomina atributo o clave. El número de valores distintos que puede tomar cada clave ki es vi.

En un Arbol Quintario hay nd niveles de nodos internos representando a las distintas claves. En cada uno de estos niveles se tiene un árbol binario cuyos nodos representan a los valores de clave, el número máximo de nodos en la estructura de árbol binario del nivel i es vi. Cada uno de estos nodos, además de tener los dos hijos característicos de la estructura de árbol binario en el mismo nivel, tienen unidos tres árboles adicionales pertenecientes a un nivel inferior.

En general, considerando el dominio del fichero F como un espacio nd-dimensional y cada clave como un eje de coordenadas, se tiene que cada nodo en el nivel i representa un subfichero en un espacio (nd-i)-dimensional, y tiene cinco ramificaciones: dos de ellas apuntando a los nodos en el mismo nivel, que representan a dos subficheros en el espacio (nd-i)-dimensional que pertenecen a la izquierda y a la derecha del hiperplano asociado al nodo padre; y otras tres, que representan a tres subficheros en el espacio (nd-i-1)-dimensional en el nivel i+1, dos de ellos son las proyecciones de los dos subficheros anteriores sobre el hiperplano, y

el tercero está constituído por los registros del hiperplano. El número total de registros de estos tres subficheros es igual al número de registros del subfichero representado por el nodo padre.

El árbol se construye como sigue: dado el fichero de n registros, se calcula la mediana de éste, rm, de tal forma que el registro rm divida al fichero en dos partes que no contengan más de n/2 registros cada una. Una vez hecho esto, se construyen tres subficheros fizquierdo, fmid y fderecho, tales que los valores de la primera clave en cada uno de ellos, sean menores, iguales y mayores que el valor de la primera clave del registro rm, respectivamente.

Se crea el nodo raíz con el valor de la primera clave del registro rm, y cinco punteros que son izquierdo, medizquierdo, central, medderecho y derecho. A partir de este punto, se generan los cinco subárboles, sabiendo que izquierdo y derecho son los punteros destinados a la estructura de árbol binario en cada nivel; y por lo tanto, apuntan a los subárboles generados a partir de los ficheros fizquierdo y fderecho en el mismo espacio nd-dimensional (representando a la misma clave). Sin embargo, los punteros medizquierdo, central y medderecho apuntan a los subárboles situados en un nivel inferior (espacio (nd-1)-dimensional), que representan, en definitiva, a los mismos ficheros fizquierdo, fmid y fderecho, pero ignorando la clave 0, al descender de nivel se pasa a representar a la clave siguiente. El mismo proceso se repite para los subficheros generados mientras no se llegue al nivel (nd-1)-ésimo.

En el nivel (nd-1)-ésimo, además de los punteros izquierdo y derecho de la estructura de árbol binario, se mantienen los tres punteros centrales, pero apuntando a los nodos hoja del árbol: central apunta a los registros representados por el nodo, y medizquierdo y medderecho apuntan a los registros de los subárboles izquierdo y derecho, respectivamente.

La figura 1 muestra el Arbol Quintario para la base de datos bidimensional representada en la tabla. Está constituido por dos niveles de nodos internos: El nivel 0 que representa la clave A y el nivel 1 que representa a la B; además del nivel de nodos hoja que apuntan a los registros. Se selecciona el nodo c como nodo raíz de la estructura de árbol binario en el nivel 0, y los nodos b y f como raíces de los subárboles izquierdo y derecho, respectivamente. Los registros que constituyen el subfichero fizquierdo son los registros del 1 al 5, los de fmid son los del 6 al 8 y los de fderecho, del 9 al 15.

Según Lee y Wong el tiempo requerido para construir un árbol quintario es O(n*(log n)**nd) y el máximo almacenamiento es O(n*(log n)**nd).

2) BUSQUEDA Y RECUPERACION

Además de proponer la estructura de árbol quintario, D.T. Lee y C.K. Wong, esbozan los algoritmos dedicados a la búsqueda y recuperación de información de acuerdo con cada uno de los cuatro tipos de interrogaciones. En los próximos apartados se resume el proceso que siguen estos algoritmos.

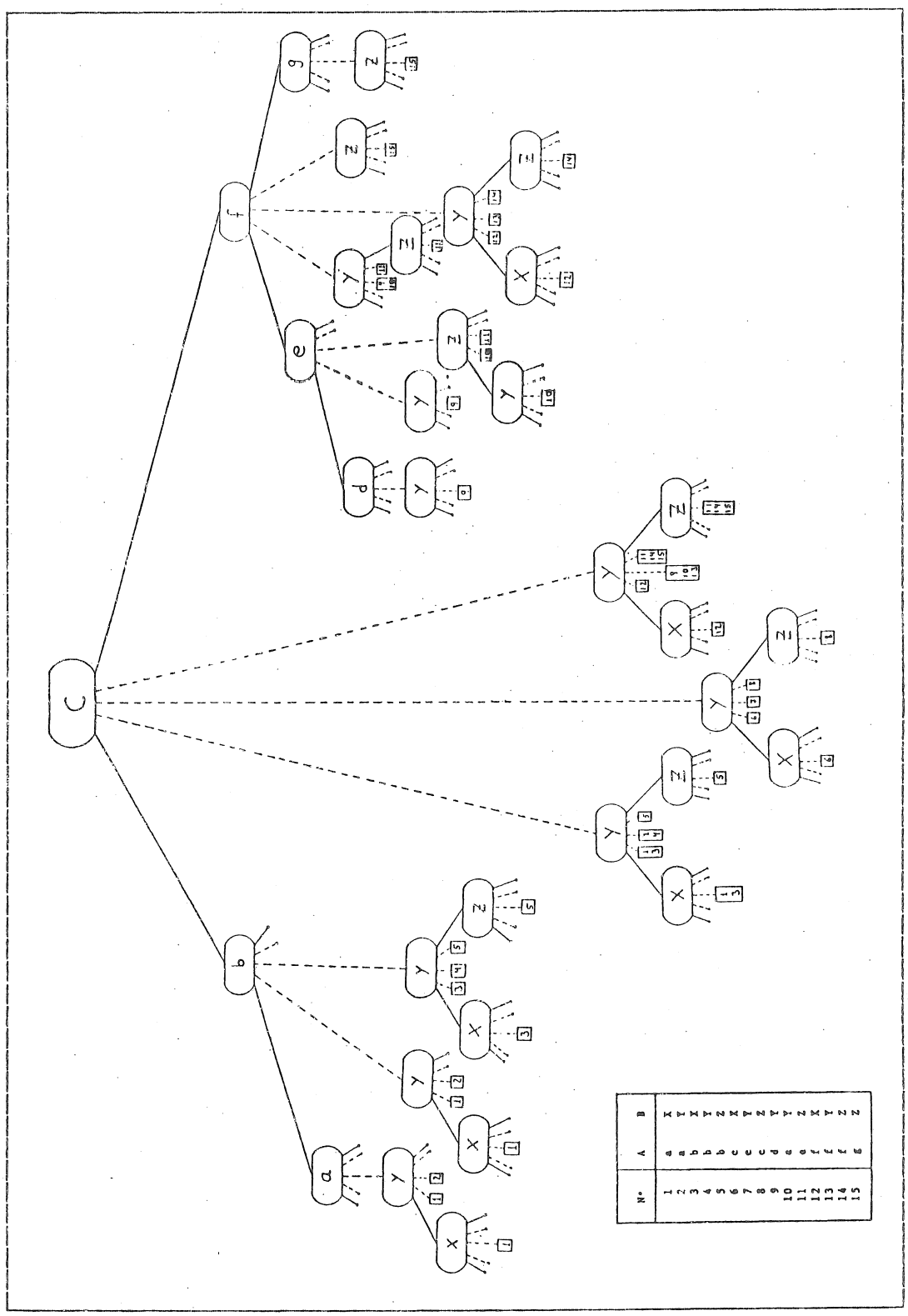


Figura 1

2.1) Búsqueda exacta

La interrogación exacta es aquella en la que se especifica un valor para cada clave. El proceso de recuperación debe ir recorriendo la estructura de árbol binario de cada nivel i, de acuerdo con el valor de la clave ki especificada, de la misma forma que una búsqueda binaria, con la salvedad que cuando el valor de un nodo coincida con el valor de la clave correspondiente, se desciende de nivel y se continúa la búsqueda a partir de la clave siguiente.

Ya que cada acceso a nodo descarta generalmente la mitad del fichero representado por el nodo, el número total de nodos accedidos es de casi O(nd*log n) [7].

2.2) Búsqueda parcial

La interrogación parcial especifica

$$s < nd$$

valores de clave, quedando las nd-s claves restantes sin especificar. El proceso a seguir para recuperaciones de este tipo es similar al anterior, excepto en que cuando no se especifica una clave se tiene que buscar a través de los tres subárboles situados en el nivel inferior. Siendo t el número de registros recuperados, el número máximo de accesos a nodos está limitado por O(3**(nd-s)*(s*log n)+t) [7].

2.3) Búsqueda en rango

La interrogación en rango especifica un intervalo de variación [q(li,i),q(ls,i)] para cada clave ki. Para recuperar la información se accede al nodo raíz del árbol en el nivel 0. Si el valor del nodo está fuera del intervalo especificado por la interrogación, se siguen los punteros izquierdo o derecho dependiendo de si el valor del nodo es mayor que q(ls,0) o menor que q(li,0). Si el valor del nodo está dentro del intervalo, se utilizan dos variables de control, I y D, que indican que el intervalo [q(li,i),q(ls,i)] ha sido partido en dos subintervalos [q(li,i),x),(x,q(ls,i)] donde x es el valor almacenado en el nodo actual. De esta forma, si el nodo raíz está dentro del intervalo, se accede recursivamente al subárbol central en un nivel inferior, y a los subárboles izquierdo y derecho en el mismo nivel, poniendo las variables de control I y D a 1. La partición inicial del intervalo da lugar al recorrido de los subárboles central, izquierdo y derecho. Por otro lado, la partición siguiente de los subintervalos da lugar al recorrido de los subárboles central, izquierdo y medderecho (derecho y medizquierdo). Ya que cada intervalo [q(li,i),q(ls,i)] se puede dividir en hasta log n subintervalos en cada nivel i, el número de accesos a nodo está limitado por O(t+(log n)**nd) [7], donde t es el número de registros encontrados.

2.4) Búsqueda en rango parcial

La interrogación en rango parcial es aquella que especifica

$$s < nd$$

intervalos de variación, dejando los nd-s restantes sin especificar. El algoritmo que proponen Lee y Wong para este tipo de búsqueda es similar al anterior, excepto en que si no se especifica una clave, se tiene que buscar a través de los tres subárboles centrales del nivel inferior.

El número máximo de nodos accedidos se puede obtener análogamente como en una búsqueda parcial, siendo t el número de registros recuperados: O(3**(nd-s)*log n)**s+t) [7].

3) ESTUDIO EXPERIMENTAL PARA INTERROGACIONES EXACTAS, PARCIALES, EN RANGO Y EN RANGO PARCIAL.

A pesar de la importancia del estudio teórico de una estructura de datos, el estudio experimental proporciona una serie de valores añadidos al conocimiento que de la misma se tiene [9].

3.1) DESCRIPCION DE LOS FICHEROS

Según Lee y Wong [7] se concluye que, debido a que el almacenamiento requerido es O(n(log n)**nd), se tiene que para un tamaño de bases de datos moderado, en el cual la dimensionalidad es alta, es muy costoso implementar esta estructura. Sin embargo, si el almacenamiento no es el problema principal o el número de claves es pequeño, la implementación es abordable.

De acuerdo con todas las limitaciones que presenta la estructura, en relación con el almacenamiento requerido, para la realización de las pruebas se crearon 11 ficheros, cuyas características se pueden observar en la tabla 1. Tales ficheros están divididos en tres grupos, de acuerdo con la variación de cada uno de los siguientes parámetros: cardinalidad, n, dimensionalidad, nd, e intervalo de variación de los valores de clave, vi.

Nombre del Fichero	Dimensionalidad	Cardinalidad	Intervalo de Variación
QTRKR0	4	500	0..14
QTRKR1	4	1.000	0..14
QTRKR2	4	2.000	0..14
QTRKR3	4	4.000	0..14
QTRKN0	4	1.000	0.. 9
QTRKN1	4	1.000	0..11
QTRKN2	4	1.000	0..14
QTRKN3	4	1.000	0..16
QTRNR0	4	1.000	0.. 9
QTRNR1	6	1.000	0.. 9
QTRNR2	8	1.000	0.. 9

Tabla 1: Ficheros utilizados

Lee y Wong [7] puntualizan que, ya que el orden de las claves no sólo afecta al tiempo de respuesta, sino también al almacenamiento requerido, en esta estructura se debe tener en cuenta que la nd-tupla de cada registro se ordene de acuerdo con algún criterio: la muestra de interrogaciones planteadas por los usuarios del sistema, o las probabilidades de las claves especificadas en una interrogación. En la interrogación parcial o en rango parcial, por ejemplo, si una clave se especifica con mucha frecuencia, debe estar en uno de los

primeros niveles en la estructura del árbol. En este experimento todas las claves de un fichero tienen el mismo intervalo de variación, y las interrogaciones utilizadas en las pruebas de búsqueda se generan aleatoriamente con distribución uniforme, puesto que no se tiene en cuenta el problema de la ordenación de las claves.

3.2) OBJETIVOS DEL EXPERIMENTO

El objetivo de este trabajo es el estudio experimental del costo de almacenamiento y el tiempo de respuesta para los tipos de interrogaciones anteriormente citados, a través de los siguientes parámetros:

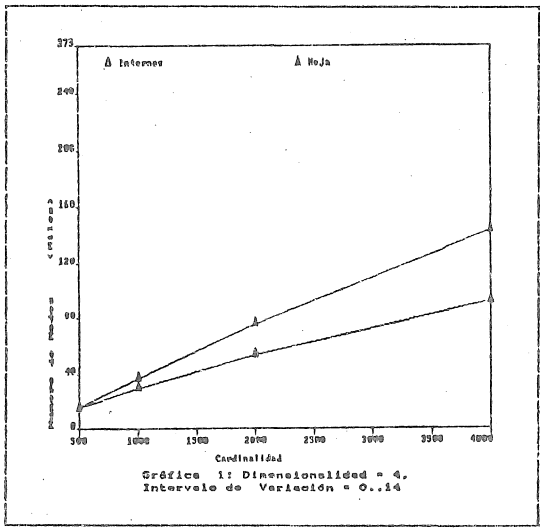
1) El número de nodos generados en la construcción de los árboles.

2) El número de accesos y comparaciones en los distintos tipos de interrogaciones, independientemente del número de claves sin especificar y el tamaño del hipercubo de interrogación.

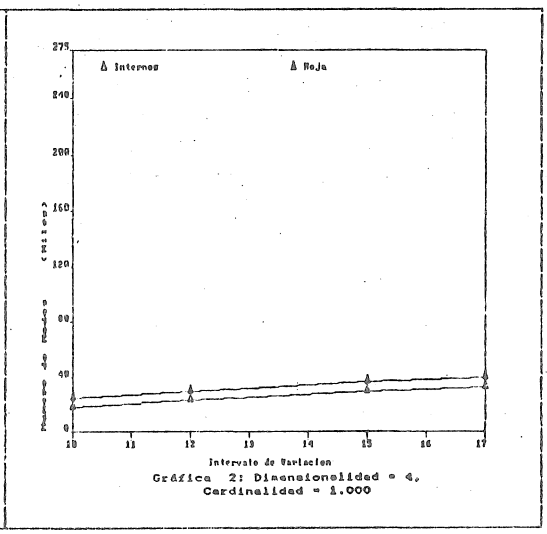
3) El número de accesos y comparaciones en las interrogaciones parciales y en rango, en función de la variación del número de claves sin especificar y el tamaño del hipercubo de interrogación.

3.3) NUMERO DE NODOS GENERADOS EN LA CONSTRUCCION DE LOS ARBOLES:

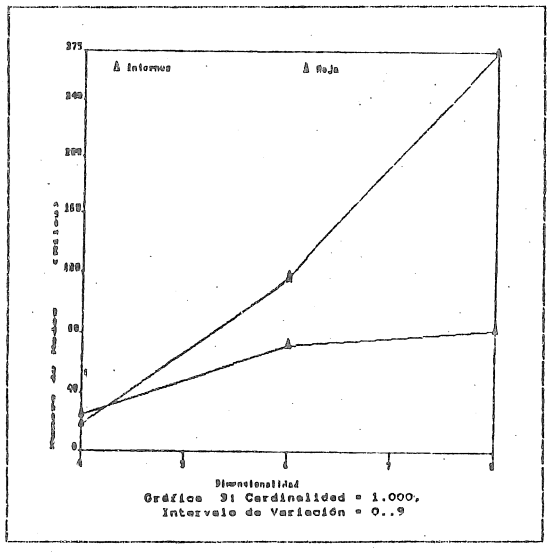
La primera parte de las pruebas consiste en la creación de los árboles quintarios correspondientes a cada uno de los ficheros y la medición del número de nodos, tanto internos como hojas. Esto da una idea de los grandes volúmenes de memoria que pueden ocupar dichos árboles quintarios, y como puede influir en este aspecto la variación de alguno de los parámetros.



Gráfica 1: Dimensionalidad = 4,
Intervalo de Variación = 0..14



Gráfica 2: Dimensionalidad = 4,
Cardinalidad = 1.000



Gráfica 3: Cardinalidad = 1.000, Intervalo de Variación = 0..9

Al observar las gráficas 1, 2 y 3 se puede comprobar, tal y como se esperaba, que el número de nodos generados se incrementa al aumentar cualquiera de los parámetros.

En la gráfica 1 el crecimiento de los nodos hoja es, prácticamente, constante; el crecimiento más suave de los nodos internos indica el grado de compartición de los valores de clave de los registros.

En la gráfica 2 la diferencia entre el número de nodos internos y el número de hojas casi se mantiene constante.

Finalmente, el número de nodos internos se ve muy afectado ante una variación del número de claves, gráfica 3. Para 4 claves es menor que el número de nodos hoja, mientras que para 6 y 8 claves el número de nodos internos crece espectacularmente respecto al número de nodos hoja. Esto se debe a que al aumentar la dimensionalidad, el árbol tiene un mayor número de niveles, por tanto el número de nodos internos es mayor; sin embargo, el número de nodos hoja se encuentra limitado por el número de registros que permanece constante, y no puede crecer en la misma proporción.

3.4) NUMERO DE ACCESOS Y COMPARACIONES EN LOS DISTINTOS TIPOS DE INTERROGACIONES:

La segunda parte de las pruebas consiste en el estudio del número de accesos, comparaciones, celdas tratadas y registros recuperados en cada uno de los cuatro tipos de interrogaciones. Se entiende por número de accesos, el número de nodos internos a los que se accede durante la búsqueda. El número de comparaciones es el número de veces que se compara el campo de información de un nodo accedido con la interrogación. El número de celdas tratadas es el número de nodos hoja visitados durante el proceso. Y el número de registros recuperados es el número de registros que se obtienen debido a que cumplen las condiciones de la interrogación.

Todos los números aleatorios utilizados se generaron con

distribución uniforme.

El número de pruebas realizadas depende, en cierta forma, del tipo de interrogación:

Para el caso de la interrogación exacta se realizaron 500 pruebas.

En el caso de la interrogación parcial se realizaron pruebas con interrogaciones con una clave sin especificar, con dos, y hasta nd-1 claves sin especificar. De cada uno de estos casos se llevaron a cabo 500 pruebas. De esta forma, el número de pruebas realizadas viene dado por 500*(nd-1).

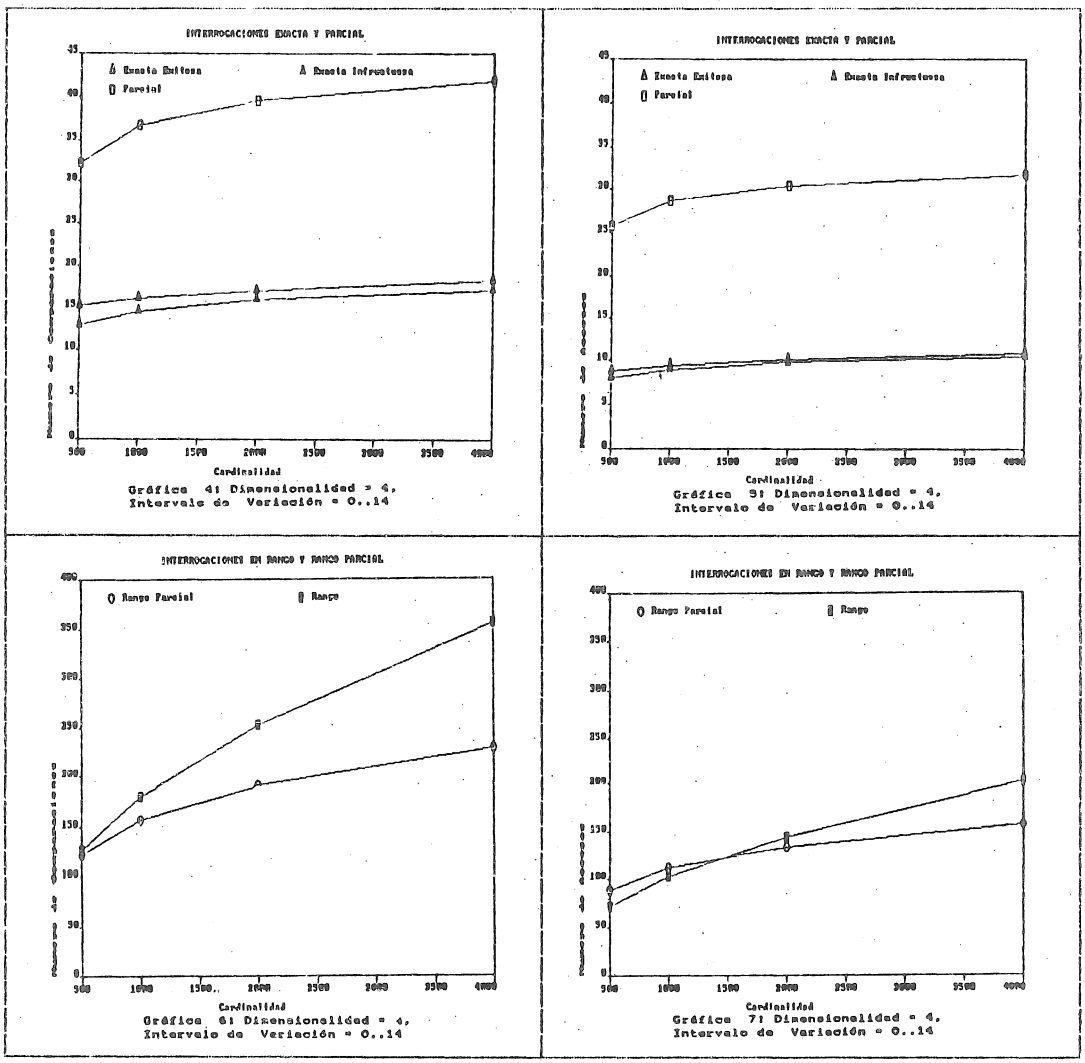
En el caso de la interrogación en rango se realizaron pruebas haciendo variar el tamaño del hipercubo desde 1 hasta 5, así el número total de pruebas realizadas fue de 500*5=2.500.

Finalmente, en el caso de la interrogación en rango parcial se varió por un lado, el número de claves sin especificar y, por otro, el tamaño del hipercubo. De acuerdo con lo anterior, el número de pruebas realizadas fue de 500*5*(nd-1).

Para cada una de las 500 pruebas se necesitan nd números aleatorios limitados por el intervalo de variación de las claves, que representan los valores especificados por la interrogación. Además, en las interrogaciones parciales, se requieren nd-1 números aleatorios necesarios para indicar las posiciones de las claves sin especificar. Para las interrogaciones en rango se pueden utilizar como límites inferiores los nd valores generados, siempre que al calcular los límites superiores ninguno de ellos se salga del intervalo de variación de las claves, en caso de que esto ocurra se deben generar números que cumplan con esta restricción. Los límites superiores se calculan a partir de los límites inferiores, sumando a cada uno de ellos, inicialmente un 1, luego un 2, y así, hasta llegar a 5, de acuerdo con el tamaño del hipercubo de interrogación.

En la interrogación exacta las pruebas se realizaron para dos casos específicos: que todas las búsquedas resultaran exitosas y que todas resultaran infructuosas. Para el primer caso, se genera aleatoriamente un número entre 1 y el número de registros del fichero y éste, precisamente, será el registro a buscar. En el segundo caso se utiliza una interrogación aleatoria que cumpla con la condición de no existir en el fichero de datos. En la interrogación exacta no tiene sentido hablar de celdas tratadas puesto que en las búsquedas exitosas siempre se visita una celda, y en las infructuosas nunca se visita ninguna.

A) El parámetro variable es la cardinalidad:

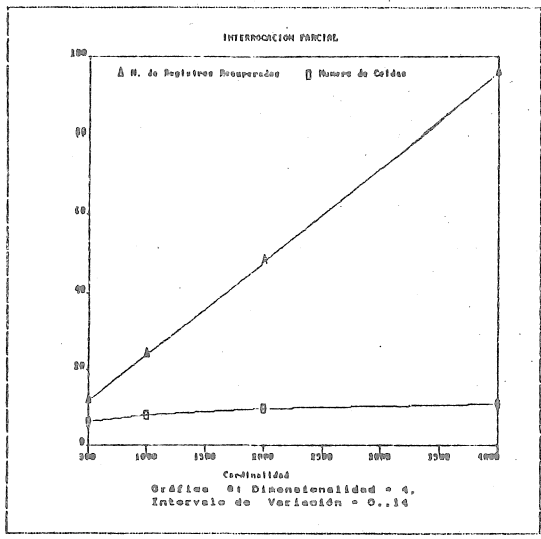


Gráfica 4: Dimensionalidad = 4,
Intervalo de Variación = 0..14

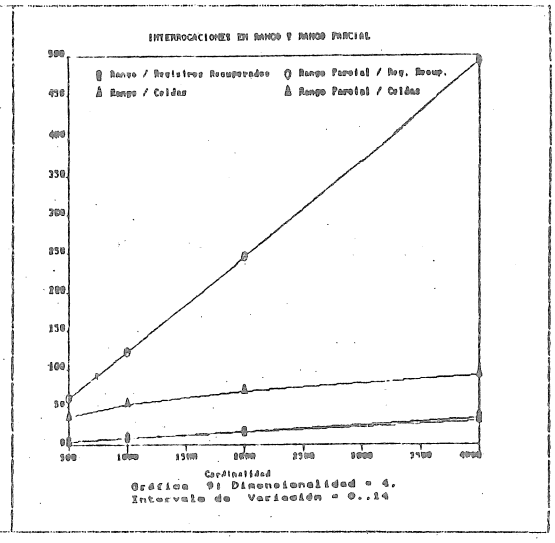
Gráfica 5: Dimensionalidad = 4,
Intervalo de Variación = 0..14

Gráfica 6: Dimensionalidad = 4,
Intervalo de Variación = 0..14

Gráfica 7: Dimensionalidad = 4,
Intervalo de Variación = 0..14



Gráfica 8: Dimensionalidad = 4.
Intervalo de Variación = 0..14



Gráfica 9: Dimensionalidad = 4.
Intervalo de Variación = 0..14

Las gráficas anteriores muestran como influye la cardinalidad en el número de comparaciones, accesos, celdas tratadas y registros recuperados en los cuatro tipos de interrogaciones. Cualquiera de estos datos aumenta con el número de registros del fichero original. Esto es lógico si se parte del hecho de que un aumento en el número de registros produce un aumento en el número de nodos de árbol. Además, en general, el crecimiento del número de comparaciones, accesos y celdas es de menor orden a medida que aumenta la cardinalidad. Esto último se debe, fundamentalmente, a que los registros comparten más nodos internos.

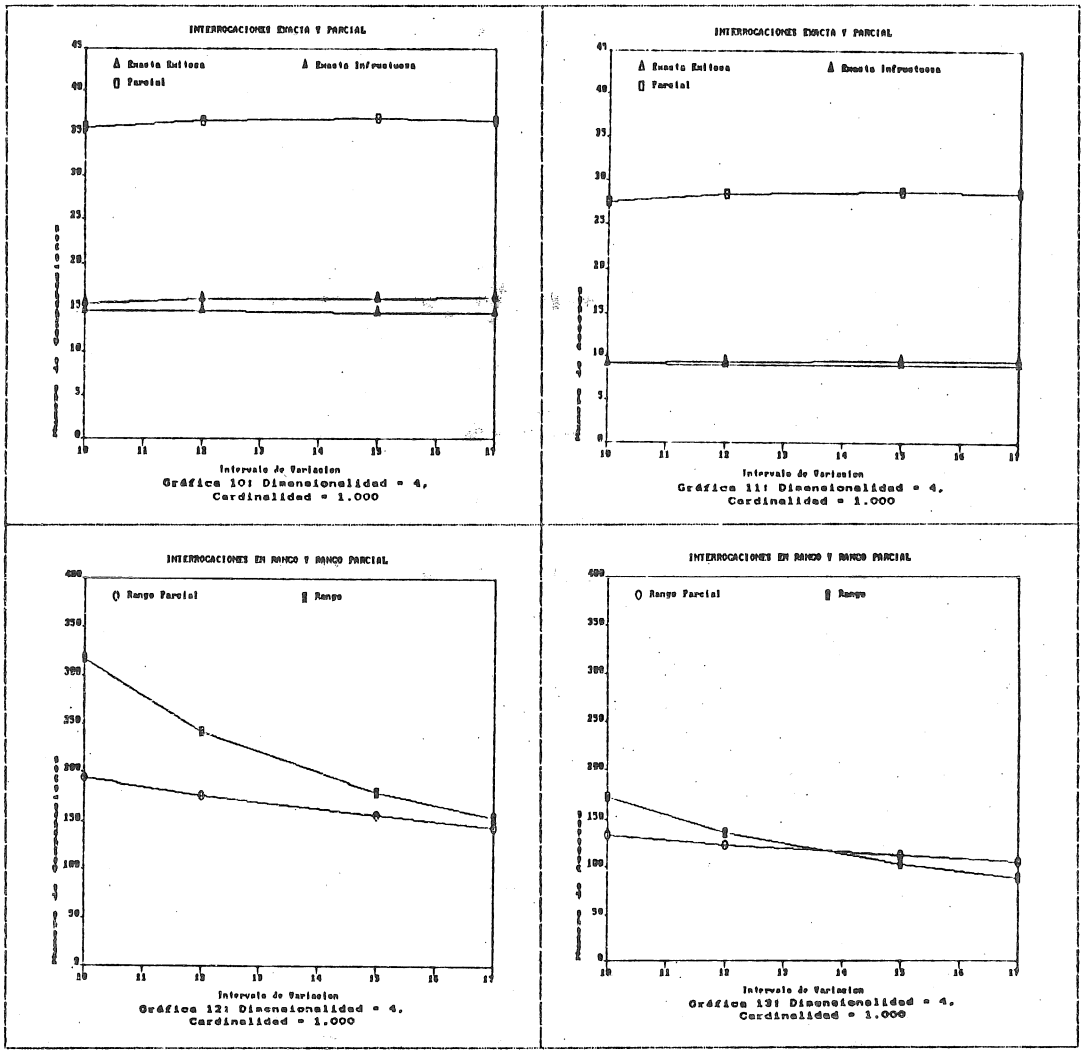
En las gráficas 4 y 5, para el caso de la interrogación exacta se obtiene que tanto el número de comparaciones como el de accesos es menor en las interrogaciones infructuosas que en las exitosas, aunque la diferencia no es muy notable. Esto se debe a que en las búsquedas infructuosas el proceso termina antes, una vez queda descartada la posibilidad de encontrar el registro; mientras que en las exitosas se llega hasta el nivel de nodos hoja para encontrar el registro solicitado. Para la interrogación parcial se observa un gran aumento tanto en el número de comparaciones como en el de accesos con respecto a la búsqueda anterior. Hecho muy natural teniendo en cuenta que en la búsqueda parcial, tras encontrar una clave sin especificar, se expande a través de los tres subárboles del nivel inferior.

En las gráficas 6 y 7 se observa que el número de comparaciones y de accesos, en la interrogación en rango presenta un gran crecimiento ante un aumento de la cardinalidad, puesto que el tamaño de los árboles binarios de cada nivel crece con éstos. Sin embargo, la interrogación en rango parcial no se ve tan afectada debido a que al existir claves sin especificar, aunque se produce una expansión, se evita el recorrido de los árboles binarios de los niveles asociados a las claves no especificadas.

En las gráficas 8 y 9 se observa que en la interrogación en rango la diferencia entre el número de registros recuperados y el de celdas es mínimo. Sin embargo, en las interrogaciones parcial y en rango parcial el número de registros recuperados tiene un crecimiento constante y notable al aumentar la cardinalidad, mientras que el

número de celdas, aunque aumenta el crecimineto es ligero, esto es un indicador de lo que favorece el Arbol Quintario a las interrogaciones parciales. Además el número de registros recuperados es comparativamente inferior al de la interrogación en rango parcial, esto es debido a que las interrogaciones en rango representan un volumen de búsqueda pequeño con respecto al número de registros existente, mientras que el de las interrogaciones en rango parcial es mayor.

B) El parámetro variable es el intervalo de variación de los valores de clave:

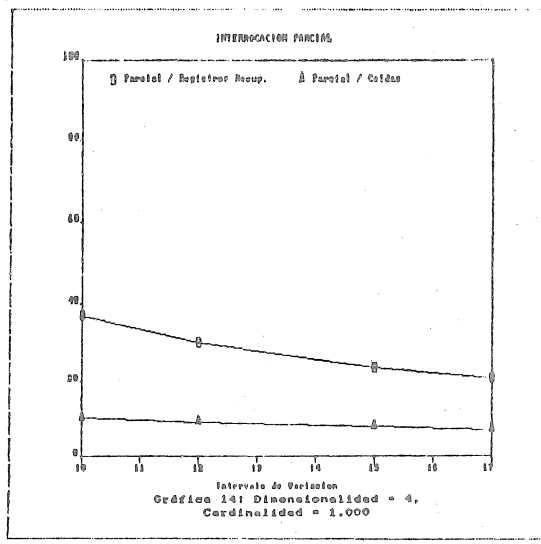


Gráfica 10: Dimensionalidad = 4, Cardinalidad = 1.000

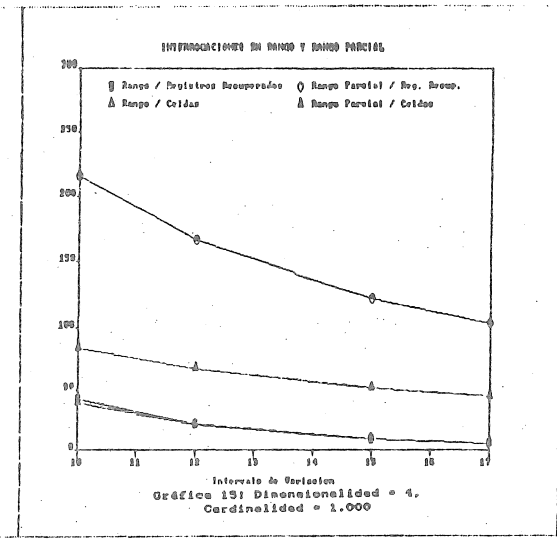
Gráfica 11: Dimensionalidad = 4, Cardinalidad = 1.000

Gráfica 12: Dimensionalidad = 4, Cardinalidad = 1.000

Gráfica 13: Dimensionalidad = 4, Cardinalidad = 1.000



Gráfica 14: Dimensionalidad = 4, Cardinalidad = 1.000



Gráfica 13: Dimensionalidad = 4, Cardinalidad = 1.000

En las gráficas 10 y 11 se muestran los resultados de las pruebas para la interrogación exacta. Las comparaciones en las búsquedas exitosas presentan un crecimiento insignificante y las infructuosas presentan, incluso, oscilaciones en torno a la media. Un hecho parecido ocurre con el número de accesos.

Estos resultados responden, por un lado, a que al aumentar el intervalo de variación, las estructuras de árbol binario de cada nivel tienden a crecer, lo que explicaría el aumento del número de comparaciones y accesos; por otro lado, debido a que el número de registros permanece constante, y no es muy elevado, el crecimiento de las estructuras de árbol binario está limitado, y éstas tienen menor altura conforme se va descendiendo por el árbol. Al descender de nivel, no se utilizan todos los valores posibles del intervalo de variación de cada clave. Se realizaron pruebas con ficheros con un número de registros mayor en las que se comprobó que, efectivamente, disminuye el efecto del segundo factor mencionado.

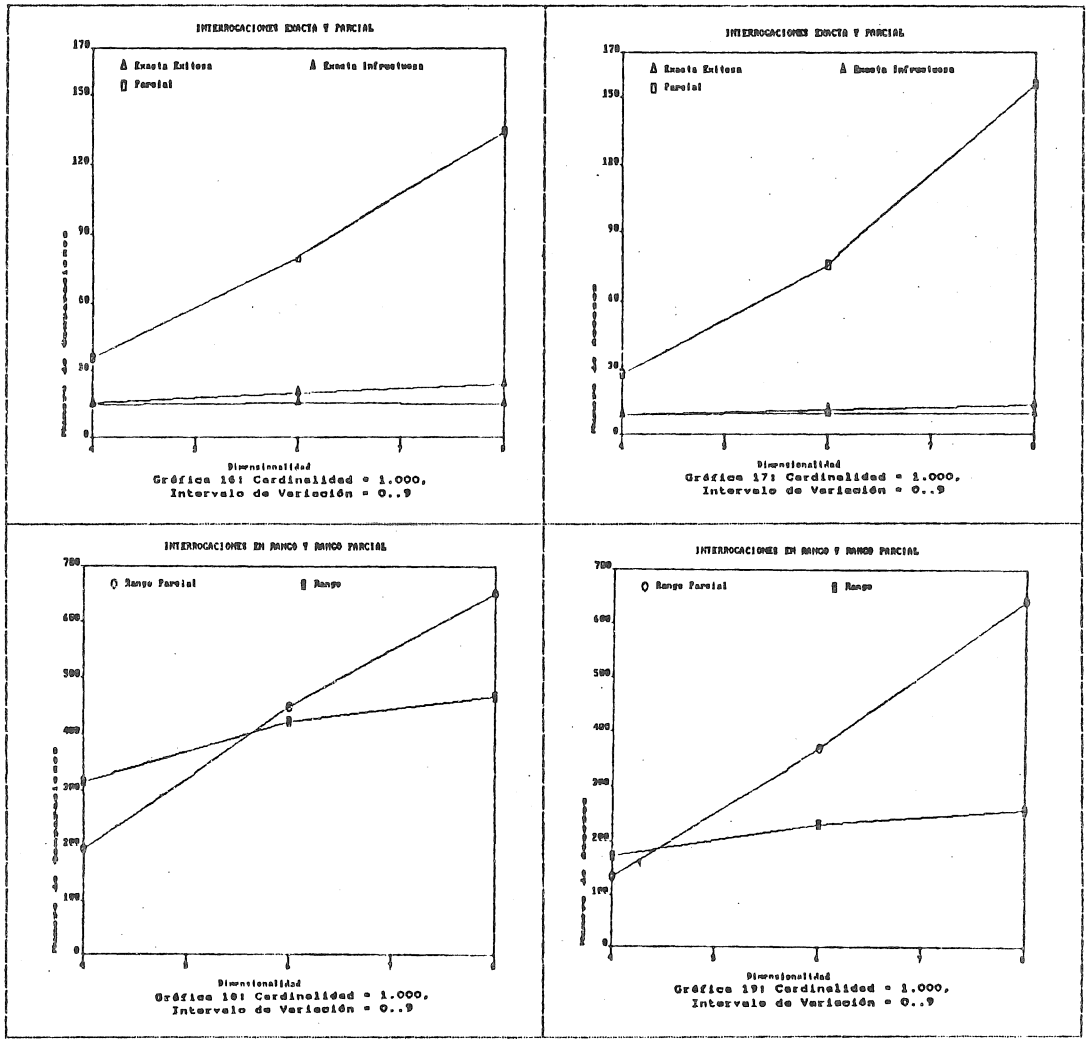
Para el caso de la interrogación parcial, influyen los mismos factores que en la interrogación exacta. Además, también hay un incremento general con respecto a ésta, por la expansión hacia los tres subárboles en el nivel inferior cuando se encuentra una clave sin especificar. Sin embargo, los registros recuperados, gráfica 14, presentan un decrecimiento a medida que el valor del intervalo de variación se hace mayor; esto se debe a que existe una mayor probabilidad de generar interrogaciones que supongan la recuperación de un menor número de registros. Así mismo, el número de celdas tratadas en la interrogación parcial, gráfica 14, presenta una ligera disminución al aumentar el tamaño del intervalo de variación de los valores de clave. Esto se debe a que aumenta la posibilidad de llegar a celdas vacías las cuales no se contabilizan.

En las interrogaciones en rango y rango parcial, observando las gráficas 12 y 13 se tiene que, al aumentar el tamaño de intervalo de variación se produce un decrecimiento tanto en el número de comparaciones como en el de accesos en ambas búsquedas. Esto es una muestra de cómo está estructurado el árbol decreciendo la altura por niveles de los árboles binarios al aumentar el intervalo de

variación. Además, en este caso, al igual que se observó en el apartado A, la interrogación en rango se ve más afectada; el orden de decrecimiento de ésta es mayor. Se realizaron pruebas con ficheros con más registros, y los resultados también se suavizaron.

En los números de celdas y registros recuperados influye, tal como se observa en la gráfica 15, que hay una mayor probabilidad de generar una interrogación que permita la recuperación de un menor número de registros al aumentar el tamaño del intervalo de variación. Se vuelve a verificar la coincidencia de los valores de los números de celdas y registros recuperados para la interrogación en rango.

C) El parámetro variable es la dimensionalidad:

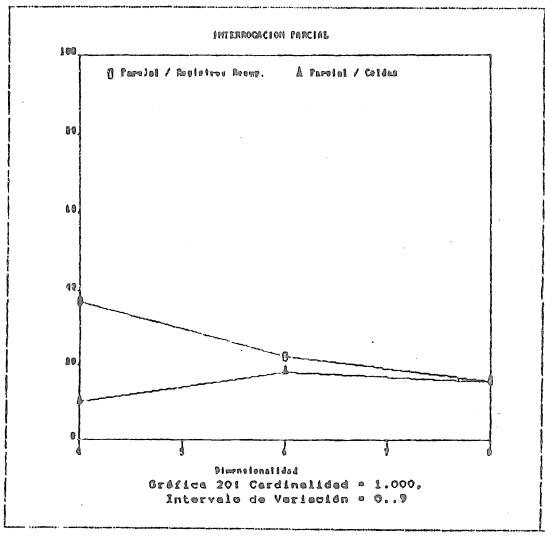


Gráfica 16: Cardinalidad = 1.000, Intervalo de Variación = 0..9

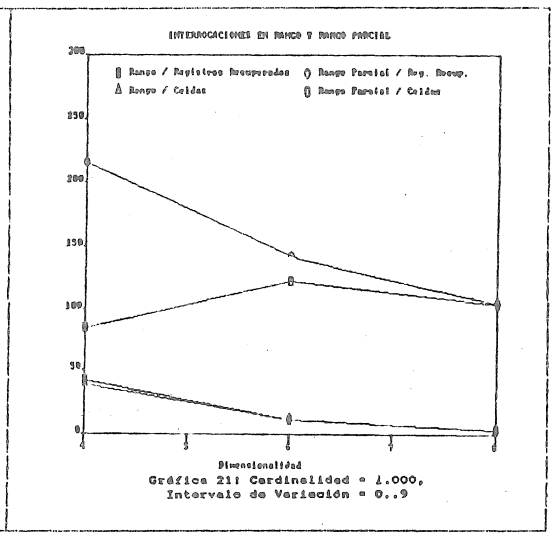
Gráfica 17: Cardinalidad = 1.000, Intervalo de Variación = 0..9

Gráfica 18: Cardinalidad = 1.000, Intervalo de Variación = 0..9

Gráfica 19: Cardinalidad = 1.000, Intervalo de Variación = 0..9



Gráfica 20: Cardinalidad = 1.000, Intervalo de Variación = 0..9



Gráfica 21: Cardinalidad = 1.000, Intervalo de Variación = 0..9

En las gráficas 16 y 17 se observa que en la interrogación exacta el número de accesos y de comparaciones aumenta con la dimensionalidad. En las búsquedas infructuosas, con un número de claves mayor es más probable detectar antes la no existencia del registro buscado, debido a ésto, tanto el número de comparaciones como el de accesos es menor que para las exitosas.

