MULTIWARE PLATFORM: SOME ISSUES ABOUT THE MIDDLEWARE LAYER

Edmundo Roberto Mauro Madeira
Unicamp-IMECC-DCC
13081-970 Campinas SP - Brazil
e-mail: edmundo@dcc.unicamp.br

Relatório Técnico DCC–95-25

Dezembro de 1995

MULTIWARE PLATFORM: SOME ISSUES ABOUT THE MIDDLEWARE LAYER

Edmundo Roberto Mauro Madeira
Unicamp-IMECC-DCC
13081-970 Campinas SP - Brazil
e-mail: edmundo@dcc.unicamp.br

Abstract

This paper presents the conceptual and implementation models to the Middleware Layer of the Multiware Platform that is under development at UNICAMP - University of Campinas, Brazil. This platform combines ideas from both the RM-ODP (Reference Model for Open Distributed Processing) and existent products, like ANSAware and CORBA (Common Object Request Broker Architecture). The platform offers functionalities to support and facilitate the development, use and management of cooperative applications, like group decision support systems for GIS (Geographical Information Systems).

Keywords: Open Distributed Processing, Middleware, CORBA, GIS, Network Communication.

1 INTRODUCTION

The advances in the communication technology with high transmission rates together with the future (and present) needs of the users for integration and cooperative work stimulated the development of open service environments. The standards proposed by the ISO (International Organization for Standardization) for Open Distributed Processing (ODP)[1] consider this need of development.

Several consortia of companies are working in these environments, although not exactly adopting the concepts of RM-ODP/ISO (Reference Model for ODP). The Open Software Foundation (OSF) produced the DCE (Distributed Computing Environment) [2], and the Object Management Group (OMG) consortium produced the CORBA (Common Object Request Broker Architecture) specifications [3].

The development of the Multiware Platform aims to combine in a unique platform the support for cooperative work in an Open Services Environment. To reach this purpose, the Multiware Platform was designed considering a stratified composition of commercial products and functional blocks developed within the project. An example of application to this platform is the GIS(Geographical Information System)-based group decision support system, where decision makers need to generate, evaluate and illustrate alternative scenarios to come to a consensus [4].

This paper only analyses the framework and the implementation of part of the Middleware layer and is organized as follows: in section 2, an overview of the Multiware Platform is presented; section 3 analyses both the framework for adding ODP functionalities upon commercial platforms for distributed processing (ANSAware and CORBA), and the specific implementation objects which are necessary to add openness to CORBA; and section 4 closes the paper with the conclusion.

2 MULTIWARE PLATFORM

The Multiware platform is composed of three layers: Basic Software/Hardware, Middleware and Groupware (Figure 1)[5].

