10

# Management of CORBA objects monitoring for the Multiware platform

*J. A. G. de Queiroz and E. R. M. Madeira*
*IC - Institute of Computing / UNICAMP - University of Campinas*
*13083-970 Campinas-SP Brazil*
*{aqueiroz,edmundo}@dcc.unicamp.br*

**Abstract**

This paper presents a distributed application monitoring system in the context of an integrated system management architecture for the Multiware platform and analyses its implementation. This architecture is based on the ODMA, on the CORBA, on the CORBAservices and on the emerging CORBAfacilities, where the target objects are the application components and the system support elements in an ODP environment. The first step in such direction was to instrument the CORBA clients and servers with sensors, enabling the monitoring. Such system was integrated to the network and system management tools, developed for the Multiware platform.

**Keywords**

Distributed System Management; Application Management; Monitoring; Instrumentation.

## 1 INTRODUCTION

The Multiware platform offers support for the distribution of applications and is being developed at the University of Campinas (Loyolla, 1994). A distributed system, as the Multiware environment, typically consists of a large number of different architecture hosts, connected by heterogeneous communication networks, various operating systems and many support services, establishing an infra-structure for developing and executing distributed applications. These applications consist of a set of co-operating software components spread over a distributed system. If the object-oriented paradigm is chosen to describe and implement this environment, the elements are objects, with well-defined interfaces, whose methods may be invoked from any node of a domain. In a client/server architecture, such objects reside in servers (processes that provide one or more services) whose interfaces are transparently called by client programs. The growth and complexity of distributed systems has made their management to become an important topic for researchers, designers, vendors and end users.

J. Rolia et al. (eds.), *Open Distributed Processing and Distributed Platforms*

The work in this area must cover all the components of distributed systems in order to understand the interactions and correlations among them. The processes that implement a distributed application should be viewed as managed objects.

In general, monitoring is part of the management that provides necessary information in order to build the model and present the observed system. Dealing with distributed systems, the problems of such activity are increased by the distribution aspects. Furthermore, the notion of monitoring is directly opposed to the encapsulation paradigm that protects the object state and associated procedures from external observation. This paper describes a Monitoring System for the Common Object Request Broker Architecture (CORBA) applications for performance and accounting management purposes, in the context of integrated system management architecture for the Multiware platform. Although the CORBA objects have both distribution and encapsulation features, the system provides a simple and modular way to instrument and monitor them.

The next section presents the Reference Model for Open Distributed Processing (RM-ODP) (ITU-T/ISO, 1995a) as a distributed system management model, because it essentially is a distributed processing activity. Section 3 describes the integrated management architecture based on the Open Distributed Management Architecture (ODMA) (ITU-T/ISO, 1995b), on the CORBA (OMG, 1995c), on the Common Object Services Specification (CORBAservices) (OMG, 1995b) and on the Common Facilities Architecture (CORBAfacilities) (OMG, 1995a) adopted in the Multiware project, while in section 4 the monitoring system model is presented. In section 5 some implementation issues are addressed and results are presented. The related work is described in section 7 and some remarks and conclusions are presented in section 8.

## 2 RM-ODP AND DISTRIBUTED SYSTEM MANAGEMENT MODELS

The reliable use of a distributed system requires a suitable management of its components: the network resources, the support services and the application elements. In object oriented distributed environments, the allocation of the application objects in the nodes affects their services, the host system load and the communication subsystem. This situation becomes more critical with migration, replication and fault recovering services. In such scenario, the requirements of Integrated Distributed System Management must be satisfied (Hegering, 1994). The main purpose is to guarantee the desired behaviour of the distributed applications, monitoring and controlling the network elements, the support services and the components of such applications, in order to avoid the risks and associated damages with the distributed processing of the organization.