En la interrogación parcial, los números de accesos y comparaciones aumentan más rápidamente a medida que aumenta la dimensionalidad, ya que el árbol tiene más niveles.

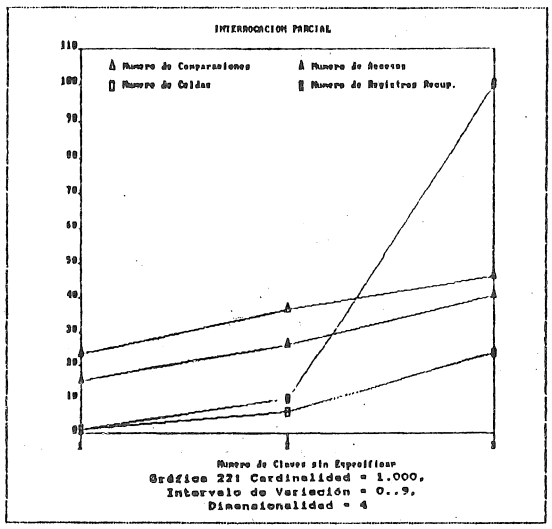
En las gráficas 18 y 19 se observa el comportamiento de la estructura en las interrogaciones en rango y rango parcial. Se muestra que en las interrogaciones en rango, aunque los números de accesos y comparaciones aumentan con la dimensionalidad, este crecimiento es menor que el registrado para la interrogación en rango parcial. Esto se debe a que en la interrogación en rango a medida que aumenta la dimensionalidad hay más búsquedas infructuosas por lo que se produce el mismo efecto que en la interrogación exacta con búsquedas infructuosas. De hecho, se realizaron pruebas con búsquedas exitosas y se produjo un incremento mayor. En la interrogación en rango parcial ocurre lo contrario, el crecimiento resultante es mayor debido a los mismos factores que influyen en la interrogación parcial.

En cuanto al número de celdas y registros recuperados, se pueden realizar las siguientes observaciones: en la interrogación en rango el número de celdas tratadas es pequeño, y disminuye conforme aumenta la dimensionalidad, debido a que las interrogaciones en rango seleccionadas de forma aleatoria, la mayoría de las veces no recuperan información por ser el número de registros muy pequeño, en comparación con el número de todas las posibles combinaciones que pueden tomar tales registros. En las interrogaciones parcial y en rango parcial se produce un crecimiento en el número de celdas al aumentar el número de claves de 4 a 6 y un ligero decrecimiento al aumentarlo de 6 a 8; ésto último se debe a la limitación del número de registros. El número de registros recuperados siempre disminuye demostrando que decrece la probabilidad de encontrar un registro al

aumentar la dimensionalidad.

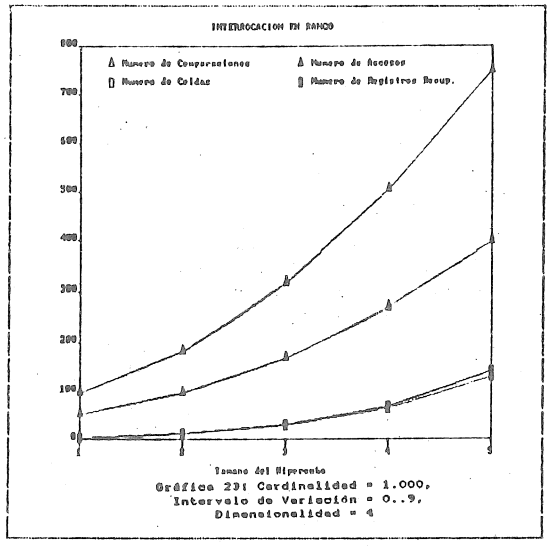
3.5) NUMERO DE ACCESOS Y COMPARACIONES EN FUNCION DEL NUMERO DE CLAVES SIN ESPECIFICAR Y EL TAMAÑO DEL HIPERCUBO:

A) Interrogación parcial en función del número de claves sin especificar:



En la gráfica 22 se observa que a medida que crece el número de claves sin especificar, crece, asímismo, el número promedio de accesos, comparaciones, celdas y registros recuperados en una proporción mucho más representativa. Estos resultados eran los esperados si se tiene en cuenta que al aumentar el número de claves sin especificar, aumenta la probabilidad de que sean más los registros que cumplan con las características especificadas por la interrogación.

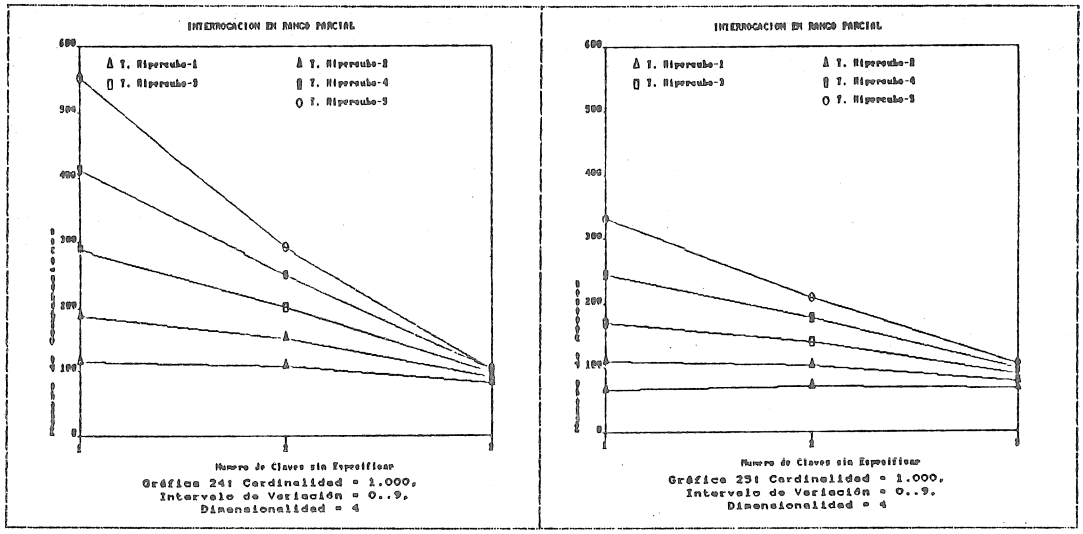
B) Interrogación en rango en función del tamaño del hipercubo:



Gráfica 23: Cardinalidad = 1.000,
Intervalo de Variación = 0..9,
Dimensionalidad = 4

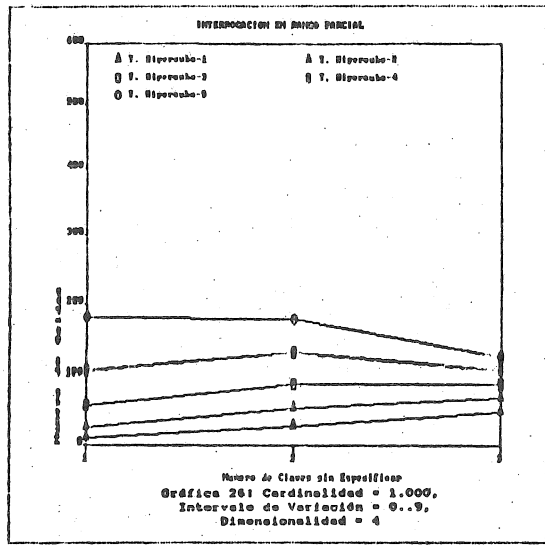
En la gráfica 23 se observa el comportamiento de la interrogación en rango ante variaciones del tamaño del hipercubo. En cuanto a la variación del número de accesos, comparaciones, celdas tratadas y registros recuperados, se tiene que, a medida que el tamaño del hipercubo aumenta, evidentemente crecen los cuatro parámetros. Nuevamente, las curvas del número de celdas y registros recuperados casi coinciden.

C) Interrogación en rango parcial en función del tamaño del hipercubo y del número de claves sin especificar:



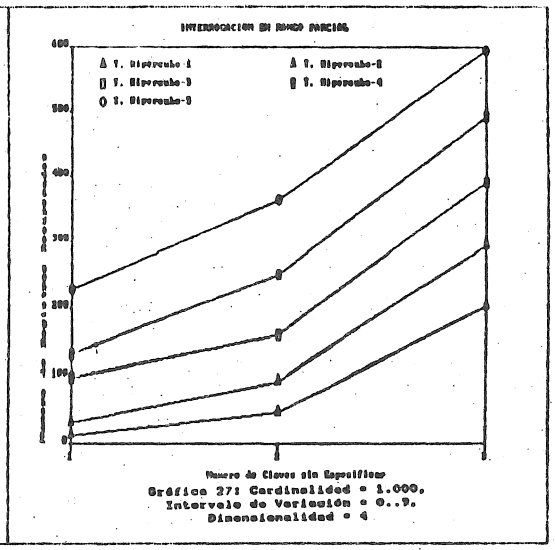
Gráfica 24: Cardinalidad = 1.000,
Intervalo de Variación = 0..9,
Dimensionalidad = 4

Gráfica 23: Cardinalidad = 1.000,
Intervalo de Variación = 0..9,
Dimensionalidad = 4



Número de Claves sin Especificar

Gráfica 26: Cardinalidad = 1.000, Intervalo de Variación = 0..9, Dimensionalidad = 4



Número de Claves sin Especificar

Gráfica 27: Cardinalidad = 1.000, Intervalo de Variación = 0..9, Dimensionalidad = 4

En las gráficas 24 y 25 se observa un incremento en el número de comparaciones y accesos al aumentar el tamaño del hipercubo, como cabe esperar ya que aumenta la información a recuperar, tal y como se observa en la gráfica 27, en la que, evidentemente, se refleja que el número de registros recuperados crece con el tamaño del hipercubo y con el número de claves sin especificar.

Para cada tamaño de hipercubo, al aumentar el número de claves sin especificar, el comportamiento general tiende a descender en los promedios de los números de comparaciones y accesos, debido a que una clave sin especificar significa tres accesos al siguiente nivel, mientras que una especificada supone un cierto número de comparaciones y accesos para situarse en todos los nodos cuyos campos de información estén dentro del rango especificado en la interrogación, y los accesos al siguiente nivel que son función del tamaño del hipercubo.

En la gráfica 26 el número de celdas visitadas crece con el tamaño del hipercubo, ahora bien, para cada tamaño la variación del número de claves sin especificar provoca un comportamiento variable de evolución continua, indicando que se accede a celdas con más de un registro.

4) ESTUDIO PROBABILISTICO DE LA INFORMACION RECUPERADA

Tras realizar un estudio probabilístico del número esperado de registros a recuperar en los tres tipos de interrogaciones, a través de:

Interrogación parcial:

$$\frac{\sum_{j=1}^{nd-1} vi^{**}j}{nd-1} * \frac{n}{vi^{**}nd}$$

Interrogación en rango:

$$\frac{\sum_{j=1}^{nd-1} \left(\sum_{p=2}^{6} p^{**}(nd-j)) * vi^{**}j \right)}{5 * (nd-1)} * \frac{n}{vi^{**}nd}$$

Interrogación en rango parcial:

$$\frac{\sum_{p=2}^{6} p^{**}nd}{5} * \frac{n}{vi^{**}nd}$$

Nótese que en este caso, vi se trata como una constante ya que tiene el mismo valor para todas las claves del fichero.

Se comparan los resultados con los obtenidos en el estudio experimental, tal y como se observa en las tablas 2, 3 y 4, en la interrogación parcial los valores, prácticamente, coinciden; y en las interrogaciones en rango y en rango parcial, la diferencia es muy pequeña.

Cardinalidad	Registros Recuperados					
	I. PARCIAL		I. EN RANGO		I. EN RANGO PARCIAL	
	Experim.	Probab.	Experim.	Probab.	Experim.	Probab.
500	11,83	11,90	4,48	4,49	61,60	62,12
1.000	23,93	23,80	8,70	8,98	122,37	124,25
2.000	47,80	47,60	17,23	17,98	245,13	248,49
4.000	95,54	95,21	35,07	35,93	492,99	496,99

Tabla 2: Dimensionalidad = 4, Intervalo de Variación = 0..14

Intervalo de Variación	Registros Recuperados					
	I. PARCIAL		I. EN RANGO		I. EN RANGO PARCIAL	
	Experim.	Probab.	Experim.	Probab.	Experim.	Probab.
0.. 9	37,02	37,00	42,17	45,48	216,14	222,67
0..11	30,29	30,29	21,24	21,93	166,86	169,75
0..14	23,93	23,80	8,70	8,98	122,37	124,25
0..16	20,86	20,83	5,17	5,45	103,73	105,16

Tabla 3: Dimensionalidad = 4, Cardinalidad = 1.000

Dimensionalidad	Registros Recuperados					
	I. PARCIAL		I. EN RANGO		I. EN RANGO PARCIAL	
	Experim.	Probab.	Experim.	Probab.	Experim.	Probab.
4	37,02	37,00	42,17	45,48	216,14	222,67
6	22,21	22,22	11,82	13,43	143,09	147,58
8	15,81	15,87	3,67	4,29	105,39	108,41

Tabla 4: Cardinalidad = 1.000, Intervalo de Variación = 0..9

Además, las tablas 5, 6 y 7 muestran los resultados de dicho estudio para el caso concreto de variación del tamaño del hipercubo y número de claves sin especificar. Se comprueba que se repite la situación anterior.

Claves sin especificar	Registros Recuperados Experim.	Probab.
1	1,01	1,00
2	10,12	10,00
3	99,93	100,00

Tabla 5: Interrogación Parcial

Tamaño de hipercubo	Registros Recuperados Experim.	Probab.
1	3,44	1,60
2	11,46	8,10
3	30,84	25,60
4	69,22	62,50
5	141,61	129,60

Tabla 6: Interrogación en Rango

Tamaño del hipercubo	Claves sin especificar	Registros Recuperados Experim.	Probab.
1	1	12,46	8,00
1	2	47,84	40,00
1	3	202,26	200,00
2	1	32,06	27,00
2	2	93,99	90,00
2	3	294,28	300,00
3	1	68,89	64,00
3	2	160,63	160,00
3	3	392,12	400,00
4	1	131,67	125,00
4	2	249,79	250,00
4	3	492,08	500,00
5	1	227,83	216,00
5	2	363,13	360,00
5	3	594,82	600,00

Tabla 7: Interrogación en Rango Parcial

CONCLUSIONES

Se presenta en este trabajo un estudio experimental de la estructura de Arbol Quintario respecto a la ocupación y al tiempo de respuesta para las interrogaciones exacta, parcial, en rango y en rango parcial.

Para valores moderados de los parámetros: cardinalidad, dimensionalidad e intervalo de variación de los valores de clave, se han obtenido árboles de grandes volúmenes; el parámetro que más influye es la dimensionalidad.

El tiempo de respuesta medido en función del número de accesos y comparaciones es bueno para interrogaciones exactas y parciales y aceptable para interrogaciones en rango y en rango parcial, siendo la dimensionalidad el parámetro que influye de forma negativa para este factor.

REFERENCIAS

1. BENTLEY, J.L., and FRIEDMAN, J.H. Data structures for range searching. Computing Surveys 11,4 (1979), 397-409.

2. CARDENAS, A.F. and SAGAMANG, J.P. Doubly-Chained Tree Data Base Organization-Analysis and Design Strategies. Comput. J. 20(1977), 15-26.

3. GUTING, H. and KRIEGEL, H.P. Multidimensional B-Tree: An Efficient Dynamic File Structure for Exact Match Queries, 1980.

4. GUTING, H. and KRIEGEL, H.P. Multidimensional B-Tree used for Partial Match Retrieval. Computer Science Technical Report, Universitat Dortmund, 1980.

5. KASHYAP, R.L.; SUBAS, S.K. and YAO, S.B. Analysis of the Multiple-Attribute-Tree Data Base Organization. IEEE Trans. Softw. Eng. SE-6 (1977) 451-467.

6. KNUTH, D. The Art of Computer Programing. Volumen 3/Sorting and Searching, Addison-Wesley Publishing Company, 1973.

7. LEE, D.T., and WONG, C.K. Quintary Trees: A file structure for multidimensional database systems, vol 5, N° 3, 1980, 339-353.

8. LIEN, Y.E., TAYLOR, C.E., and DRISCOLL, J.R. A design and implementation of a relational data base. Proc. Very Large Data Base Conf., 1975.

9. SANTANA, O.; CABRERA, J.; DIAZ, M. y MAYOR, O. Análisis Experimental de los Esquemas de Inserción del Arbol B. Informática Automática, 1986.

10. TREMBLAY, J.P. and SORENSON, P.G. An Introduction to Data Structures with Applications, Mc Graw Hill International Book Company, 1984.

ESQUEMAS Y ESTRUCTURA PARA LA BUSQUEDA DE LAS PALABRAS MAS SIMILARES A UNA DADA

O. SANTANA, M. DIAZ, O. MAYOR, J. REYES
E.U. Informática; Universidad Politécnica de Canarias
Aptdo. 550, Las Palmas de Gran Canaria, España.

En este trabajo se aborda el problema de la recuperación desde un diccionario del conjunto de palabras más similares a una palabra de búsqueda. Se introduce una distancia, DIT, invariante ante las trasposiciones, a fin de reducir el tiempo de proceso mediante la restricción del uso de la distancia direccional de Wagner y Fisher, según se muestra en el estudio experimental. Asimismo, se estudian variaciones en el esquema de búsqueda, a fin de poder desplazarse sobre el binomio exactitud-rapidez.

Seguidamente se estructuran los componentes de la DIT, produciendo con ello una disminución radical del tiempo de proceso. Se estudian las alternativas de optimización en la propia estructura, lo que conduce a nuevas reducciones del tiempo de proceso e incluso del almacenamiento utilizado por la estructura.

Aparte de las medidas estáticas tomadas durante el estudio de la estructura, se lleva a cabo un estudio experimental de su comportamiento frente al esquema de búsqueda. Al igual que en el del esquema secuencial se plantea de nuevo el dilema entre exactitud y velocidad de proceso. Por último, se introduce una optimización al esquema de búsqueda estructurado, con la consiguiente disminución del tiempo de proceso.

ESQUEMAS Y ESTRUCTURA PARA LA BUSQUEDA DE LAS PALABRAS MAS SIMILARES A UNA DADA

O. SANTANA, M. DIAZ, O. MAYOR, J. REYES
E.U. Informática; Universidad Politécnica de Canarias
Aptdo. 550, Las Palmas de Gran Canaria, España.

Resumen:

El problema que se aborda en este trabajo consiste en la recuperación desde un diccionario del conjunto de palabras más similares a una palabra de búsqueda. Se introduce el cálculo de una distancia que es independiente de la posición que ocupan los caracteres en las palabras, utilizada como un filtro para el cálculo de la distancia de Wagner y Fischer [6] a fin de mejorar el rendimiento del esquema de búsqueda cuando sólo se utiliza como criterio de similitud esta última distancia.

0.- INTRODUCCION.

Desde que surgieron los computadores, el esfuerzo investigador ha estado mucho más dirigido hacia las máquinas que hacia las personas que las usan. Uno de los factores que más complican la comunicación hombre-máquina reside en el hecho de que éstas exigen perfección en sus entradas, perfección que el hombre apenas puede darle. Los ordenadores son muy ágiles en descubrir errores, pero desgraciadamente su agilidad parece acabar ahí, su respuesta se suele limitar a un mensaje de error y una invitación para que la persona que lo maneja corrija su orden y la introduzca de nuevo. ¿Por qué los computadores han de ser tan intolerantes con los errores si son tan certeros en descubrirlos?. Cuando las personas se comunican, los errores se suceden con frecuencia, pero éstos se ignoran generalmente, permitiendo que la comunicación sea fluida. Sería sumamente interesante que los ordenadores tuvieran una tolerancia a los errores similar a la de las personas.

Alberga[2] usó matrices binarias para evaluar medidas de aproximación entre dos palabras. Szanser[4] desarrolló un proceso matemático de coincidencia elástica que era efectivo en un 95%. Morgan[3] realizaba un o-exclusivo entre dos palabras para determinar si había ocurrido un único error simple. Wagner y Fischer[6] desarrollaron un algoritmo que puede determinar la distancia entre dos palabras medida como el mínimo coste de la secuencia de operaciones de edición: sustitución, inserción y extracción.

En las secciones 1 y 2 se establecen los tipos de errores de edición que puede sufrir una palabra, se define y calcula la distancia direccional, DD (distancia de Wagner y Fischer[6]). En la sección 3 el problema se ha planteado como la recuperación secuencial a partir de un diccionario del conjunto de palabras que tengan una mínima distancia direccional a la palabra de búsqueda. En la sección 4 se define un tipo de distancia, DIT, invariante a las trasposiciones, cuyo valor es siempre menor o igual al duplo de la distancia direccional, DD. En la sección 5 el esquema de búsqueda secuencial de la sección 3 se ha optimizado debido a la inclusión del cálculo de la distancia DIT como filtro adaptivo para el cálculo de las distancias direccionales. Se propone una estructuración de los componentes que intervienen en el cálculo de DIT en la sección 6 [1,5]. En las secciones 7, 8 y 9 se estudian los parámetros más relevantes de dicha estructura, así como los criterios adoptados a fin de optimizarla. En la sección 10 se describe

el esquema de búsqueda de las palabras más similares a una dada mediante un planteamiento integrado DIT y DD recorriendo la estructura propuesta. Los tiempos promedios de búsqueda utilizando dicha estructura se presentan y se comparan en la sección 11 con los correspondientes a la recuperación secuencial. En la sección 12 se experimenta con una modificación ulterior del esquema de búsqueda propuesto, en el cual se garantiza que no existe ninguna palabra distorsionada por encima de un valor dado. La sección 13 presenta las conclusiones de este trabajo.

1.- DISTANCIA DIRECCIONAL.

1.1 ERRORES POSIBLES EN UNA PALABRA.

Wagner considera tres tipos posibles de errores básicos que pueden darse en una palabra:

1.- Sustitución de un carácter por otro.
2.- Extracción de un carácter.
3.- Inserción de un carácter.

Se considera que todas las palabras que se mencionan en lo sucesivo están compuestas por una secuencia de caracteres extraídos de un alfabeto $\{\alpha 1,..,\alpha m\}$.

Sea X una palabra, sea X<i> el carácter que está en la i_ésima posición de la palabra X; X<i:j> la secuencia de caracteres que va de X<i> a X<j> ambos inclusive, si i>j entonces $X<i:j>=\mu$, la hilera nula. |X| denota la longitud de X.

1.2 OPERACIONES DE EDICION.

Una operación de edición es el par $(\theta,\Omega)<>(\mu,\mu)$ donde θ,Ω son palabras de longitud menor o igual que 1, es decir, o son μ o son un único caracter. La palabra Y resulta de aplicar (θ,Ω) a la palabra X, que se escribe X->Y, si $X=\sigma\theta\gamma$ e $Y=\sigma\Omega\gamma$.

Dada (θ,Ω), es una operación de sustitución si $\theta<>\mu$, $\Omega<>\mu$ y $\theta<>\Omega$; es una operación de extracción si $\Omega=\mu$ y es una operación de inserción si $\theta=\mu$.

Sea S una secuencia s1, s2, ..., sn de operaciones de edición. Una derivación-S de X hasta Y es una secuencia de palabras X0, X1, ..., Xn tal que X=X0, Y=Xn y Xj-1->Xj vía sj para j=1,..,n.

Se dice que S convierte X en Y si hay una derivación-S de X hasta Y.

1.3 FUNCION DE COSTE.

Sea Γ una función arbitraria de coste que asigne a cada operación de edición (θ,Ω) un número real no negativo $\Gamma(\theta,\Omega)$. Se puede extender Γ a la secuencia S definiendo:

$$\Gamma(S)=\sum_{j=1}^{n}\Gamma(sj) \quad \text{si } n>=1 \text{ y } \Gamma(S)=0 \text{ si } n=0$$

Se llama distancia direccional de edición al mínimo coste de todas las secuencias de edición que transformen X en Y. Formalmente $\sigma(X,Y)=\min\{\Gamma(S)\}$.

2.- CALCULO DE LA DISTANCIA DIRECCIONAL.

Si X e Y son dos palabras cualesquiera, se define $X(i)=X\langle 1:i\rangle$, $Y(j)=Y\langle 1:j\rangle$ y $DD(i,j)=\sigma(X(i),Y(j))$.

$$DD(0,0)=0$$

$$DD(i,0)=\sum_{r=1}^{i}\Gamma(X\langle r\rangle,\mu) \qquad DD(0,j)=\sum_{r=1}^{j}\Gamma(\mu,Y\langle r\rangle)$$

Es decir, el coste de convertir μ en sí mismo es cero, el coste de reducir la palabra X a μ es la suma de los costes de extraer todos los caracteres de X y el coste de obtener Y a partir de μ es el la suma de los costes de añadir cada uno de los caracteres de Y.

Para calcular $DD(i,j)$ $i=1..|X|$, $j=1..|Y|$ hay que tener en cuenta tres casos.

1.- Convertir $X(i-1)$ en $Y(j-1)$ y $X\langle i\rangle$ en $Y\langle j\rangle$.
2.- Convertir $X(i-1)$ en $Y(j)$ y extraer $X\langle i\rangle$.
3.- Convertir $X(i)$ en $Y(j-1)$ y añadir $Y\langle j\rangle$.

De estas tres posibilidades se escoge aquella cuyo coste sea mínimo.

Por lo tanto:

$$DD(i,j)=\min\{DD(i-1,j-1)+\Gamma(X\langle i\rangle,Y\langle j\rangle),\ DD(i-1,j)+\Gamma(X\langle i\rangle,\mu),\ DD(i,j-1)+\Gamma(\mu,Y\langle j\rangle)\}$$

Y la distancia direccional entre X e Y viene dada por $DD(|X|,|Y|)$.

3.- CONJUNTO DE PALABRAS MAS SIMILARES USANDO SOLO DD.

Con esta distancia direccional se puede definir el conjunto de palabras similares a una palabra dada:

Sea D el conjunto de todas las palabras correctas cuyo cardinal es ND. Sea B una palabra de búsqueda, sea P un subconjunto de D definido como sigue:

$$P=\{D1,D2,\ldots,Dn/Di\in D\ i=1..n \text{ y } \sigma(B,Di)=\sigma(B,Dj)=DD\ i,j=1..n \text{ y } DD \text{ es mínimo}\}$$

Entonces P es el conjunto de palabras que más se aproximan a B.

3.1 ESQUEMA DE BUSQUEDA DDS. RESULTADOS EXPERIMENTALES.

El esquema de búsqueda DDS realiza una búsqueda secuencial en D, determinando la DD de cada palabra con respecto a B. Si la DD es inferior a la distancia mínima actual se actualiza la distancia mínima y el conjunto de palabras más similares a B. Si es igual se actualiza P.

Si es mayor se prueba la palabra siguiente en D. Si en algún momento DD es cero entonces B está en D.

En este trabajo se supone que el coste de cualquier operación de edición es uno.

Se evaluaron el porcentaje de aciertos, el porcentaje de respuesta única, así como el promedio de multiplicidad de respuesta utilizando el esquema de búsqueda DDS con un diccionario formado por las 2089 palabras de uso más frecuente del idioma Castellano.

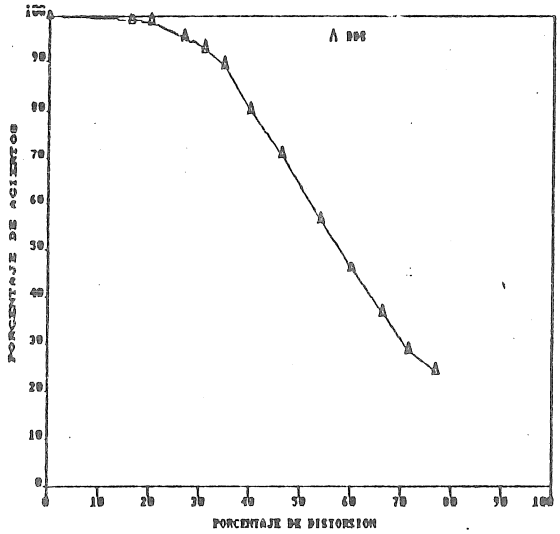


Figura 1 "Gráfica de porcentajes promedio de aciertos en el esquema de búsqueda DDS"

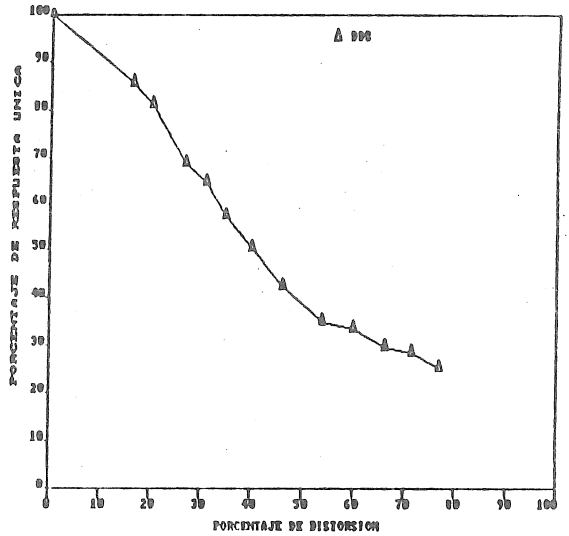


Figura 2 "Gráfica de los porcentajes promedio de respuesta única en el esquema de búsqueda DDS"

Se dice que una palabra X ha sufrido una distorsión π originando otra palabra Y si:

$$\pi = DD(X,Y)/|Y|$$

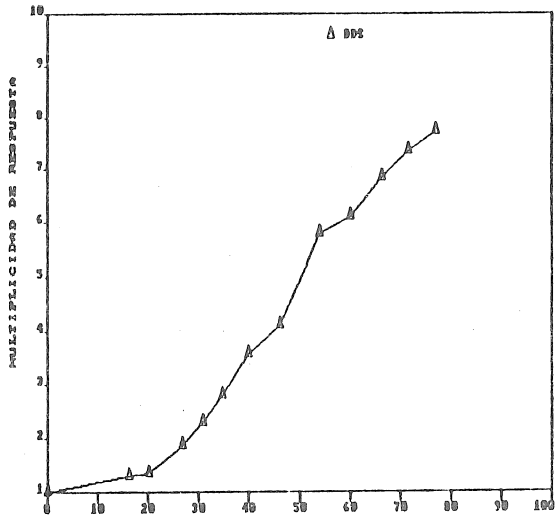


Figura 3 "Gráfica de los promedios de multiplicidad de respuesta en el esquema de búsqueda DDS".

El porcentaje de aciertos se mantiene razonablemente alto hasta distorsiones del 30%, figura 1, por allá del cual cae rápidamente. Para distorsiones del 50% la probabilidad de que el resultado sea verdadero se sitúa en torno al 0.5.

Las respuestas únicas no son demasiado abundantes incluso para distorsiones bajas, ver figura 2.

En la figura 3 se observa que el número de palabras como respuestas no-únicas aumenta rápidamente a partir del 20% de distorsión.

3.2 LIMITE SUPERIOR DE LA DD EN FUNCION DE LA DISTORSION MAXIMA PERMITIDA.

Si una palabra X ha sufrido una distorsión π originando otra palabra Y cuya DD(X,Y) se obtiene a partir de una secuencia que contiene $\Phi1$ inserciones, $\Phi2$ extracciones y $\Phi3$ sustituciones se tiene que:

$$\Phi1+\Phi2+\Phi3 = DD(X,Y)$$

despejando $\Phi2$: $\Phi2 = DD(X,Y) - \Phi1 - \Phi3$

debido a que: $\Phi1, \Phi2, \Phi3 >= 0$

se tiene que: $\Phi2 <= DD(X,Y)$

Por otra parte: $|X| = |Y| + \Phi2 - \Phi1$

$$DD(X,Y) = \pi * |X| = \pi * (|Y| + \Phi2 - \Phi1)$$
$$DD(X,Y) <= \pi * (|Y| + \Phi2)$$
$$DD(X,Y) <= \pi * (|Y| + DD(X,Y))$$

de donde:

$$DD(X,Y) <= \pi/(1-\pi) * |Y|$$

De lo que se infiere que si la palabra de búsqueda B ha sufrido una distorsión menor o igual que π la inicialización de la distancia direccional máxima es:

$$DDM = \pi/(1-\pi) * |B|$$

3.3 COSTE DE COMPUTACION Y ALMACENAMIENTO DEL ESQUEMA DE BUSQUEDA DDS.

Según Wagner la complejidad de cálculo de la DD(X,Y) entre dos palabras X e Y es $O(|X|*|Y|)$. Debido a que la búsqueda es exhaustiva, en el caso de que una palabra no se encuentre en el diccionario, la complejidad de cálculo de DDS en toda la base de datos y el coste de almacenamiento de esta estructura secuencial son respectivamente:

$$O(\ |B|* \sum_{j=1}^{ND} |Dj|\ ) \qquad O(\ \sum_{j=1}^{ND} |Dj|\ )$$

El bajo perfomance del esquema de búsqueda DDS, ver figura 4, conduce a la pregunta: ¿Es necesario el cómputo de DD(B,Dj) para todas

las palabras del diccionario D? Se verá que no.

4.- DISTANCIA INVARIANTE TRASPOSICIONAL.

4.1 DEFINICION.

Sean X e Y dos palabras, se definen xαi e yαi como el número de veces que αi esté contenido en X y en Y respectivamente. Se observa que:

$$|X| = \sum_{i=1}^{m} x\alpha i$$

Donde m es el número de caracteres que componen el alfabeto.

Se define la distancia invariante trasposicional DIT como:

$$DIT(X,Y) = \sum_{i=1}^{m} [abs(x\alpha i - y\alpha i)] + abs(|X| - |Y|)$$

Hay que tener en cuenta que si bien en el cálculo de la DD no hay nada que se pueda tener precalculado ya que todo el cálculo depende tanto de B como de Dj. En el cómputo de DIT el vector de frecuencias de caracteres de B es absolutamente independiente de la correspondiente Dj con la que se esté comparando por lo que se puede calcular sólo una vez al principio del proceso. Si además junto a cada palabra de la base de datos se almacena su vector de frecuencias de caracteres, entonces la complejidad de cálculo de DIT secuencial, DITS, es:

$$O(ND*m)$$

y el coste de almacenamiento será:

$$O(ND*m + \sum_{j=1}^{ND} |Dj|)$$

4.2 COMPORTAMIENTO DE DIT RESPECTO A DD.

Obsérvese como se comporta la distancia DIT ante sustituciones, inserciones y extracciones.

Sea X una palabra que se convierte en Y1 mediante Φ1 inserciones, Φ2 extracciones, Φ3 sustituciones. Sea C el subconjunto de los caracteres de X e Y que no han sido afectados por ninguna operación de edición, C1 el subconjunto de caracteres afectados por las inserciones, C2 el subconjunto de caracteres afectados por extracciones y C3 el

subconjunto de caracteres afectados por sustituciones. Supóngase por simplicidad que:

$$C1 \cap C2 = C1 \cap C3 = C2 \cap C3 = \phi$$

$$\sum_{\alpha h \in C} abs(x\alpha h - y1\alpha h) = 0 \qquad \sum_{\alpha i \in C1} abs(x\alpha i - y1\alpha i) = \Phi 1$$

$$\sum_{\alpha j \in C2} abs(x\alpha j - y1\alpha j) = \Phi 2 \qquad \sum_{\alpha k \in C3} abs(x\alpha k - y1\alpha k) = 2*\Phi 3$$

$$abs(|X|-|Y1|) = abs(\Phi 1 - \Phi 2) <= \Phi 1 + \Phi 2$$

$$DIT = \sum_{\alpha h \in C} abs(x\alpha h - y1\alpha h) + \sum_{\alpha i \in C1} abs(x\alpha i - y1\alpha i) +$$

$$+ \sum_{\alpha j \in C2} abs(x\alpha j - y1\alpha j) + \sum_{\alpha k \in C3} abs(x\alpha k - y1\alpha k) +$$

$$+ abs(|X|-|Y1|) <= 0 + \Phi 1 + \Phi 2 + 2*\Phi 3 + \Phi 1 + \Phi 2 = 2*(\Phi 1 + \Phi 2 + \Phi 3)$$

En este caso $DD(X,Y1) = \Phi 1 + \Phi 2 + \Phi 3$ por lo que:

$$DD(X,Y) >= DIT(X,Y)/2$$

La DIT mayor se obtiene al imponer la condición de disjunción. Como en el caso, ya estudiado, de múltiples sustituciones, si la disjunción entre los conjuntos Ci no se cumple, puede suceder que la interacción entre errores de distinto tipo que afecten al mismo carácter haga que estos influyan en la DIT en menor medida que la suma de cada uno de ellos por separado, por lo tanto:

$$DIT = \sum_{i=1}^{m} abs(x\alpha i - y1\alpha i) + abs(|X|-|Y1|) <=$$

$$<= \Phi 1 + \Phi 2 + 2*\Phi 3 + abs(\Phi 1 - \Phi 2) <= \Phi 1 + \Phi 2 + 2*\Phi 3 + \Phi 1 + \Phi 2 = 2*\Phi 1 + 2*\Phi 2 + 2*\Phi 3$$

En este caso $DD(X,Y1) = \Phi 1 + \Phi 2 + \Phi 3$ por lo que, en general:

$$DD(X,Y) >= DIT(X,Y)/2$$

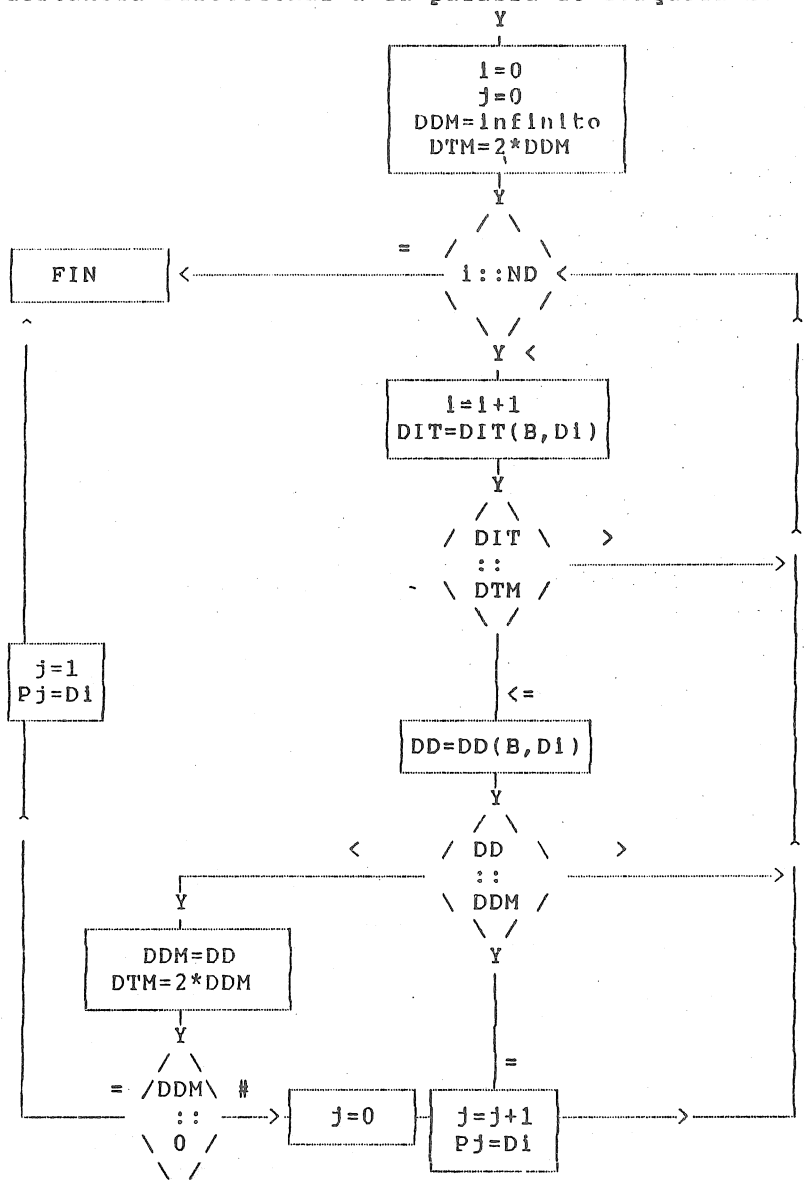
5.- CONJUNTO DE PALABRAS MAS SIMILARES USANDO DIT COMO FILTRO ADAPTIVO PARA EL CALCULO DE DD.

5.1 ESQUEMA DE BUSQUEDA DITS+DD.

De lo anterior surge la idea de disponer de un umbral DTM y calcular DD sólo para aquellas palabras cuyo DIT<=DTM. Además DTM puede ser ajustado dinámicamente. Teniendo en cuenta la relación:

$$DIT(B,Dj) <= 2*DD(B,Dj)$$

toda palabra Dj tal que su distancia DIT a la palabra de búsqueda B sea mayor que dos veces la distancia direccional mínima actual no puede tener una DD menor o igual a ésta y por tanto no se ha de calcular su distancia direccional a la palabra de búsqueda B.



Es destacable que este proceso tiene un ciclo de realimentación que le permite adaptar el filtro según la distorsión sufrida por la palabra de búsqueda.

5.2 COSTE DE COMPUTACION Y ALMACENAMIENTO DEL ESQUEMA DE BUSQUEDA DITS+DD.

El cálculo de DITS+DD presenta la siguiente complejidad:

$$O(ND*m+|B|*\sum_{j\in J}|Dj|)$$

siendo J={j/Dj verifica DIT(B,Dj)<=DTM ajustable}

El coste de almacenamiento es

$$O(ND*m+\sum_{j=1}^{ND}|Dj|)$$

5.3 RESULTADOS EXPERIMENTALES DEL ESQUEMA DE BUSQUEDA DITS+DD.

La Distancia Direccional Máxima se ha inicializado a:

1) infinito

2) |B|/2. A fin de mejorar los tiempos a costa de perder exactitud para palabras de búsqueda distorsionadas más de un 33.33%.