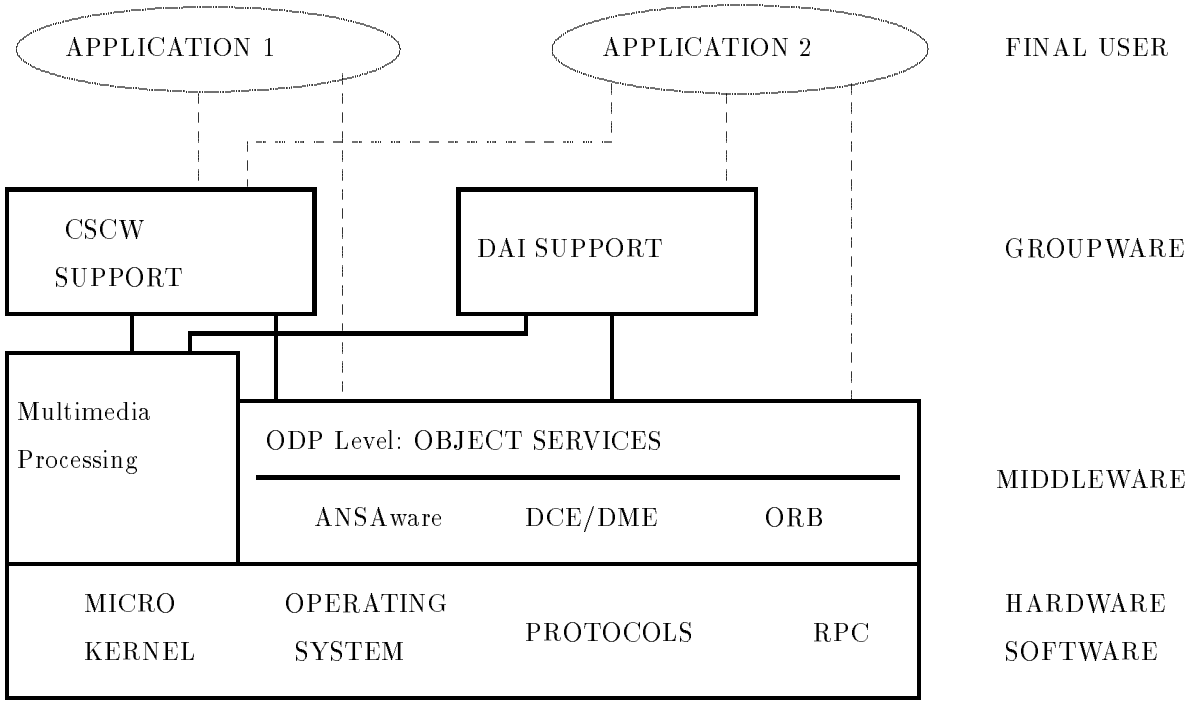

Figure 1 - MULTIWARE PLATFORM

The Basic Software/Hardware is composed of an operating system (eventually built above a microkernel), communication protocols, and so on. This layer provides no distributed system support.

The Middleware layer is responsible for providing distributed processing facilities to the Groupware layer and to the applications. The Middleware layer is composed of two sublayers:

- Multimedia Processing sublayer: allows the exchange of real-time multimedia information with a specified quality of service;

- ODP sublayer: is composed of two levels:
  - Commercial Distributed Systems: like ANSAware, ORB, and DCE;
  - ODP-level: aggregates ODP functionalities to the commercial distributed systems.

The Groupware layer provides the functionalities demanded by different classes of application, like CSCW (Computer Supported Cooperative Work)(for example, GIS-based group decision support systems), Distributed Artificial Intelligence (DAI), among others. Typical services supported by this layer are: dialog management, interaction protocols and handling of multimedia documents. Currently only the CSCW support is considered.

The testbed is composed of multimedia-workstations interconnected by an FDDI network.

3 MIDDLEWARE LAYER: THE ODP SUBLAYER

The ODP Sublayer, in the Middleware layer, is responsible for providing the infrastructure for open distributed processing to the applications, independently of the underlying basic software.

3.1 The Framework of the ODP Sublayer

The proposed model for this sublayer (Figure 2) has two levels: a lower level composed of commercially available distributed systems (ANSA, DCE and ORB) and an upper level that offers ODP services defined in the RM-ODP/ISO [5].

The ODP Level is composed of three sublevels:

- ODP Management sublevel;
- Transparency/Security sublevel;
- ODP Support sublevel.

Using the concepts of the Engineering Viewpoint from the ODP specification, the first sublevel offers the basic management services, allowing the use of BEOs - Basic Engineering Objects (that constitute the ODP applications), Clusters (units of activation, deactivation and migration) and Capsules (units of resource allocation) [6]. Each node (unit of resource independence that defines a resource management domain) has a nucleus and can support one or more capsules, while a capsule can have one or more clusters. A cluster is composed of BEOs that communicate among themselves directly or through channels (BEOs belonging to different clusters).

A distributed application is located in several capsules in different nodes. The application part that belongs to a capsule can be split in clusters depending on whether these subparts can migrate, be deactivated and be reactivated individually. A cluster encloses the BEOs that are the processes (and/or threads) of the application.