The management models and standards were developed to manage network components, whose management services were specified, designed and implemented at a different time as their normal functionality.The Simple Network Management Protocol (SNMP) model (Case, 1990) is applied to monitor and control Internet elements, like gateways and routers, through applications in management stations. Although the requirements for processing and memory are low, SNMP's polling and its trap function are not suitable for application management. The Open Systems Interconnection (OSI) management model (Yemini, 1993) provides a more high-level approach, using object oriented concepts like classes, polymorphism and inheritance. Although the Guidelines for Definitions of Managed Objects (GDMO) corresponds to the CORBA Interface Definition Language (CORBA IDL), it does not offer the

pre-compilation facility that generates interface stubs to marshal and to unmarshal parameters. SNMP and OSI management models do not recognize that the same model used to specify, design and implement distributed applications should be used for management. Recently, there has been an increase in the use of CORBA object model that motivated to map OSI/SNMP standards for CORBA through adapters, based on the fact that CORBA would become the management standard (Ban, 1995).

As the distributed management system essentially is a distributed processing activity, the RM-ODP must be used to model it. The reference model is a meta-standard that supports the distribution, inter-operability, transparency and portability of the distributed applications, using the object oriented paradigm. It addresses the distributed issues in an integrated way. Therefore, the management complexity may be analysed considering the following five viewpoints: enterprise, information, computational, engineering and technology, each of them representing a different abstraction of the process, leading towards a new top-down approach. This is a distinct way to model management processing and to move from network management dimension to integrated distributed system management. The ODMA is compliant with RM-ODP and will extend the OSI/ISO management model.

## 3 INTEGRATED DISTRIBUTED SYSTEM MANAGEMENT

The architecture described in this section is part of the Multiware platform which is being developed at the University of Campinas (Loyolla, 1994).

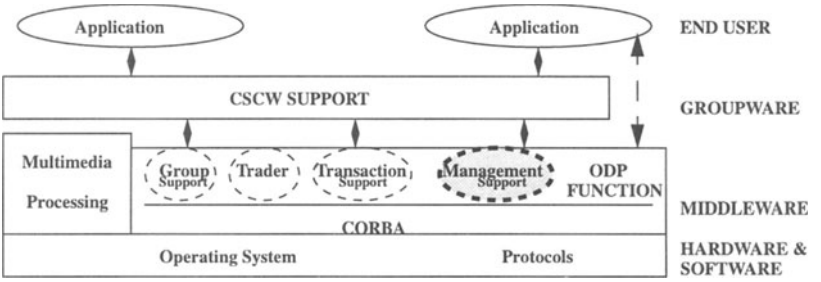


**Figure 1** Multiware Platform.

The Middleware layer of the Multiware is composed of an ORB and some ODP services, like the Trader (Lima, 1995), Transaction Support and Group Support (Costa, 1996). The Middleware is built on top of Iona Technologies Orbix. Over this layer, the Groupware layer supports different kinds of CSCW applications. Figure 1 illustrates the Multiware platform.

The adopted integrated management architecture is based on ODMA and it uses CORBA, the CORBAservices and the CORBAfacilities, providing at least as much functionality as the OSI/ISO management model. It is also influenced by the concepts described in (Sloman, 1993; Sloman, 1995; Bauer, 1994). The architecture must address the application domain, integrated with the underlying systems and networks; it must be amenable to monitoring, performing configuration actions and controlling the managed resources; it must provide a common and consistent user interface that simplifies the interaction; it must keep the associated overhead of

management as low as possible, avoiding expensive use of processing and memory; and it must support the interoperabilty and scalability requirements, common to distributed systems.

Such architecture is illustrated in Figure 2. There is not a single monolithic application, platform-centered, to perform all the issues of the management process. There is a set of cooperative applications with a common user interface which handle performance, accounting, configuration, fault and security management. They enable a logical centered view of the managed domain, independent of the physical distribution of its components. There is also a set of facilities for monitoring, control, configuration, policies and domain, used as small pieces of a *lego* for developing management applications. The support level, composed of the Multiware architecture over an ORB, the CORBAservices and CORBAfacilities, helps the development and implementation of the applications and the facilities. Moreover, this level enables managing and managed objects to be compiled in different languages, since there is a corresponded IDL mapping. The use of CORBAservices guarantees portability and reusability of code, so they allow developers to handle objects that are largely independent of the behaviour they inherit from their classes.