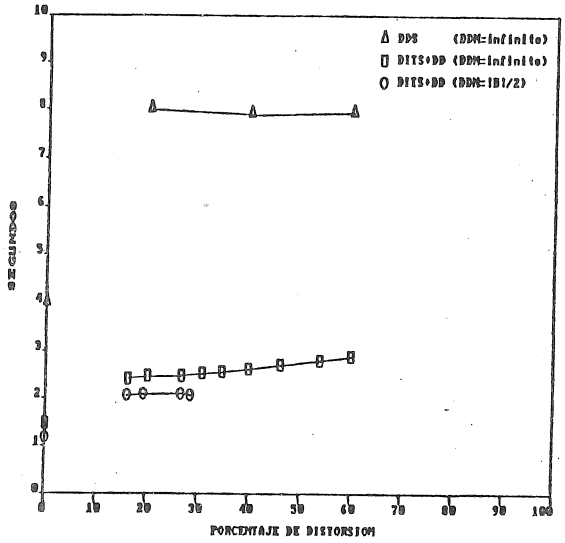


Figura 4 "Gráfica de los tiempos promedios de búsqueda para los esquemas DDS y DITS+DD"

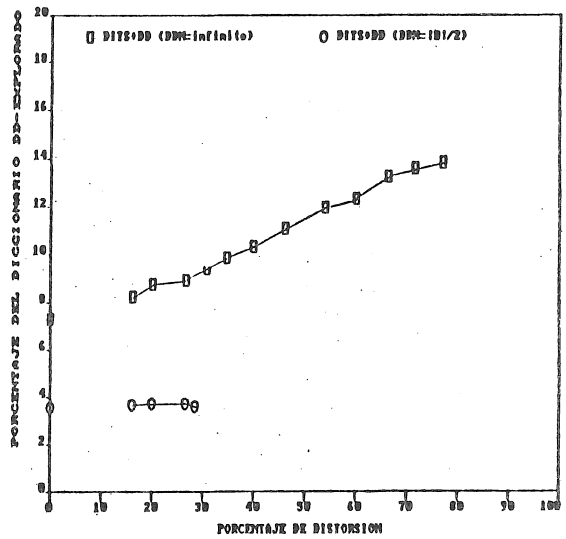


Figura 5 "Gráfica del porcentaje promedio del diccionario implicado en el cálculo de DD en el esquema DITS+DD"

Los tiempos de DDS para búsqueda de palabras correctas, aquellas que se encuentran en el diccionario, son aproximadamente la mitad que para palabras distorsionadas. Ello es debido a que durante la búsqueda lineal, si la palabra no está en el diccionario, éste ha de ser recorrido totalmente, mientras que si la palabra de búsqueda sí que está, se recorre como promedio la mitad del mismo.

La complejidad de cálculo del esquema de búsqueda DDS para palabras distorsionadas es insensible a la distorsión.

La complejidad de cálculo del esquema de búsqueda DITS+DD es del orden del 40% del esquema de búsqueda DDS.

En el esquema de búsqueda DITS+DD (DDM=infinito) aproximadamente el 30% del tiempo de búsqueda se invierte en los cálculos de DD y el 70% en los cálculos de DIT. La reducción del tiempo para este esquema con DDM=|B|/2 se debe a que el número de palabras DD-exploradas se ha reducido a un 40%. Por lo que bajo esta condición el 15% del tiempo se invierte en los cálculos de DD y el 85% restante en los cálculos de DIT.

El ligero aumento del los tiempos promedio en el esquema de búsqueda DITS+DD para palabras distorsionadas, se debe exclusivamente al pequeño incremento del número de palabras del diccionario a las que se le calcula la DD con la palabra de búsqueda.

6.- ESTRUCTURACION DE LAS COMPONENTES QUE INTERVIENEN EN EL CALCULO DE DIT.

Los vectores de frecuencias de los caracteres de las palabras de igual longitud pueden estructurarse permitiendo, conjuntamente con el esquema de búsqueda apropiado, abandonar la exhaustividad de la búsqueda, con lo que sólo se deberá tratar una parte del diccionario para la solución del problema. Se tiene, por tanto, un NODO-RAIZ que discrimina por longitudes de palabras de forma que en la posición i se tiene la dirección de la parte de la estructura que contiene las palabras de longitud i.

6.1 PARTE-ARBOL.

Se puede definir una estructura arbórea cuyos nodos son de la siguiente forma:

αi	l0	l1	..	lne

donde:

αi: carácter del alfabeto.

lk k=0..ne: un enlace que señala la parte del árbol donde se encuentran las palabras DjϵD/djαi=k.

Se llama parte-árbol a esta parte de la estructura.

6.2 PARTE-CADENA.

Cuando por una rama de la parte-árbol ya no se pueda discriminar, se utiliza una lista encadenada formada por nodos de la forma:

αi	fi	l

αi: carácter del alfabeto que no ha sido utilizado en el camino desde la raiz del árbol.
fi: frecuencia de αi.
l: enlace.

Se llama parte-cadena a esta parte de la estructura.

6.3 CADENA-SIT.

Cuando se acaban los caracteres del vector de frecuencias, tanto en la parte-árbol como en la parte-cadena, se coloca un nodo terminal con αi='/' tal que su enlace apunta a una lista encadenada de sinónimos invariantes trasposicionales, SIT, que son palabras que la estructura considera indiscernibles por tener vectores de frecuencias de caracteres idénticos, aunque éstos tienen una disposición distinta, (AMOR, ROMA, RAMO).

Se llama cadena-SIT a esta parte de la estructura.

En el caso de nuestro diccionario las longitudes de las cadenas-SIT tienen la siguiente distribución de frecuencia:

Long. cad-SIT	1	2	3
Frecuencias	2000	43	1

La longitud promedio es de 1.022

6.4 CRITERIO DE ELECCION DEL CARACTER DISCRIMINANTE EN LA PARTE-ARBOL.

Llega el momento de plantearse que caracteres αi se van eligiendo durante la creación del árbol. Se ha elegido un criterio dinámico; el árbol se va creando según se van insertando palabras. Cuando se hace necesario crear un nodo que separe los caminos de dos palabras no-SIT, se escoge el primer carácter por orden alfabético que establezca una diferencia.

7.- MEDIDAS ESTATICAS.

Creada esta estructura con el diccionario se tomaron las siguientes medidas estáticas:

1.- Longitud de caminos en la parte-árbol.
2.- Longitud de los caminos en la parte-cadena.
3.- Porcentajes de enlaces que están a nulo en cada nivel de la parte-árbol.

En las gráficas de las figuras 6 y 7 se observa una simetria especular debido a que el alfabeto tiene 26 caracteres; si existen t caminos en el árbol de longitud lt entonces también existen t listas encadenadas con longitud 26-lt.

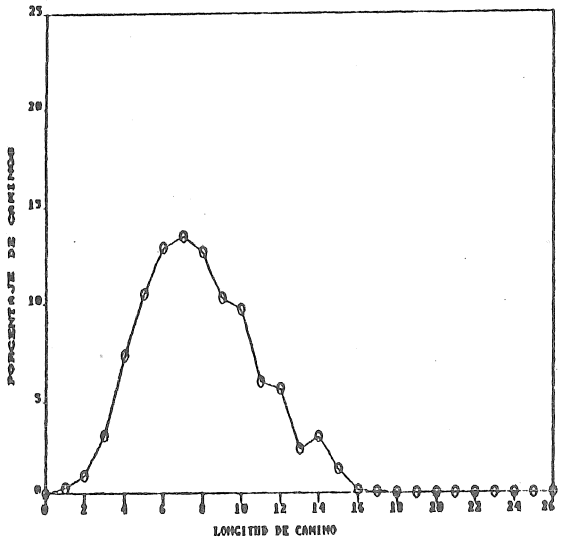


Figura 6 "Gráfica de la distribución de las longitudes de los caminos en la parte-árbol"

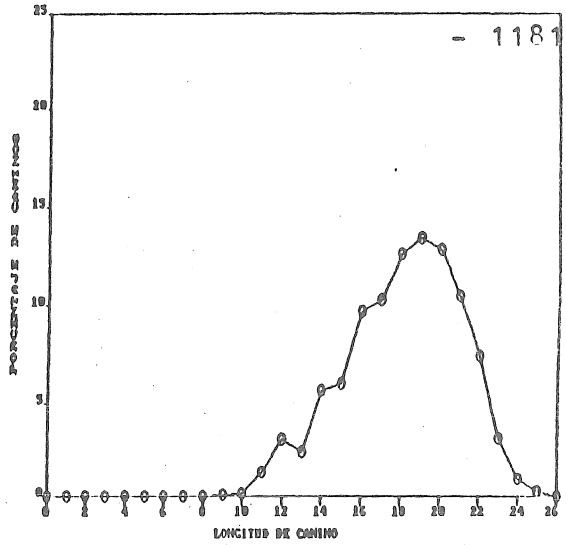


Figura 7 "Gráfica de la distribución de las longitudes de los caminos en la parte-cadena"

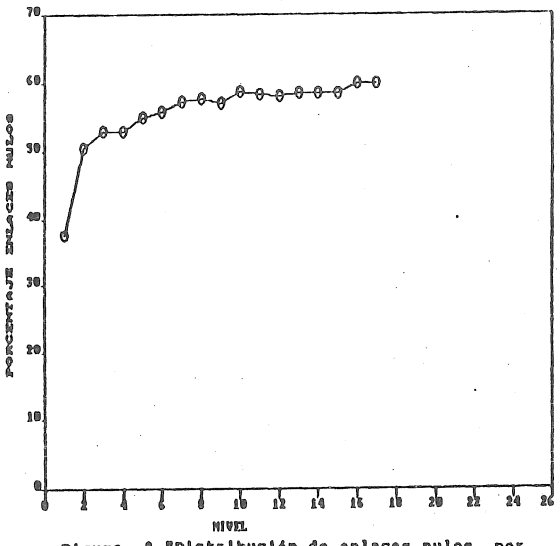


Figura 8 "Distribución de enlaces nulos por niveles en la parte-árbol con ne=4"

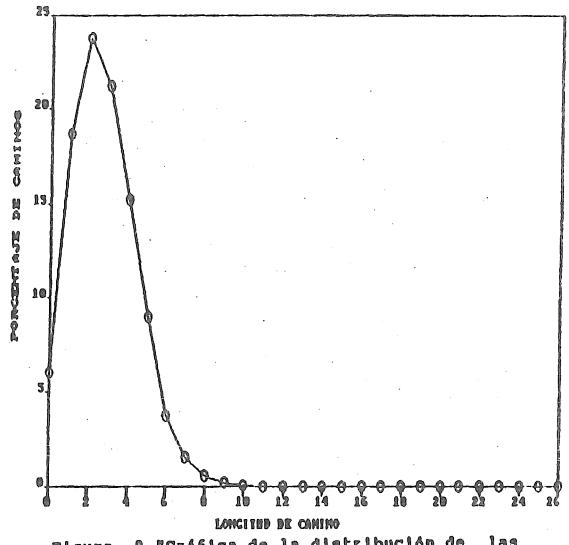


Figura 9 "Gráfica de la distribución de las longitudes de los caminos en la parte-cadena optimizada"

8.- OPTIMIZACION DE LA PARTE-CADENA.

Una mejora inmediata que se puede realizar es eliminar de la lista encadenada (parte-cadena) aquellos nodos cuya fi=0. De esta forma la figura 7 queda como refleja la figura 9.

Se observa, figura 9, que se produce un fuerte decremento de la longitud promedio de la parte-cadena; de 10.115 a 2.944.

9.- OPTIMIZACION DE LA PARTE-ARBOL.

9.1- OPTIMIZACION DEL CRITERIO DE ELECCION DEL CARACTER DISCRIMINANTE EN LA PARTE-ARBOL.

Debido a la naturaleza estática de un diccionario se puede pensar en crear el árbol conociendo a priori que palabras va a contener. De esta forma se puede escoger en todo momento el carácter que más eficazmente realicze su trabajo discriminatorio. ¿Cómo se puede hacer esta elección?. Utilizando un criterio de uniformidad.

Sea N el número de palabras que van a colgar de un nodo. N es independiente del carácter αi que se escoja como discriminante para ese nodo.

Sea αilk el número de palabras que colgarían del enlace k si se escogiese el carácter αi.

$$\sum_{k=0}^{ne} (\alpha ilk) = N \text{ para cualquier } i$$

Se escoge aquel αi tal que:

$$\sum_{k=0}^{ne} (\alpha ilk)^2 \text{ sea mínimo}$$

Esto asegura una distribución uniforme de las palabras a lo largo de cada nodo.

Las figuras 10 y 11 muestran como queda el árbol tras este proceso. Se aprecia un disminución de la altura promedio de la parte-árbol que de 7.885 pasa a 6.032. Además se da mayor agrupamiento en torno a la media como refleja la varianza:

V1=8.519
V2=3.107

Igualmente el número de nodos que tiene la parte-árbol optimizada es un 95.59% del anterior. Además con el primer criterio en la parte-árbol el 56.14% de los enlaces estaba a nulo y ahora el porcentaje es del 55.04%. Se observa igualmente un aumento en el número de listas encadenadas cuya longitud es cero. Todo esto da a entender un mayor aprovechamiento de la estructura.

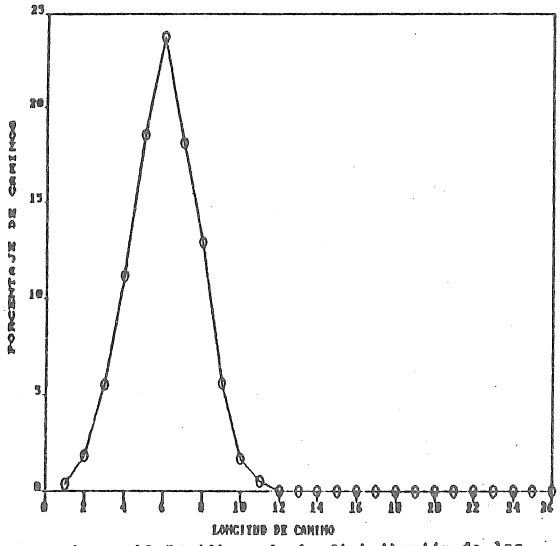


Figura 10 "Gráfica de la distribución de las longitudes de los caminos en la parte-árbol"

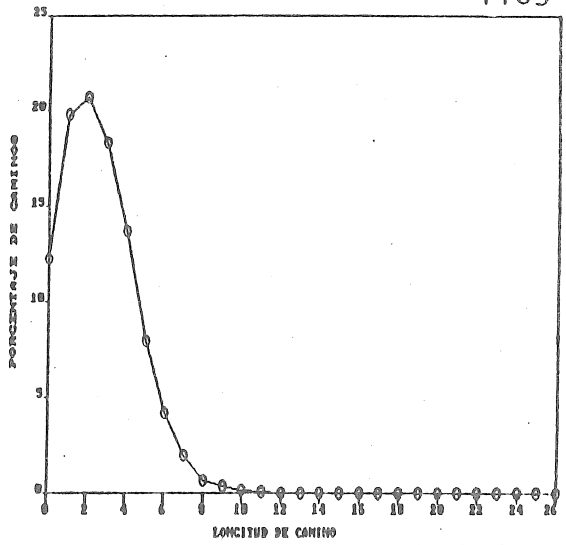


Figura 11 "Gráfica de la distribución de las longitudes de los caminos en la parte-cadena"

En las figuras 12 y 13, se tiene un reflejo de la estructura 'física' de la parte-árbol sin y con optimización.

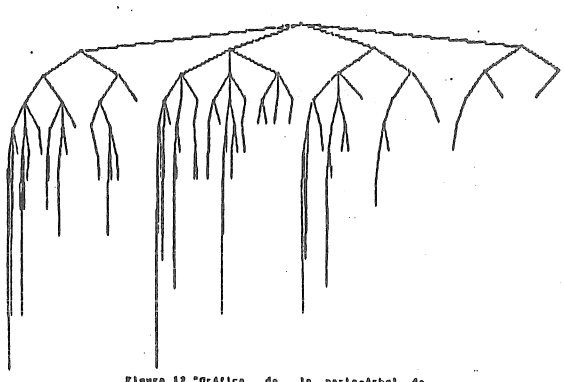


Figura 12 "Gráfica de la parte-árbol de palabras de longitud 7 sin optimización"

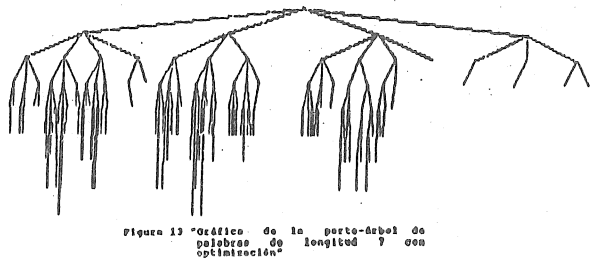


Figura 13 "Gráfica de la parte-árbol de palabras de longitud 7 con optimización"

9.2 ENLACES NULOS. TIPOS DE NODOS.

La siguiente mejora de la estructura viene de observar el elevado porcentaje de enlaces que están a nulos. ¿Podría existir alguna forma de eliminarlos?. Para ello se observó que gran parte de los nodos tenían los enlaces no-nulos agrupados, bordeados de enlaces nulos. Es decir, que de los ne enlaces disponibles en cada nodo, desde 10 hasta un cierto lF y a partir de un lR hasta lne todos los enlaces están a nulos. Por ello se dice que un nodo es de tipo [F,R] si:

$$\{lz=nulo,\ z\not\in[F..R],\ lF \text{ y } lR\#nulo\}$$

El siguiente cuadro muestra la frecuencia de los distintos tipos de nodos:

Tipo	1	2	3	4
0	73.05%	20.31%	2.95%	0.25%
1		2.34%	0.62%	-----
2			0.18%	0.06%
3			0.06%	-----

La existencia de un nodo de tipo [3,3] es debido a que en el diccionario existe una sóla palabra de longitud dieciocho, por lo que únicamente se necesita un nodo de la parte árbol con el fin de que exista esta parte. Es un caso muy particular que no se ha considerado oportuno incluir su excepcionalidad en el tratamiento de la estructura.

9.3 MODIFICACION DE LOS NODOS.

Se varia ligeramente la definición del nodo de la parte-árbol:

αi	F	R	lF	..	lR

De esta forma se consigue eliminar todos los enlaces nulos excepto los comprendidos entre F y R. Con lo que para este experimento sólo el 0.94% de los enlaces son nulos.

10.- ESQUEMA DE BUSQUEDA DE LAS PALABRAS MAS SIMILARES USANDO LA ESTRUCTURA OPTIMIZADA (DITE+DD).

En el esquema de búsqueda cabe distinguir cuatro partes perfectamente diferenciadas:

1.- Recorrido por el nodo raiz.
2.- Recorrido por la parte-árbol.
3.- Recorrido por la parte-cadena.
4.- Recorrido por la cadena-SIT.

Estas cuatro partes forman una estructura jerárquica de forma que de una parte se pasa a la siguiente por llamada y a la anterior por retorno.

1.-Recorrido por el nodo raiz.-

Se empieza entrando por la posición correspondiente a la longitud de la palabra de búsqueda y luego se van tomando alternativas por proximidad de longitud. Por ejemplo, si $|B|=3$ se explora el nodo raiz mediante la siguiente secuencia:

3,4,2,5,1,6,7,8,9,...

Estas secuencias se tienen almacenadas en una tabla de doble entrada.

ORDEN DE LA SECUENCIA

LONGITUDES	1	2	3	4	..
1	1	2	3	4	..
2	2	3	1	4	..
3	3	4	2	5	..
4	4	5	3	6	..
:	:	:	:	:	

Ahora bien en principio se tiene que DDM=INFINITO y DTM=2*DDM, pero este filtro se va ajustando cuando la búsqueda se ejecuta en los niveles inferiores.

En el cálculo de la DIT uno de los sumandos es la diferencia en valor absoluto de la longitud de la palabra de búsqueda y la longitud de la palabra del diccionario; como en cada subárbol las palabras están agrupadas por longitud, este sumando puede calcularse y entregarse a los niveles inferiores.

Como es lógico suponer, este sumando se utiliza para interrumpir la secuencia de recorrido del nodo raiz, en el caso de que sea mayor que el valor de DDM.

2.- Recorrido de la parte-árbol.-

El vector de frecuencias de B se calcula antes de empezar la búsqueda.

Si se está en un nodo cuyo carácter es α se van recorriendo sus enlaces de un modo parecido a como se hace en el nodo raiz, por alternativas. Pero además hay que tener en cuenta que los nodos están clasificados por tipos. Por ejemplo, si un nodo es de tipo [1,3] y B tiene cuatro veces el caracter α ($b\alpha$=4) se recorre este nodo según la secuencia 3,2,1; pero si $b\alpha$=2 la secuencia será 2,3,1.

Cada valor de una secuencia depende de cuatro factores:

1. Límite inferior del tipo de nodo.
2. Límite superior del tipo de nodo.
3. $b\alpha$.
4. Posición del valor dentro de la secuencia.

Por ello se utiliza una tabla de cuatro entradas.

Cada vez que se accede a un nodo, se distinguen dos casos:

1. Se direcciona por el enlace propio, es decir, por aquél que corresponde según $b\alpha$.

2. Se direcciona por un enlace alternativo.

Esta distinción es debida a que se va calculando la DIT según se baja por la parte-árbol (DITACUM). En el caso 1. la DITACUM que se tiene calculada no sufre ninguna variación, mientras que en el caso 2. hay que sumar abs($b\alpha$-i), siendo i el enlace por el que se direccione.

Si en algún momento DITACUM es superior a DTM no se continua con la toma de alternativas en este nodo, sino que se devuelve el control al nodo que está en el nivel inmediato superior para continuar la búsqueda por las alternativas apropiadas.

3.- Recorrido por la parte-cadena.-

Aquí es un sencillo recorrido secuencial, calculando en cada nodo el término correspondiente de la DIT, si α es el carácter del nodo se le suma a DITACUM el término abs(bα-fi). Si en algún momento se supera la DTM se suspende este recorrido y se le cede control al último nodo visitado de la parte árbol.

4.- Recorrido por la cadena-SIT.-

Una vez agotada la parte-cadena se calcula la DIT en base a DITACUM y a los caracteres de B que no hayan sido utilizados desde la raíz del árbol. Sólo en el caso de que la DIT no supere la DTM se pasa a calcular la DD con las palabras que estén en la cadena-SIT, se compara con DDM, a fin de ver si se ha de añadir la palabra o no al conjunto de palabras más similares y se ajusta, si procede, DDM y DTM.

11 MEDIDAS DINAMICAS.

El tiempo de búsqueda de las palabras más similares a una dada se invierte en:

1. El cálculo de las DD entre la palabra de búsqueda y las del diccionario que superen el filtro de la DIT. Esto supone alrededor del 60% del tiempo de búsqueda.

2. El recorrido por la estructura evaluando las DITs, utiliza alrededor del 40% del tiempo de búsqueda.

Por lo que el esquema DITE+DD consume más tiempo en el cálculo de las DD que en los de DIT, contrariamente a lo que ocurría en el esquema DITS+DD.

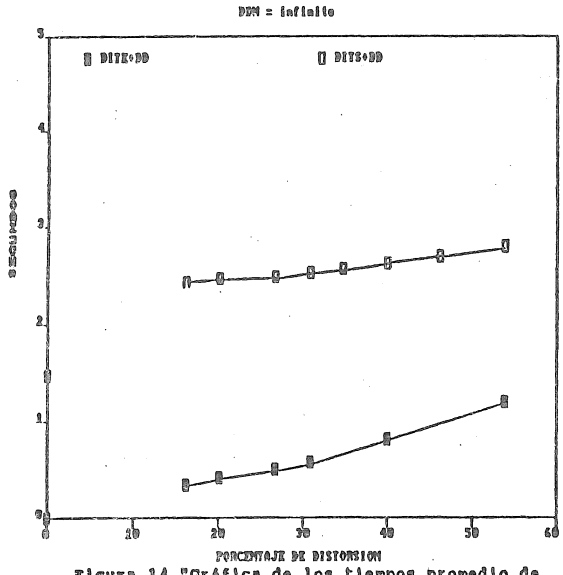


Figura 14 "Gráfica de los tiempos promedio de búsqueda para los esquemas DITS+DD y DITE+DD "

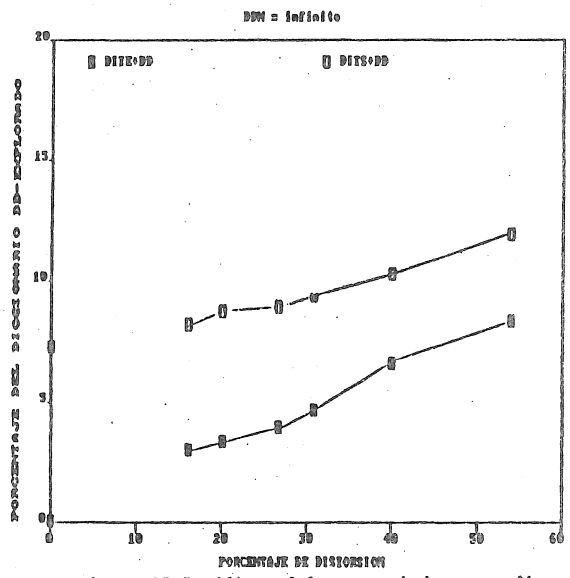


Figura 15 "Gráfica del porcentaje promedio del diccionario implicado en el cálculo de DD en los esquemas DITS+DD y DITE+DD"

Respecto al esquema de búsqueda secuencial DITS+DD (DDM=infinito) para la búsqueda de palabras distorsionadas se ha reducido el número de palabras del diccionario que han de intervenir en el cálculo de la DD, equivalente a un 20% del ahorro de tiempo por palabra de búsqueda. De lo que se deduce que alrededor del 80% de la optimización en tiempo obtenida proviene de la menor sobrecarga del cálculo de la DIT al estructurar sus componentes.

12.- MODIFICACION ULTERIOR AL ESQUEMA ESTRUCTURADO

Se introducirá una modificación al esquema de búsqueda en la estructura, proponiendo un nuevo esquema que en adelante se denominará DITEM+DD. Dicha modificación consiste en cambiar el sentido aplicado hasta ahora al juego de inicializaciones de DDM, de forma que se utilizará una afirmación mas restrictiva, garantizando que el canal no producirá una distorsión mayor que un cierto porcentaje. Por lo tanto, aquellas palabras del diccionario con respecto a las cuales la palabra de búsqueda tenga una distorsión superior a la fijada, serán eliminadas automáticamente por el nuevo esquema. Para ello, se tendrán en cuenta las características de cada rama del nodo raíz de la estructura, con respecto a la longitud de las palabras del diccionario, de forma que se inicializa el DDM al mínimo entre la distancia direccional actual y el producto de la distorsión permitida π por la longitud de las palabras a explorar.

En las figuras 16 y 17 se muestran las gráficas del tiempo promedio de búsqueda y del porcentaje promedio DD-explorado del diccionario, para los esquemas DITE+DD, en sus inicializaciones de DDM a infinito y $|B|/2$ y el esquema DITEM+DD con $\pi=1/3$ (33.33%). En ellas aparece registrado un fuerte descenso en el tiempo promedio de búqueda para la estructura DITEM+DD, debido fundamentalmete a la disminución observada en el porcentaje promedio DD-explorado del diccionario.

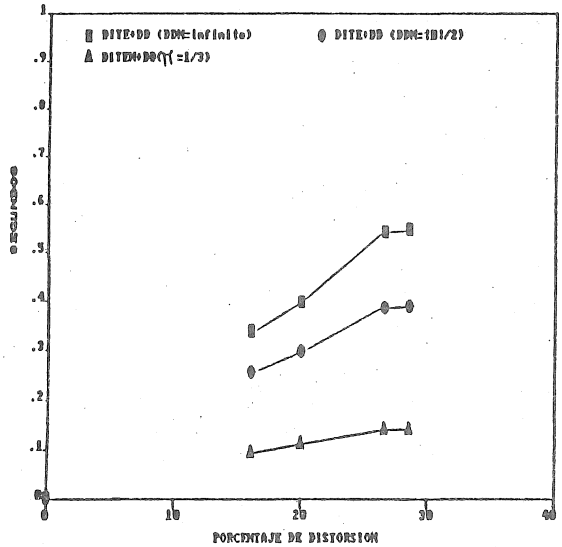


Figura 16 "Gráfica de los tiempos promedio de búsqueda para los esquemas DITE+DD y DITEM+DD"

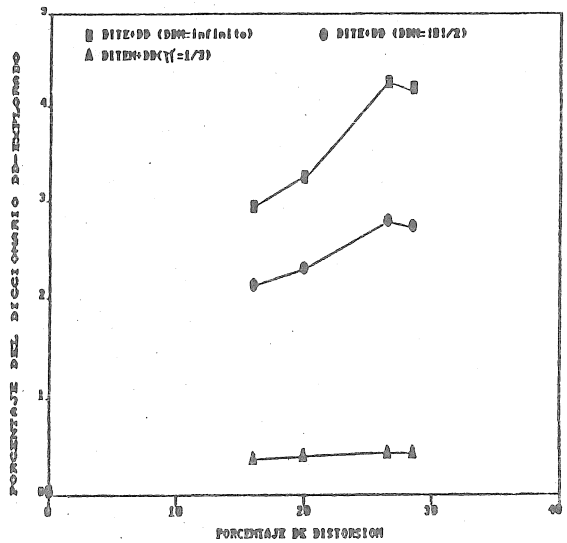


Figura 17 "Gráfica del porcentaje promedio del diccionario implicado en el cálculo de DD en los esquemas DITE+DD y DITEM+DD"

13 CONCLUSIONES.

La evolución del tema tratado aquí, parte del tratamiento más simple posible: un recorrido secuencial, exhaustivo en el caso de una palabra de búsqueda incorrecta, con calculo de la distancia direccional (DDS). Este tratamiento proporciona un porcentaje promedio de aciertos razonablemente alto hasta distorsiones del 30%, mientras que el porcentaje promedio de respuesta única no es muy alto, incluso para distorsiones bajas, y el promedio de multiplicidad de respuesta crece rápidamente a partir del 20% de distorsión.

La introducción el esquema DITS+DD, con el fin de mejorar el tiempo promedio de búsqueda utilizando un esquema más sofisticado, produce, en efecto, una disminución del mismo de alrededor del 40%, aunque una pequeña parte de la disminución se debe al aumento del coste de almacenamiento, al guardar los vectores de frecuencias. A diferencia del esquema DDS, en DITS+DD el tiempo promedio de búsqueda para palabras incorrectas se muestra sensible a la distorsión. Asimismo, la inicialización de la distancia direccional máxima a la mitad de la longitud de la palabra de búsqueda, produce una fuerte disminución del porcentaje de diccionario DD-explorado, arrastrando consigo al tiempo promedio de búsqueda.

La estructuración inicial de los datos en el esquema DITE+DD lleva asociada una serie de medidas estáticas que sufren sucesivas mejoras con las tres optimizaciones introducidas a la estructura. Hay que destacar la efectividad de la eliminación de un altísimo porcentaje de encadenamientos nulos y la progresiva disminución de la longitud promedio de los dos tipos de caminos de la estructura.

En cuanto a las medidas dinámicas tomadas al esquema DITE+DD hay que destacar una fuerte bajada del tiempo promedio de búsqueda, con respecto al del esquema DITS+DD, así como la inversión de la importancia relativa de sus componentes: al contrario que en el esquema anterior, en el DITE+DD tiene más peso el tiempo invertido en el cálculo de las distancias direccionales. Otro factor a destacar es la disminución del porcentaje promedio del diccionario que es DD-explorado, con respecto al esquema DITS+DD. Este factor equivale a una pequeña parte del total de la disminución del tiempo de búsqueda, por lo que ha de entenderse que el peso de esta disminución recae sobre la menor sobrecarga en el cálculo de las distancias invariantes trasposicionales, a causa de la estructuración. De nuevo, tanto el tiempo promedio de búsqueda, como el porcentaje DD-explorado mejoran al inicializar DDM a la mitad de la longitud de la palabra de búsqueda.

La modificación introducida en el esquema DITEM+DD conlleva una fuerte disminución de los tiempos de búsqueda y una bajada del porcentaje DD-explorado con respecto al esquema DITE+DD. Por último, la respuesta al uso de diferentes distorsiones máximas sigue las líneas trazadas en los anteriores esquemas con el uso de diferentes inicializaciones de DDM, con la salvedad de que en este caso, habrán de ser rechazadas las posibles palabras propuestas como solución por tener una distancia direccional baja, que superen la distorsión máxima permitida.

BIBLIOGRAFIA

{1} AHO, A.V.; HOPCROFT, J.E.; ULLMAN, J.D.: Data Structures and Algorithms. Addison-Wesley Publishing Company. 1983

{2} ALBERGA, C.N.: String Similarity and Misspellings. CACM 1967, 10 (5),302 313.

{3} MORGAN, H.L.: Spelling Correction in System Programs. CACM 1970, 13 (2), 90-94

{4} SZANSER, A.J.M.: Automatic Error Correction of Natural Text. Part I Computer Science N. 46 National Physics Laboratory, 1973.

{5} TREMBLAY, J.P.; SORENSON, P.G.: An Introduction to Data Structures with Applications. McGraw-Hill Book Company. 1984.

{6} WAGNER, R.A.; FISCHER, M.J.: The String-to-String Correction Problem. JACM 1974, 21 (1), 168-173.

UM SISTEMA PARA AUXILIO AO *PROJETO E INTEGRAÇÃO DE* ESQUEMAS CONCEITUAIS SEMANTICOS

Jano Moreira de Souza *1

SUMARIO

Este trabalho apresenta as caracteristicas gerais de um sistema para auxílio ao projeto e integração heterogênea de Bases de Dados descritas usando modelos semânticos de dados. Detalhes da implementação, conceitos e algoritmos utilizados podem ser encontrados em [Souz86]. Uma taxonomia de tipos de integração de banco de dados é fornecida graficamente. A importância da tarefa de comparação semântica dos esquemas conceituais que descrevem os banco de dados, a serem integrados, é salientada e conceitos desenvolvidos em seis outras áreas da Ciência da Computação são sugeridos para uso nessa comparação.

ABSTRACT

This paper outlines a Computer Aided Design and Integration System for databases described using different Semantic Data Models. Details of the implementation, underlying concepts and algorithms used can be found in [Souz86]. A taxonomy of integration types is presented graphically. The paper stresses the importance of the *semantic comparison* of the conceptual schemata describing the database to be integrated. Concepts borrowed from six others fields of the Computer Science are suggested for use in the comparison task.

---

1. Professor Adjunto - COPPE/UFRJ
Caixa Postal 68.511 - CEP 21945 - Rio de Janeiro - RJ - Brasil

1 — INTRODUÇÃO

Como consequência do intenso debate ocorrido na área de Banco de Dados, da década de 70 até o começo da década de 80, sobre a conveniência de um modelo de dados sobre os outros [Date74, Sibl76], emergiu um consenso em torno da necessidade de suportar a coexistência de diversos modelos de dados [Smit81, Mann83, Mcle80, Melo83]. Os requisitos das aplicações e/ou a preferência dos usuários devem ser os determinantes da escolha do modelo de dados. O enfoque adotado no presente trabalho baseia-se nas seguintes premissas:

i) Existe uma necessidade real de utilizarem-se diferentes modelos semânticos de dados e de integrar (física e/ou logicamente) bases de dados descritas por seu intermédio;
ii) a completa equivalência (semântica ou sintática) de construções de ambos esquemas a ser integrados são muito raras em banco de dados reais;
iii) A integração (e principalmente a comparação) de grandes esquemas, ou de esquemas ricos em descrição semântica, manualmente é virtualmente impossível.

No estudo da integração de esquemas conceituais semânticos três aspectos foram considerados:

i) Uma taxonomia dos tipos de integração (ou das arquiteturas dos banco de dados após a integração);
ii) algoritmos e conceitos úteis para a comparação e mapeamentos dos esquemas conceituais;
iii) a arquitetura e funcionalidade necessárias a um sistema especialista que auxilie na tarefa de integração.

Os três aspectos acima foram estudados em profundidade em [Souz86]. No presente artigo seus tópicos mais relevantes serão apresentados nas seções 2, 3 e 4.

2 - UMA TAXONOMIA DOS TIPOS DE INTEGRAÇÃO

A integração de banco de dados pode tomar várias formas dependendo de diversos fatores. Embora cada tipo de integração tenha suas especificidades, há suficiente em comum entre elas para justificar um tratamento geral. Os dois fatores mais importantes do tipo de integração são: a) o fato de existir ou não uma integração física dos dados, além da integração lógica dos meta-dados (esquemas), e, b) a ocasião em que se dá a integração. Esta pode ser:

i) Depois que os bancos de dados já se encontram em uso. Isto é, integração de banco de dados já populados;

ii) durante o desenvolvimento de um banco de dados de grande porte e/ou complexidade, situação em que várias pessoas ou equipes trabalham em paralelo e necessitam consolidar ('integrar'), periódicamente suas porções do projeto;

iii) no momento da consulta. O usuário executa uma integração rápida, dinâmica, válida sòmente para aquela consulta.

Os dois appectos determinantes citados acima são por sua vez função das características das aplicações que utilizam esses bancos de dados ('integrado' e a integrar). As características que consideramos mais importantes são:

a) o padrão de utilização previsto para o banco de dados integrado - (ex. somente consultas, atualizações frequentes, atualizações locais x globais, etc...). O padrão de utilização previsto pode tornar proibitiva algumas das arquiteturas (que serão apresentadas a seguir), seja por impedir certas funcionalidades (por exemplo, atualizações em vistas) ou por tornar proibitivo o desempenho de certas transações;

b) consideraçöes geográficas, dado que algumas arquiteturas podem ser mais apropriadas para um banco de dados distribuido do que outras;

c) custos da integração física, isto é, a integração física das bases de dados pode não ser compensadora por envolver custos de cópia e/ou transmissão de dados elevados;

d) custos da perda da produção, pois certos tipos de integração podem exigir a completa paralização do sistema existente, durante a integração, por um período inaceitável;

e) custos ou tempo para re-escrever transações/aplicações pode ser excessivos;

f) dificuldade em definir os mapeamentos entre os esquemas antigos e o integrado (incluindo mapeamentos inversos, para atualizaçöes).

g) 'propriedade dos dados', pois devido a razões de confidencialidade ou segurança, pode ser importante não efetuar a integrção física dos dados;

h) heterogenidade do banco de dados integrado.

Um dos problemas encontrados no estudo da integração de banco de dados é a confusão existente na literatura quanto ao problema que cada pesquisador tenta resolver. Mais adiante será apresentada, em forma gráfica, uma classificação dos tipos de integração encontrados na literatura ou em sistemas comerciais. A partir dos dois bancos de dados apresentados na figura 1, seis arquiteturas para o banco de dados integrado podem ser obtidas. Elas são apresentadas nas figuras 2 até 7. O leitor estudioso da área não terá dificuldades em identificar alguns sistemas a partir dessa representação (ex. MULTIBASE [Daya82] e Multidatabase [Litw82] são exemplos de integração tipo 'Super-Vista' (fig.4)). A classificação foi desenvolvida levando em consideração oito características do banco de dados integrado, resultante, que são listadas a seguir. Uma caracterização mais precisa de cada tipo, bem como seu estudo comparativo pode ser encontrado em [Souz86].

Características Usadas Para Classificar Tipos de Integração:

i) Esquema antigo torna-se vista;
ii) esquema integrado é uma vista;
iii) problemas com atualisações de transações locais;
iv) problemas com atualisações de transações globais;
v) integração física;
vi) transações locais são afetadas;
vii) transações globais são re-escritas;
viii) presta-se a integração heterogênea.

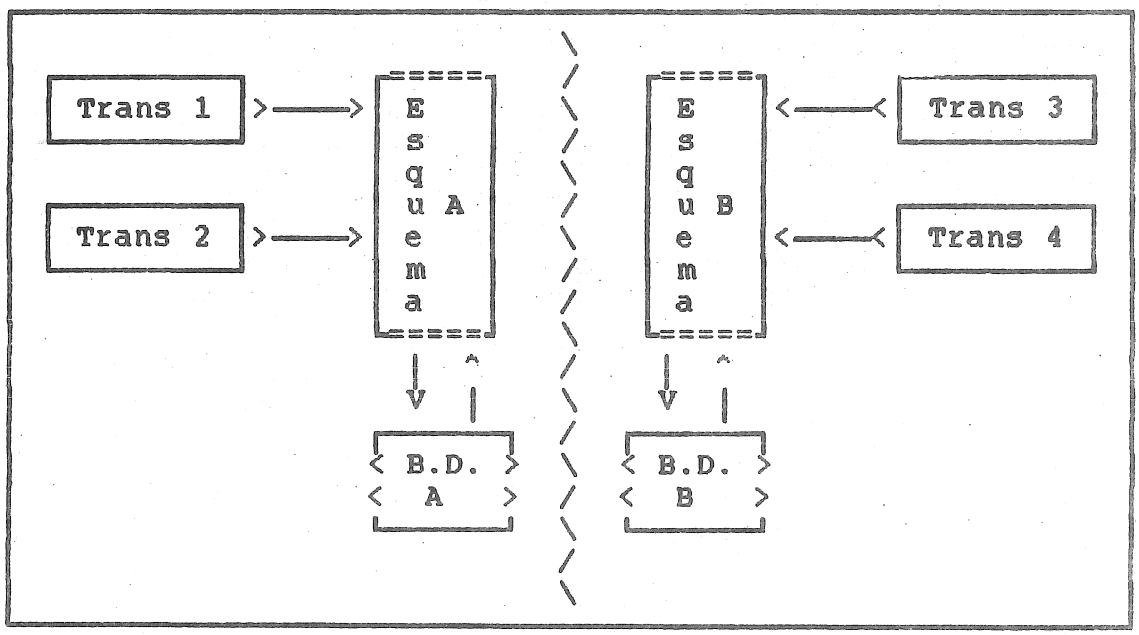


Figura 1 – Bancos de Dados Antes da Integração

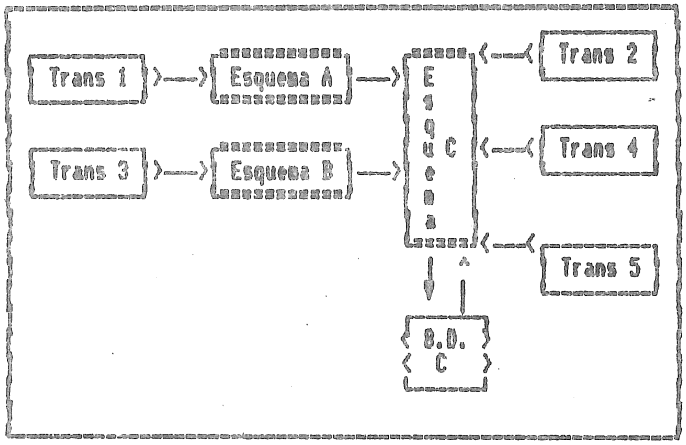


Figura 2 - Integração Total Com Sub-Esquemas

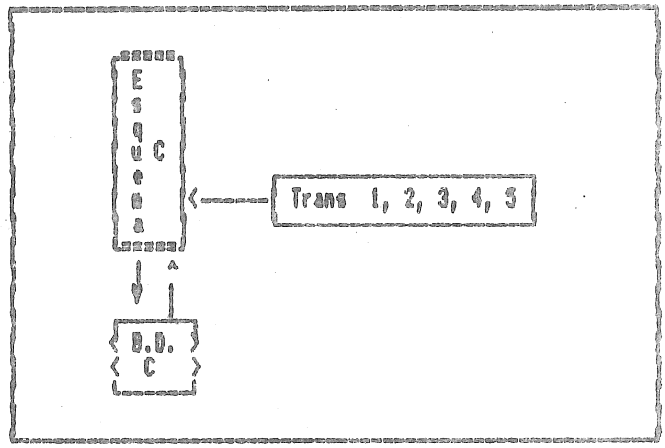


Figura 3 - Integração Total

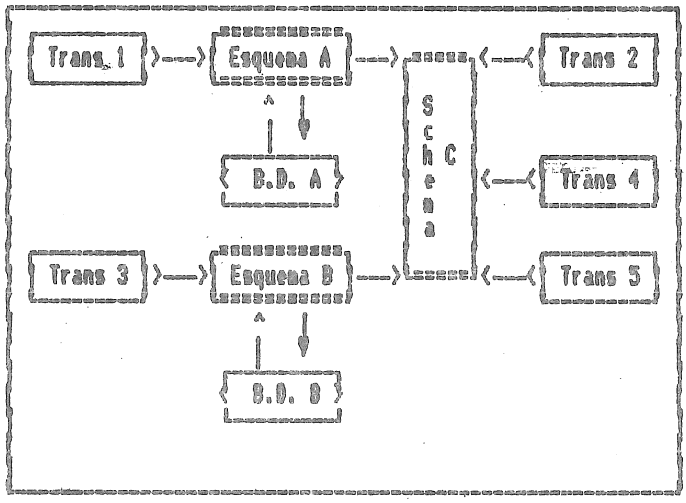


Figura 4 - Super-Vista

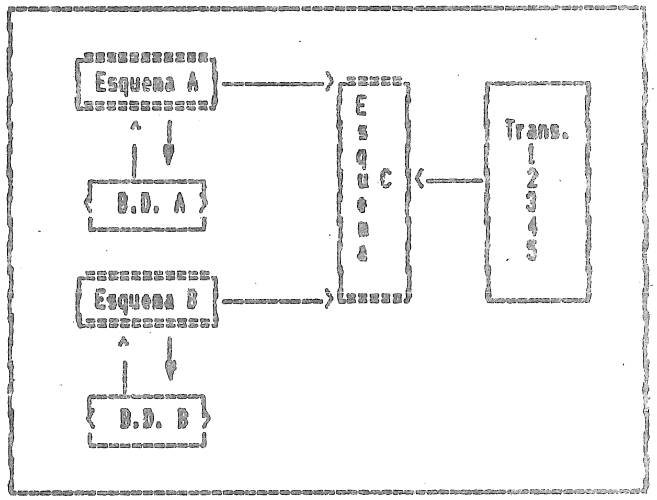


Figura 5 - Banco de Dados Distribuido com Esquema Global

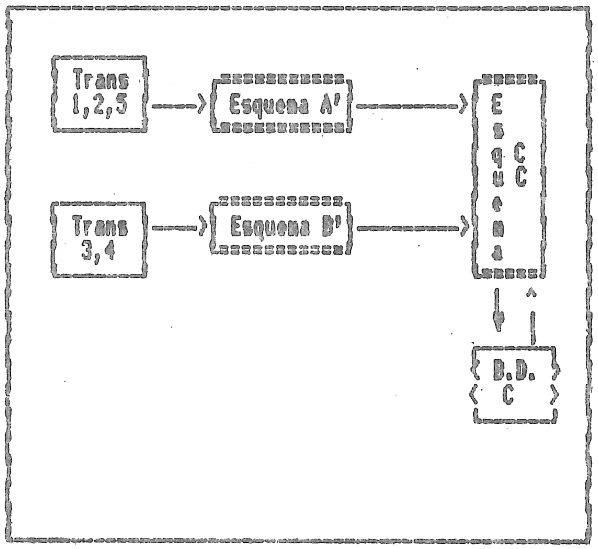


Figura 6 - Integração Total com Sub-Esquemas Modificados

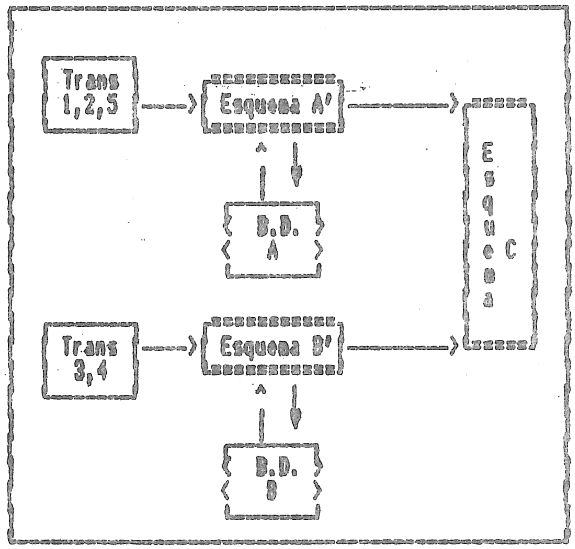


Figura 7 - Banco de Dados Distribuido com Esquema de Integração

3 - COMPARAÇÃO DE ESQUEMAS

Para grandes esquemas, ou para aqueles ricos em descrição semântica, a comparação manual torna-se praticamente impossível. O problema é agravado pelo fato de que, na prática, dificilmente se encontrará um par de construções com exatamente a mesma descrição, tanto do ponto de vista sintático (nomes por exemplo) como semântico. A solução encontrada foi utilizar o conceito de equivalência ao invés do de semelhança. O objetivo da comparação é encontrar subconjuntos de construções em um esquema A que pareçam o máximo com um outro subconjunto de contruções do esquema B.