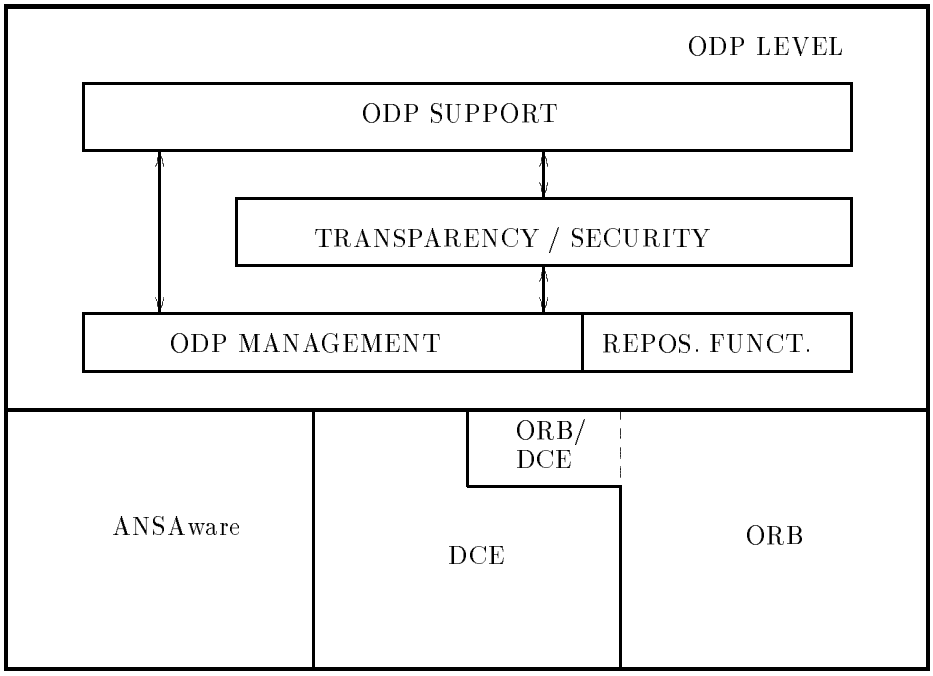

Figure 2 - ODP SUBLAYER

The ODP Management is supported by the repository functions defined in the ODP specification. The second sublevel offers the transparencies and the security functions of the ODP specification, and the last sublevel provides the ODP functionalities to the applications. The functional blocks of the ODP support sublevel are:

- Application Support: provides the basic functionalities of an open service platform, such as the definition and instantiation of the objects (BEOs) (processes) that compose the applications and how these objects are structured in subsystems, the definition of desirable transparency and security requirements, and the establishment of the necessary interactions (binding) among these objects;

- Trading: offers the service negotiation between servers (exporters) and clients (importers). Examples of services are: to export a service, to search for a service type, to select the best service according to parameters, among others;

- Group Support: provides cooperation support among members of a group, as for instance, the transmission of a client invocation to the appropriate server members of the group, and the guarantee of sending invocations to group members in a determined order;

- Transaction Support: ensures that a transactional operation invocation has the required ACID properties.

3.2 The Implementation of the ODP Sublayer

In the current implementation, CORBA is used as the basic infrastructure. In the ORB architecture, a client requests an operation of an object implementation through a stub (if the client knows the object implementation in the compilation time) or through a dynamic invocation interface (if the client only knows the object implementation in the execution time). The ORB has all the mechanisms to find the object implementation and to interact with it. The ORB provides the access, location and relocation transparencies.

Object services, common facilities and application objects can be put upon an ORB. The object services constitute a set of services (interfaces and objects) that provides the basic functions to use and to implement objects. The common facilities form a set of services that offers useful functions to several applications, and the application objects are specific objects of the end user application.

The reasons for the choice of the ORB are: ORB is a simple platform, it covers the main concepts proposed by the ODP specification, and it allows adding of new objects with ODP functionalities.

The object services offer only some ODP functionalities and support only some ODP transparencies [7]. In this project, the ODP functionalities and transparencies obtained from the object services are initially used, and these functionalities and transparencies are completed with new objects that are being developed in the project, to compose the ODP Level.

Some objects are defined to implement the Management and Transparency Sublevels. Initially the ODP Level has the access, location, and relocation transparencies and some management and repository functions (Figure 3).