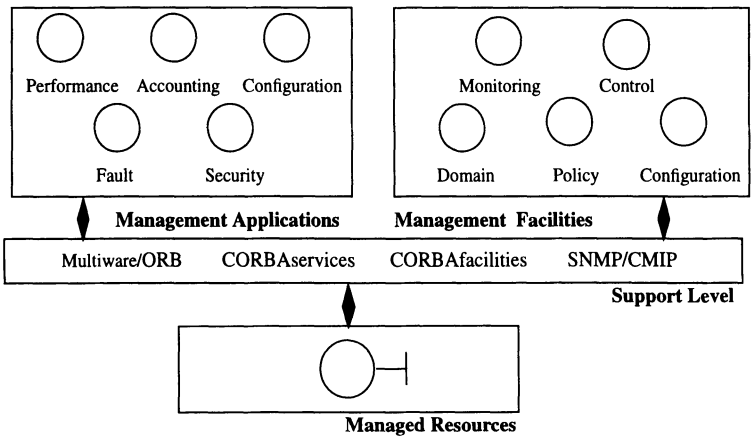


**Figure 2** Management Architecture.

The monitoring function consists of the observation of the activities from the distributed system components and of the collection of static and dynamic data by requests and notifications. The Monitoring Facility is defined as a dynamic process to acquire, collect and present management information from distributed system components, essential to management decision making. A filtering service is necessary for selecting collected data in advance. The proposed distributed object-oriented monitoring model includes the generation, processing, dissemination and presentation of monitored information (Bauer, 1994). The Event Management Service is used to provide asynchronous communication between managed objects and managing entities. Thresholds may be established to generate events, using the push or pull model through an event channel. A Java-applet is used to create dynamic, interactive Web presentations to display collected and filtered management information through graphics, tables and dialogues.

The Control Facility performs actions to avoid the functional degeneration of a distributed

system, controlling the behaviour of managed objects. Such situations are triggered by state information from a component, requiring reactive, preventive and pro-active actions. The conditions are usually detected by the monitoring facility. The main clients are the Configuration Facility and performance management applications. A very important aspect is to ensure that only actions allowed by the Policy Facility will be issued by authorized and authenticated users or systems.

The Configuration Facility is able to establish the initial configuration of a distributed system, to track the changes and to carry out modifications, when desirable. In large systems, human interventions can create errors and cause faults. Such facility may be a derived class of Life Cycle Service to create, delete, copy and migrate managed objects. An important aspect is the specification of which management operations a manager can carry out by permission, obligation and prohibition in order to guide the decision making process. There is thus a need to specify, represent and manipulate policies. The specification begins with abstract policies which are refined, after several interactions, into actions that can be interpreted by computer systems. The Policy Facility then provides operations to create and delete policies, to read and write policies attributes, to get the policies which refers to a domain and to determine the domains to which a policy is applied.

In an ODP environment, there are multiple management views and different responsibility limits (Yemini, 1993). Management must be structured to split responsibility and authorization amongst different managers (Sloman, 1995). Such structuring may reflect the physical network connectivity, a platform domain or the administration structure of an organization. The domains do not encapsulate their members. They are passive entities which hold references to their members, that are specified explicitly and not in terms of grouping criteria. The Domain Facility provides operations to read and write domain attributes; create and delete domains; list their membership; and insert or remove objects as members of domains.

The support level, composed of the Multiware platform which is over an Orb and the CORBAservices, allows the development of management applications, simplifying the implementation and guaranteeing portability. The Time service is desirable to provide an accurate and precise time in order to synchronize management activities and timestamp messages. The Naming service is employed to specify management interfaces for binding and resolve names. The Event Management service is conceived to decouple communication between supplier objects that generate events and consumer objects that receive and process them, supporting the push and pull model. Finally, the Life-Cycle service is used to enable the operations for creating, deleting, copying and moving objects.

## 4 MODELLING AND DESIGN OF THE MONITORING SYSTEM

This work is related to the performance and accounting management. It was motivated by questions that arise during a distributed system operation, concerning the reasons for an excessive response delay and for identifying the users who are responsible for the resources consumption in heterogeneous environment of the Multiware. Much of the data obtained from measurement and observation for performance purposes is also used for accounting goals. Our focus is on the operational interactions between objects in client/server roles.