Seis conjuntos de conceitos tradicionalmente utilizados para lidar com informações imprecisas foram estudados, a saber:

i) fuzzy Sets;
ii) reconhecimento de Padrões;
iii) teoria da informação;
iv) comparação sintática Uni-dimensional;
v) comparação usando gramáticas Bi-dimensionais, tais como gramática de grafos;
vi) teoria de Grafos, (isomorfismo de sub-grafos aliado à um algoritmo similar à programação dinâmica).

4 - UM SISTEMA PARA AUXILIO À INTEGRAÇÃO

O sistema desenvolvido objetivou auxiliar a integração permanente de banco de dados já existentes ou em desenvolvimento (conforme mencionado na seção 1). Entretanto muitas das técnicas e algoritmos empregados ou sugeridos podem ser utilizados no momento da consulta a banco de dados fracamente acoplados [Mac180]. O sistema é totalmente interativo e possui módulos para definir, comparar, modificar e visualisar construções de diversos esquemas, bem como para verificar sua consistência e definir mapeamentos intra e inter-esquemas. O módulo de comparação utiliza algoritmos heurísticos para lidar com similaridades ao

mapeamentos intra e inter-esquemas. O módulo de comparação utiliza algoritmos heurísticos para lidar com similaridades ao invés de equivalências.

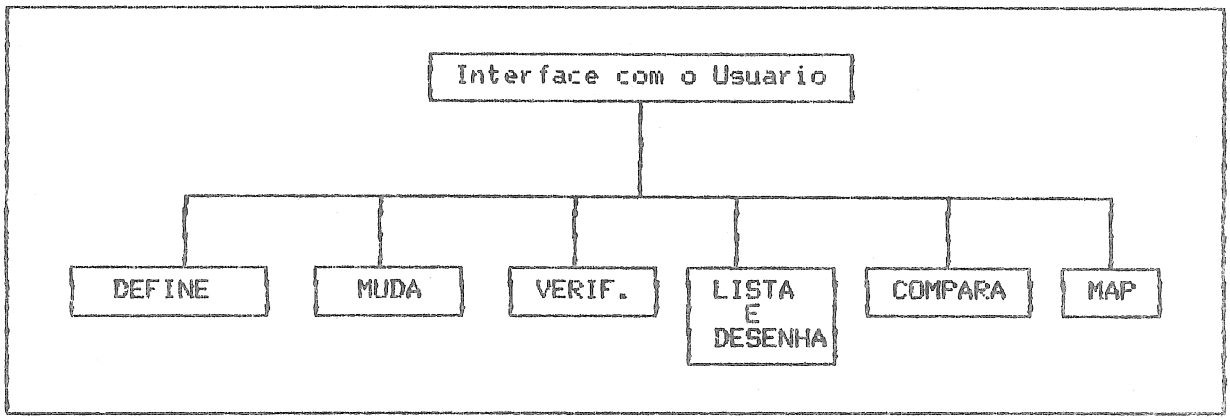


Figura 1 - Módulos do sistema

Embora o sistema possa ser utilizado para integrações heterogêneas, a adoção de uma arquitetura de quatro níveis (uma extensão direta da arquitetura de de três níveis proposta pela ANSI/SPARC [ANSI75]), juntamente com um modêlo de dados canônico permite a integração elegante de banco de dados descritos por modelos de dados semânticos tais como NIAM [Verh82], CIM [Gust82] ou ACM/PCM [Brod82].

5 - CONCLUSÃO

O trabalho destaca três aspectos considerados importantes para a integração de esquemas conceituais semânticos: a) uma taxonomia dos tipos de integração, b) algoritmos e conceitos para a comparação e mapeamento e c) uma ferramenta automatizada para a comparação e integração. Os três aspectos são importantes para a integração, e tem sido muito pouco pesquisados até o momento. Devido à exigüidade de espaço para um aprofundamento dos três tópicos, a taxonomia mereceu maior destaque por ser pré-requisito para o entendimento dos outros, que serão objeto de futuros trabalhos.

REFERENCIAS

[ANSI75] ANSI/SPARC, Study Group on Data Base Management Systems: Interim Report, ACM FDT Vol 7, No 2, New York, 1975.

[Brod82] Brodie M.L. and Silva E., Active and Passive Component Modelling: ACM/PCM, in [29].

[Cloc81] Clocksin, K.L. and Mellish, C., Programming in Prolog, Springer-Verlag, 1981.

[Date74] Date, C.J., The Relational and Network Approaches: Comparison of The Application Programming Interfaces, IBM Technical Report RJ 1401(#21706), June 1974.

[Daya82a] Dayal, U. and Hwang, H., View Definition and Generalization for Database Integration in Multibase: A System for Heterogeneous Distributed Databases. In Proc 6th Berkeley Workshop on Distr Data Managmt and Comm, Feb 1982, pp203-238

[Gust82] Gustafson, M.R., Karlsson T. and Bubenko T.A. Jr, A Declarative Approach to Conceptual Information Modelling,in [Olle82].

[Litw83] Litwin, W. e Kabbaj, K. Manipulation of Relational Multidatabases, INRIA Report, 1983.

[Mann83] Mannino, M.V., A Methodology for Global Schema Design, PhD Thesis, University of Arizona, 1983.

[McLe8o] McLeod,D. and Heimbigner, D. A Federated Architecture for Data Base Systems. In AFIPS National Computer Conference, Vol 39, 1980.

[Melo83] Melo, R.N., Pena, J.C.C. and Zakimi, B., Comparando Modelos Semanticos de Dados, Revista Brasileira de Computacao, Vol 3, No 1, pp 75-89, 1983.

[Motr80] Motro, A. and Buneman, P., Automatically Merging Databases, Proc Compcom Conf Distributed Computing (IEEE), Fall 1980.

[Motr81] Motro, A., Constructing Superviews, in Proc International Conference on Very Large Databases, Cannes, France, 1981.

[Olle82] Olle, T.W. et al (eds.), Information Systems Design Methodologies: A Comparative Review, Noth-Holland Pub Comp, 1982.

[Saha83] Saha, A.B., A Set Oriented Abstract Conceptual Schema Based on Logic, Internal Report, School of Information Systems, University of East Anglia, Dec 1983.

[Schr75] Schreider,J.A., Equality, Resemblance and Order, Mir Publishers, Moscow 1975.

[Smit81] Smith, J.M. et al MULTIBASE, Integrating Heterogeneous Distributed Databases Systems, in National Computer Conference, 1981.

[Souz86] Souza, J. M., "*Software Tools for Conceptual Schema Integration*", PhD Thesis, University of East Anglia, 1986.

RESUMEN

BASES DE DATOS : OPTIMIZACION DEL MODELO INTERNO

Rodolfo Schmal S (*)
Yanko Ossandón N.(*)

Cuando se está ante la tarea de diseñar una base de datos -BD- debe abordarse la tarea de captar toda la semántica contenida en la realidad que interesa recoger, y por otro lado debe procurarse que el rendimiento sea aceptable. Por lo general la atención ha estado centrada en el primer objetivo, descuidándose el segundo de ellos. Al tratar la temática del rendimiento estamos haciendo referencia al tiempo de respuesta que involucre todo acceso que se desee realizar sobre la BD. Una BD puede satisfacer las exigencias lógicas propias de la realidad percibida, pero no será implementable si a la hora de hacer uso de ella los tiempos de respuestas son excesivos. Este trabajo presenta una metodología destinada justamente a optimizar la representación interna considerando las actividades que se estiman se van a realizar sobre la BD.

Arica, Julio 1987

(*)Masters en Informática Universidad Politécnica de Madrid
Académicos del Departamento de Computación e Informática
Universidad de Tarapacá - Arica - Chile

BASES DE DATOS : OPTIMIZACION DEL MODELO INTERNO

Rodolfo Schmal S (*)
Yanko Ossandón N.(*)

1.0 INTRODUCCION

El diseño de una base de datos (BD) es una tarea compleja, que puede tomar meses de trabajo, incluso años. Hasta ahora, el diseño ha sido un arte más que una ciencia, carente de estándares y normas de procedimiento, sujeto a la intuición y experiencia de sus diseñadores.

Determinar la semántica de los datos es uno de los mayores obstáculos a enfrentar, y una vez recogida esta semántica, las estructuras diseñadas pueden ser tan complejas que resulte impracticable su operación, dado el imperativo de eficiencia que exige su implementación. En este sentido la BD debe ser capaz de proporcionar la información requerida con una eficiencia razonable. Un diseño simplista puede dificultar la programación al necesitar una lógica más complicada para acceder a la información, influyendo además en un mayor tiempo y costo de implementación, y posteriormente en mayor tiempo de proceso.

Aspectos como la integridad y la consistencia de los datos, la privacidad y la seguridad, amén de otras consideraciones, son las que nos invitan a buscar diseños sistemáticos que nos pueden ayudar a reconocer y resolver muchas de estas exigencias, antes que la BD sea instalada.

Una de las consecuencias más graves que ocasiona un diseño deficiente es el costo prohibitivo que implica reestructurar una base ineficiente.

Generalmente, las mayores dificultades que se presentan las podemos agrupar en las siguientes categorías :

- Análisis insuficiente de las necesidades de los usuarios.
- Estructuración más allá de la capacidad humana para su análisis manual.
- Dificultades para predecir la incidencia que la implantación del sistema ocasionará en el medio ambiente existente (coexistencia con otras BD).

(*)Masters en Informática Universidad Politécnica de Madrid
Académicos del Departamento de Computación e Informática
Universidad de Tarapacá - Arica - Chile

- Exigencias de cumplimientos de plazos por parte de la administración.

La búsqueda de metodologías de diseño sistemático, y de técnicas de diseño asistido por computador han significado importantes contribuciones para minimizar estos problemas, además de mejorar la calidad y acortar el tiempo de diseño.

En este contexto utilizaremos el formalismo individual para modelar la realidad que nos interesa automatizar, pero colocando énfasis en la optimización del modelo interno lógico.

El formalismo individual es una metodología propuesta por un grupo de trabajo de la Universidad de Marsella (H. Tardieu, D. Nanci, D. Pascot [1]) que ha tenido una amplia acogida en Francia y Europa, que tiene sus raíces en el modelo entidad-relación de P.P.Chen[4].

2.0 FORMALISMO INDIVIDUAL

Nuestro objetivo básico es representar aspectos del mundo real que nos interesa estudiar, en un medio susceptible de ser manipulado electrónicamente. Para llegar, desde los objetos de ese mundo real, a este otro de los datos es preciso realizar una importante labor de abstracción y estructuración que se ha de apoyar en un modelo que nos proporcione un conjunto de métodos, reglas y símbolos.

El formalismo individual permite definir y describir el real percibido utilizando tres elementos [1]:

- los INDIVIDUOS-TIPO que son los objetos percibidos, concretos o abstractos, equivalentes a las ENTIDADES del modelo entidad-relación[2];

- las RELACIONES-TIPO que son las asociaciones entre los objetos percibidos;

- las PROPIEDADES que son las características que permiten describir a los individuos y sus relaciones.

Nuestro trabajo, como informáticos, consistirá en acotar el espacio de información al subespacio que corresponde al fenómeno que se está estudiando con el objeto de conseguir minimizar el tiempo de acceso a los datos y reducir al máximo el soporte físico necesario para el almacenamiento. Normalmente estos dos objetivos se contraponen entre sí, por lo que suele tener que alcanzarse alguna solución de compromiso.

Una de las dificultades claves reside en describir directamente el real percibido en el lenguaje de descripción del SGBD que se vaya a utilizar; por ello es necesario superar las siguientes etapas:

1. ELABORAR EL MODELO CONCEPTUAL.

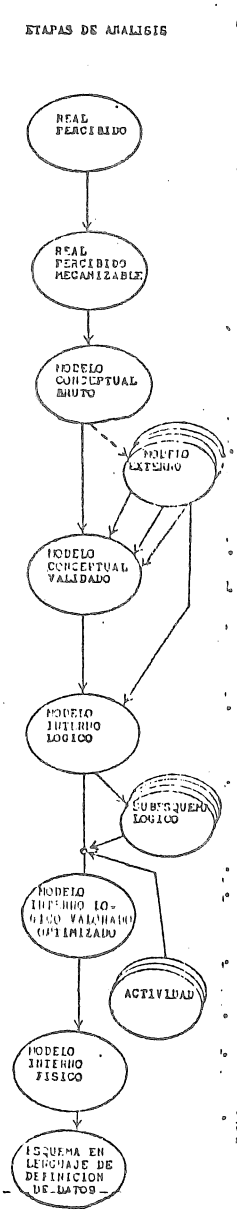
2. VALIDAR EL MODELO CONCEPTUAL.

3. TRANSFORMAR EL MODELO CONCEPTUAL EN UN MODELO INTERNO LOGICO

4. TRANSFORMAR EL MODELO INTERNO LOGICO EN UN MODELO INTERNO FISICO.

El significado de los términos modelo conceptual, modelo interno lógico y modelo interno físico es el dado en [11].

Una síntesis de ellas se presenta en la siguiente figura :



OBJETIVOS BASICOS	INFORMACION OBTENIDA	FORMALISMO
• Determinar las funciones a automatizar ya sean previstas o en curso de realización. • Inventariar las informaciones o memorizar que describen la actividad de la organización. • Seleccionar de entre las funciones ya automatizadas las que forman parte de la BD.	Individuos tipo Propiedades Relaciones Reglas de gestión Limitaciones de integridad	• Metodología de diseñar. Otras técnicas.
• Expresar los datos obtenidos en la etapa anterior en forma mecanizable con una codificación adecuada y un vocabulario y ortografía normalizados.	Datos informatizables.	
• Obtener una expresión estructurada de las informaciones comunes a todos los utilizadores de la BD de la organización. • Determinar todos los accesos posibles, actuales y potenciales. • Verificar sintácticamente el modelo obtenido.	Modelo conceptual verificado.	• Técnicas de clasificación que permiten obtener grupos candidatos e individuos. • Técnicas de concepción asistidas por ordenador. • Algoritmos de verificación de compatibilidad de las cardinalidades.
• Crear los modelos externos utilizando el vocabulario del modelo conceptual bruto.	Subconjunto de datos utilizados para actualizar alguna ocurrencia. Subconjunto de datos utilizados para consultas a la base.	
• Validar el modelo conceptual bruto para cada modelo externo. • Obtener una estructura de datos sin limitaciones informáticas, no representable directamente en LDD, pero expresado en un formalismo mecanizable.	Modelo conceptual validado.	• Utiles disponibles para asegurar que los modelos externos son deducibles del modelo conceptual.
• Traducir el modelo conceptual a un modelo adaptado a las limitaciones de la informática sin perder el significado expresado en el modelo conceptual. • Traducir los modelos externos en subesquemas lógicos a partir del subconjunto del modelo conceptual equivalente al modelo externo.		• Reglas para respetar las limitaciones de las relaciones del DC o BD utilizado. • Reglas ad-hoc para asignar al modelo interno.
• Evaluar la frecuencia de utilización de los modelos externos. • Valorar en forma global los accesos a nivel del modelo interno lógico en función del análisis de la actividad de cada subesquema lógico. • Seleccionar el mejor modelo lógico interno.		• Primitivas de acceso. • Función económica de optimización que considere la actividad y el volumen de los datos a memorizar.
• Considerar las limitaciones físicas de los SGBD para la implantación física de los datos. • Investigar la reagrupación de los registros físicos de las informaciones relacionadas lógicamente.	Base de datos con una actividad total reducida.	
• Traducir el modelo interno físico utilizando el LDD del SGBD disponible.		• Normativas propias del SGBD utilizado.

De la metodología propuesta nos centraremos en el tema que denominamos optimización del modelo interno.

3.0 OPTIMIZACION DEL MODELO INTERNO

3.1 ESTUDIO DE LA ACTIVIDAD DE LOS MODELOS EXTERNOS

Los programas que actúan sobre una BD están destinados a satisfacer las necesidades de los usuarios expresadas mediante modelos externos -subesquemas en el modelo Codasyl-. En la gran mayoría de los casos las acciones que con ellos se realizan apuntan a crear, eliminar, consultar y modificar datos. Su campo de acción puede ser muy variado y afectar a diferentes volúmenes de datos.

Para poder evaluar esta actividad -entendiendo por tal al producto de acciones elementales por la frecuencia de utilización del programa-, usaremos un conjunto de primitivas de acceso a la BD, que representan funciones específicas, y que definiremos sobre el modelo interno lógico[1].

Algunas de estas funciones son :

1. Accesar una ocurrencia de un registro
2. Incorporar una ocurrencia de un registro
3. Suprimir una ocurrencia de un registro
4. Modificar una ocurrencia de un registro

Dado un esquema de BD, la acciones a realizar para poblar la base de datos se expresan en términos de primitivas valorizadas considerando las cardinalidades medias de cada set y la cantidad de registros nuevos a incorporar. Para obtener una valoración global de los accesos a nivel de modelo interno logico, deberá analizarse y valorar cada uno de los subesquemas lógicos e ir acumulando el conjunto de las actividades. De esta manera se obtiene un modelo interno lógico valorado.

El modelo interno obtenido es una traducción del modelo conceptual a algún modelo implementable, en este caso Codasyl, sin considerar el rendimiento del modelo; por lo tanto debemos tratar de minimizar una función económica (FE) que nos permitirá seleccionar el mejor modelo interno.

Una expresión para esta función FE podría ser:

$$FE = (NA \times PA) + (NC \times PAL)$$

donde

NA = número de accesos por unidad de tiempo
PA = precio del acceso por unidad de tiempo
NC = número de caracteres almacenados
PAL= precio del almacenamiento de un caracter por unidad de tiempo

La utilización de esta función supone considerar las siguientes hipótesis :

- La actividad de una aplicación puede medirse en número de accesos, independientemente de la programación de dicha aplicación;
- La medida determinada no considera los efectos borde, entendiendo como tales a los accesos extras producidos por el uso de pistas de desbordamiento o por falta de reorganización de los datos;
- El volúmen de datos accesorios (punteros) necesarios en la implantación física de la BD es constante, cualquiera que sea la técnica utilizada; y
- Las máquinas empleadas tienen jerarquía de memoria simple: principal y secundaria; el volúmen almacenado y el número de accesos a la memoria secundaria son los factores antagónicos para los cuales es necesario buscar un equilibrio óptimo.

Por otra parte, y a nivel de modelo interno, se supone :

* que los volúmenes almacenados provienen de los campos contenidos en los registros; y
* que los accesos a la memoria secundaria provienen únicamente de la actividad definida por los ME sobre el modelo interno en término de primitivas.

3.2 VALORACION DE LAS PRIMITIVAS EN TERMINOS DE ACCESOS

El objetivo de esta etapa es obtener una medida uniforme de la actividad del modelo interno lógico.

A cada primitiva de acceso se le asigna un número equivalente, en términos de acceso, y que sea compatible con los accesos del sistema de gestión que se utiliza [5].

3.3 PROCEDIMIENTOS DE OPTIMIZACION

Para optimizar el modelo podemos utilizar varios criterios :

Crear redundancia

- Repetir algunas informaciones con el fin de disminuir la actividad;

- Hacer emigrar algunos campos que provocan accesos contínuos a los registros cabecera, desde un registro cabecera hacia un registro miembro;

- Crear sets redundantes para evitar ciertos recorridos; esto corresponde a crear sets equivalentes a la composición de dos o más sets que forman una jerarquía. De esta manera se economizan accesos, especialmente en las operaciones de acceso a la cabecera.

Abandonar algunos sets ordenados

- A la luz de los casos analizados, está claro que mantener sets ordenados implica realizar un considerable número de accesos cuando se efectúa una actualización. Si se elimina el ordenamiento de algunos sets de cardinalidad media muy elevada, se disminuye notablemente la actividad. Debe considerarse además que esta optimización se hace sin incrementar el almacenamiento.

Revisar el modelo conceptual

- Otra posibilidad consiste en reestudiar la definición de aquellas entidades que se utilizan en forma disjunta por dos subesquemas lógicos de muy elevada tasa de actividad, separándolas en dos entidades, lo que nos obligará a definir una relación adicional, si algún otro subesquema utiliza la entidad original, para indicar que las dos nuevas entidades están a su vez relacionadas en forma biunívoca.

Revisar las cardinalidades

- La posibilidad de definir en el modelo conceptual relaciones n-arias o binarias del tipo 0-n - 0-n, es una riqueza semántica de la cual puede abusarse. Dimensionar adecuadamente el modelo reducirá considerablemente la actividad del modelo interno.

Conociendo la valoración del modelo interno lógico, se reestructuran los registros y sets en base a los criterios expuestos anteriormente, redefiniendo los subesquemas que correspondan y valorizando el nuevo modelo interno resultante.

El modelo final será aquel que minimice la función económica y que mantenga un equilibrio entre el número de accesos y el volúmen de datos almacenados.

Este es un proceso tedioso y sólo será eficaz en la medida que se disponga de un producto utilitario que permita obtener y valorar subesquemas lógicos.

Al término de esta fase de optimización se podría afirmar que el diseñador ha alcanzado un compromiso(trade-off) entre volúmen y acceso, independiente del SGBD a usar.

4.0 CONCLUSIONES

En múltiples ocasiones hemos estado frente a BD que satisfacen en plenitud las exigencias lógicas de los usuarios en cuanto a la definición de los individuos, sus asociaciones, y sus propiedades para que recojan la semántica incorporada en la realidad a percibir; sin embargo a la hora de la implementación la problemática del rendimiento pasa a ser crucial. Esto último obliga a buscar modelos internos que maximicen el rendimiento de una BD dada. Este trabajo se inscribe justamente en la obtención de alguna metodología que nos permita determinar las estructuras internas mas adecuadas para una realidad dada. Esta última dada no solo por los elementos que la componen, sus características, y sus relaciones, sino incorporando como parte de esa realidad lo que en el presente trabajo hemos llamado el nivel de actividad de sus elementos.

Queda pendiente lo relativo a la automatización de la metodología aquí sugerida, un afinamiento de ella, su extensión a sistemas de gestión de BD basados en modelos distintos al CODASYL, y su aplicación fuera de un marco académico.

Entre las limitaciones que presenta lo sugerido no podemos ignorar la exigencia que información que no siempre está disponible cuando se aborda la tarea de diseñar una BD. El conocimiento de las cardinalidades medias de los individuos y sus asociaciones, es clave en la aplicación de esta metodología. También lo es el conocimiento del comportamiento futuro de estas cardinalidades. Prever el comportamiento que tendrán los elementos constitutivos de la BD nos ahorrará futuras reorganizaciones altamente indeseables por lo que conlleva en consumo de recursos de toda índole. Otra limitación que debemos mencionar radica en su caracter determinístico. Una alternativa más acuciosa nos debe conducir al manejo de modelos probabilísticos. No obstante lo anterior hemos optado por un conocimiento determinístico de las variables comprometidas en aras de su simplicidad y facilidad de implementación.

5.0 BIBLIOGRAFIA

[1] CONCEPTION D'UN SYSTEME D'INFORMATION
H. Tardieu, D. Nancy, D. Pascot
Les Editions d'Organisation, Paris, 1979.

[2] UNA METODOLOGIA DE DISENO LOGICO DE BASES DE DATOS
Rodolfo Schmal S., Tesis de Maestría, 1985.

[3] COMPUTER-ASSISTED DATA BASE DESIGN
George V. Hubbard
Van Nostrand Reinhold Company, 1981.

[4] ENTITY-RELATIONSHIP MODEL-TOWARD A UNIFIED VIEW OF DATA
P.P. Chen, ACM TODS, VOL 1 Nro 1, Marzo 1976.

[5] DISENO CONCEPTUAL DE UNA BASE DE DATOS DE SEGUIMIENTO CURRICULAR Y CONTROL ACADEMICO
Yanko Ossandón N., Tesis de Maestría, 1984.

MODELO BORROSO DE UN SISTEMA DE RECUPERACION DE INFORMACION*

Miguel Francisco Buitrago Botero
Ingeniero de Sistemas
Depto. Análisis y Programación
Empresas Públicas de Medellín
A.A. 4928 Medellín, Colombia.

William Jaramillo Jaramillo
Ingeniero de Sistemas
Depto. de Sistemas Delima
A.A. 4928 Medellín, Colombia.

Resumen

En la mayoría de los sistemas de recuperación de información el usuario debe expresar la pregunta en forma de expresiones booleanas. Aunque este tipo de sistemas trabajan, tienen una serie de desventajas que es necesario estudiar y tratar de remediar.

En la literatura conocida hay intentos para aplicar la teoría de conjuntos borrosos en la recuperación de información. En este trabajo explicamos las características de un sistema boolenao de recuperación de información y sus mayores desventajas; posteriormente y dado que el concepto de conjunto borroso es una generalización de la noción convencional de conjunto, proponemos un modelo borroso de un sistema de recuperación de información que soluciona algunas de las desventajas del booleano tradicional. El modelo incluye la forma como se deben expresar las preguntas al sistema y la función de recuperación que es de naturaleza borrosa.

* El presente trabajo es un extracto del proyecto de grado presentado por los autores a la universidad EAFIT.

1. Sistema booleano de recuperación de información

Un sistema booleano de Recuperación de Información (SBRI) es una quintupla [Radecki (R8)]:

$$I = (D, T, P, F, \sigma)$$

En donde D es un conjunto finito de documentos y T un conjunto finito de términos indices o descriptores. Los elementos de P son preguntas que un usuario potencial tiene para el sistema; la sintaxis de las preguntas la conforman los elementos de T conectados por los operadores lógicos Y, O, NO. El siguiente elemento de la quintupla es una relación en el producto cartesiano DxT. Formalmente:

$$F : DxT \longrightarrow \{0,1\}$$

donde F es una función que toma el valor de uno si el documento d trata el tema t y cero si no lo trata.

El último elemento, σ, es una función de la forma:

$$\sigma : DxP \longrightarrow \{0,1\}$$

que asigna para cada par ordenado (d,p); $d \in D$, $p \in P$ un valor en $\{0,1\}$ llamado relevancia formal* del documento para la pregunta p. La relevancia formal expresa el grado de similaridad entre un documento y una pregunta; en otras palabras, indica si el documento d satisface o no la pregunta p. Si $\sigma = 1$, el documento es relevante para la pregunta y el sistema lo recupera; si $\sigma = 0$, el documento no es relevante para la pregunta y por lo tanto no lo recupera.

* Existe diferencia entre relevancia y pertinencia [Radecki (R8)]. Relevancia es una medida de que tan adecuado es el contenido de un documento con respecto al contenido de la pregunta que el usuario formula al sistema. En un sentido amplio, pertinencia es una medida de que tan adecuado es el contenido de un documento con respecto a la necesidad de información de un usuario. El grado de relevancia de cada documento es asignado por un especialista en el área; mientras que el grado de pertinencia es asignado por el usuario. Un documento puede ser muy relevante pero poco pertinente porque el usuario ya esta familiarizado con su contenido. La relevancia formal denota el grado de similaridad, medido por una función de matching, entre la representación del documento y la pregunta formulada al sistema. En otras palabras, el grado de relevancia formal de un documento en la colección lo determina la función de matching que usa el sistema de recuperación de información.

Cuando la pregunta consta de sólo un elemento de T, el cálculo de para cada documento de D es inmediato; cuando la pregunta consta de varios elementos de T unidos por operadores lógicos, el cálculo se hace aplicando las reglas de la lógica matemática:

$$\sigma(d, t_1 \wedge t_2) = \text{Min}[\sigma(d,t_1), \sigma(d,t_2)]$$

$$\sigma(d, t_1 \vee t_2) = \text{Max}[\sigma(d,t_1), \sigma(d,t_2)]$$

$$\sigma(d, \neg t) = 1 - \sigma(d,t)$$

La forma de procesar la pregunta $p_1 = t_1 \wedge t_2$ es encontrar el conjunto de documentos que tratan el tema t_1 e interceptarlo con el conjunto de documentos que tratan el tema t_2. Aquellos documentos que pertenecen a la intersección (valor de relevancia formal igual a 1) se presentan como respuesta a la pregunta p_1. Hemos asociado a cada componente (término índice) de la pregunta un subconjunto de documentos de D y el sistema de recuperación efectua las bien conocidas operaciones de unión, intersección y complementación para corresponder al Y, O, NO de la expresion Booleana. Esta propiedad, que es una de las mayores ventajas de un SBRI se conoce con el nombre de separabilidad.

Despues de calcular la relevancia formal de todos los documentos en la coleccion para una pregunta p, el Sistema selecciona aquellos documentos cuyo valor de σ es igual a 1 y los entrega al usuario; este analiza la salida dada por el sistema y escoge algunos documentos o reformula la pregunta hasta satisfacer completamente su necesidad de información.

El sistema de Recuperación descrito recibe el nombre de Booleano porque el conjunto potencia de D con las operaciones unión ($\cup$) e intersección ($\cap$) satisface las condiciones necesarias para ser un Algebra de Boole, Whitesitt (W2).

2. Fallas de un SBRI

Aunque un SBRI es sencillo, fácil de entender y a nivel operativo el de uso más difundido, tiene ciertas desventajas, Salton (S1), que se resumen en:

2.1. Es difícil controlar la cantidad de documentos que dan respuesta a una pregunta. Dependiendo de la frecuencia de los descriptores en la colección de documentos y la combinación que se use para formular una pregunta, se pueden obtener muchos documentos que la respondan. El sistema no dispone de un mecanismo que permita restringir el número de documentos recuperados; y en caso que existiera, no se puede garantizar que los recuperados son los más importantes.

2.2. La salida que se obtiene en respuesta a una pregunta carece de cualquier orden de presunta importancia para el usuario.

Recordemos que la relevancia formal para un documento recuperado es igual a uno; es decir, todos son teoricamente de igual importancia.

2.3. No es posible asignar algun factor de relevancia de los descriptores con respecto a los documentos. Es posible que tanto el documento d_1 como el documento d_2 traten el tema t pero también es posible que el tratamiento que hace d_1 de t sea cuantitativa y/o cualitativamente diferente del que hace d_2; un SBRI no es capaz de reflejar esta diferencia. Un caso similar se presenta con los términos en las preguntas.

3. Sistema borroso de recuperación de información

Dado que el concepto de conjunto borroso es una generalización de la noción convencional de conjunto, Zadeh (Z1), la generalización de un SBRI a uno con características borrosas se puede hacer fácil y naturalmente, redefiniendo las funciones F y σ.

Las definiciones para D, P y T permanecen invariantes. Las únicas variaciones, aunque muy importantes, ocurren sobre los rangos de F y σ; en vez de definirlos en el conjunto $\langle 0,1 \rangle$ como haciamos en el booleano, lo haremos ahora en el intervalo $[0,1]$.

Un valor real de F en $[0,1]$ detalla la importancia del término t en la repesentación de un documento d. De igual manera, un valor de σ en $[0,1]$ indica la relevancia formal del documento d para la pregunta p. El problema consiste en encontrar un valor apropiado que represente con certeza el grado con que un documento d es relevante para una pregunta p.

Los valores de F parecen fáciles de calcular. En el ambiente de la Recuperación de Información es común escuchar expresiones como: "Este documento trata muy bien el tema", "Este documento es muy simple, te recomiendo mejor este libro", etc. En el primer caso es casi seguro que el valor de la función de pertenencia del documento al tema es alto; en el segundo, el adjetivo simple debe hacer referencia a un valor bajo en el valor de la función de pertenencia, mientras que el libro debe tener un valor un poco más alto.

La situación anterior nos hace pensar que, intuitivamente, tenemos claro el concepto de relevancia de un documento para con un término o viceversa; aunque debemos notar que se presenta un problema al definir el nivel de detalle que se debe alcanzar en el proceso de indización y por lo tanto en el calculo de F.

En un sistema borroso de recuperación, como en el caso booleano, se puede asociar a cada $t \in T$ un conjunto S, borroso en este caso, de documentos relacionados con dicho término; por lo tanto el sistema borroso también satisface la propiedad de separabilidad.

Disponemos ahora de un modelo mucho más flexible, con la garantía que tenemos las mismas propiedades de un SBRI tradicional; de hecho,

en los puntos extremos su comportamiento es análogo. La única excepción es que, en general; $S \cap S' \neq \emptyset$ y $S \cup S \neq D$, debido a la definición de conjunto borroso, pero que no tiene importancia para una pregunta p en el sentido general de la Recuperación de Información.

Indiscutiblemente esta generalización abre todo un mundo de posibilidades a los usuarios de un sistema de recuperación de información.

4. Pesos y umbrales

La generalización de un SBRI a un sistema boleano de recuperación con características borrosas inmediatamente proporciona la facilidad de entregar los documentos que más se acomoden a las necesidades reales del usuario. Otra ventaja es la de poder ordenar la salida por valor de relevancia formal. Esta última le sugiere por ejemplo a un usuario, leer primero aquellos documentos con mayor valor de relevancia formal, pues es donde se sospecha esta la mayor parte de lo que el necesita.

Hasta el momento no hemos hablado de la posibilidad de expresar preguntas del tipo αt, donde $t \in T$ y α es un número posiblemente en $[0,1]$. Existen en la literatura conocida dos interpretaciones del valor α; algunos autores lo llaman umbral y otros lo denominan peso.

Las ideas de peso y umbral nacen del hecho que es muy importante que el usuario pueda precisar la importancia relativa, dentro de una pregunta, de un término con respecto a los demás (peso), o quiera especificar ciertos niveles mínimos de información contenida en los documentos por recuperar (umbrales).

En el presente trabajo y por limitaciones espacio-temporales nos centraremos en el concepto de umbrales. Para mayor información sobre pesos sugerimos leer: Buitrago y Jaramillo (B4), Buell y Kraft (B7), Bookstein (B8).

5. Umbrales

Intuitivamente un umbral es un punto de corte, un tope, un "lugar" de transición. En recuperación de información es común escuchar expresiones del tipo: "Quiero un libro que contenga mucho de Bases de Datos", "El documento d trata poco el tema t", etc. Aquí, las palabras mucho y poco pueden interpretarse como umbrales. Un umbral es entonces un valor linguístico por encima o por debajo del cual sucede algo; en nuestro caso, la existencia o no existencia de documentos.

El concepto de umbral se basa en la noción intuitiva que $F(d,t) > \alpha$, es cualitativamente diferente del caso en el cual $F(d,t) < \alpha$; con α =

umbral. En la primera situación, σ debe reflejar el grado con que la función de pertenencia del documento d al tema t, [F(d,t)], sobrepasa el umbral α; en el segundo, el grado con que se acerca al umbral.

La manera más sencilla de usar umbrales en las preguntas consiste en definir un valor fijo α (umbral) para todos los términos de la pregunta; si un documento tiene una función de pertenencia para el término t mayor o igual que el umbral α ,[$F(d,t) > \alpha$], se le asigna un valor a σ de uno; Si la función de pertenencia es menor que α,σ toma el valor de cero. La salida es muy similar a la de un SBRI tradicional; todos los documentos recuperados tienen el mismo valor de relevancia formal ($\sigma = 1$).

Buell y Kraft (B9) hacen énfasis en que σ debe reflejar el mayor o menor grado con que el documento satisface el umbral asociado a un término t. Con base en este supuesto estudian las características que debería tener la función σ de manera análoga a como lo hace Bookstein (B8) para los pesos. De nuevo las limitaciones de espacio y tiempo nos impiden mostrar los detalles matemáticos de dichas características.

A continuación proponemos un modelo de un sistema de recuperación de información, que utiliza conceptos booleanos, borrosos y de umbrales. A más de resolver algunos problemas del SBRI, es sencillo e intuitivamente pragmatico.

6. Modelo

Como explicamos antes un sistema de recuperación de información es una quintupla de la forma (D,T,P,F,σ). En nuestro modelo, las definiciones para D, T, P y F son las mismas que las descritas en 1.; los elementos de P son, en notación BNF, del tipo:

$$p = (q \ op \ q) / q$$

$$q = at / \neg at$$

$$op = \vee / \wedge$$

$$\alpha \in [0,1]$$

Ejemplos de elementos validos de P son:

$$p_1 = at_1 \vee bt_2$$

$$p_2 = \neg at_1 \wedge bt_2$$

Un umbral α en preguntas del tipo at , indica el límite mínimo de información que debe contener un documento acerca del tema t para que sea recuperado por el sistema; un umbral α en preguntas del tipo $\neg at$, indica el límite máximo de información que debe contener un documento para que sea recuperado.

La definición anterior se basa en el hecho que si at indica el deseo de documentos que hablen de t con un valor de función de pertenencia mayor o igual que a, $\neg at$ debe indicar que no se desean documentos que hablen del término t con un valor mayor o igual que a; o sea que se desean aquellos que hablen de t con un valor menor que a.

En el caso del SBRI, si el término t ($t \in T$) aparece en una pregunta, este hace referencia a todos los documentos que hablen de el (función de pertenencia = 1); si el término t aparece negado, $\neg t$, este indica que no se desean documentos que hablen de él (función de pertenencia = 0). Si en nuestro modelo los umbrales se restrigen a 0 y 1, su comportamiento es análogo al Booleano. De lo anterior se infiere que el sistema booleano es un caso particular de un sistema borroso de recuperación de información.

Esta definición de umbrales permite al usuario expresar preguntas del tipo: "Quiero documentos que hablen mucho de t_1 y no mucho de t_2". En el lenguaje de consulta esta pregunta podría tener la siguiente sintaxis:

$$p = at_1 \wedge \neg bt_2 \text{ ; con } a \geq 0.9 \text{ y } b \leq 0.8$$

Como en el caso booleano, el sistema busca los documentos asociados al término t_1 que cumplen el nivel de umbral exigido (0.8) y los asociados a t_2 que satisfacen el umbral respectivo (0.8); luego efectua la intersección entre estos dos subconjuntos y la entrega al usuario en respuesta a la pregunta inicial, p.

El ultimo elemento de la quintupla, σ, lo definimos así:

$$\sigma = \begin{cases} F(d,t) & \text{si } F(d,t) \geq a \\ 0 & \text{si } F(d,t) < a \end{cases}$$

Para preguntas tipo at

$$\sigma = \begin{cases} 1 - F(d,t) & \text{si } F(d,t) < a \\ 0 & \text{si } F(d,t) \geq a \end{cases}$$

Para preguntas tipo $\neg at$

Para preguntas tipo at, si la pregunta esta conformada por sólo un término, asignamos como valor de relevancia formal el valor F(d,t) que tiene el documento; si la pregunta la conforman varios términos unidos por operadores lógicos, el valor de relevancia formal es el que resulta de aplicar las operaciones de máximo y mínimo. De manera similar se puede razonar para preguntas tipo $\neg at$. Un ejemplo ilustrará mejor esta situación.

Supongamos el siguiente conjunto de documentos y sus valores de función de pertenencia para el término t_1 y t_2

d_1:	$0.8\,t_1$	$0.62\,t_2$	d_4:	$0.9\,t_1$	$0.83\,t_2$
d_2:	$0.5\,t_1$	$0.5\,t_2$	d_5:	$0.6\,t_1$	$0.68\,t_2$
d_3:	$0.7\,t_1$	$0.8\,t_2$			

Si el usuario formula la siguiente pregunta:

$p_1 = 0.15\, t_1$

El sistema respondería:

$d_4 = 0.9$

$d_1 = 0.8$

Obviamente, el documento d_4 como el documento d_1 sobrepasan sus necesidades iniciales; en el primer caso en 0.15 y en el segundo en 0.05.