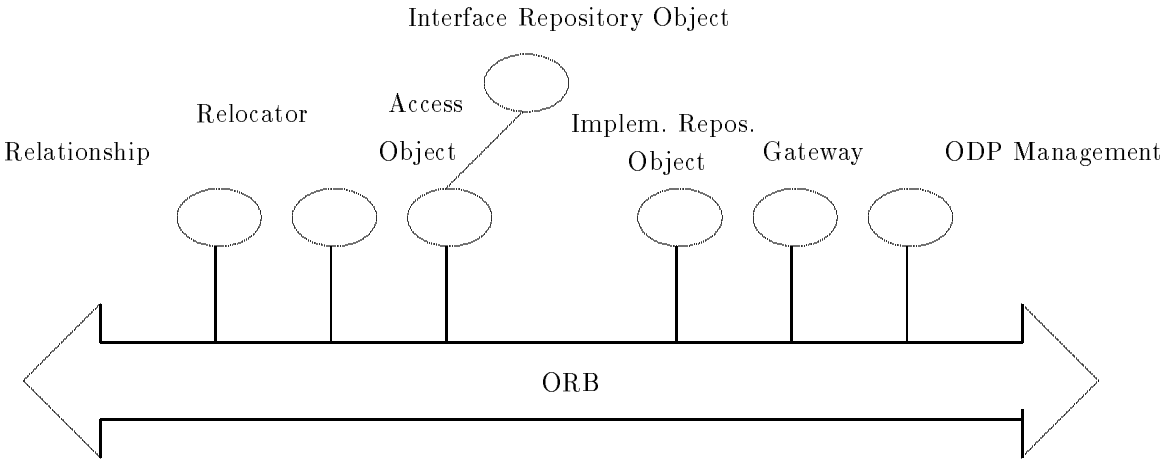

Figure 3 - Objects belonging to the Management

and Transparency Sublevels

Interface Repository:

The interface repository makes a consistent store of interfaces definitions. The ORB can access the interface repository by two ways. In the first way, the ORB obtains access by the stub routines, and in the second way by the DII (Dynamic Invocation Interface).

To reduce the problems of access to the interface repository, a framework of access to the interface repository was created. This scheme has two objects (access and repository), two data bases (files) (access file and interface repository), and a memory cache. The goal of the access object is to direct the requests from ORB to the correct interface repository, searching for additional information about the location where the required information is stored. The repository object has the function to get the information from the interface repository. The access file has all information about identifiers of interfaces types, addresses and identifiers of interfaces repositories. This file enables the access object to obtain the information required in the repository object. The memory cache has all information about the scope of one specific machine in the ORB environment.

The advantages using this framework of access to the interface repository are: the framework has one access object for each ORB environment, concentrating all requests from the ORB to send to the correct repository object; the framework can support more than one repository object in the ORB environment; and the framework uses a memory cache to reduce the time to search the information in the interface repository.

Implementation Repository:

The Implementation Repository contains the necessary information that enables the ORB to locate and to activate the object implementations.

Interoperability:

Interoperability as defined in [8] is the ability for a client on ORB A to invoke an OMG IDL (Interface Description Language) defined operation on an object on ORB B, where ORB A and ORB B are independently developed. Interoperability thus extends distribution transparency (currently access and location) to multiple ORB environments and the interoperability, by itself, is to be transparent.

In order to pass a request between ORB domains, it is necessary to translate the Object Reference (which uniquely identifies a requested object within an ORB and has an ORB-specific implementation) to a form that can be understood by the target ORB.

The idea is to use a Gateway, placed between the ORBs, that will translate the object reference of one ORB for a neutral representation and to other ORB's object reference representation. A table can make the mapping between both object references. Each ORB has to provide the gateway with the object references that will be available to be used by another ORB.

It is also necessary to introduce some way to deliver requests from an ORB to an Object Implementation (that implements proxy objects that use the translated object reference) because their types are not known at compilation time. We will achieve this through extending CORBA to support the dynamic interpretation of the requests [4].

Although, in order to get it, both ORBs need to agree on the types of their interfaces and the operation identifiers. This issue can be solved by providing some kind of agreement between their Interface Repositories (initially the new types will be assumed as being registered in the same form in both ORB's repositories). To deal with operations a new object is introduced to map the operations between ORBs.

Relocation:

Firstly, it is important to emphasize that our relocation concept deals only with the moving of objects when they are not interacting.

In order to provide this transparency we introduce a Relocator. This object will be able to store the new address from moving objects. It will consist in a table mapping old to new addresses and being used when the searched object is not found on the original address.

Dealing with the same ORB, this ORB does not have to worry about its relationships. Being necessary to cross ORBs' domains the gateways services are introduced to store relocated objects addresses (it will be very useful later for the garbage collection). At this situation, it is possible to map directly the original address from the object to the gateway address on the Relocator. The request will be switched for the Relocator and then for the gateway that will do the redirection to the other ORB.