A modular approach is necessary to design a flexible, generalized and efficient monitoring system. The proposed architecture, derived from an ODP model, is composed of generation,

observation, collection, logging, processing and presentation modules (Figure 3).

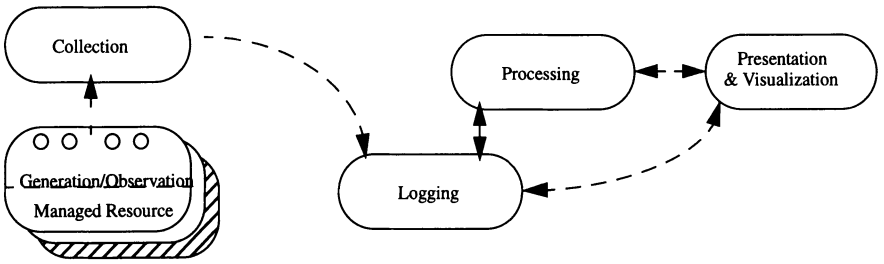


**Figure 3** Monitoring Model

The generation module is responsible for the gathering of raw monitoring data as a result of an important event. This activity is performed by instrumented sensors for a particular metric. The observation forwards monitoring messages from one or more sources to one or more destinations and provides an instrumentation control point. The collection module collects data from various instrumented managed objects to enable subsequent logging and processing. It is executed by collection processes, distributed over the managed domain. The logging module is responsible for the storing and recovery management of data, using the available Persistent Service. Processing is necessary to transform, combine, correlate, filter and analyse different management data. This activity depends on the kind of the functional area and the scope of the monitoring. Although there is a specific module processing, its functions are executed by all previous modules if possible. The components of the presentation and visualization module deal with human user interfaces. This module is integrated with other tools that manage some part of the distributed system. The GUI resources of Java and its Abstract Window Toolkit provide the required facilities. Java applets, embedded in a Web browser, allow the manager to navigate using the tools, according to the concept of Intranet applications. In general, an applet's execution requires methods for initialization, starting, painting, repainting, stopping and destroying graphics objects. The background drawing is composed of rectangles that represent the nodes, over which running objects are animated by circles. Components like simple dialogues display management information.

A management system requires a way to obtain values from the instrumented components of distributed applications. The data is generated by sensors, created within an address space, which are specialized objects incorporated into client/server processes to provide relevant metrics. Dynamic data are gathered by sensors, as counters and timers, instantiated to acquire a particular metric. Such sensors must be installed where significant event transition occurs. Another kind of sensor, a composer, is defined to return specific attributes such as to identify the managed resource, its operational state and to gather operating system data. The first step in the implementation phase was to investigate a way to enable the instrumentation of CORBA applications.

Figure 4 shows the static schema from the information viewpoint, used to specify the monitoring system that focuses on the meaning of the management information manipulated by and stored within the system, according to the graphical notation of Rumbaugh's Object Modelling Technique.