En el caso que la pregunta sea:

$p_2 = 0.7\, t_1 \wedge 0.7\, t_2$

La respuesta es:

$d_4 = 0.83$

$d_3 = 0.74$

Aquí hemos aplicado operación Min para calcular los valores 0.83 = [Min (0.9,0.83)] y 0.74= [Min (0.74,0.8)].

Veamos ahora en más detalle algunos aspectos interesantes de las respuestas a las preguntas p_1 y p_2.

La respuesta a la pregunta p_1 fueron los documentos d_4 y d_1; pero acaso el documento d_3 no parece ser importante para esta pregunta?. La respuesta es que sí parece ser importante, puesto que casi llega al umbral requerido. sólo le falto 0.01 para cumplirlo ($\alpha = 0.75$; $F(d_3, t_1) = 0.74$).

En el caso de la pregunta p_2 sucede algo similar con el documento d_5 sólo le falto 0.03 para cumplir con el umbral de t_1 y 0.02 para cumplir con el de t_2.

Para obviar la situación anterior, hemos definido una función $\mathcal{E}$, que incluye en los documentos por recuperar aquellos que estan cerca del umbral requerido. Dado que el concepto de cerca es subjetivo y no debe ser el mismo para umbrales altos que para umbrales bajos definimos a $\mathcal{E}$ de tal manera cumpla las siguientes propiedades:

- Para preguntas del tipo αt (positivas), fijamos a $\mathcal{E}$ alto para valores altos de α y bajo para valores bajos de α.

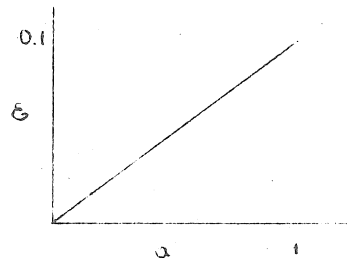
- Para preguntas del tipo $\neg \alpha t$ (negativas), $\mathcal{E}$ debe ser bajo para valores altos de α y alto para valores bajos de α.

La justificación es la siguiente: Sea $S(\alpha, t)$ el conjunto de documentos que satisfacen el umbral α para la pregunta αt; entonces, la cardinalidad de $S(\alpha, t)$ varia inversamente con α; esto es:

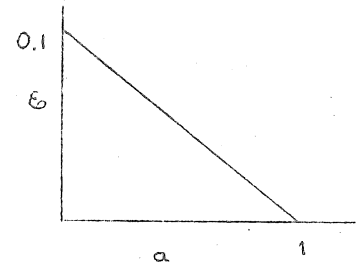
$$|S(a,t)| > |S(b,t)| \quad \text{si } a < b$$

lo cual indica que el número de documentos que satisfacen un umbral alto es menor que el número de documentos que cumplen un umbral bajo; es decir, es más fácil alcanzar un umbral de 0.2 que un umbral de 0.9. Este razonamiento nos lleva a pensar que $\frac{\partial \varepsilon}{\partial a} > 0$ (o sea: ε es una función creciente de a).

Una función que cumple con las características anteriores y que proponemos como punto de partida es:



para preguntas tipo at



para preguntas tipo ¬at

El máximo valor que toma la función epsilon es 0.1. Este valor corresponde a una apreciación subjetiva de los autores.

Un aspecto interesante de la definición de ε es que el caso en el cual en una pregunta aparece un término de la forma ¬at ($a = 0$) es diferente al caso en el cual el término no aparece en la pregunta. En el primero, el sistema recupera aquellos que estan cerca de 0; en el segundo, el sistema no toma ninguna acción al respecto. Por ejemplo,

$$p_1 = at \wedge \neg bt \; ; \; b = 0$$

$$p_2 = at$$

aunque a primera vista las preguntas son iguales, los conjuntos de documentos que las responden no lo son debido a la definición de ε.

Las ventajas que presenta el modelo antes explicado son:

- Conserva las propiedades del SBRI. El usuario puede formular preguntas de una forma más o menos natural por medio de expresiones conformadas por los elementos de T con el nivel de información que nececita (umbral) y los operadores lógicos Y, O, NO. Además, conserva la propiedad de separabilidad que permite descomponer una pregunta en sus términos individuales, seleccionar los subconjuntos correspondientes a cada término y

aplicar luego las operaciones de Max y Min para encontrar el conjunto por recuperar.

La propiedad de separabilidad también permite encontrar formas equivalentes, más fáciles de manipular, para una pregunta. Esto es especialmente útil al momento de procesar preguntas complejas o que involucran muchos operadores lógicos.

- Permite expresar la importancia de un documento para un término. Esta característica hace que el proceso de indización sea más preciso pues ya no es suficiente asignar términos a documentos sino que es necesario cuantificar el grado con que el documento trata el tema.

- De manera análoga, permite expresar, en las preguntas, el grado de información que debe contener un documento para que sea efectivamente recuperado. La consecuencia de esta propiedad es que la cardinalidad del conjunto de documentos recuperados es menor que la cantidad de documentos recuperados en respuesta a la misma pregunta, sin umbrales, por un SBRI.

- Permite ordenar la salida por valor de relevancia formal. Esta es una ventaja muy importante, puesto que el usuario dispone ahora de un orden recomendado de lectura que no proporcionaba el caso Booleano. Ante una cantidad de documentos recuperados, el usuario tiene más elementos para decidir cual debe leer primero.

- Por último, el modelo es general o sea que puede ser manejado como un SBRI tradicional.

Dentro de las desventajas tenemos:

- El hecho que el sistema permita una indización cualitativa además de ser una gran ventaja, puede llegar a convertirse en una desventaja, ya que realizar el proceso de indización con valores de función de pertenencia bajos se puede convertir en una tarea monumental.

- En el caso que se formule una pregunta del tipo τ_{at} con $\alpha = 0$ el sistema recupera aquellos documentos cerca de cero dependiendo del valor que tome μ (en nuestro caso aquellos documentos que tratan el tema t con un valor < 0.1). Si un usuario definitivamente no quiere estos documentos, esto, que inicialmente se planteó como una facilidad, se convierte en una desventaja.

  Una solución es pensar en expresar las preguntas como ternas del tipo: (t, α, β), conectadas por operadores lógicos. En la forma anterior de expresar la pregunta, β es un valor real en el intervalo $[0, 1]$, que opera como un multiplicador de μ. Un valor de β igual a uno no tiene ningún efecto

sobre ω . Un valor de γ^a en (0,1) disminuye el valor de hasta una cantidad igual a $\omega \times \gamma^a$. Un valor de γ^a igual a cero elimina el efecto de ω . De esta manera, un usuario que este interesado en documentos que traten el tema t_1 y no traten el tema t_2 podría expresar la pregunta así:

$$p = (t_1, a, 1) \wedge (\neg t_2, b, 0) \quad \text{con } b = 0$$

7. Conclusiones

El hecho de que la Teoría de Conjuntos clásica sea un subconjunto de la Teoría de Conjuntos Borrososos tiene implicaciones muy importantes en el proceso de conocimiento humano y no debe pasar desapercibido.

En este trabajo mostramos como se puede aplicar la teoría de conjuntos borrosos a la recuperación de información. El modelo descrito es sencillo y aparentemente de fácil implementación.

Es necesario que nuestros paises presten mayor atención a las nuevas metodologías y formas de trabajo, fomenten la creatividad y hagan uso de sus propios recursos.

Bibliografía y referencias

B1 Bookstein, A; Cooper, W. "A MODEL FOR INFORMACION RETRIEVAL SYSTEM". Library Quartely V.46, N.2, pp 153-167, 1976

B2 Bookstein, A. "THE PERILS OF MERGING BOOLEAN AND WEIGHTED RETRIEVAL SYSTEMS", ASIS, V.29, N.3, pp 136-157, 1978.

B3 Bookstein, A. "FUZZY REQUEST: AN APPROACH TO WEIGHTED BOOLEAN RETRIEVAL" ASIS V.31, pp 240-247, Julio 1980.

B4 Buitrago, Miguel F. y Jaramillo W. "CONJUNTOS BORROSOS Y RECUPERACION DE INFORMACION", Proyecto de grado, Universidad EAFIT, 1987

B5 Burgerss, R.S. "WHAT IS INFORMATION RETRIEVAL?", Sociedad de Bibliotecarios de Puerto Rico, Enero-Junio 1972

B6 Buell, Duncan A. "A GENERAL MODEL OF QUERY PROCESSING IN INFORMATION RETRIEVAL SYSTEMS", Information & Management, V.17, N.5, pp 249-262, 1981

B7 Buell, Duncan A. y Kraft, Donald H. "A MODEL FOR WEIGHTED RETRIEVAL SYSTEM", ASIS, V.32, N.1, pp 211-216, 1981

B8 Bookstein, A. "A COMPARISON OF TWO SYSTEMS OF WEIGHTED BOOLEAN RETRIEVAL" ASIS, Julio 1981.

B9 Buell, Duncan A. y Kraft Donald H. "THRESHOLD VALUES AND BOOLEAN RETRIEVAL SYSTEMS", Informacion & Management, V.17, pp 127-136, 1981

C1 Cleverdon, C. W., Mills, J. y KEEN, M., "FACTORS DETERMINING THE PERFORMANCE OF INDEXING SYSTEMS", V.1 -Desing, V.11-Test Result, AQSLIB Cranfield Project, Cranfield, 1966

C2 Craft, W.Bruce. "BOOLEAN QUERIES AND TERM DEPENDENCIES IN PROBABILISTIC RETRIEVAL MODELS", ASIS, V.37, N.2, pp 71-77, 1986

C3 Crouch, D. "A CLUSTERING ALGORITHM FOR LARGE AND DYNAMIC DOCUMENT COLLECTION", Ph.D. Tesis, Southern Methodist University, 1972

M3 Molina, J.F. "APORTES DE LA TEORIA DE SUBCONJUNTOS BORROSOS EN INVESTIGACION OPERACIONAL". Comunicación presentada en el X Congreso Colombiano de Calculo Electronico e Investigación Operacional, Bucaramanga, 1986

R2 Radecki, Tadeusz. "ON THE INCLUSIVENESS OF SYSTEMS FOR RETRIEVAL DOCUMENTS INDEXED BY UNWEIGHTED DESCRIPTORS", Information & Management, V.17, N.5 pp 227-237, 1981

R3 Radecki, Tadeusz. "FUZZY SET THEORICAL APPROACH TO DOCUMENT RETRIEVAL", Information & Management, V.15, pp 247-259, 1979

R4 Radecki, Tadeusz. "GENERALIZED BOOLEAN METHODS OF INFORMATION RETRIEVAL", Int. J. Man-Machine Studies, 1983, V.18, pp 407-439, 1983

R8 Radecki,Tadeusz. "A THEORICAL BACKGROUND FOR APPLYING FUZZY SET THEORY7 IN INFORMATION RETRIEVAL", Fuzzu sets and systems, V.2, pp 169-183, 1983

S1 Salton, Gerard, "EXTENDED BOOLEAN INFORMATION RETRIEVAL", Comunications of the ACM, V.26, N.12, pp 1022-1036, 1983

V1 Van Rijsbergen, C.J. y Sperk Jones, K. "A TEST FOR THE SEPARATION OF RELEVANT AND NON-RELEVANT DOCUMENTS IN EXPERIMENTAL RETRIEVAL COLLECTIONS", Journal of Documentation, V.29, pp 251-257, 1973

V2 Van Rijsbergen, C.J. "INFORMATION RETRIEVAL", Butterworths, 1979

W1 Waller, W.G. y Kraft, Donald H. "A MATHEMATICAL MODEL OF WEIGHTED BOOLEAN RETRIEVAL SYSTEM", Information & Management, V.15, pp 235-245, 1979

W2 Whitesitt, J.E. "ALGEBRA BOOLEANA Y SUS APLICACIONES", C.E.C.S.A, 1975

Z1 Zadeh, L. A. "THEORY OF FUZZY SETS", Memorandum No UCB/ERL M77-1, University of califirnia at Berkeley, Enero 4, 1977

Z2 Zadeh, L. A. "THE CONCEPT OF A LINGUISTIC VARIABLE AND ITS APPLICATION TO APPROXIMATE RESONING", Part I: Information Science, V.8, pp 301-357, 1977, Part III: Information Science, V.9, pp 43-80, 1976

Z3 Zadeh, L. A. "FUZZY SETS", Information and Control, V.8, pp 338-353, 1965

COMPARACION ESTRUCTURAL DE INGRES Y ORACLE EN EL AMBIENTE UNIX

Por:
Roy Cahn-Speyer Valenzuela
Alfredo Hernández Cifuentes
Andrés Suárez Rosselli

Ministerio de Hacienda y Crédito Público

Bogotá, Colombia, Agosto de 1987

Presentado a la XIII Conferencia Latinoamericana de Informática.

En 1970 E.F. Codd propuso el modelo relacional para diseño de bases de datos. Luego de una década de trabajo investigativo tendiente a llevar a la práctica este modelo, ORACLE e INGRES fueron anunciados, en 1979 y 1980 respectivamente, como los primeros Sistemas Manejadores de Bases de Datos Relacionales (SMBDR) comerciales. Desde entonces se han lanzado al mercado múltiples implementaciones del modelo relacional. A pesar de ello, INGRES y ORACLE siguen siendo líderes de la tecnología relacional.

También a comienzos de la década del 70 se desarrolló y tomó altura comercial el sistema operacional UNIX. Entre las características de UNIX se destacan las siguientes: (1) Está escrito en un 95% en un lenguaje de alto nivel ("C"), lo que le da flexibilidad para el desarrollo de nuevo "software" y lo hace independiente en alto grado de los dispositivos de "hardware". (2) Provee un conjunto extenso de herramientas orientadas hacia el desarrollo incremental de "software". UNIX tiende a ser un estándar en sistemas operacionales. Recientemente UNIX ha salido de las universidades y los laboratorios científicos hacia ambientes gubernamentales y comerciales.

El presente artículo compara la implementación de INGRES y ORACLE en el ambiente UNIX desde el punto de vista de su arquitectura de archivos y procesos.

COMPARACION ESTRUCTURAL DE INGRES Y ORACLE EN EL AMBIENTE UNIX

1. INTRODUCCION

1.1 Presentación.

El presente artículo examina la implementación sobre UNIX de dos sistemas manejadores de bases de datos relacionales (SMBDR): INGRES, de Relational Technologies, y ORACLE, de Oracle Corporation.

Para cada SMBDR se estudió su arquitectura particular sobre UNIX, ubicando dos aspectos:

a. Estructura de procesos del SMBD como parte de la estructura de procesos de UNIX.
b. Estructura de archivos del SMBD como parte del "file system" de UNIX.

Tanto INGRES como ORACLE están clasificados como SMBD "totalmente relacionales" [CODD79][CODD83], en el sentido de que satisfacen las siguientes condiciones:

1. Toda la información de la base de datos está representada como valores en tablas.

2. No hay enlaces de navegación visibles por el usuario entre las tablas.

3. Los sistemas soportan todos los operadores del álgebra relacional, sin recurrir a comandos especiales para iteración y recursión, y sin restricciones impuestas por los métodos de acceso predefinidos.

4. Soportan las dos reglas generales de integridad del modelo relacional básico.

En la actualidad INGRES y ORACLE son los dos SMBDR con mayor éxito comercial. Su estrategia de mercadeo es la de ofrecer total transportabilidad a través de distintas marcas y tamaños de computadores. INGRES es el resultado de un proyecto de investigación de la Universidad de California, Berkeley, e inició su vida comercial en 1980. ORACLE se comenzó a desarrollar en 1977. La primera copia comercial de ORACLE se produjo en 1979 [SCHM83]. Su desarrollo teórico y práctico ha sido paralelo, aunque ha resultado en arquitecturas diferentes, como se muestra en este artículo.

A continuación se presentan los aspectos generales del manejo de archivos y procesos de UNIX. Las dos secciones siguientes hacen una descripción de la implementación sobre UNIX de INGRES y de ORACLE, respectivamente.

1.2 Generalidades del ambiente UNIX.

a. Sistema de archivos de UNIX ("file system").

Tiene estructura de àrbol. Cada archivo es o un directorio, con referencias a archivos del siguiente nivel, o un archivo de datos.

Cada archivo està dividido fìsicamente en bloques (generalmente de 512 o 1024 bytes), llamados pàginas. UNIX utiliza un àrea global de memoria llamada "buffer pool" para guardar temporalmente las pàginas a medida que son escritas o leìdas. El tamaño de este "buffer pool" se define al generar el sistema operacional.

Para responder una solicitud de lectura, UNIX mueve una o màs pàginas de memoria secundaria al "buffer pool" (si es necesario, haciendo que otras pàginas sean escritas al disco) y luego le retorna al usuario la cadena de bytes deseada. Si la misma pàgina es referenciada otra vez (por el mismo u otro usuario) cuando todavìa està en el "buffer pool", entonces no hay entrada/salida real al disco. Una solicitud de escritura simplemente mueve los datos escritos al "buffer pool"; en algùn momento posterior el manejador de "buffers" graba las pàginas en el disco.

El manejador de "buffers" de UNIX utiliza la estrategia de reemplazo de la "pàgina menos recientemente usada" (LRU). Cuando UNIX detecta acceso secuencial a un archivo, hace una prebùsqueda de las pàginas antes de que èstas sean solicitadas.

Las pàginas presentes en el "buffer pool" son escritas en el disco por varios mètodos:

- Cuando se emite el llamado del sistema "fsync".
- Cuando se ejecuta el comando de UNIX "sync".
- Cuando el "buffer pool" se aproxima a su lìmite de capacidad.
- Aproximadamente cada 30 segundos por el mismo UNIX.

b. Estructura de procesos de UNIX.

Un proceso en UNIX es un espacio de direcciones (de mìnimo 64K) que se asocia a un "user-id" y es la unidad de trabajo que maneja el "scheduler" de UNIX. El tamaño de un proceso està limitado por la capacidad de manejo de memoria virtual de la CPU.

Los procesos se pueden bifurcar en subprocesos; asì, un proceso dado puede ser la raìz de un àrbol de subprocesos. Varios procesos se pueden comunicar por medio de "pipes", señales, semàforos y memoria compartida.

UNIX permite que los procesos ejecuten còdigo reentrante, para que varios procesos compartan segmentos ejecutables. También se pueden definir segmentos de datos compartidos por varios procesos.

2. ARQUITECTURA DE INGRES SOBRE CENTIX *

2.1. Generalidades.

INGRES (Interactive Graphics and Retrieval System) es un sistema manejador de bases de datos relacional implementado originalmente sobre UNIX. La implementación está desarrollada primariamente en C, lenguaje de alto nivel en el que también está escrito UNIX [STON76]. INGRES soporta el lenguaje de interrogación QUEL (QUEry Language), y a partir de la versión 4 también soporta SQL, el cual es un estándar (ANSI e ISO).

La arquitectura de INGRES se mantuvo sin variaciones importantes desde su concepción inicial, durante el lanzamiento comercial de INGRES en 1980, hasta la versión 2 de 1984, similar a la que actualmente se conoce como INGRES/CS (para sistemas compactos). La versión 3 y posteriores tienen cambios significativos, principalmente para operar en equipos más poderosos y mejorar el "performance" de INGRES.

Las versiones de INGRES disponibles comercialmente no proveen documentación detallada de su arquitectura interna, particularmente en lo que a la estructura de procesos se refiere. Por esta razón, la principal fuente fué la observación directa sobre dos implementaciones diferentes, utilizando las herramientas de monitoreo que provee el sistema UNIX. Las implementaciones observadas fueron las siguientes:

- INGRES versión 3, bajo CENTIX versión 5.01, en un equipo Burroughs XE-550-3.
- INGRES/CS (versión 2), bajo UNIX Sistema V, en un equipo Tower/XP de NCR.

Esta sección describe la estructura de archivos y procesos de INGRES versión 3 desde el punto de vista de la implementación, es decir, INGRES operando bajo un ambiente específico de UNIX.

2.2. Estructura de archivos de INGRES.

Es igual en las dos versiones observadas, y se acoge a la arquitectura descrita en [STON76], aunque algunos nombres de directorios han cambiado.

(*) CENTIX es la forma comercial de UNIX Sistema V que ofrece UNISYS en la línea del desaparecido Burroughs.

2.2.1. Estructura del directorio "ingres":

INGRES manipula un subàrbol del "file system" de UNIX, cuya raíz es el directorio del usuario "ingres". El usuario "ingres" es el superusuario de INGRES. En dicho subàrbol residen por defecto las bases de datos además de los programas, librerías y utilidades de INGRES.

Los subdirectorios más importantes del directorio "ingres" son:

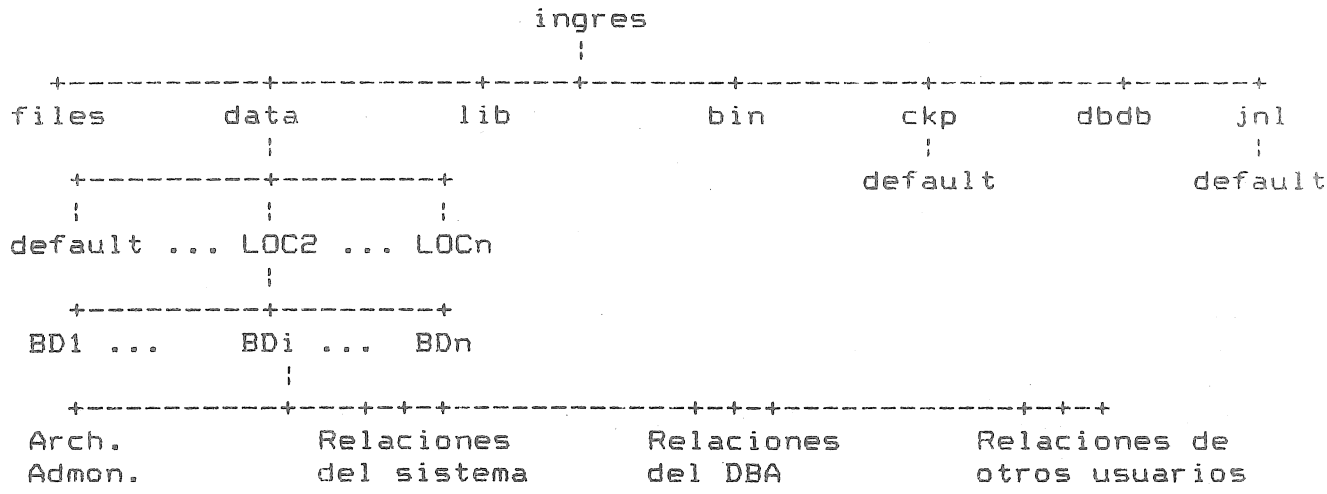


Figura 2.1

- files: Contiene los archivos con los textos de ayuda y los mensajes de error de todas las herramientas de INGRES.

- bin: Incluye todos los màdulos ejecutables de INGRES.

- lib: Contiene las librerías para compilación con EQUEL (el precompilador de QUEL).

- data: Tiene un subdirectorio "default", en el cual se mantienen subdirectorios con todos los archivos correspondientes a cada base de datos, a menos que el usuario "ingres" especifique otras "locaciones" (ver sección 2.2.2.) para algunas base de datos o para algunas tablas (estas locaciones alternas se muestran en la figura 2.1 como LOC2 ... LOCn). Otros subdirectorios de "data" corresponden a locaciones diferentes de la locación "default".

- data/default: Tiene un subdirectorio con el diccionario de datos y las relaciones de cada base de datos.

- ckp y jnl: Contienen un subdirectorio por cada locación (además del subdirectorio "default"), y en cada uno de ellos un subdirectorio por cada base de datos que utilice esa locación. En cada subdirectorio residen los archivos de "checkpoint" y de "journal" (ver sección 2.2.2.), respectivamente, para la base de datos correspondiente.

- dbdb: Contiene el diccionario de datos de las bases de datos, es decir, es la base de datos administrativa. Es una base de datos especial mantenida por INGRES. Se utiliza para almacenar información sobre bases de datos y usuarios autorizados de INGRES, y para especificar las bases de datos que pueden ser utilizadas por cada usuario. También mantiene la información de los directorios y dispositivos en los cuales residen las bases de datos particulares (cuando no se desea usar el "default").

2.2.2. Tipos de archivos de INGRES:

El sistema INGRES se compone de las imágenes ejecutables de INGRES, los archivos de mensajes y las librerías, además de las bases de datos. Una base de datos INGRES consiste de tres componentes: tablas, "checkpoints" y "journals".

En disco, una base de datos está repartida en un conjunto de directorios, y es constituida por una colección de archivos. Dichos directorios son creados como subdirectorios de locaciones INGRES predefinidas. Una locación es un directorio de UNIX (CENTIX) que el sistema INGRES identifica como un directorio en el cual pueden ser colocados componentes de las bases de datos de INGRES. Existe una utilidad especial ("accessdb") para definir locaciones.

i. Tablas:

Una tabla de una base de datos INGRES corresponde a un archivo UNIX. Las tablas de cada base de datos son de cuatro tipos:

a. Archivo de administración. Contiene el "user-id" del DBA ("Data Base Administrator" o administrador de la base de datos, es decir, el usuario que la creó) e información administrativa.

b. Relaciones del sistema. Tienen nombres predefinidos y existen para cada base de datos. Pertenecen al DBA. Junto con el archivo de administración constituyen el diccionario de datos de la base de datos. Pueden ser interrogadas por cualquier usuario autorizado por medio de QUEL. Sólo los puede modificar INGRES de manera automática, o excepcionalmente, el usuario "ingres", quien es el superusuario de todas las bases de datos (i.e., el DBA de la base de datos "dbdb").

c. Relaciones del DBA. Estas son relaciones de la base de datos creadas por el DBA, pero son compartidas en el sentido que cualquier usuario autorizado puede acceder a ellas. El DBA puede modificar los niveles de protección de estas relaciones para cada usuario y para cada relación.

d. Otras relaciones. Son relaciones de la base de datos creadas por usuarios diferentes del DBA. No son compartidas.

El DBA tiene poderes no delegables adicionales a los de los demàs usuarios: crear relaciones compartidas y especificar controles de acceso a ellas; destruir cualquier relaciòn de la base de datos (excepto las del sistema).

A excepciòn del archivo de administraciòn, cada archivo es tratado como una relaciòn. Mantener los archivos del sistema como relaciones trae las siguientes ventajas:

- Se economiza còdigo de acceso.
- Se pueden usar las mismas estructuras de informaciòn que se usan para las demàs relaciones.
- Se pueden examinar vìa QUEL.

Cada relaciòn està separada en un archivo fìsico diferente, i.e., no se intenta colocar en el mismo archivo tuplas de diferentes relaciones.

INGRES maneja sus propios mètodos de acceso para las relaciones, y administra el espacio libre dentro de cada archivo UNIX.

Tanto las tablas del diccionario de datos como las tablas definidas por los usuarios son colocadas en las locaciones como archivos individuales de UNIX. Las relaciones del sistema siempre son creadas en la locaciòn "home" asociada a cada base de datos. La locaciòn "home" por defecto para una base de datos es la locaciòn "default" (que corresponde al directorio "ingres/data/default" segùn se viò en la secciòn 2.2.1.), y se define cuando se crea la base de datos (mediante el comando "createdb").

Las tablas de los usuarios son tambièn creadas en la locaciòn "home" para la base de datos respectiva, a menos que se especifique alguna locaciòn diferente en el comando QUEL o SQL que crea la tabla. Ademàs pueden ser reùbicadas en una nueva locaciòn utilizando el comando "relocate".

ii. Checkpoints:

Un checkpoint es una imagen UNIX "tar" de una base de datos, con un significado especial (ver TAR(C) en la documentaciòn de UNIX). El DBA crea un "checkpoint" en un momento en el cual se sabe que la base de datos es consistente. Asì, el "checkpoint" se convierte en un "backup" seguro del cual se puede regenerar la base de datos.

Cada "checkpoint" consiste de mìnimo dos archivos "tar", que residen en la respectiva locaciòn "default" (que corresponde al directorio "ingres/ckp/default"). Para cada "checkpoint", INGRES crea un archivo "tar" que contiene informaciòn sobre las locaciones en las que la base de datos reside, màs un archivo "tar" asociado a cada locaciòn con imàgenes de las tablas presentes en ella.

iii. Journals:

Un journal es un registro de los cambios realizados a una base de datos desde el último "checkpoint". El "journal" se mantiene en un formato que permite al DBA rehacer o deshacer algunas de las operaciones si es necesario.

INGRES mantiene "journals" para todas las tablas sobre las que se ha habilitado el "journaling". Hay un archivo de "journal" por cada base de datos que tenga tablas con "journaling". Los archivos de "journal" son colocados en su locación "default" (que corresponde en este caso al directorio "ingres/jnl/default").

2.2.3. Sistema de recuperación de INGRES:

INGRES tiene un sistema de "journaling" que, valiéndose de los archivos de "checkpoint" y "journal" permite realizar las siguientes operaciones:

- Tomar copias "instantáneas" de la base de datos ("checkpoints") en momentos en que se sabe que está consistente.

- Rehacer la base de datos, en caso de desastre lógico o físico, desde el último "checkpoint" hasta cualquier momento antes de la falla.

- Deshacer transacciones sobre la base de datos hasta cualquier momento posterior al último "checkpoint".

- Producir pistas de auditoría, las cuales son relaciones tipo INGRES, sobre cualquier tabla que tenga "journal".

INGRES recomienda, por lo tanto, tener en dispositivos físicos diferentes los directorios de "checkpoint", "journaling" y de datos de cada base de datos.

2.2.4. "Buffer pool":

INGRES provee manejo de "buffers" privado para cada usuario, sin utilizar el "buffer pool" de UNIX. Esto se debe a que los mecanismos de reemplazo y prebúsqueda de bloques que UNIX tiene degradan en general el desempeño de INGRES, a pesar de que operan correctamente para el uso normal del sistema operacional [STON81]. Uno de los procesos que INGRES lanza para cada usuario ("backend") reserva un "buffer pool" privado y lo administra.

2.3. Estructura de procesos de INGRES.

Los diseñadores de INGRES quisieron que sus componentes fueran procesos típicos de UNIX; si así no fuera, el sistema operacional

tendría que ser alterado, perdiendo portabilidad. No existe un proceso común para todos los usuarios, sino que cada usuario tiene su propio conjunto de procesos.

2.3.1. Manejo de memoria central:

INGRES no tiene procesos comunes que actúen de "kernel" para todos los usuarios. Cuando un usuario invoca INGRES, se crean dos procesos por cada utilidad o aplicación de INGRES que éste invoque. Uno de ellos es el programa de aplicación o el paquete de INGRES, y el otro es el proceso "backend", el cual se describe en la sección 2.3.2.

2.3.2. Procesos lanzados por INGRES:

Al invocar INGRES, ya sea desde la interfaz QUEL, desde alguno de los paquetes de INGRES desde un programa en EQUEL (QUEL anidado en un lenguaje de programación convencional), se crea la siguiente estructura de procesos:

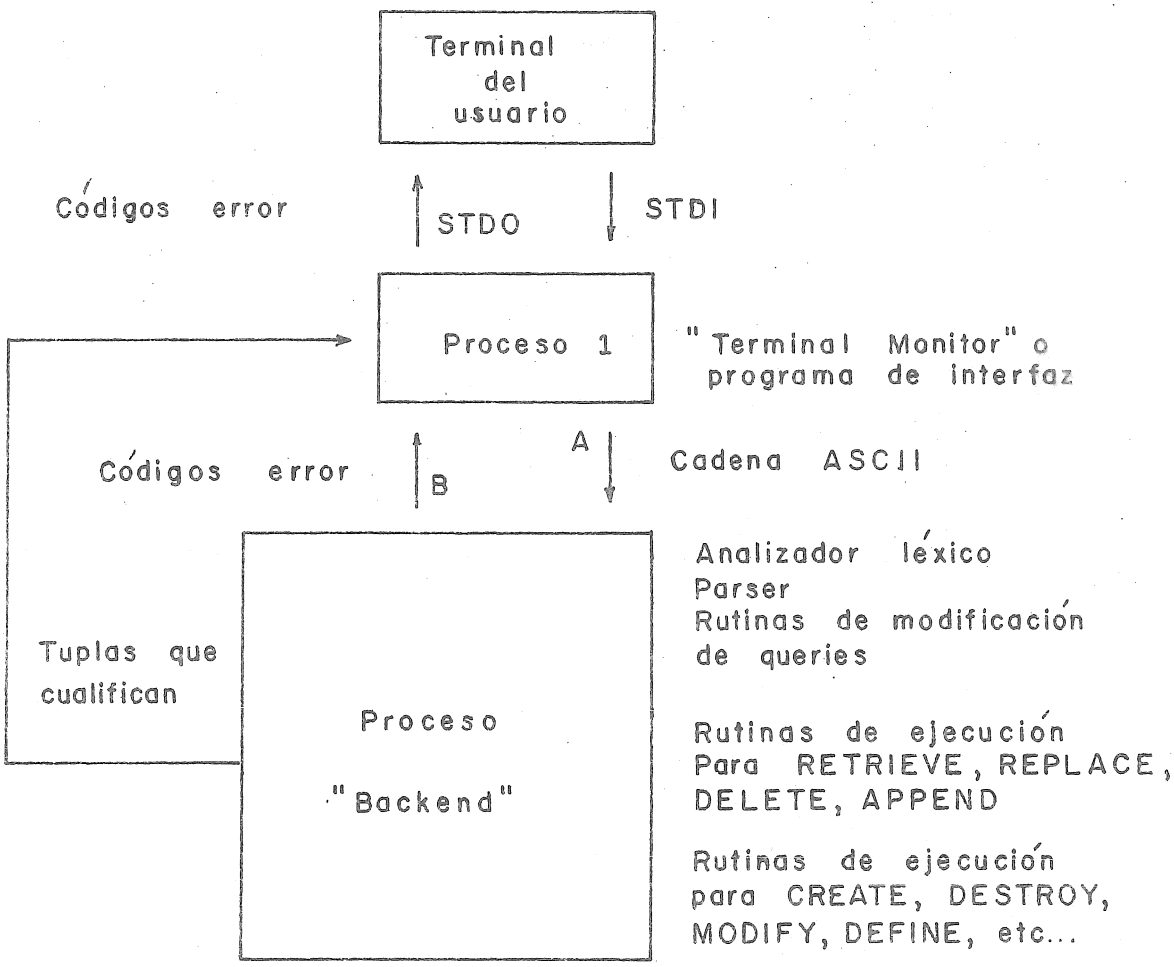


Figura 2.2

Proceso 1: Es el proceso correspondiente al programa que sirve de interfaz con la base de datos. Este puede ser:

a) El "terminal monitor", que permite al usuario editar, listar y ejecutar conjuntos de comandos de QUEL;
b) Alguno de los paquetes de interfaz proveídos por INGRES (INGRES/FORMS, INGRES/MENU, INGRES/QUERY, etc.);
c) Un programa de aplicación escrito en EQUEL (EQUEL/C, EQUEL/Pascal, EQUEL/COBOL, etc.).

Los comandos QUEL que genera el programa de interfaz son transmitidos al proceso "backend" por medio del "pipe" A, como cadenas de caracteres ASCII, en el momento de la ejecución.

Proceso "backend": Contiene un analizador lèxico, un "parser", rutinas de modificación de "queries" para control de integridad y soporte de vistas, rutinas de ejecución para los comandos RETRIEVE, REPLACE, DELETE, APPEND y para los demàs comandos de QUEL. Cualquier modificación a una relación se convierte en un comando RETRIEVE que aisla las tuplas a modificar. Copias de las tuplas revisadas se almacenan en un archivo especial (colocado temporalmente en el directorio "home" de la base de datos), que luego es procesado por un "procesador diferido de actualizaciones", también presente en este proceso.

Los còdigos de error que se generen en el proceso "backend" y las tuplas que resultan de "queries" sin relación resultado son devueltos al proceso 1 por medio del "pipe" B.

El proceso "backend" maneja además el "buffer pool" de cada usuario. INGRES coloca la tabla de "locking" para el control de concurrencia en el interior del sistema operacional.

La principal razón para esta estructura de procesos, comparada con la de versiones anteriores (que era de cuatro o màs procesos y un conjunto de "overlays"), es la del tamaño del espacio de direcciones para un proceso. En el caso de CENTIX, la memoria es virtual, alcanzando un màximo de 3,5 MB por proceso. En este espacio hay capacidad suficiente para unir todos los procesos que en versiones anteriores (INGRES/CS, por ejemplo) se repartían las tareas del proceso "backend".

2.3.3. Concurrencia y sicronización:

El control de concurrencia (en el caso de CENTIX) es llevado a cabo mediante un "device driver" especial instalado en el "kernel" del sistema operacional en el momento de la primera instalación de INGRES. Este "driver" provee niveles màs altos de "locking" que los de CENTIX, permitiendo a INGRES bloquear pàginas y tablas.

La sincronización y la comunicación entre el proceso 1 y el "backend" se hace utilizando las facilidades de UNIX, a través de "pipes".

3. ARQUITECTURA DE ORACLE SOBRE UNIX

3.1. Generalidades.

ORACLE fue implementado originalmente en un computador VAX corriendo el sistema operacional VMS. Al igual que INGRES se desarrollò en C. El SMBDR consta de dos mil módulos de los cuales ciento veinte tuvieron que ser modificados para la implementaciòn sobre UNIX. La primera versiòn apareciò en 1979 y con èsta la primera implementaciòn comercial de SQL. La primera versiòn sobre UNIX se instalò en un PDP-11 en 1981.

Esta secciòn describe los procesos y archivos usados por ORACLE bajo UNIX. Puesto que las implementaciones de ORACLE disponibles al escribir este artìculo, o bien no corrìan sobre UNIX o bien no estaban lo suficientemente documentadas, esta secciòn se basa primordialmente en la documentaciòn de Oracle Corporation para la versiòn 5 de ORACLE sobre un UNIX sistema V genèrico.

3.2. Estructura de archivos de ORACLE .

3.2.1. Estructura del directorio "oracle":

La figura 3.1 muestra la estructura de los directorios de ORACLE sobre UNIX. No existe una estructura rìgida de los directorios. Por medio de variables ambientales y el archivo de inicializaciòn de una base de datos, ORACLE detecta la localizaciòn de los distintos archivos.

Ya que ORACLE maneja varias bases de datos concurrentes la labor del administrador de la base de datos es importante. Se sugiere que cada tipo de archivo ("after image", "before image" y base de datos) resida en un disco fìsico diferente.

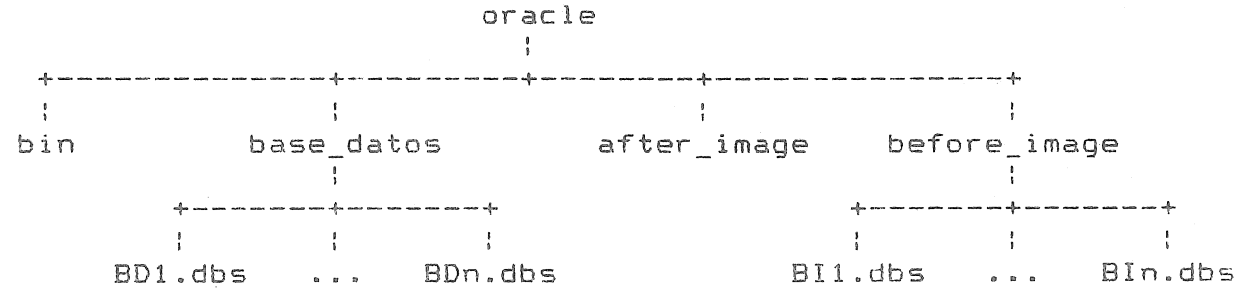


Figura 3.1

3.2.2. Tipos de archivos:

Una base de datos ORACLE consta de dos tipos de archivos: archivos de base de datos y archivos de recuperación.

i. Archivos de base de datos:

Los archivos de base de datos contienen el diccionario de datos y las tablas de los usuarios al igual que todos los índices. Desde el punto de vista de UNIX los archivos de base de datos son de tamaño estático. Se crean de un tamaño predeterminado por el administrador de la base de datos con el utilitario "Create Contiguous File" (CCF) de ORACLE. Sobre UNIX el tamaño máximo de un archivo de base de datos es una unidad física de disco. ORACLE se encarga de asignar espacio a las tablas dentro del archivo. Para direccionar e incrementar el tamaño de una base de datos, ORACLE utiliza el concepto de particiones lógicas. Una partición lógica consta de un máximo de 32 archivos de base de datos. Los archivos de una partición lógica pueden residir en cualquier "file system" (y por consiguiente en cualquier disco) y deben conformarse de bloques contiguos del disco.

Un archivo de base de datos no se puede ampliar dinámicamente. El mecanismo usado para adicionar espacio a la base de datos es crear un archivo nuevo (usando CCF) del tamaño deseado y añadírselo a una partición de la base de datos.

En la figura 3.2a se muestra la configuración de una base de datos con dos particiones y cuatro archivos distribuidos a traves de tres unidades de disco. La figura 3.2b muestra los comandos ORACLE necesarios para llegar a esta configuración. Primero se crean las particiones. Luego se crean los archivos iniciales de cada partición con el utilitario CCF dando como parámetro el número de bloques de UNIX que se quieren separar. Finalmente se hace la conexión entre cada partición y el archivo mediante el comando "alter partition". Por medio de vistas del diccionario de datos, el administrador de la base de datos puede detectar el momento en que se haga necesario crear un nuevo archivo. En tal caso se repite la anterior secuencia de comandos para los archivos /disco3/arch2pers y /disco2/arch2sist. El papel del administrador de la base de datos es muy importante ya que este estima el tamaño inicial y crecimiento de la base de datos.

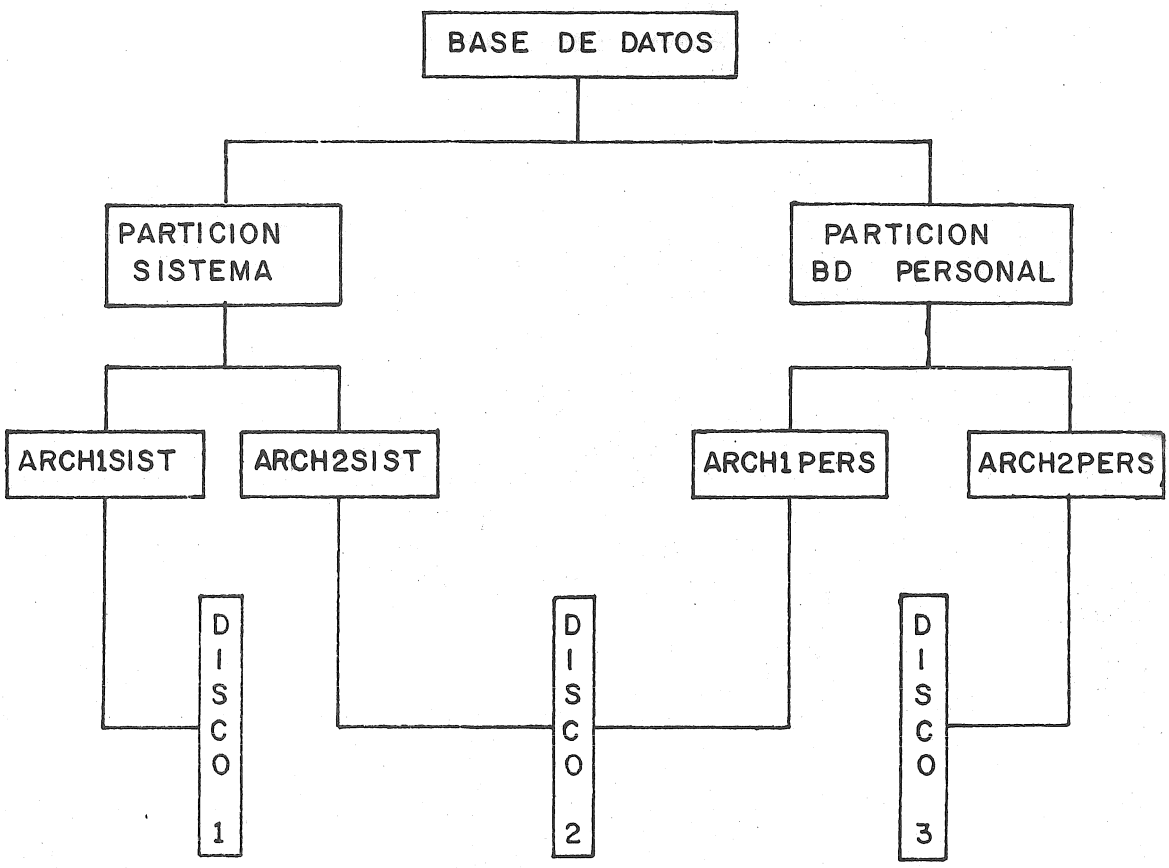


Figura 3.2a

```
CREATE PARTITION SISTEMA
CREATE PARTITION BD_PERSONAL
CCF /disco1/db/arch1sist  10000
CCF /disco2/db/arch1pers  20000
ALTER PARTITION SISTEMA ADD FILE '/disco1/db/arch1sist'
ALTER PARTITION DB_PERSONAL ADD FILE '/disco2/db/arch1pers'
CCF /disco2/db/arch2sist  15000
CCF /disco3/db/arch2pers  25000
ALTER PARTITION SISTEMA ADD FILE '/disco2/db/arch2sist'
ALTER PARTITION DB_PERSONAL ADD FILE '/disco3/db/arch2pers'
```

Figura 3.2b

ORACLE utiliza el sistema de archivos de UNIX únicamente para adquirir grandes archivos compuestos de bloques físicamente contiguos. Una vez hecha esta operación, ORACLE se encarga de asignar el espacio dentro del archivo a las tablas de la base de datos sin involucrar a UNIX en el proceso.