When there are different ORBs, the problem of the relocated object that does not know how to interact with the old relationships, that have stayed on the original ORB, emerges. The proposed solution was to use the object called Relationship which will contain for each ORB object all the other objects related. After that, it is enough to make the related objects available at the gateway.

ODP Management:

The executable unit of the prototype is the basic engineering object that is mapped in an Unix process. The basic modules of the prototype are the follows: the Node Management is provided by the nucleus, that performs the creation of capsules, and by the interface reference manager that provides an unique identifier to the objects of the model in a set of nodes; the Capsule Manager carries out the creation, reactivation and recuperation of the clusters, and performs checkpoint, deactivation and termination of the capsules; and the Cluster Manager performs the creation of the basic engineering objects and provides checkpoint, deactivation and termination of the clusters.

In addition, one module (part of the nucleus) is implemented to handle the interface references to the object identifiers, allowing the establishment of the dynamic channels among objects.

Other objects are defined to implement the ODP Support Sublevel (Figure 4). The Application Support Object performs the high level functions of the application support block discussed in the section 3.1. This object requests services from the presented objects to carry out its high level functions. The Transaction Support object is not specified yet.

Group Support:

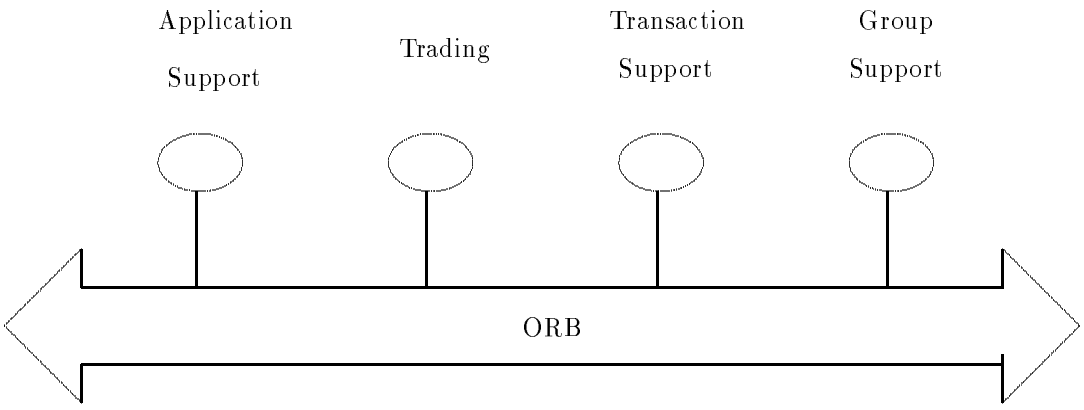

Figure 4 - Objects belonging to the ODP Sublayer

In the proposed model, groups are closed, in the sense that only members of the group can participate in the interactions within it. Interactions between group members takes place in a way such that each member can send messages to a specified subset of the group members, not necessarily to everyone. Each group member has associated a set of roles that represent the functions the member plays in the context of the group application. Roles are used for addressing and authorization purposes. The behavior of a group as a whole is prescribed by a set of group policies that impose rules to the execution of each group service.

The implementation of the group support services is structured in such a way that almost all functionalities are distributed across the system. Only functions that require global coordination are placed in a centralized element, called the group coordinator. The distributed functions are replicated in objects called local group objects that are located near each group member for efficiency. For each group, there is one and only one active coordinator, while the local group objects are associated with all groups the user is a member. Interactions between group members are mediated by these objects. When global ordering or any sort of global coordination is required, the interaction goes through the group coordinator. When there are real time requirements such as in multimedia information exchange, interactions takes place through communication channels, which are established and controlled by the group support, but provided by the Multimedia Processing sublayer. The services of the Group Support object are: Group Creation: makes a new group based on a list of potential members (which are consulted for participation) and their roles. A set of policies is associated with the group; Group Finishing: terminates the specified group, using a mechanism determined by the finishing mode; Join Group: integrates a new user to the group, giving it the needed context information to interact in the group; Leave Group: extracts the specified user from the group with a notification to the other group members; Message Distribution: multicasts a message to a specified subset of the group members. The messages may or may not be subject to the ordering of events in the group; Channel Creation and Maintenance: causes the establishment of a communication channel linking