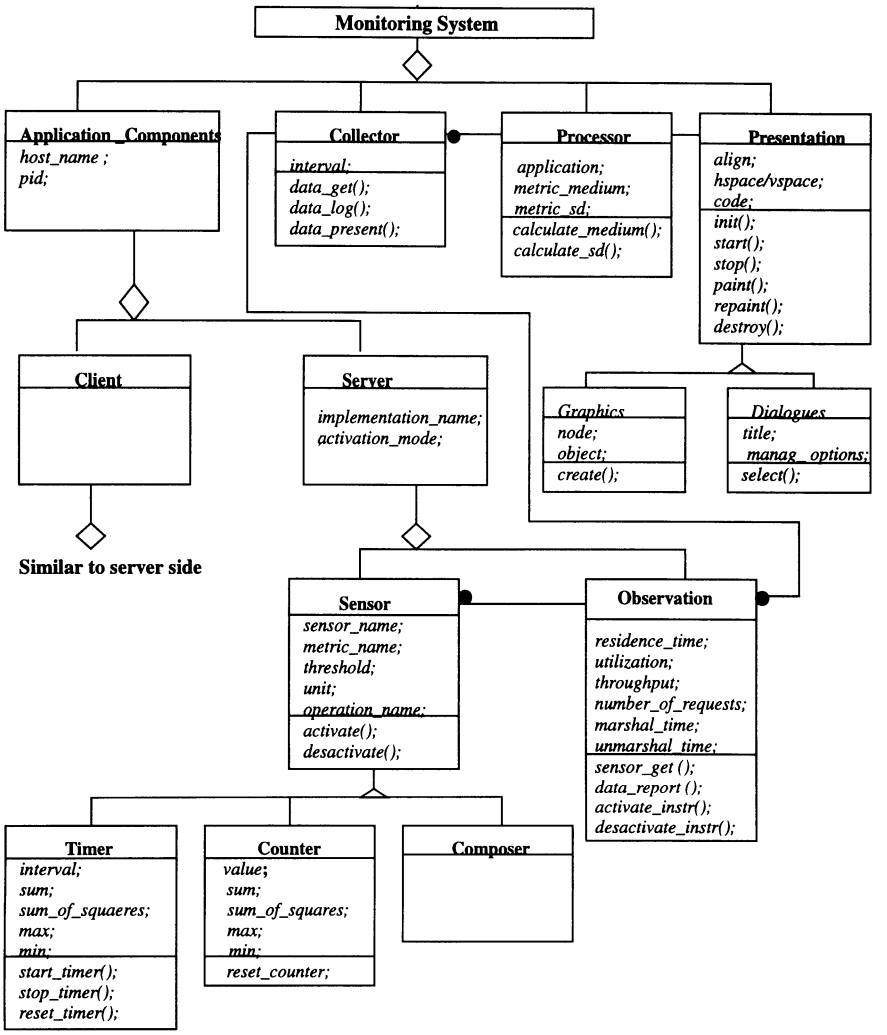


**Figure 4** Static Schema.

The main performance metrics chosen were response time, throughput and utilization (Rolia, 1993), translated to the CORBA environment. Such translation was based on the specification for monitoring DCE based applications presented in (Friedrich, 1995b), requiring counters, timers and composers. From a client perspective, the collected metrics are: number of requests sent, parameters unmarshalling time, number of responses received, marshalling time and response time. From a server perspective, they are number of requests received, parameters marshalling time, residence time, number of responses sent, unmarshalling time, utilization and throughput. The instrumentation also computes statistical quantities as

minimum and maximum values, the sum and the sum of squares to compute mean and variance during a time interval. The services provided by each application component must be monitored and characterized separately. Other data is also required for performance and accounting purposes. Such data is: operation name, server name, marker name, activation mode, host_name, pid, and operation caller user name. Table 1 presents which data is available in CORBA or in Orbix specifications. Unlike the operation *get_principal()*, they are available just in Orbix. The specified interface is shown in Annex A.

**Table 1**

| Data | Operation | Class |
|---|---|---|
| operation name | getOperation() | CORBA::Request |
| implementation name | myImplementationName() | CORBA::BOA |
| | myServer() | CORBA::ORB |
| marker name | myMarkerName() | CORBA::ORB |
| activation mode | myactivationmode() | CORBA::ORB |
| hostname | myHost() | CORBA::ORB |
| caller user name | get_principal() | CORBA::BOA |
| pid | getpid() | Unix system call |

## 5 PROTOTYPE IMPLEMENTATION

The Multiware platform has been implemented over IBM RS/6000 and Sun Sparc nodes, running AIX, SunOS and Solaris operating systems, connected by ethernet, fast ethernet and FDDI networks. This environment is dispersed in two labs, interconnected by a 10Mbits Internet link. Therefore, the monitoring system is portable to such heterogeneous systems.

The per-process filters, only available in Orbix, allow one to specify that additional code is to be executed before and after the normal code of an operation (IONA, 1995), monitoring all incoming and out-going requests/responses to and from a client or server address space. Such filters may be installed in chain. The elements of a chain perform the required sensor functions. Such feature, not available in the CORBA specification, allows the modular feature required by instrumentation. A filter on a client side can add extra data to an out-going request buffer, so that this is made available to the corresponding filter on the server side. Correspondingly, a server side filter can add data to an out-going reply.