La figura 3.3 muestra la estructura lógica de una partición. Una partición se compone lógicamente de tablas. Una tabla se compone lógicamente de segmentos de datos y de índices. Un segmento se compone físicamente de "extents" donde un "extent" es la mínima cantidad de espacio asignada. Cada "extent" es un conjunto físico de bloques de disco. El tamaño del "extent", dado en bloques de ORACLE, lo asigna el administrador de la base de datos cuando se crea una tabla. Para la asignación de espacio a las tablas de una partición, ORACLE maneja dos listas de espacio libre. Una a nivel de partición y otra a nivel de segmento lógico.

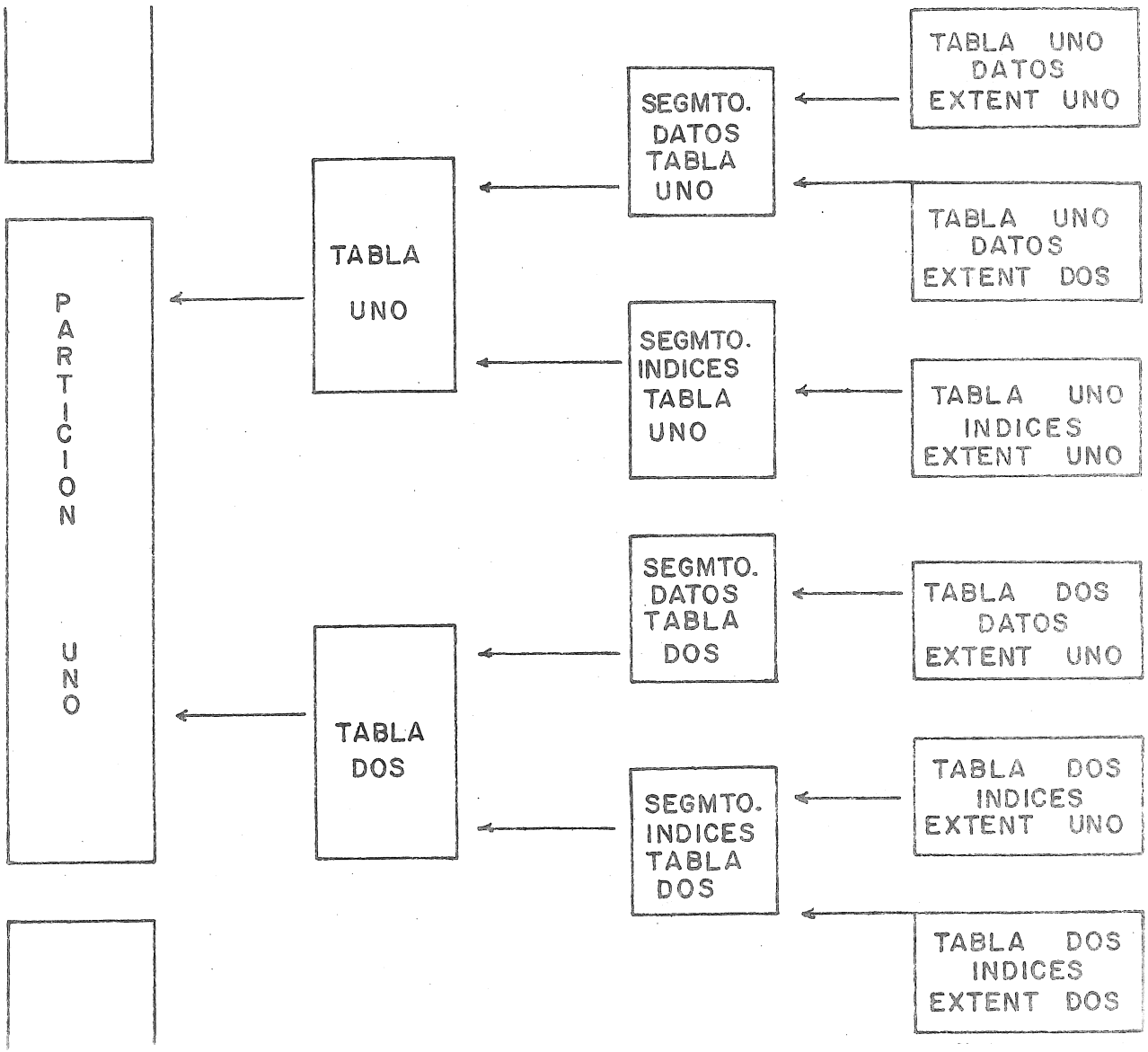


Figura 3.3

ii. Archivos de recuperación:

ORACLE maneja dos tipos de archivos de recuperación el "Before Image" (BI) y el "After Image" (AI). El BI se utiliza para recuperación de transacciones y el AI en caso de falla de una unidad física de disco.

En ORACLE una transacción (unidad lógica de trabajo) tiene tres estados: activa, finalizada o abortada. Cuando se finaliza una transacción las modificaciones que esta haya hecho a la base de datos se vuelven permanentes. Cuando una transacción aborta los cambios parciales que haya hecho se pueden deshacer por medio del comando "rollback". Mientras que una transacción no haya finalizado o abortado su estado se define como activo.

a. Archivo Before Image:

Al igual que los archivos de la base de datos, el archivo BI es un archivo de tamaño estático creado con CCF. Su función principal es permitir el "rollback". El "before image" tiene copias de los bloques de la base de datos antes de estos ser modificados. En el caso de que se ejecute un "rollback" los datos de las tablas afectadas en el "before image" se reescriben al archivo de la base de datos.

El "before image" también cumple la función de controlar lecturas concurrentes. Si un usuario está actualizando mientras otros están leyendo, los lectores leen la copia del "before image". El BI está implementado como un archivo circular de modo que la transacción más vieja es substituida por la transacción corriente cuando el archivo se llena. Se recomienda que el "before image" esté en un disco diferente a la base de datos para minimizar la contención de accesos.

b. Archivo After Image:

El archivo "after image" tiene como única función permitir la recuperación de una base de datos en caso de falla del disco en que se almacena dicha base de datos. Por consiguiente el "after image" debe residir en una unidad de disco diferente a la base de datos. En el "after image" se lleva un registro de todas las escrituras a los archivos de la base de datos y puede ser habilitado o deshabilitado a nivel de base de datos por el administrador. Para la recuperación el archivo de "after image" se aplica a una copia de respaldo de la base de datos bajo control del utilitario AIJ. El resultado es una copia exacta de la base de datos perdida. Este procedimiento se llama "roll-forward". Al habilitar "after image journaling" cada transacción genera una escritura adicional al disco lo cual degrada el desempeño del sistema aproximadamente en un 10% [ORAC86].

3.2.3 Manejo del "buffer pool" de UNIX

Una base de datos ORACLE puede almacenarse en disco bajo dos modalidades: en el "file system" de UNIX o en un "raw device". Un raw device es un disco o sección de un disco que ha sido configurado para no ser controlado por UNIX.

El uso de raw devices conlleva a que:

- La entrada/salida al disco no utiliza el "buffer pool" de UNIX. Las lecturas y escrituras se hacen directamente de los "buffers" de ORACLE (descritos en la sección 3.3) al disco.

- Se evita el "overhead" de UNIX el cual lee varios bloques del disco por cada pedido de lectura.

- Baja la carga del "file system" de UNIX.

- Los "buffer pools" de UNIX y de ORACLE se graduan independientemente.

- El desempeño del sistema mejora de 10 a 20% [ORAC86].

Cuando los archivos de la base de datos se almacenan en el "file system" de UNIX, ORACLE maneja el "buffer pool" de UNIX con una facilidad no documentada de UNIX sistema V [ORAC86a]: el "cache write-thru". Este permite que ORACLE pueda forzar la escritura de un bloque al disco por medio del "buffer pool" de UNIX y asi garantizar la integridad de la base de datos.

3.3. Estructura de procesos de ORACLE .

3.3.1 Manejo de memoria principal:

ORACLE utiliza el "System Global Area" (SGA) para implementar el manejo de memoria compartida. Se utilizan las facilidades de memoria compartida de UNIX sistema V para que varios procesos puedan accesar los mismos datos. Ya que UNIX 4.X de Berkeley no implementa memoria compartida, ORACLE no corre bajo estas versiones de UNIX. EL SGA contiene los "buffers" de lectura y escritura al igual que las tablas de "locks". UNIX tiene tres parámetros que interactúan con el manejo de memoria de ORACLE los cuales se asignan cuando se genera el sistema.

- SHMMAX - El máximo número en bytes de un segmento de memoria compartida.

- SHMSEG - El número máximo de segmentos de memoria compartida que puede accesar un proceso.

- SHMALL - El número máximo de páginas de memoria compartida que soporta el sistema.

Para optimizar el manejo de memoria compartida se recomienda que SHMMAX sea mínimo 131K Bytes y que SHMALL no sea más de 25% de la memoria real de la máquina para evitar "swapping" excesivo de los procesos. ORACLE impone un límite al tamaño del SGA de 10 Mbytes (10 segmentos de 1Mbyte cada uno).

3.3.2. Procesos lanzados por ORACLE:

ORACLE maneja cuatro procesos compartidos para lectura y escritura y dos adicionales por cada usuario bajo UNIX. La figura 3.4 muestra la estructura de los procesos y su relación con el SGA.

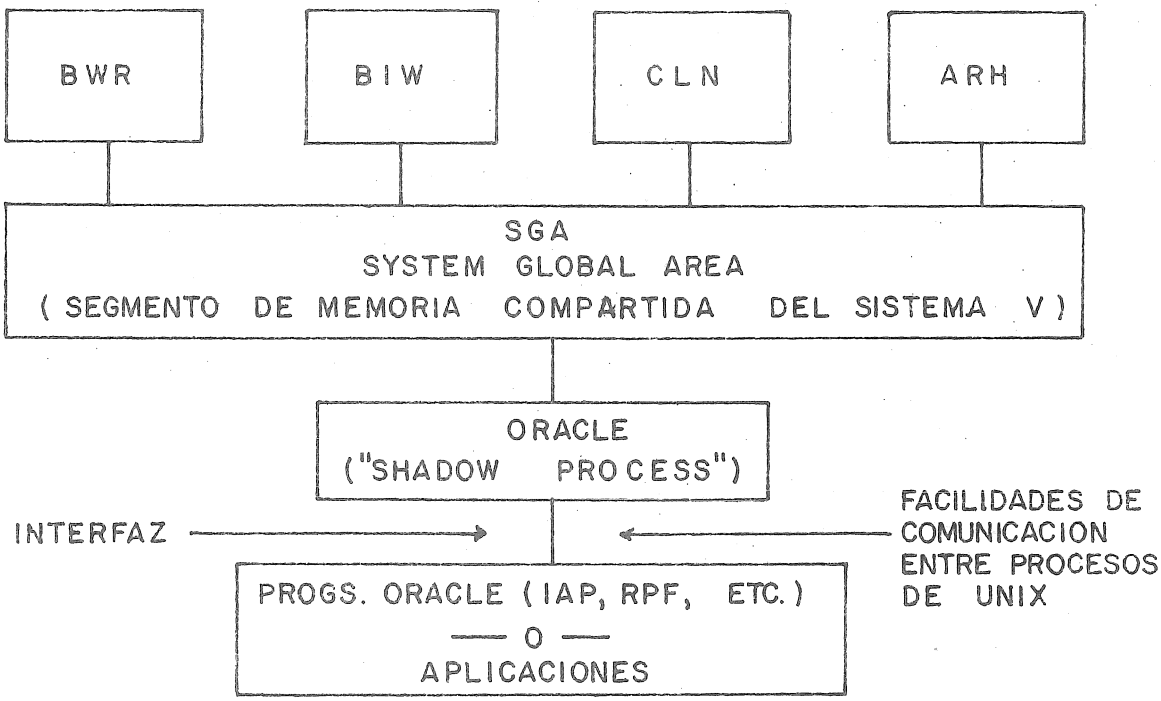


Figura 3.4

Por cada usuario que se conecta al "kernel" de ORACLE se crea un proceso principal y uno que corre en "background". Los utilitarios de ORACLE o aplicaciones hechas por el usuario conforman el proceso principal. El segundo proceso ("shadow process") es una interfaz que se encarga de filtrar los pedidos del proceso principal para evitar la corrupción del area de memoria compartida.

ORACLE usa código reentrante. Cada usuario utiliza el mismo segmento de código cuando hace llamados a las rutinas internas de ORACLE. El apuntador al "stack" de cada proceso mantiene la posición de éste en el código.

Los siguientes son los procesos que hacen interfaz con los archivos de la base de datos.

IOR - Initialize ORACLE

Lanza los cuatro procesos descritos a continuación e inicializa el SGA de acuerdo con los parámetros dados por el administrador de la base de datos. Utilizado también para crear nuevas bases de datos y bajar ("shutdown") una base de datos.

ARH - Asynchronous Read Ahead

Copia bloques del archivo de la base de datos al (SGA). Para mejorar el desempeño de la base de datos lee el bloque que pidió el programa al igual que los bloques vecinos. La lectura se hace en paralelo con la ejecución del programa cliente.

BIW - Before Image Writer

Copia bloques del SGA al archivo BI ("Before Image"). Es el único proceso que tiene acceso de escritura sobre el archivo BI. Los bloques copiados se usan para hacer recuperación en caso de transacciones abortadas ("rollback") y para asegurar la consistencia de los datos leidos por un proceso mientras que otro actualiza los mismos datos.

BWR - Buffer Writer

Escribe los bloques que han sido modificados en el SGA a la base de datos y al archivo AI ("After Image"). Es el único proceso que puede escribir a estos archivos. Los archivos se escriben cuando una transacción termina exitosamente.

CLN - Cleanup

Identifica periódicamente los procesos que han terminado anormalmente y hace la recolección de basura. Tomando la identidad de los procesos abortados, borra las tablas temporales, deshace las transacciones incompletas, libera los "buffers" y finalmente mata el proceso.

3.3.3. Concurrencia y sincronización:

Para manejo de concurrencia y sincronización ORACLE utiliza las facilidadaes de comunicación entre procesos de UNIX. Los procesos se comunican por medio de memoria compartida, señales y semáforos. Las señales se utilizan para informar a los procesos en què dirección de la memoria compartida se encuentran sus datos. Los semáforos se utilizan para controlar la concurrencia en los accesos a la memoria compartida. ORACLE versión 5 no utiliza "pipes". Aunque los "pipes" implementan las funciones de memoria compartida y señales, ORACLE considera que èstos son lentos y poco confiables [ORAC86].

4. CONCLUSIONES

INGRES y ORACLE tienen la misma arquitectura relacional (ver figura 4.1), en la cual los comandos de definición y manipulación de datos pertenecen al mismo lenguaje y pueden ser mezclados libremente [SCHM83].

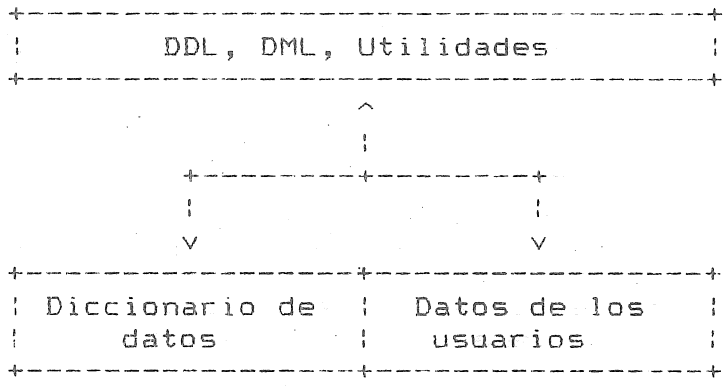


Figura 4.1

Sin embargo, sus arquitecturas físicas, tanto en archivos como en procesos son notablemente diferentes:

1. Archivos: La disparidad entre INGRES y ORACLE en la representación física de las bases de datos puede deberse a los diferentes conceptos de portabilidad establecidos por los creadores. El equipo que diseñó INGRES quería que su SMBDR fuera transportable a cualquier máquina UNIX, por lo cual podía apoyarse en su "file system". Los diseñadores de ORACLE, por su parte, buscaban portabilidad independiente del sistema operacional. De allí que hayan decidido implementar su propio sistema de administración de espacio en disco.

   Cada relación de INGRES corresponde a un archivo UNIX. El tamaño de este archivo varía de acuerdo al tamaño y la estructura interna de la tabla correspondiente, bajo el control de UNIX. ORACLE requiere que el DBA reserve explícitamente archivos UNIX de tamaño fijo en los cuales se almacenan conjuntos de tablas. El tamaño de los archivos UNIX permanece invariable. ORACLE se encarga de la administración del espacio dentro de estos archivos. UNIX no puede hacer uso del espacio libre de ORACLE. Por lo tanto es importante que el DBA de ORACLE planee cuidadosamente el uso del disco, para no reservar espacio que no utilizará la base de datos y que tampoco podrá utilizar UNIX.

2. Procesos: INGRES lanza procesos para cada usuario (incluyendo el "buffer pool"). No hay procesos que actúen como un núcleo común para todos los usuarios. ORACLE tiene cuatro procesos que se encargan de la entrada/salida del disco de todos los usuarios.

Tanto INGRES como ORACLE han escogido manejar su propio "buffer pool" para mejorar el desempeño del SMBDR, en lugar de utilizar el "buffer pool" de UNIX. Sin embargo, sobre el Sistema V de UNIX, ORACLE deja la opción de utilizar los "buffers" de UNIX con la facilidad "cache write-thru" para mantener la integridad de la base de datos, o evitarlos del todo por medio de "raw devices".

Uno de los principales objetivos de los diseñadores de INGRES y ORACLE fuè garantizar la portabilidad de sus SMBDRs. En las implementaciones estudiadas se distinguieron tres niveles de portabilidad:

1. Portabilidad de UNIX: Por estar implementados sobre UNIX, INGRES y ORACLE disfrutan del amplio rango de equipos y marcas que soportan este sistema operacional.

2. Portabilidad del SMBDR: Existen versiones de ambos SMBDRs sobre una gran variedad de sistemas operacionales. INGRES presenta mayor dependencia de UNIX que ORACLE, porque los diseñadores de INGRES consideraron suficiente desde un comienzo el nivel de portabilidad de UNIX y querían aprovechar a fondo su ambiente de desarrollo [STON80]. En particular, la arquitectura de archivos de INGRES se basa en el "file system" de UNIX.

3. Portabilidad del modelo relacional: De los tres principales modelos de bases de datos (red, jerárquico y relacional), el modelo relacional es el que màs alto nivel de independencia de datos provee, a la vez que permite una forma natural de visualizarlos [CHEN76]. Estas dos características garantizan en todo momento transportabilidad de diseño y de datos. De diseño a través de distintas implementaciones del modelo relacional, y de datos a través de arquitecturas diferentes de "hardware" y "software".

Por último, la tendencia de la industria del "software" es ceñirse a estándares. En cuanto a sistemas operacionales, UNIX, y en cuanto a modelos de bases de datos, el modelo relacional. INGRES y ORACLE, además de tener fuertes nexos con UNIX, han constituido la fuerza impulsora de la estandarización del modelo relacional.

REFERENCIAS Y BIBLIOGRAFIA

[BACH86] Bach, M. "The Design of the UNIX Operating System". Prentice-Hall, 1986.

[BURR86a] "XE 500 CENTIX System INGRES: Installation and Implementation Guide". Burroughs Corporation, 1986.

[BURR86b] "XE 500 CENTIX System INGRES: Application Development Guide". Burroughs Corporation, 1986.

[BURR86c] "XE550 INGRES Burroughs Multi-AP Release, version 3.0.7". Burroughs Corporation, 1986.

[CHEN76] Chen, P. P. "The Entity-Relationship Model - Toward a Unified View of Data". ACM TODS, Vol. 1, No. 1, 1976.

[CODD79] Codd, E.F. "Extending the Relational Data Base Model to Capture More Meaning". ACM TODS, Vol. 4, No. 4, 1979.

[CODD83] Codd, E.F. Introducción al libro "Relational Database Systems", editado por Joachim W. Schmidt y Michael L. Brodie. Springer-Verlag, 1983.

[ORAC86a] "ORACLE for UNIX General Information Manual Version 5". ORACLE Corporation, 1986.

[ORAC86b] "ORACLE Data Base Administrator's Guide Version 5 for AOS/VS". ORACLE Corporation, 1986.

[ORAC86c] "ORACLE Tricks of the Trade". ORACLE Corporation, 1986.

[ORCE85] "SQL, The quiet revolution". ORCE Systems Software, 1985.

[SCHM83] Schmidt, J.W. et al. "Relational Database Systems - Analysis and Comparison". Springer-Verlag, 1983.

[STON76] Stonebraker, M. et al. "The Design and Implementation of INGRES". ACM TODS, Vol. 1, No. 3, 1976.

[STON80] Stonebraker, M. "Retrospection on a Database System". ACM TODS, Vol. 5, No. 2, 1980.

[STON81] Stonebraker, M. "Operating System Support for Database Management". Communications of the ACM, Vol. 24, No. 7, 1981.

SISTEMA ESPECIALISTA PARA VALIDAÇÃO DE BASES ARUANDA

Angela Maria Bosísio
Sedes/Arinf/Disul
292-4100 ramal 592

Marcia Pacitti Teles
Sfgts/Digis/Deat-2
292-4100 ramal 208

SERVIÇO FEDERAL DE PROCESSAMENTO DE DADOS - SERPRO

Rua da Lapa, nº 40 - CEP: 20.000

Rio de Janeiro - BRASIL

RESUMO

O Sistema Especialista destina-se a verificar se uma determinada base definida para o Sistema Aruanda, está obedecendo às normas de adequação da arquitetura Aruanda. Além disto, são geradas recomendações para que o projetista possa melhorar e/ou corrigir a definição da base fornecida para o sistema.

ABSTRACT

The purpouse of the expert system is to verify if a given base defined to the Aruanda System is suitable to the norma and adequate to the Aruanda arquitecture. Besides that, the expert system generates recommendations to the projetist so that he can optimize and/or correct his original definition of the base furnished to the system.

1. ORIGEM HISTÓRICA DO PROJETO

O Aruanda é um sistema de gerenciamento de Banco de Dados capaz de armazenar, administrar e distribuir um conjunto teoricamente não limitado de bases de dados com formatos e conteúdos diversificados, consultados e acessados segundo algum critério de seleção.

A falta de vivência da cultura Aruanda acarreta dificuldades no projeto de incorporação de uma nova base. Verificamos que seria necessário investir na disseminação desta cultura. A nossa experiência revela que o tempo de maturação de um projetista de bases não sofisticadas é bastante longo.

O que nos pareceu uma solução interessante foi criar um Sistema Especialista para ajudar ao projetista validar sua base. O sistema contem na sua base de conhecimento as regras de validação necessárias para a incorporação de uma nova base ao Aruanda. Na base de fatos do sistema, a estrutura das bases são armazenadas de modo que possam ser recuperadas e consultadas pelo projetista.

O desenvolvimento deste projeto teve início em Março de 1987. No momento ele se encontra em fase final de testes.

2. DEFINIÇÃO ARUANDA

2.1 Características Gerais

O Aruanda é um sistema de acesso a bases de dados, que tem como objetivo principal ser um sistema de disseminação de informação de uso geral e público.

Foi desenvolvido no Serpro em 1978, utilizando a linguagem PL/1, e os sistemas Adabas e Intercomm.

Para se alcançar o objetivo principal do Aruanda sua filosofia de recuperação incorpora os seguintes quesitos básicos:

a) Generalidade

O Aruanda não se destina a lidar apenas com um determinado tipo de informação. É capaz de tratar arquivos de dados com os mais diversos conteúdos lógicos e formatos físicos, ou seja, é capaz de acessar diferentes bases de dados com uma única linguagem.

b) Conversacionalidade

A comunicação usuário / sistema remete o usuário de uma situação onde ele dispõe apenas de seu vocabulário habitual para fazer a consulta, até uma outra situação onde ele finalmente recupera um registro ou um campo contendo a informação desejada.

A filosofia de recuperação supõe para isto, tanto a utilização de um vocabulário "livre" (o próprio conteúdo da informação) como de um vocabulário controlado cujos termos indexam a informação e funcionam como chaves de recuperação.

A fim de atender a estes quesitos básicos, foi definida uma determinada estrutura lógica para o sistema, e também criada uma linguagem de acesso a ser usada para recuperação em todas as bases de dados componentes do Aruanda, linguagem esta chamada LA (Linguagem Aruanda).

2.2 Arquitetura Aruanda

Todo projeto e funcionamento do Aruanda se baseia, portanto, numa estrutura lógica de dados, constituida dos seguintes elementos:

BASE DE DADOS
ARQUIVO DE DADOS
UNIDADE DE INFORMAÇÃO
ÍNDICE DE SELEÇÃO
DICIONÁRIO
DESCRITOR
VARIÁVEIS RECUPERÁVEIS
FORMATO DE IMPRESSÃO

Onde,

a) UNIDADE DE INFORMAÇÃO é o conjunto de dados que constitui a informação básica a ser recuperada.

b) ARQUIVO DE DADOS é o conjunto onde cada elemento é uma unidade de informação.

c) ÍNDICE DE SELEÇÃO - os índices de seleção são chaves de pesquisa que são utilizadas na recuperação do Aruanda. Na recuperação da informação existem dois metodos de recuperação. O primeiro utiliza um critério de seleção acessando o arquivo de dados diretamente. O segundo utiliza um critério de seleção indireto, acessando um outro conjunto de informação (Dicionário), que de alguma forma estará indexando as informações do arquivo de dados. Por este motivo, os índices de seleção são classificados em dois tipos segundo o metodo de recuperação no qual ele será utilizado.

d) DICIONÁRIO é um conjunto determinado de todos os descritores que compõem um vocabulário controlado. Quando o dicionário é utilizado o meio de acesso as informações é indireto.

e) DESCRITOR é cada elemento do dicionário. Cada descritor é identificado padronizadamente por seu código e por sua descrição. Cada descritor está associado a uma ou mais unidades de informação do arquivo de dados.

f) VARIÁVEIS RECUPERÁVEIS é o conjunto de itens da unidade de informação escolhidos para serem utilizados no metodo de seleção direto.

g) FORMATO DE IMPRESSÃO é um subconjunto de dados da Unidade de Informação escolhidos para exibição ao usuário.

h) BASE DE DADOS ARUANDA é constituida pelos elementos definidos acima, ou seja, a tripla (Arquivo de Dados, Conjuntos de índices de Seleção, Formatos de Impressão).

2.3 Validações Necessárias

2.3.1 Definição da Base Aruanda para fins de Validação

A fim de validar a arquitetura de uma base, o sistema especialista deve analisar as entidades contidas na sextupla (Arquivo de Dados , Conjunto índices Dicionário , Conjunto índices Variáveis Recuperáveis , Formatos de Impressão , Dicionário , Associação dos índices), que foi definida em função da arquitetura vista no item 2.2.

Os índices foram divididos da seguinte forma :

- Entidade Conjunto índices Dicionário, constituida de chaves de recuperação, que estão indexando indiretamente itens da Unidade de Informação, onde para cada índice existe um Dicionário associado.

- Entidade Conjunto índices Variáveis Recuperáveis, constituida de chaves de recuperação, que estão associadas diretamente

a itens da Unidade de Informação, escolhidos como índices.

Esta divisão foi efetuada pois o Aruanda trata de modo diferente estes dois tipos de índices, o que acarreta validações distintas para eles.

Além disto, ao Conjunto índices Dicionário está associada a entidade Dicionário.

Criamos uma outra entidade que denominamos de ASSOCIAÇÃO DOS ÍNDICES, que indica a associação entre um elemento do Conjunto de índices Variáveis Recuperáveis e a correspondente variavel na Unidade de Informação, ou a associação que representa a indexação entre um elemento do Conjunto de índices Dicionário e um item da Unidade de Informação.

A validação a qual nos propusemos , foi verificar se as entidades de uma dada sextupla satisfazem aos requisitos necessários ao funcionamento do Sistema Aruanda.

2.3.2 Regras de Validação

O conjunto de regras a seguir, usadas na validação de uma Base de Dados Aruanda, representa a base de conhecimento do Sistema. O número de regras não depende de cada Base, pois são regras para verificar se uma Base de Dados qualquer satisfaz à arquitetura Aruanda.

Definiremos inicialmente a linguagem L, que representa as regras de validação Aruanda, como uma linguagem de primeira ordem, com notação

< PL , FL , CL > onde V,PL,FL,CL
são conjuntos assim constituídos:

V = conjunto de variáveis

F = (campo,listacampo,listaíndices,listavarecus,
L listaformatos,estruturaformato,associação,
listaassociação,listanomes...)
conjunto de funções de L

C = N U (cadeia de caracteres)
L U (alfa,num,descritor,periódico)
conjunto de constantes de L

P = (∈,inteiro,=,<=,>=,enhargumento) U
L (formato,atributo1,atributo2,atributo3,
base_aruanda,unidade_de_informação,
arquivo_de_dados,índices_em_lu,...)
conjunto de predicados de L

Apresentaremos a seguir os termos de L, que representam os elementos da sextupla Aruanda.

1) Arquivo de Dados, notação LU

LU, termo de L é assim definido:

listacampos(campo(n ,t ,f ,at1 ,at2 ,at3),
1 1 1 1 1 1
campo(n ,t ,f ,at1 ,at2 ,at3), ...,
2 2 2 2 2 2
campo(n ,t ,f ,at1 ,at2 ,at3))
m m m m m m

onde n ∈ V, t ∈ V, ... etc
1<=i<=m 1<=i<=n

Observe que o termo campo(n ,t ,f ,at1 ,at2 ,at3) está
i i i 1 1 1
associado ao elemento unidade de informação do Aruanda.

2) Conjunto índices Dicionário, notação LID
LID, termo de L, é assim definido:
listaíndices(n ,n ,...,n), onde n ∈ V
1 2 m 1<=i<=m
3) Conjunto índices Variáveis Recuperáveis, notação LVR
LVR, termo de L,é assim definido:
listavarecus(n ,.....,n), onde n ∈ V
1 m 1<=i<=m
4) Formatos de Impressão, notação LFR
LFR, termo de L, é assim definido:
listaformatos(estruturaformato(m ,listanomes(n ,...,n)),
1 11 1p
estruturaformato(m ,listanomes(n ,...,n)),
2 21 2p
.
.
estruturaformato(m ,listanomes(n ,...,n)))
l l1 lp
onde m ∈ V, n ∈ V, com 1 <=j<=l, 1<=k<=p
1<=i<=l jk
5) Dicionário, notação DIC
DIC, termo de L, é assim definido:
regdic(campocodigo (n ,t ,f ,at1 ,at2 , at3),
1 1 1 1 1 1
campodescrição1(n ,t ,f ,at1 ,at2 , at3),
2 2 2 2 2 2
campodescrição2(n ,t ,f ,at1 ,at2 , at3),
3 3 3 3 3 3
campovps (n ,t ,f ,at1 ,at2 , at3),
4 4 4 4 4 4
campotg (n ,t ,f ,at1 ,at2 , at3),
5 5 5 5 5 5
campotc (n ,t ,f ,at1 ,at2 , at3),
6 6 6 6 6 6
campotr (n ,t ,f ,at1 ,at2 , at3),
7 7 7 7 7 7
camposn1 (n ,t ,f ,at1 ,at2 , at3),
. 8 8 8 8 8 8
.
.
camposn6 (n ,t ,f ,at1 ,at2 , at3))
m m m m m m
6) Dicionários, notação LDIC
LDIC, termo de L, é assim definido:
listaregdic(DIC ,DIC ,... ,DIC)
1 2 n
onde DIC, é termo de L definido no item 5.
1<=i<=n
7) Associação índice, notação AID
AID, termo de L, é assim definido:
associação(m,campo(n,t,f,at1,at2,at3))

8) Associação dos índices, notação LAID

LAID, termo de L, é assim definido:

listaassociação(AID ,AID ,...,AID)
1 2 n

onde AID é termo de L definido no item 7.
1<=i<=n

Uma base de dados Aruanda, será definida pelo seguinte predicado:

base_aruanda(LU,LID,LVR,LFR,LDIC,LAID), onde LU,LID,LVR, LFR,LDIC,LAID são termos de L.

Diz-se, que uma base de dados Aruanda está "correta", se dado LU,LID,LVR,LFR,LDIC,LAID, conseguirmos provar através das regras de validação, que o predicado base_aruanda(LU,LID,LVR,LFR, LDIC,LAID) é verdadeiro.

Mostraremos, a seguir, um pequeno subconjunto de critérios utilizados na validação de uma base Aruanda.

Regras de Validação

I) Dado LU,LID,LVR,LFR,LDIC,LAID

arquivo_de_dados(LU)
⌢
índices_em_lu(LID,LVR,LDIC,LAID)
⌢
conj_formato_impressão(LFR)

==> base_aruanda(LU,LID,LVR,LFR,LDIC,LAID)

II) Dado LU=listacampo(campo(n ,t ,f ,at1 ,at2 ,at3),...,
1 1 1 1 1 1
campo(n ,t ,f ,at1 ,at2 ,at3))
l l l l l l

$\forall$ n ,t ,f ,at1 ,at2 ,at3 , 1<=i<=l
l l l l l l

[unidade_informação(campo(n ,t ,f ,at1 ,at2 ,at3))
i i i i i i

==> arquivo_de_dados(LU)]

III) Dado campo (n,t,f,at1,at2,at3)

$\exists$ (n,{cadeia de caracteres})
⌢ inteiro(t)
⌢ formato(t)
⌢ atributo1(at1)
⌢ atributo2(at2)
⌢ atributo3(at3)
⌢

[=(at2,multiplo) ==> inteiro(at3)]

==> unidade_de_informação(campo(n,t,f,at1,at2,at3))

IV) $\forall$ f [=(f,alfa) $\veebar$ =(f,num) ==> formato(t)]
$\forall$ at1 [=(at1,descritor) $\veebar$ =(at1,$\emptyset$) ==> atributo1(at1)]
$\forall$ at2 [=(at2,multiplo) $\veebar$ =(at2,$\emptyset$) ==> atributo2(at2)]
$\forall$ at3 [(inteiro(at3) $\veebar$ =(at3,$\emptyset$) ==> atributo3(at3)]

V) Dado LID,LVR,LDIC,LAID

conjunto_indice_dicionário(LID,LDIC,LAID)
⌣ conjunto_indice_varecu(LVR,LAID)

==> indices_em_lu(LID,LVR,LDIC,LAID)

VI) Dado LU,
LID=listaindices(n ,...,n),
1 m
LDIC=listaregdic(DIC ,...,DIC)
1 x
LAID=listaassociação(AID ,...,AID)
1 k
∀ n [∃ DIC ,
1<=i<=m 1<=p<=x
∃ AID=associação(n ,campo(m ,t ,f ,at1 ,at2 ,at3))
1<=q<=k i j j j j j j
[dicionário_para_índice_em_lu(DIC ,n)
⌢ p i
associação_para_indice_dic_em_lu(AID)]]

==> conjunto_indice_dicionário(LID,LDIC,LAID)

Obs: Cada ∃ "termo de L", está sendo usado para simplificar. Na verdade, teríamos que usar vários ∃s para cada variavel, constituinte do termo quantificado.

VII) Dado LU,
LVR=listavarecus(v ,v ,...,v),
1 2 k
LAID=listaassociação(AID ,...AID)
1 m
∀ v [∃ AID = associação(v ,campo(n ,t ,at1 ,at2 ,at3))
1<=i<=k 1<=j<=m i k k k k k
[associação_para_indice_varecu_em_lu(AID)]]

==> conjunto_indice_varecu(LVR,LAID)

3. IMPLEMENTAÇÃO DO SISTEMA

3.1 Enfoque Adotado na Implementação das Entidades Principais

A linguagem adotada para desenvolver o sistema foi o Prolog. O projetista deve fornecer ao sistema as informações relevantes a cada entidade. O sistema irá requerer do projetista a configuração de cada entidade, pedindo os campos de entrada de cada entidade separadamente. Com isso, o projetista passa a compreender melhor as entidades básicas do Aruanda.

A primeira entidade a que chamamos de Unidade de Informação contem os campos básicos que podem ser recuperados no Sistema Aruanda. Para cada campo o projetista deve fornecer seu nome, tamanho, formato e atributos. Os atributos servem para designar

se um campo é descritor, isto é, se ele pode ser utilizado como chave de pesquisa, ou ainda se é um campo múltiplo, ou seja, ocorre várias vezes dentro de uma unidade de informação. Uma lista de estruturas chamada LU, é construida a medida que os campos são fornecidos. Cada campo possui a seguinte definição :

campo (X,T,F,At1,At2,At3)

onde:

X - nome do campo
T - tamanho
F - formato
At1 - informa se é campo chave (será referenciado como Atributo 1)
At2 - informa se é um campo múltiplo (será referenciado como Atributo 2)
At3 - informa o numero de ocorrências do campo múltiplo (será referenciado como Atributo 3)

A segunda entidade representa os índices que são utilizados para recuperação de Informação no Sistema Aruanda. Estes índices são agrupados em dois conjuntos distintos, como visto anteriormente.

Os elementos do primeiro grupo são os índices chamados de variáveis recuperáveis.

Uma lista chamada LVR é construida, contendo os nomes das variáveis recuperáveis, a medida que estes índices são fornecidos.

Os elementos do segundo grupo são chamados de índices dicionários. Estes índices são representados no sistema como uma lista de estruturas, denominada LID, e construida a medida que o projetista fornece estes índices.Cada elemento desta lista foi assim representado :

estrudic (X,Ndic)

onde:

X - nome do índice dicionário
Ndic - nome do arquivo dicionário associado

Para o Aruanda, como já foi visto, a um índice dicionário está associado a entidade Dicionário. Esta terceira entidade é associada explicitamente quando definimos o índice dicionário.

A entidade dicionário esta definida no sistema da seguinte forma:

Dic (M,Ndic,descritor_aruanda(L))

onde:

M - nome da base
Ndic - nome do dicionário
descritor_aruanda - estrutura que compreenderá uma lista de itens, necessários para que este índice seja tratado no Aruanda.

A estrutura descritor_aruanda poderá compreender alguns ou todos dos sequintes itens:

.campo_cod - campo que identifica um descritor Aruanda
.campo_desc1 - campo que será necessário para descrição do conteúdo da ocorrência de um índice (do descritor Aruanda)
.campo_desc2 - continuação da descrição acima
.campo_vocab - necessário para a descrição dos vocábulos pertinentes de um descritor Aruanda

.campo_tg - mostra se existe alguma relação hierárquica entre os descritores Aruanda em um índice

.campo_tc - mostra se existe alguma relação de associação entre os descritores Aruanda em um índice

.campo_tr - mostra se existe alguma relação de equivalência entre os descritores Aruanda em um índice

Cada uma das estruturas acima contem os seguintes elementos: X-nome, T-tamanho, F-formato, At1-atributo1, At2-atributo2, At3-atributo3 igual aos argumentos que definem um campo da Unidade de Informação.

A quarta entidade representa os formatos de exibição definidos para uma Base Aruanda. Esta entidade é representada através de uma lista de estruturas, denominada "estrufor" , onde cada elemento é definido por :

estrufor (Nformato , Lista_nomes)

onde:

Nformato - identifica o nome do formato

Lista_nomes - uma lista que contem os nomes dos campos pertencentes ao formato

Existem dois tipos de formatos. O formato estático é definido quando todos os campos de Lista_nomes são nomes de campos da dita LU. Caso algum campo não pertença à LU, será denominado formato dinâmico.

3.2 Validação

A validação é realizada em cada entidade separadamente. Este enfoque é importante, pois quando existir uma falha detetada, e não for possível continuar validando aquela entidade, passamos a validar a próxima entidade. Veja bem, isto só é realizado se for possível continuar a validação das outras entidades, ou dela própria.

Verificamos que em certos pontos da validação, com o nosso conhecimento do Aruanda, seria oportuno gerar recomendações que possibilitariam ao projetista melhorar o desempenho de recuperação das informações na Base. Em termos da estrutura Aruanda, o que foi fornecido poderá estar correto, porém uma recomendação será emitida para que o usuário possa avaliar se uma outra estrutura não seria mais eficiente. Outras recomendações poderão implicar num alerta para problemas que poderão surgir na recuperação da informação, não implicando que a estrutura seja modificada.

3.3 Inserção de Fatos

A partir do momento que uma determinada base é validada pelo sistema e nenhum diagnóstico de erro é enviado , são inseridos fatos na base de fatos do Prolog, de modo que apenas estes fatos permaneçam após a validação. Os fatos armazenados são aqueles que representam as entidades já definidas anteriormente, vinculadas à base validada.

4. UTILIZAÇÃO DO SISTEMA

A entrada dos dados necessários para que a validação seja iniciada é executada antes do início da validação. A única entidade que é requisitada durante a validação é a que representa a entidade dicionário, caso o projetista necessite que seu índice sejá dicionarizado.

Os campos são requisitados um de cada vez e as informações são indagadas explícitamente ao projetista de forma que este não omita nenhum dado relevante.

As mensagens decorrentes de falhas na validação ou de recomendação são exibidas na tela no momento em que estas são detetadas, porem o processo de validação continua até o final.

Como já foi visto anteriormente, a partir do momento que uma base é validada sem nenhum diagnóstico de falha, são inseridos fatos na base de fatos que representam as entidades associadas à base Aruanda em questão.

Estes fatos são depositados num arquivo para que, a qualquer momento, um projetista do Aruanda possa consultar uma imagem de seu próprio banco, que compreende todas as entidades de todas as bases já implantadas.

Isso proporciona uma ferramenta de apoio muito interessante para os projetistas. Por exemplo, caso uma nova base necessite de um índice dicionário que já tenha sido utilizado anteriormente para outras bases, ela pode ser consultada da seguinte forma:

```
?-Arq_dic ( Base , nome_indice , Dic ).
```

Com isso, todas os Arq_dics para onde este índice já tenha sido definido serão instanciados, e o projetista conhecerá todas as descrições dos índices já existentes.