a subset of the group members, allowing immediate information exchange between them; Policy Change: changes the group policy for the specified service, altering its behavior; Role Change: allows a user to change the role(s) of a given member of the group. The roles can be replaced, added or removed; Authorization Change: allows a user to change the set of authorizations related to the specified service. This can impose certain restrictions on the access of the service; Event Ordering: imposes a global order to the events within the group. It is an internal service, not present in the service interface; Response Collation: performs the collation of the various related responses caused by a previous multicast. The collation is made according to a specified mechanism and allows the user to receive a single response; and Failure Detection and Recovery: detects the failures of the underlying system that supports a group member or the group coordinator, and initiates a recovery action.

In the geographical applications, this object is fundamental to define groups of decision makers that observe, manipulate and evaluate alternative scenarios in the group decision support systems.

Trading:

In this project, a trader was built. This trader supports federated operations, besides the normal operations. Every trader has an administrator that decides about the creation of federations, and proposes and evaluates contracts. The operations of the prototype of this object are [9]: common operation interface: search, export, withdraw, list_offer_details; type repository interface: add_service_type, display_repository; context management interface: authorize, create_context, delete_context; federation contract interface: distribute_catalogue, request_catalogue, establish_federation, exchange_contract; and administrator interface: send-_catalogue, request_catalogue, evaluate_contract.

In the geographical applications, this object is important because it can register the services used by the decision makers to create possible scenarios.

There is another object, called Multimedia Support Object (related to the Multimedia Processing Sublayer) in the Middleware Layer that offers services both to the applications and to the ODP Sublayer. The Group Support Object requires resources (for example, multimedia channels) from this object.

4 CONCLUSION

In spite of the differences between ODP and ORB concepts, their combination in the Multiware Platform is very fruitful. The RM-ODP model is a framework for the conceptual phase of functional specifications, and the OMG-CORBA is already an implementation model with products in the market.

The development of a Trader and a basic protocol between the Trader and its Administrator (to establish federations with exchange of contracts that contain service offers) were very important to allow interworking in the open service environments.

The Groupware layer (for example, a GIS-based group decision support system) uses the communication functionalities offered by the group support object in the ODP Support

Sublevel. The proposed framework is useful to manage conflicting concepts involved in cooperative work within an open environment where the user autonomy is very important.

Acknowledgments:

The author wishes to thank Fábio M. Costa, Nuccio Zuquello, Luiz A.P. Lima, Cláudio M. Garcia, and Luiz Otvio B. Lento who are working (or worked) in this implementation, and Manuel J. Mendes and Waldomiro P.D.C. Loyolla for the helpful discussions about this work.

This work was in part supported by FAPESP (Grant 92/3507-0) and by CNPq (Project PROTEM GEOTEC 680061-94-0).

5 REFERENCES

[1] - ISO/IEC JTC1/S C21, Basic Reference Model ODP - Part 1: Overview and Guide to Use; Part 2: Descriptive Model; Part 3: Prescriptive Model - July 1994

[2] - OSF, Distributed Computing Environment, September 1990

[3] - Object Management Architecture Guide, Version 2.0, OMG TC Document 92.11.1, September 1993

[4] - M.P. Armstrong - "Requirements for the Development of GIS-Based Group Decision-Support Systems" - Journal of the American Society for Information Science, October 1994, pp. 669-677

[5] - W.P.D.C. Loyolla; E.R.M. Madeira; M.J. Mendes; E. Cardozo and M.F. Magalhães - "Multiware Platform: an Open Distributed Environment for Multimedia Cooperative Applications" - IEEE COMPSAC'94, 18th Annual International Computer Software and Applications Conference, Taipei, Taiwan, November 1994

[6] - M.J. Mendes; W.P.D.C. Loyolla and E.R.M. Madeira - "DEMOS: A Distributed Decision Making Open Support System" - 4th Workshop On Future Trends of Distributed Computing Systems do IEEE Computer Society, Lisbon September 1993, pp. 208-214

[7] - Object Services Request for Proposal 1/2, OMG TC Documents 92.8.6 and 93.6.1 Oct/1992, Nov/1993

[8] - Object Management Group: ORB 2.0 Interoperability and Initialization Request for Proposals, September 15,1993, OMG TC Document 93.9.15.

[9] - L.A.P. Lima and E.R.M. Madeira - "A Model for a Federative Trader", International Conference on Open Distributed Processing ICODP'95, Brisbane, Australia, February 1995, pp. 155-166

Relatórios Técnicos – 1992

92-01 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

92-02 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

92-03 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

92-04 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

92-05 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

92-06 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

92-07 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

92-08 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

92-09 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

92-10 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

92-11 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

92-12 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

93-01 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

93-02 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

93-03 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

93-04 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

93-05 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

93-06 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

93-07 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

93-08 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

93-09 **Codificação de Seqüências de Imagens com Quantização Vetorial,** *Carlos Antonio Reinaldo Costa, Paulo Lício de Geus*

93-10 **Minimização do Consumo de Energia em um Sistema para Aquisição de Dados Controlado por Microcomputador,** *Paulo Cesar Centoducatte, Nelson Castro Machado*

93-11 **An Implementation Structure for RM-OSI/ISO Transaction Processing Application Contexts,** *Flávio Morais de Assis Silva, Edmundo Roberto Mauro Madeira*

93-12 **Boole's conditions of possible experience and reasoning under uncertainty,** *Pierre Hansen, Brigitte Jaumard, Marcus Poggi de Aragão*

93-13 **Modelling Geographic Information Systems using an Object Oriented Framework,** *Fatima Pires, Claudia Bauzer Medeiros, Ardemiris Barros Silva*

93-14 **Managing Time in Object-Oriented Databases,** *Lincoln M. Oliveira, Claudia Bauzer Medeiros*

93-15 **Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures,** *Nabor das Chagas Mendonça, Ricardo de Oliveira Anido*

93-16 $\mathcal{LL}$ – **An Object Oriented Library Language Reference Manual,** *Tomasz Kowaltowski, Evandro Bacarin*

93-17 **Metodologias para Conversão de Esquemas em Sistemas de Bancos de Dados Heterogêneos,** *Ronaldo Lopes de Oliveira, Geovane Cayres Magalhães*

93-18 **Rule Application in GIS – a Case Study,** *Claudia Bauzer Medeiros, Geovane Cayres Magalhães*

93-19 **Modelamento, Simulação e Síntese com VHDL,** *Carlos Geraldo Krüger e Mário Lúcio Côrtes*

93-20 **Reflections on Using Statecharts to Capture Human-Computer Interface Behaviour,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-21 **Applications of Finite Automata in Debugging Natural Language Vocabularies,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-22 **Minimization of Binary Automata,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-23 **Rethinking the DNA Fragment Assembly Problem,** *João Meidanis*

93-24 EGOLib — **Uma Biblioteca Orientada a Objetos Gráficos,** *Eduardo Aguiar Patrocínio, Pedro Jussieu de Rezende*

93-25 **Compreensão de Algoritmos através de Ambientes Dedicados a Animação,** *Rackel Valadares Amorim, Pedro Jussieu de Rezende*

93-26 GeoLab: **An Environment for Development of Algorithms in Computational Geometry,** *Pedro Jussieu de Rezende, Welson R. Jacometti*

93-27 **A Unified Characterization of Chordal, Interval, Indifference and Other Classes of Graphs,** *João Meidanis*

93-28 **Programming Dialogue Control of User Interfaces Using Statecharts,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-29 EGOLib – **Manual de Referência,** *Eduardo Aguiar Patrocínio e Pedro Jussieu de Rezende*

Relatórios Técnicos – 1994

94-01 **A Statechart Engine to Support Implementations of Complex Behaviour,** *Fábio Nogueira de Lucena, Hans K. E. Liesenberg*

94-02 **Incorporação do Tempo em um SGBD Orientado a Objetos,** *Ângelo Roncalli Alencar Brayner, Claudia Bauzer Medeiros*

94-03 **O Algoritmo KMP através de Autômatos,** *Marcus Vinícius A. Andrade e Cláudio L. Lucchesi*

94-04 **On Edge-Colouring Indifference Graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

94-05 **Using Versions in GIS,** *Claudia Bauzer Medeiros and Geneviève Jomier*

94-06 **Times Assíncronos: Uma Nova Técnica para o Flow Shop Problem,** *Hélvio Pereira Peixoto e Pedro Sérgio de Souza*