5. CONCLUSÃO

Esta foi a nossa primeira experiência no desenvolvimento de um sistema especialista, e achamos que foi muito gratificante. O sistema irá ajudar a preservar e disseminar a cultura Aruanda, que foi o nosso principal objetivo, facilitando ao projetista a utilização do Sistema Aruanda.

SOFTWARE Y HARDWARE

DE BASE

El Computador Personal como Analizador de Estados Logicos de bajo costo

Gerónimo Bellassai

Laboratorio de Electrónica Digital
Universidad Catolica Nuestra Señora de la Asunción
Asunción - Paraguay

Se describe en este artículo un ANALIZADOR DE ESTADOS LOGICOS (A.E.L.) muy simple y económico que conectado a una Computadora Personal (P.C.) permite el analisis de circuitos digitales.

En efecto, este A.E.L. es en realidad una interfase con una pequeña memoria auxiliar donde se guardan en forma muy veloz los estados lógicos de diferentes puntos del circuito digital en análisis; para pasar posteriormente dichos datos a la memoria de la P.C. para su estudio detallado y conveniente.

El analizador, en sus dos versiones, fue diseñado en el Laboratorio de Electrónica Digital (L.E.D.) de la Universidad Católica Nuestra Señora de la Asunción, para uso de los estudiantes de Ingeniería Electrónica e Informática. Es ideal para fines didácticos por su bajo costo, suponiendo que ya se posee una P.C., sencillez en el uso y simplicidad del circuito que permite la fácil comprensión de su funcionamiento

1- ANALIZADOR DE ESTADOS LOGICOS

1.1-PRINCIPIO DE FUNCIONAMIENTO

En sistemas analógicos, cada punto contiene señales con la información "completa", o sea que cada señal, al ser observada en un osciloscopio nos proporciona en sí misma "una información" acerca del funcionamiento del sistema en estudio; por lo que generalmente es suficiente poder visualizar simultáneamente hasta dos señales.

En sistemas digitales en cambio, la información viaja en varias lineas (ocho, dieciseis o mas) llamadas "bus", en un código (generalmente binario) siendo imprescindible observar simultáneamente todas las líneas para obtener "la información". Por otra parte raras veces las señales son sincrónicas lo que elimina por completo al osciloscopio común para el análisis.

Aparece entonces el analizador de estados lógicos, como el instrumento propio para el análisis de los sistemas digitales, así como el osciloscopio lo es para sistemas analógicos.

Un analizador de estados lógicos consta basicamente de una memoria en la que se almacenan los estados lógicos de 8, 16, 24 o 32 puntos del sistema analizado simultaneamente, en sincronismo con los pulsos de un reloj (clock) durante el funcionamiento de dicho sistema. Los datos así almacenados pueden luego ser visualizados en tablas de distintos formatos, como binario, octal, y hexadecimal o en forma gráfica en diagramas de tiempos.

La presentación binaria o gráfica es útil cuando se desea verificar una secuencia de estados en cada punto individualmente; mientras la presentación hexadecimal u octal cuando se trabaja con el bus de datos o direcciones de microprocesadores.

Todo A.E.L., generalmente incluye en los circuitos que acoplan los datos de entrada a la memoria, un circuito de palabra de disparo (P.D.). Esto sirve para dar la posibilidad de que la secuencia de datos se graben en memoria solamente a partir del momento en que aparezcan los datos en una combinación dada, seleccionable, llamada palabra de disparo.

1.2-ESQUEMA GENERAL

Las partes fundamentales de un A.E.L. son, como se ve en el diagrama en bloques de la Fig. 1; la **memoria** donde se almacenan los datos que llegan a travez de las puertas de **entrada de datos**, el **circuito comparador palabra de disparo** que compara los datos de entrada con las **llaves de selección de palabra de disparo** para habilitar o no, la lectura de la memoria, y finalmente el **sistema procesador** que tiene dos funciones; controlar el funcionamiento del sistema por medio de los **circuitos de control de Muestreo y Lectura** y procesar los datos una vez terminado el muestreo, para su visualización y analisis.

Podríamos decir que mientras la **memoria** constituye el corazón del A.E.L., el **sistema procesador** es el cerebro ya que debe ser un sistema inteligente, capaz de ser programado para procesar los datos tomados y mostrarlos de modo que puedan ser comprendidos y analizados.

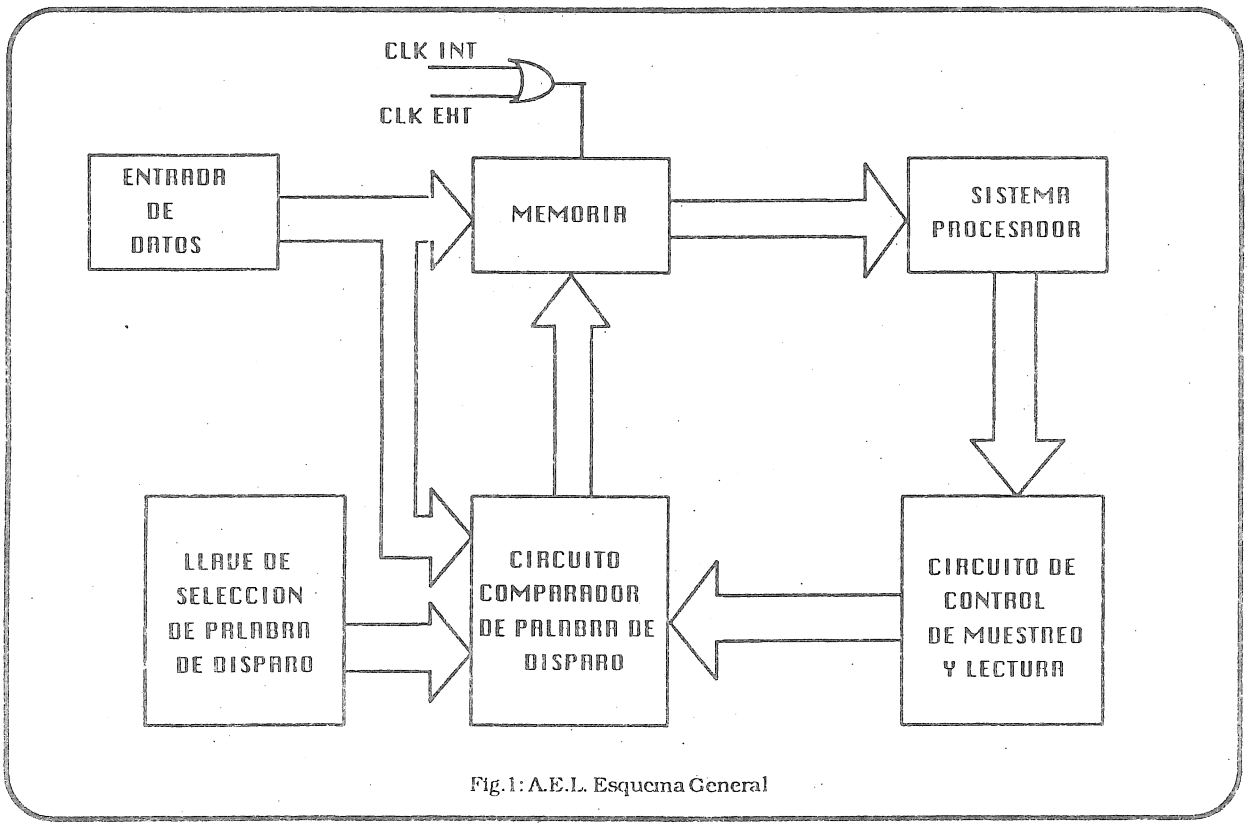


Fig.1: A.E.L. Esquema General

1.3-COSTOS

El costo de un A.E.L. comprado estaría alrededor de los 5.000 (cinco mil) dólares americanos; lo que vuelve imposible poseer varios para fines didácticos, por la suma que representa, no siempre accesible, y por el riesgo que significa poner en manos de aprendices un aparato tan costoso.

Por lo ya expuesto, es facil deducir que en todo A.E.L. el sistema procesador representa la parte más costosa. El objeto de este trabajo es eliminar ese costo, aprovechando el sistema procesador de una **Personal Computer** (que se supone ya se tiene), y simplificar algunas cosas del resto del analizador.

La interfase para la P.C. aquí desarrollada representa un costo de aproximadamente solo 100 (cien) dólares americanos. Además es sumamente simple, tanto en el **Hardware** como en el **Software**, de modo que el estudiante puede conocer y comprender en profundidad la herramienta que utiliza y hasta modificarla. Sirve de ese modo, como una excelente introducción al uso de A.E.L.s más complejos que estarán a disposición sólo para estudiantes avanzados e investigadores.

2- LA INTERFASE PARA P.C.

Hemos desarrollado dos soluciones para utilizar una P.C. como A.E.L..

La **versión A**, es más sencilla y menos engorrosa en su conexionado a la P.C.; la **versión B** en cambio es mucho mas veloz en el paso de los datos a la P.C. y por consiguiente más agil para su uso.

2.1.-VERSION A

2.1.1.-DESCRIPCION GENERAL

En esta versión la P.C. se comunica con la interfase por medio de instrucciones de *entrada/salida* (I/O). Se utiliza solo el bus de datos para los intercambios de información.

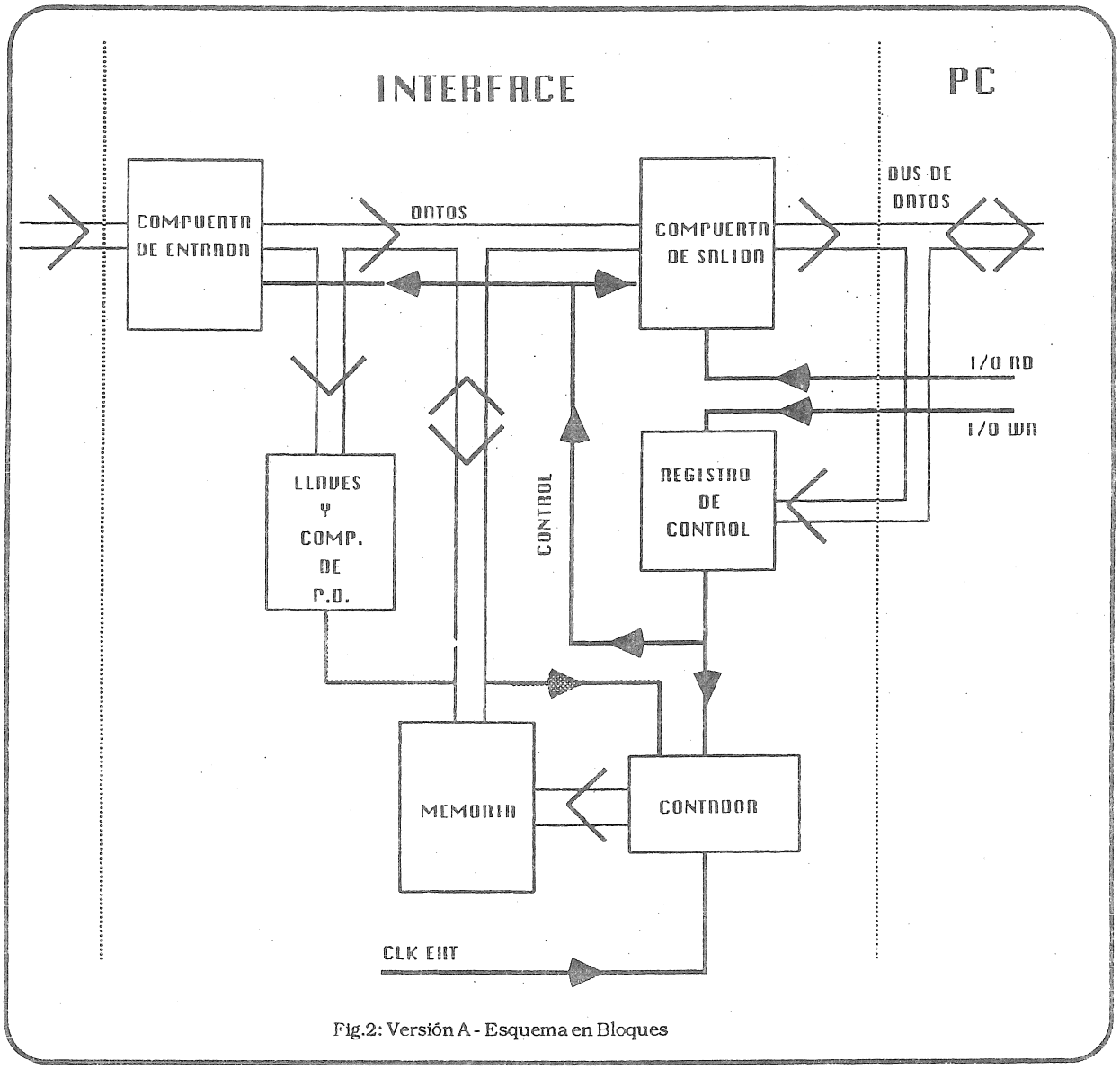


Fig.2: Versión A - Esquema en Bloques

La **P.C.** accede a los datos de **memoria** a través de la **compuerta de salida** (en Fig. 2), utilizando instrucciones de *lectura*; y controla el funcionamiento del sistema a través de un **registro de control** utilizando instrucciones de *escritura*.

El funcionamiento básico es el siguiente:

- *Durante el muestreo* (ver Fig.3), la **memoria** guarda los estados lógicos del sistema en estudio en las direcciones dadas por el **contador**. Este, a su vez, cuenta los pulsos de un reloj (**CLOCK**) que pueden venir del mismo sistema.

El **comparador de palabra de disparo**, que no se ha dibujado, determina el momento en que será habilitado el **contador** y por tanto, el momento en que se iniciará efectivamente el muestreo.

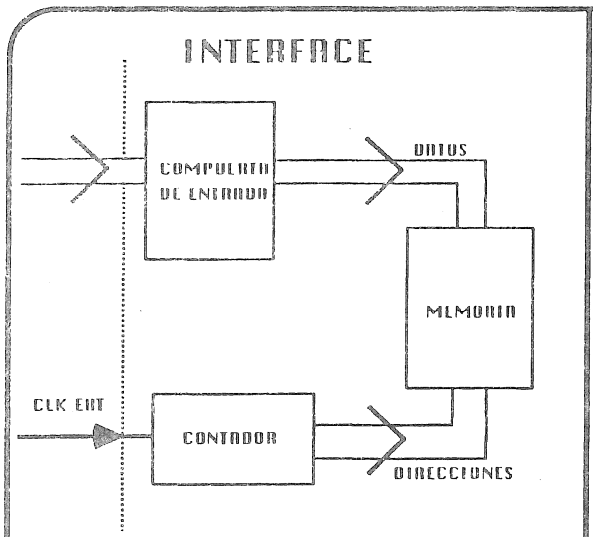


Fig.3

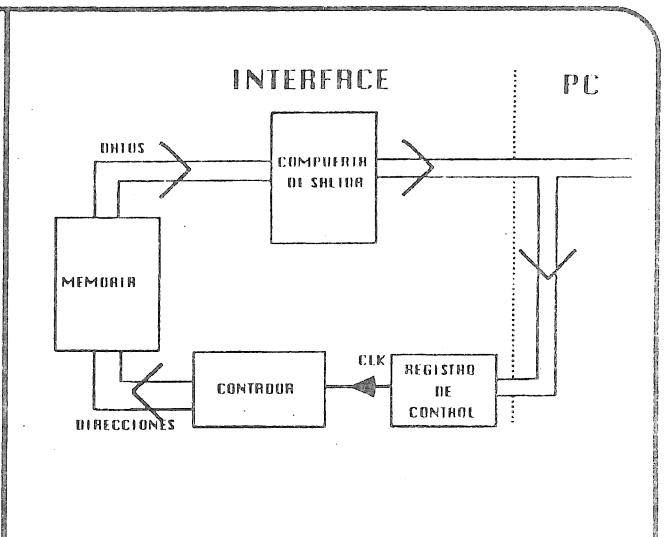


Fig. 4

- *Durante la lectura de datos* (ver Fig. 4) la **P.C.** recibe los datos que ya se encuentra en **memoria** en las direcciones dadas por el **contador** que recibe el **CLK**, esta vez de la misma P.C. por medio de instrucciones de *escritura de entrada/salida*.

En la Fig. 5, vemos el diagrama de flujo simplificado del programa ejecutado por la P.C. durante la lectura. Como se puede apreciar el gran número de instrucciones de *entrada/salida* hacen lento el paso de los datos 'de la **memoria de la interfase** a la **memoria de la P.C.**; tiempo al que habrá de sumar, el necesario para procesar dichos datos y mostrarlos en pantalla; solo luego, podrán ser analizados por el usuario.

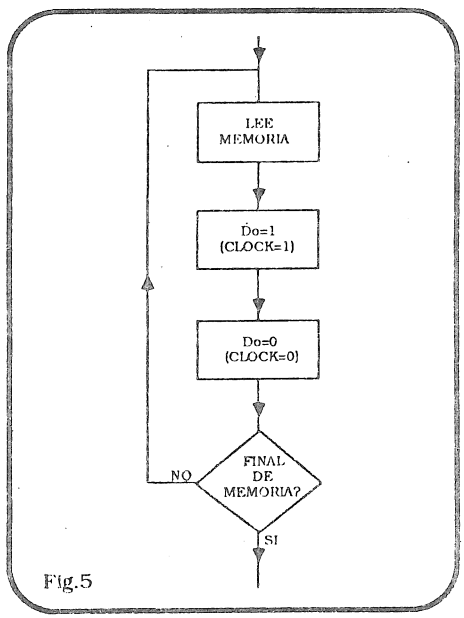


Fig.5

2.1.2.-DESCRIPCION DETALLADA

Vamos a hacer una descripción detallada de la versión A en base al esquema de la Fig. 6.:

PALABRA DE CONTROL

La P.C. escribe, usando instrucciones de *entrada/salida*, en el **registro de control** (6), un dato de cuatro bits llamado **palabra de control** que sirve para controlar el funcionamiento de la interfase. Estos cuatro bits son:

D0.....CLOCK
D1...../CLEAR
D2...../READ
D3.....P.D.

El registro de control esta constituido por cuatro FLIP-FLOPs (F.F.) tipo D correspondientes a cada bit.

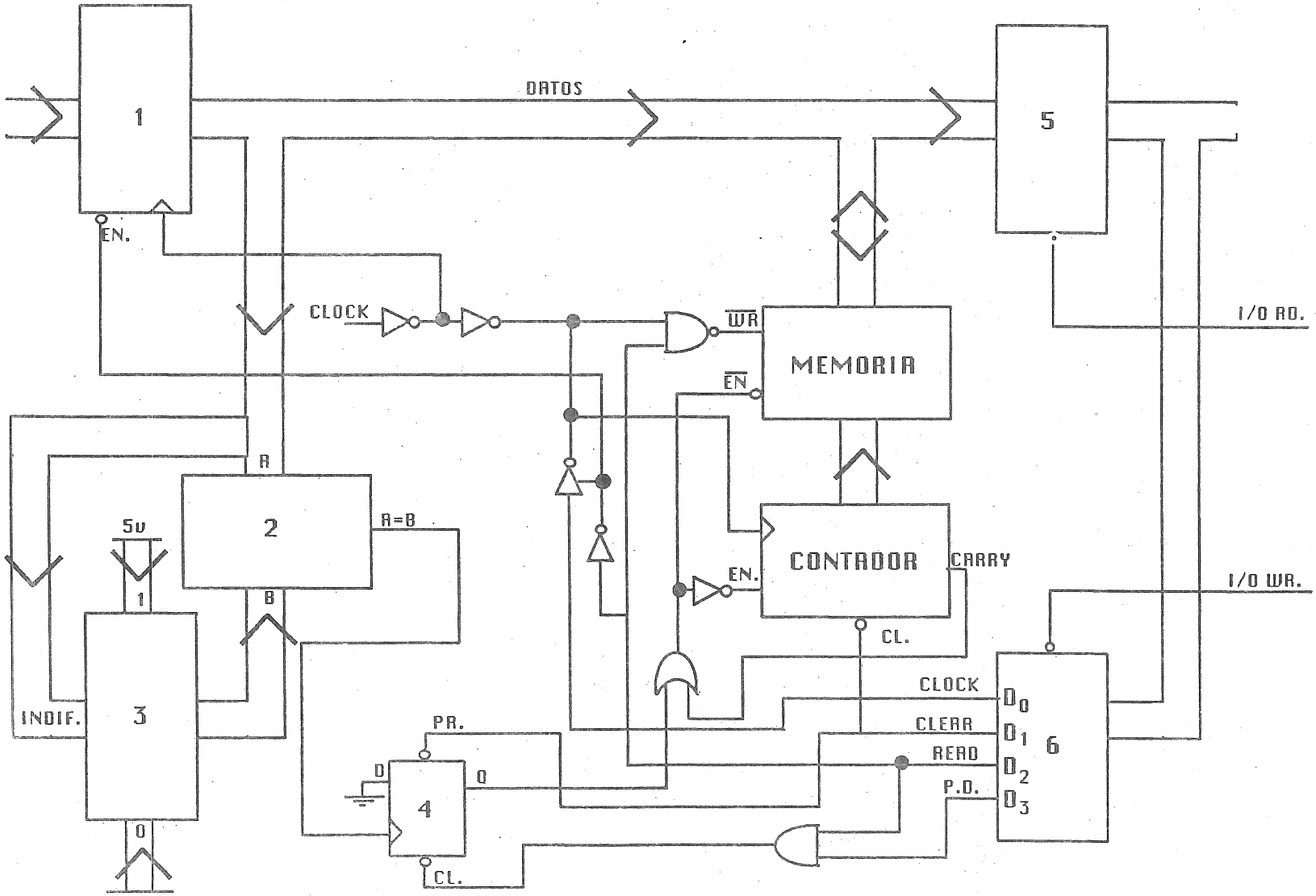


FIG.6- ESQUEMA DE A.E.L. VERSION A

ENTRADA DE DATOS

La **compuerta de entrada (1)** esta formada por F.F.s tipo D que aprovechan la trancisión positiva del **CLK** para congelar los datos en el bus de datos mientras el **contador** incrementa la dirección. Durante la trancisión negativa del mismo **CLK** la **memoria** (RAM estática) graba el dato presente en el bus en la direccion indicada por el contador. Cuando el contador finaliza el conteo se produce un acarreo (carry) que lo autodeshabilita.

PALABRA DE DISPARO

Si se desea usar P.D. , el *bit D3* de la **palabra de control** debe valer 1. De este modo el **F.F. de habilitación (4)** deshabilita el **contador** y la **memoria** hasta que **el comparador (2)** encuentre en el bus de datos una palabra igual a aquella seleccionada por las **llaves (3)**.

Cuando esto ocurra el **F.F.** cambiará de estado y habilitará el **contador** y la **memoria** para la entrada de datos hasta que se reinicie el muestreo. Cada bit de la P.D. se puede seleccionar, en forma independiente, en uno, indiferente o cero por medio de una llave. Para selecionar el estado indiferente se conecta el bit de la entrada **B** del **comparador (2)** al correspondiente bit de la entrada **A**.

LECTURA DE DATOS

Poniendo el *bit D2* de la **palabra de control** en 0, el **contador** y la **memoria** quedan permanentemente habilitados. La **memoria** queda inhibida para grabar nuevos datos y solo es posible leerla.. De este modo la P.C. por medio del *bit D0* de la **palabra de control** envía los pulsos de **CLK** al **contador** para incremenntar la dirección y leer los datos a través de la **compuerta** de **salida (5)** (BUFFERS DRIVER 3 - STATE), segun el digrama de flujo de la Fig. 5.

INICIALIZACION

EL *bit D1* en pone al **contador** en cero, y al **F.F. de habilitación (4)** en posición deshabilitar si la opción P.D. se ha elegido.

2.2.-VERSION B

En esta versión, a diferencia de la A, se utiliza el bus de direcciones a más del bus de datos de la P.C. (ver Fig. 7).

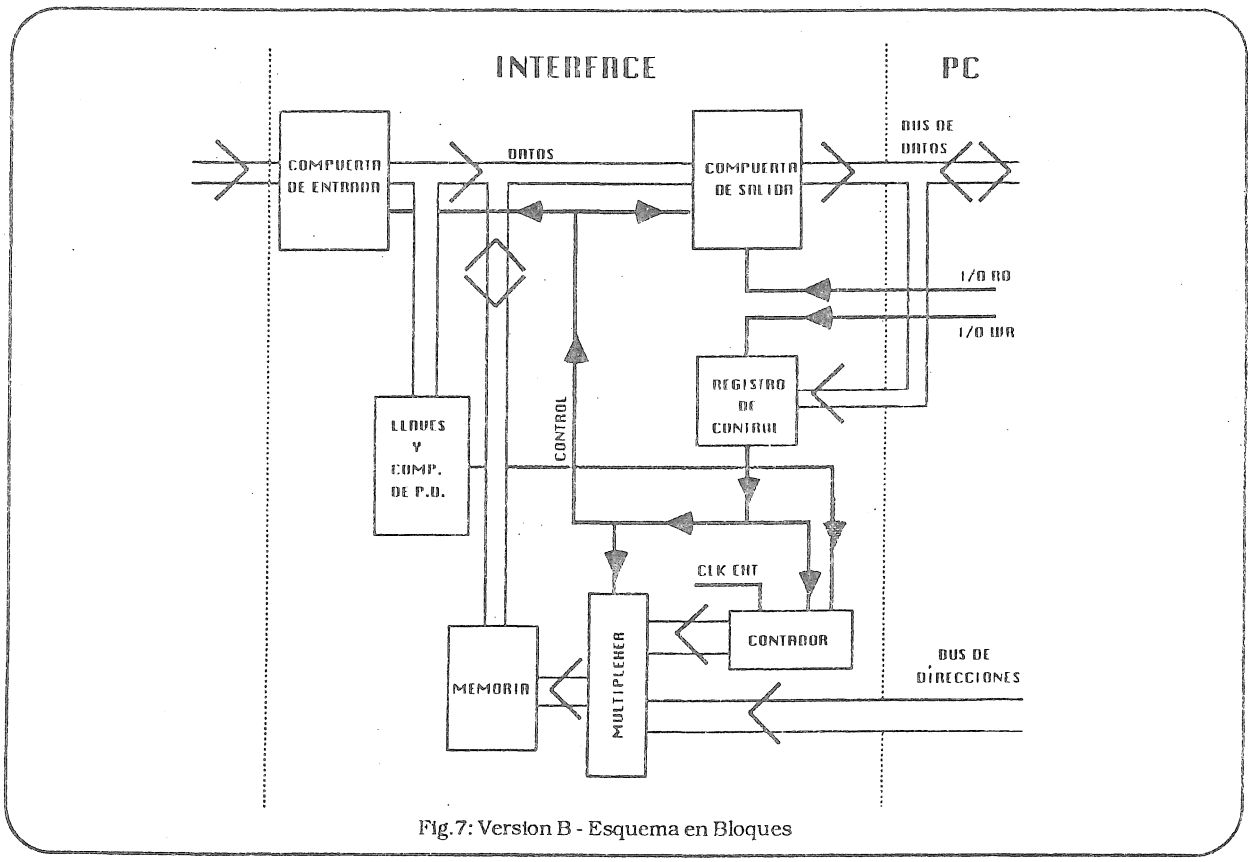


Fig.7: Version B - Esquema en Bloques

Aquí al igual, que en la versión A, la P.C. controla el funcionamiento del sistema a través de un **registro** utilizando instrucciones de *entrada/salida de escritura*. Sin embargo, el acceso a los datos de **memoria** se efectúa por medio de instrucciones de *lectura de memoria*, lo que se realiza muy velozmente.

El funcionamiento básico se resume de la siguiente forma:

-*Durante el muestreo* funciona exactamente de la misma forma que la versión A (ver Fig.8).. Por medio del **multiplexer** se conecta el bus de direcciones de la **memoria** al **contador** que se incrementa por medio del **CLK**.

-*Durante la lectura de datos*, en cambio, (ver Fig.9), por medio de una instrucción de *entrada/salida de escritura* la P.C. envía una **palabra de control** a la interfase que conecta el bus de direcciones de la **memoria** al bus de direcciones de la P.C. y de ese modo la **memoria de la interfase** pasa a formar parte del **mapa de memoria de la P.C.** y por tanto el acceso a los datos para el procesamiento es inmediato.

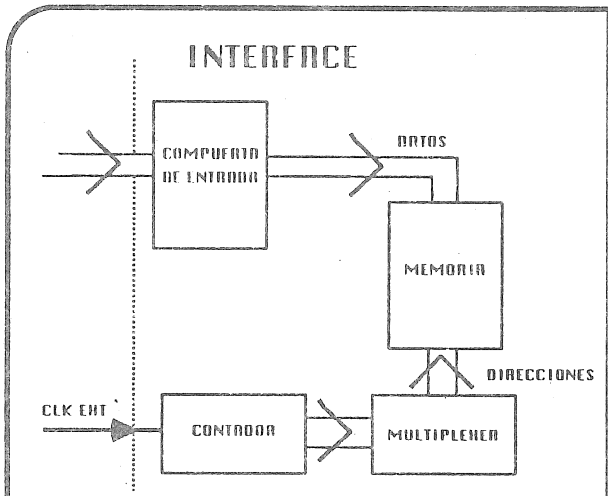


Fig.8

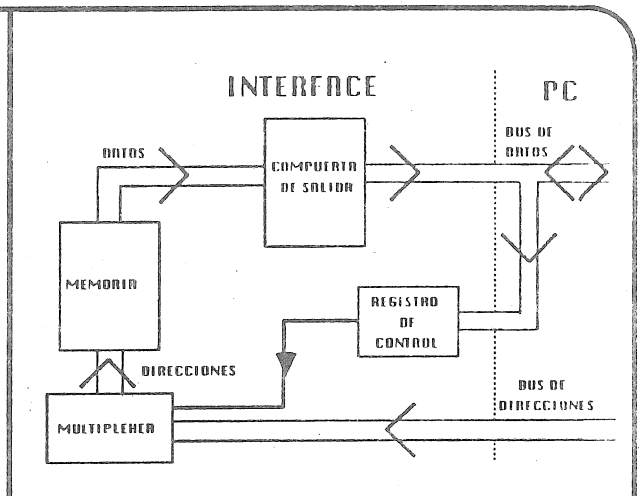


Fig.9

Para esto, debe disponerse de una zona de memoria no utilizada (por ejemplo una zona destinada a una ROM no instalada).

Con ésto, sólo es necesaria una instrucción de *entrada/salida de escritura* para que toda la información esté ya "dentro de la P.C.". Por tanto, el programa queda enormemente simplificado y el tiempo de acceso reducido.

Para una descripción más detallada podemos decir que a la versión A, Fig. 6, habría que hacerle las siguientes modificaciones:

-Agregar el circuito multiplexor de modo que las direcciones de la memoria provengan, ya sea del contador, como del bus de direcciones de la P.C.
-Este multiplexor debe seleccionar también los controles **/WR** y **/EN** de la memoria que podrán provenir de la interfase misma, como en la versión A, o de los decodificadores de la P.C. correspondientes a la zona de memoria en la que se va ha ubicar.
-Agregar un F.F. más al registro de control que servirá para controlar al multiplexor.

A cambio de las enormes ventajas, la versión B no representa un Hardware mucho más complejo ni el porcentaje de aumento en el costo es muy grande.

3- CONCLUSION

Atravez de este artículo, se ha mostrado ésta interfase para P.C. con la que se logra, con mucha sencillez, analizar el funcionamiento de un sistema digital. Es especialmente útil para fines didácticos por su muy bajo costo, y por la fácil comprención por parte del alumno que puede así apropiarse de la tecnología, usándola, analizándola y, por último, modificándola para sus necesidades particulares.

Al mismo tiempo se apunta más lejos; con este ejemplo se trata de desmitificar "las costosas instrumentaciones para la enseñanza de la Electrónica", mostrando como no siempre lo más caro y sofisticado es la mejor solución didáctica.

BIBLIOGRAFIA

[1] Almaini A.E.A., "Electronic Logic Systems", Prentice Hall International, UK, 1986

[2] Cannon D.L., "Understanding Digital Troubleshoting", The Texas Instrument Learning Center, USA, 1984

[3] Slater M., Bronson B., "I Microprocessori in Pratica, HW SW e Ricerca Guasti", Jackson Italiana Editrice, I, 1980

[4] Eccles W.J., "Microprocessors Systems a 16-bit Approach", Addison Wesley Inc., 1985

[5] Osborne A., Kane J., Rector R., Jacobson S., "Programmazione dello Z80 e Progettazione Logica", Jackson Italiana Editrice, I, 1981

[6] Schuchner J., Utard M., Ortiz C., "Analizador de Estados Logicos", Revista Telegrafica Electronica, Julio 1983, pp. 632-637

[7] Artwick B.A., "Microcomputer Interfacing", Prentice Hall Inc., NJ USA, 1980

PROJETO E IMPLEMENTAÇÃO DO BANCO DE DADOS DE UM SISTEMA PARA PAC DE MICROCIRCUITOS

Karla Parreiras Polanczyk
Nilton de Oliveira Junior
Alberto Henrique Frade Laender

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Caixa Postal 702
30161 - Belo Horizonte - MG
Brasil

SUMÁRIO

Este artigo descreve o projeto e a implementação do banco de dados de um sistema para PAC de microcircuitos. São apresentados os principais problemas de modelagem encontrados durante o projeto, as soluções adotadas, e a estratégia geral utilizada na implementação.

1. INTRODUÇÃO

A maioria dos sistemas para PAC (Projeto Assistido por Computador) consiste de um conjunto de ferramentas isoladas, cada uma exercendo uma função específica dentro do contexto da aplicação. Esforços têm sido feitos no sentido de integrar estas ferramentas em um único sistema. Uma das maneiras de se fazer esta integração é através de um banco de dados que atenda de forma centralizada às necessidades dos vários componentes do sistema. Desta forma, é possível manter a consistência entre as possíveis representações dos objetos de trabalho, bem como a integração de outras ferramentas disponíveis [5,6,7,9,20]. Sistemas para PAC, entretanto, trabalham com objetos de alta complexidade, diferentes daqueles existentes em aplicações convencionais. Isto requer técnicas especiais de modelagem de dados, o que torna os modelos de dados contemporâneos [18] e, consequentemente, a maioria dos sistemas de gerenciamento de banco de dados disponíveis comercialmente, inadequados para tais aplicações [6,7,14].

No caso de PAC de microcircuitos, a integração é fundamental para facilitar a manutenção da consistência dos diversos produtos resultantes das várias etapas de projeto. A utilização de um banco de dados como mecanismo de integração

permite o armazenamento consistente de todos os dados referentes ao circuito, desde aqueles que representam informações gráficas até os que são necessários na etapa de produção do "layout" (Veja a Figura 1).

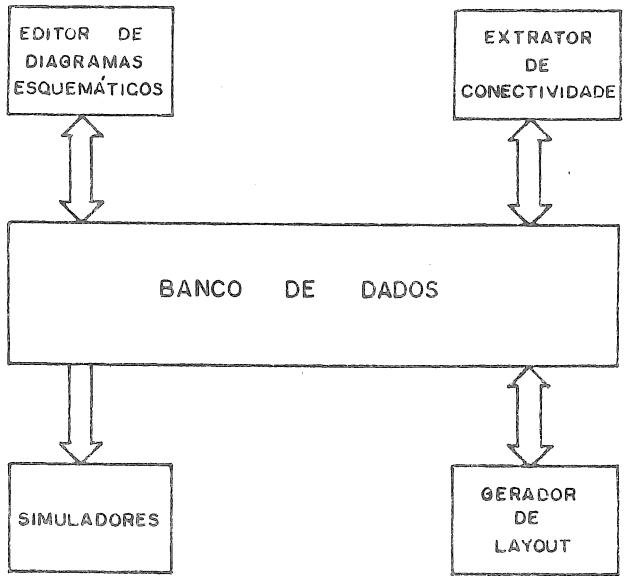


Figura 1

Este artigo descreve o projeto e a implementação do banco de dados utilizado pelo sistema AIPIM (Ambiente Integrado para Projeto Interativo de Microcircuitos) [2] em desenvolvimento no Departamento de Ciência da Computação da UFMG, através de convênio estabelecido com o Centro de Pesquisa e Desenvolvimento da TELEBRÁS. O AIPIM é um sistema para PAC de microcircuitos, utilizando células padrão, que provê, entre outras, facilidades para captura de diagramas esquemáticos [11], posicionamento de células [19], traçado automático de rotas [10] e desenho de "layouts" [21]. O banco de dados do AIPIM armazena todos os dados necessários aos seus vários componentes, possibilitando a total integração do sistema.

O restante do artigo se divide da seguinte maneira: na Seção 2 são analisados alguns aspectos referentes ao projeto de microcircuitos; na Seção 3 é descrito o projeto conceitual do banco de dados do sistema AIPIM; na seção 4 é apresentada uma breve descrição da implementação; finalmente, na Seção 5 são apresentadas algumas considerações finais sobre o trabalho.

2. CONSIDERAÇÕES ACERCA DO PROJETO DE MICROCIRCUITOS

As aplicações de PAC envolvem geralmente dados de natureza complexa e que variam de aplicação para aplicação. É imprescindível, portanto, que ao se desenvolver um sistema para PAC, se compreenda claramente o seu ambiente de aplicação. Assim, analisaremos aqui algumas características inerentes ao projeto de microcircuitos. Esta análise não pretende ser exaustiva e tem como objetivo caracterizar as necessidades básicas de um ambiente de projeto de microcircuitos, tendo como base a visão do sistema AIPIM.

Um microcircuito é constituído basicamente de componentes, interligados através de sinais, cujo projeto lógico é expresso através de um diagrama esquemático. Neste diagrama, os componentes são representados graficamente por figuras geométricas e correspodem a instâncias de macros, que são as funções pré-definidas disponíveis para a elaboração do circuito. As macros podem ser utilizadas em um diagrama tantas vezes quantas necessárias, sendo que as instâncias correspondentes não precisam ser necessariamente iguais, uma vez que o projetista pode alterar algumas de suas características básicas (por exemplo, os parâmetros elétricos).

As macros podem ser de dois tipos: aquelas que constituem as células mais básicas de um circuito, as macros padrão, e aquelas criadas pelo projetista a partir de outras, as macros de projeto. As macros de projeto possuem um projeto interno que representa estruturalmente a função lógica ou elétrica que elas sintetizam. Estas macros compreendem, portanto, o projeto hierárquico de um circuito. Este é um aspecto muito importante, uma vez que possibilita ao projetista produzir o seu circuito hierarquicamente e por etapas. Uma macro de projeto pode ainda apresentar diversos "layouts" e ser considerada como uma célula básica para a geração de outros "layouts". Desta forma, o projetista pode acoplar ao seu circuito blocos já testados. Estes blocos constituem as macros de layout.

Existe um outro aspecto importante a se considerar no que diz respeito às macros. Uma macro de projeto pode ter vários projetos internos associados, sendo que qualquer um deles pode ser utilizado em qualquer uma de suas instâncias presentes em um diagrama esquemático. Isso caracteriza a existência de múltiplas versões de uma macro de projeto, possibilitando ao projetista várias opções para elaboração do circuito.

Na seção seguinte veremos como o banco de dados do sistema AIPIM foi projetado para atender a estas necessidades. Em particular, mostraremos como foram modelados os aspectos referentes ao projeto hierárquico de circuitos e à existência de múltiplas versões de um circuito ou macro de projeto.

3. O BANCO DE DADOS DO SISTEMA AIPIM

O banco de dados do sistena AIPIM foi projetado para atender aos seguintes objetivos:

- suportar o editor de diagramas esquemáticos, armazenando todos os dados que representam informações gráficas e de contexto necessárias ao seu funcionamento.

- permitir o projeto hierárquico de circuitos, através do armazenamento das definições de macros de projeto e macros de layout.

- suportar a existência de diferentes versões de um mesmo circuito, macro de projeto ou macro de layout.

- possibilitar uma pesquisa espacial eficiente, viabilizando a exibição pelo editor gráfico dos objetos situados numa determinada área ou janela de visualização.

- suportar o posicionamento de células e o traçado de rotas na etapa de elaboração do "layout".

- permitir que outras ferramentas já existentes (simuladores, por exemplo) possam ser facilmente integradas ao sistema AIPIM quando necessário.

O projeto conceitual [3] do banco de dados envolveu basicamente duas etapas: (1) o desenvolvimento do modelo conceitual, no qual são descritos os objetos envolvidos e os relacionamentos entre eles, e (2) a especificação das funções primitivas de manipulação dos dados (criação, remoção, atualização e recuperação). A relação destas funções é apresentada no Apêndice 1. Uma descrição mais detalhada pode ser encontrada em [12,14].

A Figura 2 apresenta o modelo conceitual produzido com base no modelo de entidades e relacionamentos estendido (MER+), uma simplificação do modelo proposto em [3]. Antes de apresentarmos os detalhes do modelo conceitual, faremos uma breve descrição do MER+ e de sua simbologia.

3.1. O Modelo de Entidades e Relacionamentos Estendido (MER+)

O MER+ é uma extensão do modelo de entidades e relacionamentos proposto por Chen [4]. No modelo de entidades e relacionamentos, as primitivas básicas de modelagem são entidades, atributos e relacionamentos. As entidades correspondem aos objetos do mundo real inerentes à aplicação em questão. Os atributos descrevem as propriedades destes objetos e os relacionamentos representam associações significativas entre entidades. Os relacionamentos podem ser dos tipos "um-para-um" (1:1), "um-para-muitos" (1:N), "muitos-para-um" (N:1) e "muitos-para-muitos" (M:N) [18].

O MER+ estende o modelo de Chen, adicionando outras construções que permitem expressar, por exemplo, hierarquias de generalização [16,18] e restrições de existência (totalidade) nos relacionamentos [18]. A Figura 3 apresenta a simbologia básica adotada pelo MER+.

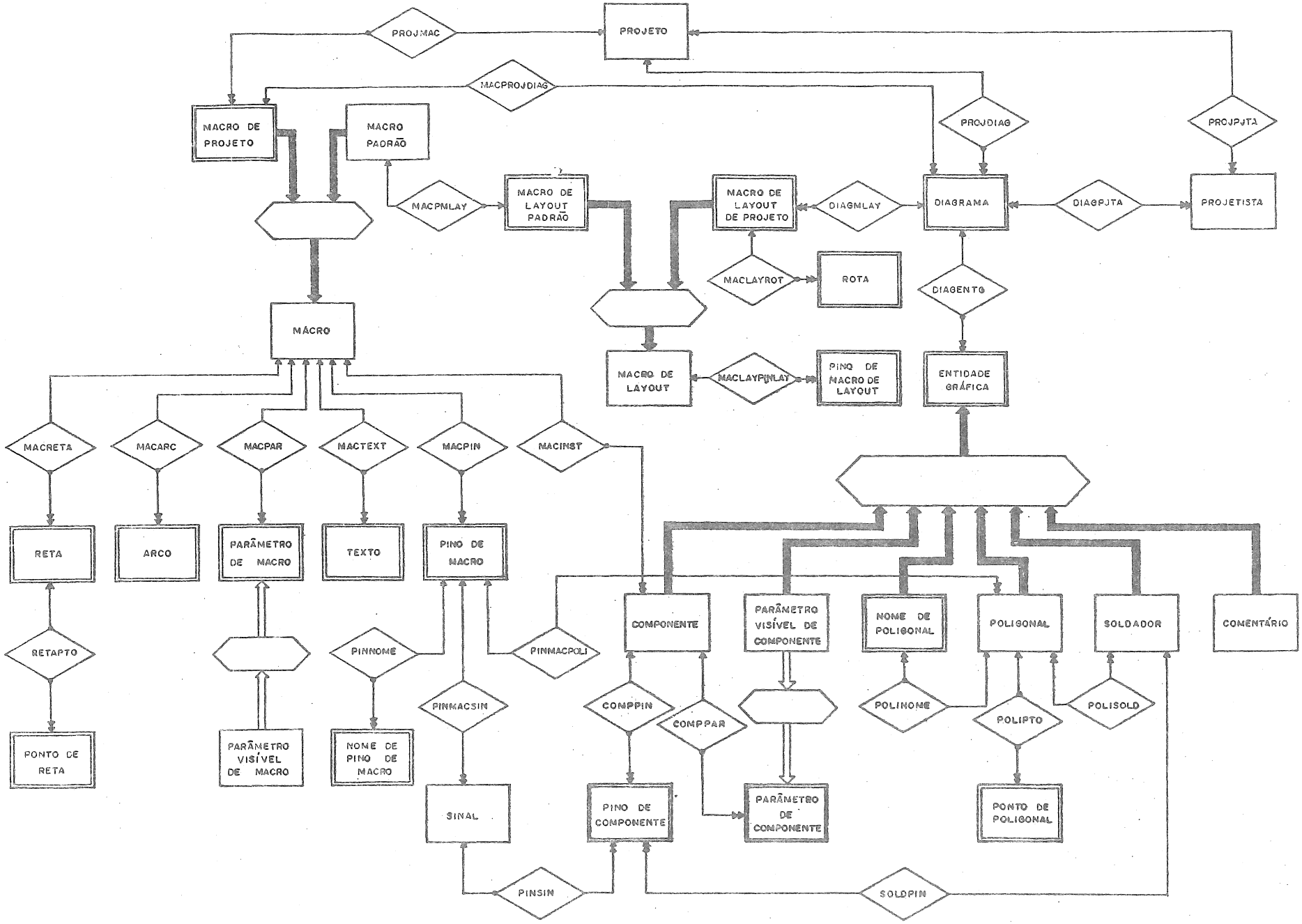

PROJMAC
PROJETO
MACPROJDIAG
MACRO DE PROJETO
MACRO PADRÃO
PROJDIAG
PROJPJTA
MACPMLAY
MACRO DE LAYOUT PADRÃO
MACRO DE LAYOUT DE PROJETO
DIAGMLAY
DIAGRAMA
DIAGPJTA
PROJETISTA
MACLAYROT
ROTA
DIAGENTO
MÁCRO
MACRO DE LAYOUT
MACLAYPINLAY
PINO DE MACRO DE LAYOUT
ENTIDADE GRÁFICA
MACRETA
MACARC
MACPAR
MACTEXT
MACPIN
MACINST
RETA
ARCO
PARÂMETRO DE MACRO
TEXTO
PINO DE MACRO
RETAPTO
PINNOME
PINMACPOLI
COMPONENTE
PARÂMETRO VISÍVEL DE COMPONENTE
NOME DE POLIGONAL
POLIGONAL
SOLDADOR
COMENTÁRIO
PONTO DE RETA
PARÂMETRO VISÍVEL DE MACRO
NOME DE PINO DE MACRO
PINMACSIN
COMPPIN
COMPPAR
POLINOME
POLIPTO
POLISOLD
SINAL
PINO DE COMPONENTE
PARÂMETRO DE COMPONENTE
PONTO DE POLIGONAL
PINSIN
SOLDPIN

Figura 2

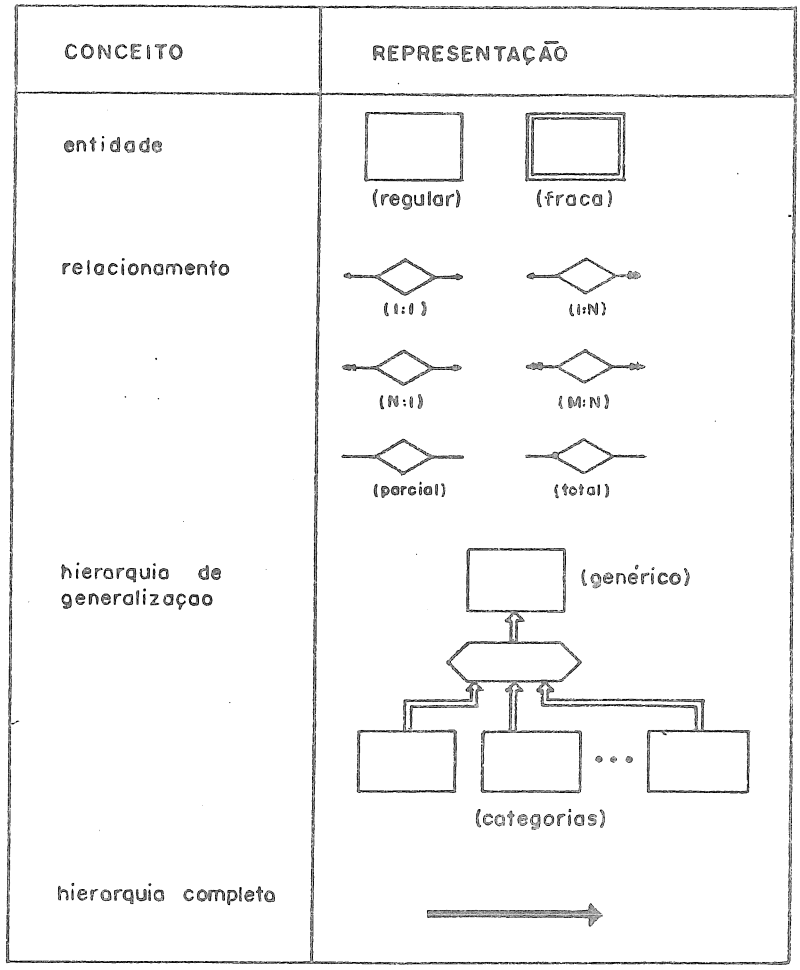


Figura 3

3.2. Descrição do Modelo Conceitual

O modelo conceitual apresentado na Figura 2 descreve a estrutura lógica do banco de dados do sistema AIPIM, levando-se em conta os objetivos citados anteriormente. Discutiremos aqui como os problemas inerentes à aplicação foram

Primeiramente, era necessário possibilitar uma pesquisa espacial eficiente, uma vez que o editor de diagramas esquemáticos precisa ter acesso a todos os objetos graficamente presentes em uma determinada janela de visualização (a janela de visualização é uma área retangular, determinada pelas coordenadas de seus cantos inferior esquerdo e superior direito) [11]. Note que estes objetos podem corresponder a entidades diferentes no banco de dados, como por exemplo, componentes, poligonais, soldadores, etc. Para solucionar este problema, foi criada uma entidade especial, a ENTIDADE GRÁFICA, que é uma generalização de todos os objetos que podem aparecer na tela do editor. A hierarquia de

generalização, neste caso, é uma hierarquia completa [3], uma vez que cada ocorrência da ENTIDADE GRÁFICA pertence a uma de suas categorias. Desta forma, é possível identificar todos aqueles objetos presentes em uma determinada área do circuito, independentemente de seus tipos.

Um outro aspecto importante modelado refere-se às macros de projeto. Durante uma sessão de trabalho com o editor, o projetista pode definir macros de projeto, criando o seu circuito de forma hierárquica. Para representar isto conceitualmente, foi criada a entidade MACRO DE PROJETO. Através do relacionamento MACPROJDIAG, podemos obter o(s) diagrama(s) esquemático(s) correspondente(s) ao(s) projeto(s) interno(s) de uma determinada macro de projeto. Desta forma, foi também resolvido o problema de se armazenar diferentes versões de projeto interno para uma mesma macro de projeto.

Assim como as macros de projeto, as macros de layout também precisavam ser modeladas. Note, no entanto, que uma macro de layout pode estar associada ou ao projeto interno de uma macro de projeto ou ao projeto interno de uma macro padrão. Por isto, foi criada a entidade MACRO DE LAYOUT como uma generalização das entidades MACRO DE LAYOUT PADRÃO e MACRO DE LAYOUT DE PROJETO. Assim, através do relacionamento DIAMGLAY, pode-se obter as possíveis macros de layout de um determinado diagrama esquemático. Com isto podemos obter também as possíveis versões de um diagrama esquemático.

Finalmente, cabe ressaltar que várias das entidades definidas no modelo conceitual são entidades fracas [4,18], ou seja, entidades cuja existência é dependente de outra. Esta é uma característica inerente à própria aplicação. Assim, por exemplo, toda ocorrência da ENTIDADE GRÁFICA (por exemplo, um componente) é dependente da existência do diagrama ao qual está associada, que por sua vez é dependente de um projeto.

4. IMPLEMENTAÇÃO

O sistema AIPIM está sendo implementado em Pascal Microsoft, em um microcomputador do tipo PC-AT com 2 MBytes de memória principal, duas unidades de disco flexível, um disco rígico com capacidade de 20 MBytes, e facilidades para entrada e saída gráficas (monitor gráfico, tablete, "plotter" de mesa e impressora gráfica). Para suporte ao banco de dados está sendo utilizado o sistema Btrieve [16], que é um sistema de gerenciamento de arquivos que possui algumas características relacionais. O acesso ao banco de dados é feito através de uma série de procedimentos que implementam as funções de manipulação especificadas na fase de projeto conceitual e relacionadas no Apêndice 1. Estes procedimentos provêm uma interface padrão, através da qual são mantidas as restrições de integridade definidas para o banco de dados.

O esquema relacional, gerado a partir do modelo conceitual como sugerido em [3], se encontra detalhado no Apêndice 2. Os nomes das relações e dos atributos são auto-explicativos e dão uma idéia da semântica envolvida. Note que, para cada entidade, um identificador gerado internamente foi definido como sendo a chave primária da relação correspondente. Desta forma,

estabeleceu-se um mecanismo de identificação simples e eficiente (em geral um dos principais problemas neste tipo de aplicação [5, 20]), evitando-se também a propagação de chaves compostas para representação dos relacionamentos.

Como ilustração da interface do banco de dados, é apresentada abaixo a definição dos procedimentos que implementam as principais funções primitivas para manipulação de uma poligonal. A definição inclui a lista de relações às quais cada um dos procedimentos tem acesso. RemPolig, por exemplo, tem acesso às relações POLIGONAL, ENTIDADE_GRÁFICA, NOME_DE_POLIGONAL, PONTO_DE_POLIGONAL e POLISOLD. Estas relações correspondem às entidades de mesmo nome e ao relacionamento POLISOLD representados no modelo conceitual (Figura 2). Isto significa que ao removermos uma poligonal do banco de dados, este procedimento não só remove a tupla correspondente da relação POLIGONAL, como também remove as tuplas associadas das relações ENTIDADE_GRÁFICA, PONTO_DE_POLIGONAL, NOME_DE_POLIGONAL e POLISOLD.

Procedimento CriaPolig
descrição: insere uma nova poligonal no banco de dados;
parâmetros de entrada: iddiag, xmin, ymin, xmax, ymax, nomepolig, estilolinha, cor, marcaexcon;
parâmetros de saída: idpolig, resultado;
relações manipuladas: ENTIDADE_GRÁFICA, POLIGONAL;

Procedimento AchaPolig
descrição: recupera os valores dos atributos de uma poligonal;
parâmetros de entrada: idpolig;
parâmetros de saída: nomepolig, xmin, ymin, xmax, ymax, estilolinha, cor, marcaexcon, resultado;
relações manipuladas: ENTIDADE_GRÁFICA, POLIGONAL;

Procedimento RemPolig
descrição: remove uma poligonal do banco de dados;
parâmetros de entrada: idpolig;
parâmetros de saída: resultado;
relações manipuladas: ENTIDADE_GRÁFICA, POLIGONAL, NOME_DE_POLIGONAL, PONTO_DE_POLIGONAL, POLISOLD;

Procedimento AtPolig
descrição: atualiza os valores dos atributos de uma poligonal;
parâmetros de entrada: idpolig, seleção, nomepolig, estilolinha, cor, marcaexcon, xmin, ymin, xmax, ymax;
parâmetros de saída: resultado;
relações manipuladas: ENTIDADE_GRÁFICA, POLIGONAL, NOME_DE_POLIGONAL;

5. CONCLUSÕES

Um banco de dados é uma forma bastante eficiente de se integrar diferentes ferramentas e assegurar a consistência dos dados de projeto em um sistema para PAC de microcircuitos [5,6,9,20]. O projeto de um banco de dados para tal aplicação não é, entretanto, uma tarefa fácil. As características da aplicação e a falta de ferramentas adequadas para modelagem dos dados impõem uma série de dificuldades ao projetista do banco de dados.

No caso particular do banco de dados do sistema AIPIM, estas dificuldades foram parcialmente superadas através da utilização do MER+. Apesar de não possuir facilidades para a modelagem de objetos complexos, o MER+ possibilitou uma abordagem simples através da qual a semântica da aplicação foi capturada de forma satisfatória. Desta forma, foi possível gerar um esquema relacional consistente e especificar uma interface que assegura as restrições de integridade definidas, possibilitando a utilização do sistema Btrieve de forma sistemática e eficiente.

AGRADECIMENTOS

Os autores gostariam de agradecer a todos o membros do projeto AIPIM pelas sugestões e comentários. Este trabalho foi desenvolvido com apoio financeiro do convênio UFMG/TELEBRÁS.

REFERÊNCIAS

[1] BATORY, D. S. & KIM, W. Modeling concepts for VLSI CAD objects. ACM Transactions on Database Systems 10, 3 (September 1985), 322-346.

[2] CAMPOS. I. M. AIPIM - Ambiente Integrado para Projeto Interativo de Microcircuitos. Anais do I Congresso da Sociedade Brasileira de Microeletrônica, Campinas, Julho 1986, pp. 351-360.

[3] CERI, S. (ed.) Methodology and Tools for Database Design. North-Holland, Amsterdam, 1983.

[4] CHEN, P. P. S. The entity-relationship model - Toward a unified view of data. ACM Transactions on Database Systems 1, 1 (March 1976), 9-36.

[5] CHU, K. et al. Vdd - A VLSI design database system. Databases for Engineering Aplications, ACM Database Week (May 1983), 39-50.

[6] KATZ, R. H. A database approach for managing VLSI design data. Proc. ACM/IEEE Design Automation Conference, Las Vegas, Nevada, June 1982, pp. 274-282.

[7] McLELLAN, P. Effective data management for VLSI design. Proc. ACM/IEEE

Design Automation Conference, Las Vegas, Nevada, 1985, pp. 652-657.

[8] McLEOD, D. et al. An approach to information management for CAD/VLSI applications. Databases for Engineering Applications, ACM Database Week (May 1983), pp. 39-50.

[9] NEUMANN, T. On representing the design information in a common database. Databases for Engineering Applications, ACM Database Week (May 1983), pp. 81-87.

[10] OLIVEIRA, E. Um sistema para o traçado automático de rotas em circuitos integrados baseados em "standard-cells". Anais do I Congresso da Sociedade Brasileira de Microeletrônica, Campinas, Julho 1986, pp. 310-319.

[11] OLIVEIRA, N. Jr. & POLANCZYK, K. P. EDDIE - Um Editor Gráfico de Circuitos a Nível Lógico. Anais do I Congresso da Sociedade Brasileira de Microeletrônica, Campinas, Julho 1986, pp. 419-429.

[12] POLANCZYK, K. P. Banco de Dados do AIPIM - Descrição Funcional da Visão do Subsistema EDDIE. Relatório Técnico, DCC-ICEX-UFMG, 1986.

[13] POLANCZYK, K. P. Descrição do Esquema Relacional do Banco de Dados do AIPIM. Relatório Técnico, DCC-ICEX-UFMG, 1986.

[14] POLANCZYK, K. P. Projeto e implementação de um banco de dados para suporte a PAC de microcircuitos. Dissertação de mestrado, DCC-ICEX-UFMG, 1987 (em elaboração).

[15] SIDLE, T. W., Weakness of commercial database management systems in engineering applications. Proc. ACM/IEEE Design Automation Conference, Minneapolis, Minesotta, June 1980.

[16] SMITH, J. M. & SMITH, D. Database abstractions: aggregations and generalizations, ACM Transactions on Database Systems 1, 2 (June 1977), 105-133.

[17] SOFTCRAFT, INC. Btrieve Version 3.15. Austin, Texas, 1984.

[18] TSICHRTZIS, D. & LOCHOVSKY, F. Data Models, Prentice-Hall, Englewood Cliffs, N.J., 1982.

[19] TRILLO, J. L. R. Um posicionador de células padrão em "layouts" de circuitos integrados. Anais do I Congresso da Sociedade Brasileira de Microeletrônica, Campinas, Julho 1986, pp. 320-328.

[20] WONG, S. & BRISTOL, W. A. A Computer-aided design data base. Proc. ACM/IEEE Design Automation Conference, San Diego, California, June 1979, pp. 398-402.

[21] ZOLA, W. M. N. CIFDRAW - Um programa para "check-plot" de circuitos

integrados. Anais do I Congresso da Sociedade Brasileira de Microeletrônica, Campinas, Julho 1986, pp. 463.

APÊNDICE 1

FUNÇÕES DE MANIPULAÇÃO DO BANCO DE DADOS DO SISTEMA AIPIM

1) Funções para manipulação da entidade Projeto:
CriaProj, CriaProjPjta, AchaProjPjta, AchaProj, RemProj, RemProjPjta, AtProj;

2) Funções para manipulação da entidade Diagrama:
CriaDiag, AchaDiagPjta, AchaDiagProj, AchaDiag, RemDiag, AtDiag;

3) Funções para manipulação da entidade Projetista:
CriaPjta, CriaPjtaDiag, AchaPjtaAloc, AchaPjtaDiag, RemPjta, RemPjtaDiag;

4) Funções para manipulação das entidades Macro Padrão e Macro de Projeto:
CriaMacroPadrão, CriaMacroProjeto, CriaReta, CriaPtoReta, CriaArco, CriaTexto, CriaPinoMacro, CriaNomePinoMacro, CriaParamMacro, CriaParVisMacro, AchaMacroPadrão, AchaMacroProjeto, AchaRetaMacro, AchaPontoReta, AchaArcoMacro, AchaParamMacro, AchaParVisMacro, AchaPinoMacro, AchaNomePinoMacro, AchaTextoMacro, RemMacroProj, AtMacProj, AtReta, AtPtoReta, AtArco, AtTexto;

5) Função para manipulação da entidade Entidade Gráfica:
AchaEntGraf;

6) Funções para manipulação da entidade Componente:
CriaComp, CriaParamComp, CriaParVisComp, CriaPinoComp, AchaParamComp, AchaParVisComp, AchaPinoComp, AchaComp, RemComp, RemParVisComp, RemParamComp, AtComp, AtParComp, AtParVisComp;

7) Funções para manipulação da entidade Poligonal:
CriaPolig, CriaNomePolig, CriaPtoPolig, AchaPolig, AchalNomePolig, AchaNomesPolig, AchaPtoPolig, RemPolig, RemNomePolig, AtPolig, AtNomePolig, AtPtoPolig;

8) Funções para manipulação da entidade Soldador:
CriaSold, CriaSoldPin, CriaSoldPoli, AchaSold, AchaSoldPoli, AchaSoldPoli, AchaSoldPin, AchaPinSold, RemSoldPin, RemSoldPoli, RemSold, AtSold;

9) Funções para manipulação da entidade Comentário:
CriaComent, AchaComent, RemComent, AtComent;

APÊNDICE 2

ESQUEMA RELACIONAL DO BANCO DE DADOS DO SISTEMA AIPIM

PROJETO (idproj, nomeproj, nomeresp, dcriação)

PROJETISTA (idpjta, nomepjta)

PROJPJTA (idproj, idpjta)

DIAGRAMA (iddiag, nomediag, dcriação, dultatu, respultatu, xmin, ymin, xmax, ymax, nmens, netlist, refdiagx, refdiagy, raiox, raioy, idproj, idmacroproj)

DIAGPJTA (iddiag, idpjta)

ENTIDADE_GRÁFICA (identgraf, xmin, ymin, xmax, ymax, categoria, iddiag)

COMENTÁRIO (idcoment, texto, cor, orientação, tamtexto)

COMPONENTE (idcomp, nomecomp, cor, teta, escala, translacao, posicaox, espelhamento, idmacro)

NOME_DE_POLIGONAL (idnomepolig, nomepolig, cor, orientação, tamtexto, idpolig)

PARAMETRO_VISÍVEL_DE_COMPONENTE (idparviscomp, completo, orientação, tamtexto, cor)

POLIGONAL (idpolig, nomepolig, estilolinha, cor, npontos, idpinomacro)

SOLDADOR (idsold, cor, translacao, escala, teta)

PARÂMETRO_DE_COMPONENTE (idparamcomp, nomepar, valor, visibilidade, idcomp)

PINO_DE_COMPONENTE (idpinocomp, idpinomacro, idsinal, idsold, idcomp)

PONTO_DE_POLIGONAL (idptopolig, pontox, pontoy, idpoli)

POLISOLD (idpolig, idsold)

MACRO (idmacro, nomeresp, dcriação, refx, refy, xmin, ymin, xmax, ymax, categoria)

MACRO_PADRÃO (idmacropadrao, nomemacro)

MACRO_DE_PROJETO (idmacroproj, nomemacro, idproj)

RETA (idreta, npontos, idmacro)

PONTO_DE_RETA (idptoreta, pontox, pontoy, idreta)

ARCO (idarco, arc1x, arc1y, arc2x, arc2y, arc3x, arc3y, idmacro)

TEXTO (idtexto, texto, txtxrel, txtyrel, cor, orientação, tamtexto, idmacro)

PINO_DE_MACRO (idpinomacro, xrel, yrel, nomepino, tipo, idmacro)

NOME_DE_PINO_DE_MACRO (idnomepinmacr, xrel, yrel, cor, orientação, nomepino, tamtexto, idpinomacro)

PARÂMETRO_DE_MACRO (idparammacro, nomeparam, visibilidade, valor, idmacro)

PARÂMETRO_VISÍVEL_DE_MACRO (idparvismacro, xrel, yrel, completo, orientação, nomepar, tamtexto, cor)

SINAL (idsinal, nomesinal, crítico, capacitância, idpinomacro)

MACRO_DE_LAYOUT (idmacrolayout, altura, largura, pad, categoria)

PINO_DE_MACRO_DE_LAYOUT (idpinmaclay, xinf, xsup, nomepino, intercamb, idmacrolayout)

MACRO_DE_LAYOUT_PADRÃO (idmaclaypadrao, nomemacrolayout, idmacropadrao)

MACRO_DE_LAYOUT_DE_PROJETO (idmaclayproj, nomemacrolayout, iddiag)

ROTA (idrota, canal, máscara, x, y, comprimento, idmacrolayproj)

Prof. Celso Maciel da Costa
Prof. Simão Sirineo Toscani
Prof. Thadeu Botteri Corso

Curso de Pós-Graduação em Ciência da Computação
Universidade Federal do Rio Grande do Sul
Caixa Postal 1501
90.001 - Porto Alegre - RS
Telefone (0512) 21-2161
BRASIL

Este trabalho descreve o projeto MICRO-BIS, em desenvolvimento na Universidade Federal do Rio Grande do Sul, que consiste na construção de uma máquina multiprocessadora simples e na implementação do software para essa máquina. Dá-se detalhes sobre as chamadas do sistema, o tratamento de interrupções, a exclusão mútua no acesso aos dados globais e a inicialização do sistema.

1. INTRODUÇÃO

Alguns anos atrás os grupos de pesquisa em arquitetura de computadores e sistemas operacionais do Curso de Pós-Graduação em Ciência da Computação (CPGCC) da Universidade Federal do Rio Grande do Sul (UFRGS) decidiram construir um sistema multiprocessador que fosse adequado para a execução de programas escritos na linguagem Pascal Concorrente. O projeto, denominado Multimicro /TOS 82/, foi concebido a partir de sugestões apresentadas por Brinch Hansen /HAN 78/ e correspondia a uma arquitetura envolvendo vários processadores e vários módulos de memória. Infelizmente, devido à complexidade do projeto, à inexistência de laboratórios adequados, às dificuldades para se obter os recursos materiais necessários e também ao afastamento de alguns pesquisadores do projeto, o mesmo foi descontinuado após 2 anos de trabalho.

O motivo principal do insucesso do projeto Multimicro foi a grande complexidade do sistema a ser construído. Esta constatação levou parte dos pesquisadores envolvidos no projeto à decisão de construir um novo sistema bem mais simples, mas que pudesse resultar em um protótipo acabado, em um curto espaço de tempo. A existência de um protótipo em funcionamento serviria de estímulo para a elaboração de projetos mais ambiciosos no futuro.

O resultado foi a definição de uma máquina bimicroprocessadora, denominada MICRO-BIS, com arquitetura simples, de fácil implementação, que é adequada para a execução de programas Pascal Concorrente.

A máquina MICRO-BIS, conforme é mostrado na figura 1, é formada por dois microprocessadores MC68000, cada qual com uma memória local de 128 KB e ambos compartilhando uma memória global também de 128 KB. Estes componentes estão dispostos em uma placa ligada a um microcomputador PC/XT que serve como processador de entrada e saída (PES).

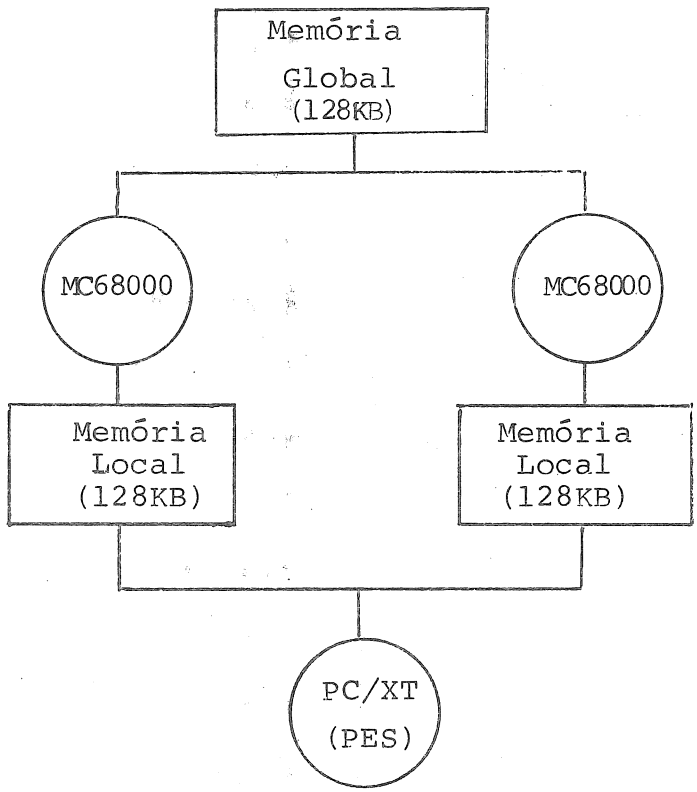


Figura 1. Arquitetura da máquina MICRO-BIS

O PES realiza suas funções de entrada e saída através do acesso às memórias locais dos microprocessadores MC68000. O PES pode interromper a ambos os microprocessadores e esses podem interromper ao PES. Também, os microprocessadores podem interromper-se mutuamente.

As características básicas dessa máquina, assim como a organização do sistema Pascal Concorrente e do sistema de entrada e saída, já foram apresentadas na XII Conferência Latino Americana de Informática /TOS 86/. O presente trabalho é uma complementação daquele.

2. JUSTIFICATIVA DA ARQUITETURA DA MÁQUINA

Faz parte do projeto MICRO-BIS, além da construção da máquina bimicroprocessadora e da implementação do sistema Pascal Concorrente, também a implementação de um sistema operacional multiusuário e a implementação de um sistema Lisp concorrente. Estes dois últimos sistemas justificam plenamente a arquitetura escolhida para a máquina MICRO-BIS, conforme será mostrado nesta seção.

O sistema Lisp concorrente, denominado COLISP /GEY 86/, é composto por dois módulos concorrentes: um simulador de uma máquina virtual para a interpretação de uma variante da linguagem Lisp e um coletor de lixo. A figura 2 mostra a interação desses módulos através do espaço de memória compartilhado por eles. Esses módulos devem ser executados em processadores diferentes por questões de eficiência. A máquina MICRO-BIS representa, portanto, uma resposta adequada a essa exigência.

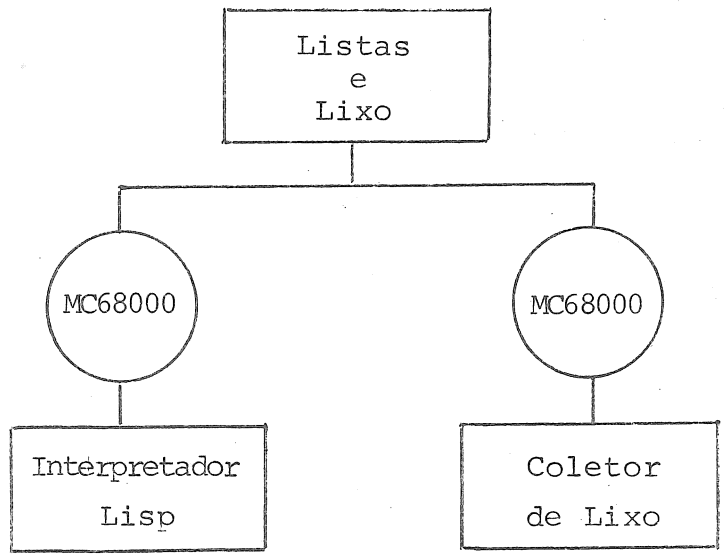


Figura 2. O sistema Lisp concorrente

Por outro lado, a arquitetura é conveniente para a execução de programas concorrentes cujos processos se intercomuniquem através de monitores. Este é o caso de várias linguagens de programação, entre as quais encontra-se o Pascal Concorrente /HAN 77/. Nessas linguagens, os processos executam a maior parte do tempo as suas próprias instruções e somente quando eles precisam se intercomunicar (trocar informações ou acessar dados compartilhados) é que eles executam nos monitores. A figura 3 mostra como ficariam distribuídos os componentes de um programa Pascal Concorrente na máquina MICRO-BIS.

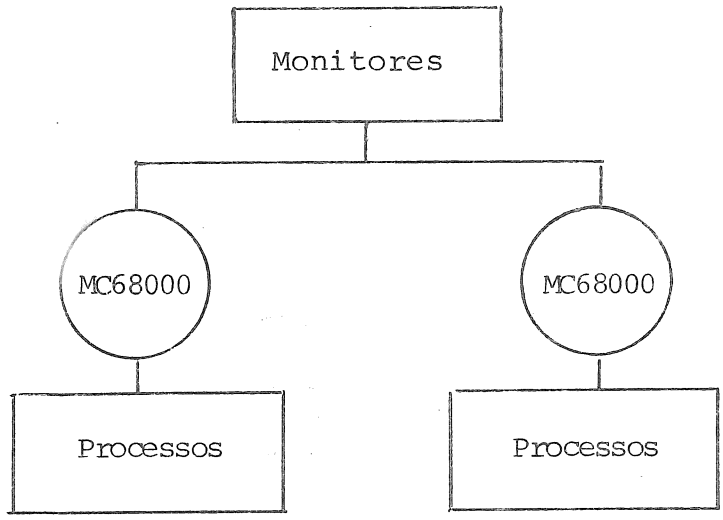


Figura 3. Distribuição dos componentes de um programa Pascal Concorrente

Na realidade, ficam nas memórias locais dos micro processadores cópias do segmento comum aos processos (núcleo, interpretador e código virtual) e na memória global, as estruturas que implementam os monitores e os dados compartilhados. O segmento privativo de um processo é alocado na memória local do microprocessador que for executá-lo. Mais detalhes sobre a distribuição desses componentes na máquina MICRO-BIS podem ser encontrados em /TOS 86/.

O sistema operacional multiusuário para a máquina MICRO-BIS, denominado SOMBRA /OLI 87/, foi escrito em Pascal Concorrente e tem as seguintes características principais:

- multiusuário interativo;
- linguagem de comandos com construções de alto nível;
- sistema de arquivos organizado em árvore;
- acesso controlado ao sistema de arquivos;
- execução de programas em background;
- spooling de saída para impressora;
- controle de acesso ao sistema através de senhas.

3. CHAMADAS DO SISTEMA

Como se sabe, os compiladores do Sistema Pascal Concorrente geram código para uma máquina virtual, a qual é formada por um interpretador e um núcleo.

Um programa de usuário somente pode ser executado quando ele é carregado para a área privativa de algum processo. Quando o programa necessita de alguma função do núcleo, o mesmo invoca um procedimento do processo no qual ele está encapsulado, sendo esse processo o que efetivamente executa a chamada.

Existem, portanto, três entidades envolvidas numa chamada de sistema: o programa de usuário, o processo e a rotina do núcleo.

Os processadores MC68000 da máquina MICRO-BIS possuem dois modos de operação: modo supervisor e modo usuário.

O programa de usuário e o processo são executados em modo usuário, o núcleo em modo supervisor. A mudança de modo usuário para supervisor é feita automaticamente pela instrução TRAP e pelas interrupções do hardware.

A instrução TRAP possui como operando um número entre Ø e 15 que identifica a entrada no vetor de tratamento de exceções (vetor de interrupções) que contém o endereço da rotina do núcleo que deve ser executada.

A implementação das chamadas do sistema foi feita da seguinte forma:

Um programa seqüencial em execução tem seu código executado pelo interpretador da máquina virtual. Eventualmente, o interpretador encontra a invocação de um procedimento do processo e passa então a executá-lo. Esse procedimento pode conter a chamada de uma função do núcleo que, ao ser identificada, ocasiona a execução da instrução TRAP. Com isso, o modo de operação do processador passa para supervisor e a execução prossegue no endereço contido no vetor de tratamento de exceções.

Existem 12 funções do núcleo que podem ser chamadas pelos processos, cada uma implementada por uma rotina específica. Como o vetor de tratamento de exceções do processador possui 16 entradas para TRAPS, sobram ainda quatro entradas desse vetor.

A última instrução de uma rotina do núcleo deve ser RTE (RETURN FROM EXCEPTION). Essa instrução restaura o contexto anterior à chamada da função do núcleo. Deve ser observado que após a execução dessa instrução o processador poderá ficar no modo supervisor ou no modo usuário. Ficará no modo usuário se a chamada do sistema tiver sido feita por um processo.

A passagem dos parâmetros necessários para a execução das diferentes funções do núcleo é feita através de uma área de memória comum ao interpretador e ao núcleo.

4. TRATAMENTO DAS INTERRUPÇÕES

As fontes de interrupção de hardware da máquina MICRO-BIS são três: interrupção de relógio, a cada 3,2 ms, interrupção do outro processador e interrupção do processador de entrada e saída (PES). O PES também pode ser interrompido pelos processadores principais. Para cada interrupção existe uma rotina específica de tratamento.

As interrupções de relógio são usadas para fazer escalonamento de processos e para implementar contagem de tempo. A cada cinco interrupções é atualizada a tabela de contagem de tempo e verificado, em uma fila, se o tempo de espera de algum processo já se esgotou. Em caso afirmativo, o processo é inserido na fila de processos aptos de acordo com sua prioridade. Após isso, é chamada a rotina que faz o escalonamento. O processo selecionado pelo scheduler é o de maior prioridade, podendo ser, inclusive, o que estava executando no momento da interrupção.

Um processador interrompe o outro através de um registrador de comandos programável. O processador interrompido desvia para uma rotina que acessa uma estrutura de dados, na memória global, usada para troca de informações.

De maneira análoga um processador interrompe o PES. Esse, quando interrompido desvia para uma rotina que acessa uma estrutura de dados na memória local do processador que o interrompeu, a qual contém a fila de requisições de entrada e saída.

Finalmente, o PES pode interromper a qualquer um dos dois processadores principais, o que de fato faz ao término de uma operação de entrada e saída. Um processador principal, ao ser interrompido, pelo PES, desvia para uma rotina que acessa a estrutura de dados em sua memória local, a qual contém o resultado da operação. Se a operação foi bem sucedida, o processo que a tinha solicitado é inserido na fila de aptos, de acordo com sua prioridade.

5. EXCLUSÃO MÚTUA NO ACESSO AOS DADOS COMPARTILHADOS

Conforme já foi visto, na máquina MICRO-BIS os dados compartilhados ficam na memória global. Existem três estruturas de dados compartilhadas nessa memória: um buffer que serve para troca de mensagens entre os processadores, um contador de requisições de entrada e saída, usado para que o atendimento das requisições seja feito na ordem da solicitação, e uma tabela com os registros descritores dos processos. O registro descritor de um processo contém, entre outras coisas, a identificação do processador que executa esse processo.

O acesso a essas estruturas deve ser feito de forma exclusiva pelos processadores. Para o controle do acesso

foram utilizados três semáforos: um para a tabela de descritores de processos, outro para o buffer de troca de mensagens e o terceiro para o contador de requisições de entrada e saída.

Existe ainda uma área destinada à conter 25 outros semáforos, os quais são utilizados para controlar o acesso aos monitores que, como se sabe, são os mecanismos usados pelos processos para sincronização e troca de mensagens.

A implementação dos semáforos é feita com a instrução TAS (Test And Set). Esta instrução permite testar e modificar o valor do operando de forma indivisível (em uma única operação).

Para obter acesso a uma estrutura de dados compartilhada, o processador executa a operação P. A liberação do acesso é feita através da operação V. Supondo um semáforo S inicializado com 1, as operações P e V são implementadas da seguinte maneira:

```
P:   TAS   S
     BNE   P
     RTS

V:   MOVE  $1, S
     RTS
```

6. SISTEMA DE ENTRADA E SAÍDA

A organização e o funcionamento do sistema de entrada e saída já foram apresentados na XII Conferência Latino Americana de Informática /TOS 86/. Entretanto, houve uma pequena mudança em relação ao que foi lá apresentado. Essa mudança foi a troca do PES, motivada pela intenção de dar maior portabilidade para o sistema multiprocessador. O papel do PES passou a ser desempenhado por um microprocomputador PC, mais universal que o minicomputador ED-281 previsto originalmente. Assim, a placa com o sistema multiprocessador (2 microprocessadores MC68000 e 3 módulos de memória) poderá ser instalada facilmente em qualquer microcomputador PC, no futuro.

7. INICIALIZAÇÃO DO SISTEMA

O processo de inicialização do sistema acontece em duas fases distintas. Na primeira delas, o PES carrega, a partir do byte 1024 de cada memória privativa, o núcleo do sistema operacional (o núcleo é carregado logo após o vetor de interrupções que ocupa os primeiros 1024 bytes). Carrega ainda, na entrada do vetor de interrupções correspondente à interrupção do PES, o valor do stack pointer do sistema e

o endereço da rotina de inicialização (valor inicial do PC) correspondente ao processador. A seguir, através de seu registrador de comandos, o PES executa a operação RESET nos processadores principais.

A partir daí inicia a segunda fase do processo de inicialização. Cada processador acessa seu vetor de interrupções para buscar o valor do stack pointer e do PC e iniciar sua execução.

A rotina inicial de um dos processadores inicializa o restante do seu vetor de interrupções (colocando o endereço da rotina de tratamento de cada TRAP e de cada interrupção), inicializa os dados globais e, por último, faz a chamada da rotina do núcleo que cria o processo ancestral. A partir daí, o processo ancestral, ganha o processador e passa a ser executado.

Simultaneamente, o outro processador executa uma rotina que inicializa o restante do seu vetor de interrupções e executa a instrução STOP. Esta instrução faz com que o processador fique em estado de espera até ser interrompido pelo outro processador.

O processo ancestral é o responsável pela criação dos demais processos do sistema. As informações que indicam quais processos devem ser executados por cada processador estão contidas em uma tabela. No momento em que o processo ancestral identifica um processo que deve rodar no outro processador, ele coloca todas as informações necessárias para a criação desse processo no buffer de comunicação dos processadores e interrompe o outro processador.

A rotina de tratamento dessa interrupção acessa o buffer de comunicação e, de posse das informações alí existentes, chama o procedimento do núcleo que faz a inicialização de um processo.

Passada a fase de inicialização, cada processador terá a seu encargo a execução de um conjunto de processos.

8. CONCLUSÃO

O projeto MICRO-BIS deverá terminar ainda este ano. O hardware da máquina multiprocessadora foi concluído e testado no ED-281 (antigo PES) e atualmente a placa está sendo adaptada para ser ligada ao microcomputador PC/XT (novo PES).

O sistema Lisp concorrente /GEY 86/ foi todo escrito em C, tendo sido construído um programa Pascal Turbo para fazer a carga dos códigos do interpretador e do coletor de lixo nas memórias locais dos processadores MC68000 (códigos de máquina para o MC68000 gerados pelo compilador C). O sistema foi desenvolvido e testado num computador

ED-680, esperando-se que com pouco esforço o mesmo possa ser colocado em funcionamento na máquina MICRO-BIS.

O sistema operacional SOMBRA /OLI 87/ foi todo escrito em Pascal Concorrente, tendo sido testado no minicomputador LABO 8034, que dispõe de um sistema Pascal Concorrente /MED 81/. A próxima etapa será a transferência e o teste do SOMBRA na máquina MICRO-BIS.

O núcleo do sistema operacional descrito neste trabalho foi programado na linguagem assembly do MC68000 e está em fase de depuração. O sistema de entrada e saída a ser executado pelo PES está sendo desenvolvido como dissertação de mestrado por um aluno do CPGCC, devendo ser concluído até o final do corrente ano. Este sistema deverá permitir a execução simultânea de várias operações de entrada e saída (funcionamento simutâneo de vários periféricos).

Espera-se que em um novo projeto os pesquisadores dos grupos de arquitetura de computadores e sistemas operacionais venham a se envolver na construção de uma máquina mais poderosa, com um número maior de processadores, o que permitirá o desenvolvimento de outras pesquisas em arquiteturas multiprocessadoras e em novos softwares para essas arquiteturas.

9. BIBLIOGRAFIA

/GEY 86/ GEYER, C.F.R. Implementação de um interpretador para uma linguagem funcional com coletor de lixo concorrente. Porto Alegre, CPGCC/UFRGS, 1986. (Dissertação de mestrado).

/HAN 77/ HANSEN, P.B. The architecture of concurrent programs. Englewood Cliffs, N.J. Prentice-Hall, 1977.

/HAN 78/ HANSEN, P.B. Multiprocessor architectures for concurrent programs. In: ACM 78, Washington, D.C., Dec. 1978. Proceedings. New Yor, ACM, 1978. p.317-323.

/HOA 74/ HOARE, C.A.R. Monitors: an operating systems structuring concept. ACM, 17(10):549-557, Oct. 1974.

/MED 81/ MEDEIROS, G.C.R. Implementação do sistema PASCAL CONCORRENTE no Labo 8034. Porto Alegre, CPGCC da UFRGS, 1981. (Dissertação de mestrado).

/OLI 87/ OLIVEIRA, R.S. SOMBRA: um sistema operacional multiusuário. Porto Alegre, CPGCC/UFRGS, 1986. (Dissertação de mestrado).

/TOS 82/ TOSCANI, S.S. et alii. Multimicro Pioneiro: protótipo de sistema multiprocessador orientado para execução de programas Pascal Concorrente. In: CONGRESSO NACIONAL DE INFORMÁTICA, 15., Rio de Janeiro, 1982. Anais. Rio de Janeiro, SUCESU/ /RJ, 1982.

/TOS 86/ TOSCANI, S.S. et alii. Implementação do sistema Pascal Concorrente na máquina MICRO-BIS. In: CONFERÊNCIA LATINO AMERICANA DE INFORMÁTICA - PANEL' 86, 12., Montevidéu, 1986. Anais. Montevidéu, Uruguai, Grafiservice S.R.L., 1986.