94-07 **Interfaces Homem-Computador: Uma Primeira Introdução,** *Fábio Nogueira de Lucena e Hans K. E. Liesenberg*

94-08 **Reasoning about another agent through empathy,** *Jacques Wainer*

94-09 **A Prolog morphological analyser for Portuguese,** *Jacques Wainer, Alexandre Farcic*

94-10 **Introdução aos Estadogramas,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

94-11 **Matching Covered Graphs and Subdivisions of K_4 and $\overline{C_6}$,** *Marcelo H. de Carvalho and Cláudio L. Lucchesi*

94-12 **Uma Metodologia de Especificação de Times Assíncronos,** *Hélvio Pereira Peixoto, Pedro Sérgio de Souza*

Relatórios Técnicos – 1995

95-01 **Paradigmas de algoritmos na solução de problemas de busca multidimensional,** *Pedro J. de Rezende, Renato Fileto*

95-02 **Adaptive enumeration of implicit surfaces with affine arithmetic,** *Luiz Henrique de Figueiredo, Jorge Stolfi*

95-03 **W3 no Ensino de Graduação?,** *Hans Liesenberg*

95-04 **A greedy method for edge-colouring odd maximum degree doubly chordal graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

95-05 **Protocols for Maintaining Consistency of Replicated Data,** *Ricardo Anido, N. C. Mendonça*

95-06 **Guaranteeing Full Fault Coverage for UIO-Based Methods,** *Ricardo Anido and Ana Cavalli*

95-07 **Xchart-Based Complex Dialogue Development,** *Fábio Nogueira de Lucena, Hans K.E. Liesenberg*

95-08 **A Direct Manipulation User Interface for Querying Geographic Databases,** *Juliano Lopes de Oliveira, Claudia Bauzer Medeiros*

95-09 **Bases for the Matching Lattice of Matching Covered Graphs,** *Cláudio L. Lucchesi, Marcelo H. Carvalho*

95-10 **A Highly Reconfigurable Neighborhood Image Processor based on Functional Programming,** *Neucimar J. Leite, Marcelo A. de Barros*

95-11 **Processador de Vizinhança para Filtragem Morfológica,** *Ilka Marinho Barros, Roberto de Alencar Lotufo, Neucimar Jerônimo Leite*

95-12 **Modelos Computacionais para Processamento Digital de Imagens em Arquiteturas Paralelas,** *Neucimar Jerônimo Leite*

95-13 **Modelos de Computação Paralela e Projeto de Algoritmos,** *Ronaldo Parente de Menezes e João Carlos Setubal*

95-14 **Vertex Splitting and Tension-Free Layout,** *P. Eades, C. F. X. de Mendonça N.*

95-15 **NP-Hardness Results for Tension-Free Layout,** *C. F. X. de Mendonça N., P. Eades, C. L. Lucchesi, J. Meidanis*

95-16 **Agentes Replicantes e Algoritmos de Eco,** *Marcos J. C. Euzébio*

95-17 **Anais da II Oficina Nacional em Problemas Combinatórios: Teoria, Algoritmos e Aplicações,** *Editores: Marcus Vinicius S. Poggi de Aragão, Cid Carvalho de Souza*

95-18 **Asynchronous Teams: A Multi-Algorithm Approach for Solving Combinatorial Multiobjective Optimization Problems,** *Rosiane de Freitas Rodrigues, Pedro Sérgio de Souza*

95-19 **wxWindows: Uma Introdução,** *Carlos Neves Júnior, Tallys Hoover Yunes, Fábio Nogueira de Lucena, Hans Kurt E. Liesenberg*

95-20 **John von Neumann: Suas Contribuições à Computação,** *Tomasz Kowaltowski*

95-21 **A Linear Time Algorithm for Binary Phylogeny using PQ-Trees,** *J. Meidanis and E. G. Munuera*

95-22 **Text Structure Aiming at Machine Translation,** *Horacio Saggion and Ariadne Carvalho*

95-23 **Cálculo de la Estructura de un Texto en un Sistema de Procesamiento de Lenguaje Natural,** *Horacio Saggion and Ariadne Carvalho*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

`reltec@dcc.unicamp.br`