The implemented sensors are derived from the superclass *CORBA::Filter,* allowing the instrumentation in two points of a request and in two other points of a response, before and after parameter marshalling and unmarshalling, on the server and client sides. We redefined some subset of the methods to carry out the required sensors operations. On the server side, just five sensors were instantiated, three as timers, one as a counter and the last one to return arbitrary data (Figure 5). One timer was installed to measure server's residence time and two more were instantiated to compute marshalling and unmarshalling times. Clock objects were designed and instantiated in order to provide a customized and low overhead time service. An incoming request starts the clock and an out-going reply stops it to measure server's residence time. Moreover, five sensors were installed on the client side: three timers, one counter and one composer. We estimate that the potential number of sensors can be much larger.

At first, a timer was instantiated on the client side just to calculate the response time between an out-going request and an incoming reply in order to measure the impact of the

inserted instrumentation. Such action was useful to evaluate the instrumentation overhead during all phases of implementation. We also noted that the largest response times are from requests to *orbixd* daemon servers. Such values are tied to static and dynamic factors of the node, the running operating system and the network. The heterogeneous and overloaded environment of the Multiware allowed to confirm such observation.

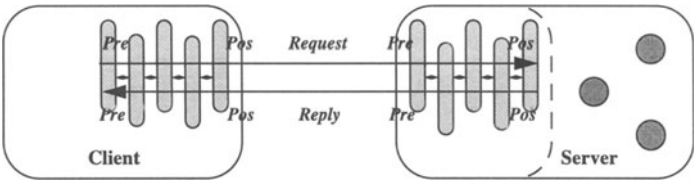


**Figure 5** Sensors Chain.

Besides the development, we found out the possibility of getting some management information, useful to other functional areas: data as port number, activation mode and marker for configuration or operation caller name for authentication, exploiting the modular feature of filters. The IDL interface attributes (Annex A) of an instrumented server show the essential data for our purposes, avoiding larger amounts of data. We integrated the monitoring system to network and system management tools in a Web browser through Java-applets, providing data about CPU load, disk and paging activities and workload of the nodes over which the components of CORBA applications are running. The heterogeneous Multiware environment allowed the monitoring of instrumented applications in different configurations. The integrated management concept was used to guarantee the specified quality of service and behaviour of the components of a distributed system.

The raw displayed data about a server is: its name, host_name, pid, utilization percentage, residence time mean (ms) and standard deviation, and throughput (number of requests per unit of time). Table 2 shows an example of sample data. The information is complemented with clients's response times which are compared with server's residence time. Additional data can be showed about the most invoked operation like the number of requests, the marshalling time and the medium unmarshalling time for a sample time.

**Table 2**

| Server Name | Hostname | pid | utilization | residence time mean | residence time sd | throughput | most invoked Operation |
|---|---|---|---|---|---|---|---|
| matrix | marumbi | 4483 | 0.11 | 0.32 | 0.11 | 7.8 | set |
| matrix | marumbi | 4483 | 0.13 | 0.35 | 0.10 | 8.0 | set |
| grid | trancoso | 1181 | 0.21 | 0.29 | 0.13 | 9.8 | get |
| grid | trancoso | 1181 | 0.20 | 0.30 | 0.12 | 8.0 | get |

Table 3 displays the differences between the client's response times and the server's residence times. The mean and standard deviation of differences of response times of a complete request/reply is shown. Furthermore, each of these conditions is shown both with the client and server on the same machine, on different machines and on different labs. The server's residence time is just one component of client's response time. The channel loading, distributed through its objects, degrades the client's response time but it does not affect the

residence time. For instance, on different machines, the mean and standard deviation values of client's response time were 5,24 ms and 0,81, while the values of server's residence time were 0,36 ms and 0,12 respectively.

**Table 3**

| localization: | same machine | different machines | different LAN's |
|---|---|---|---|
| Mean | 4.18 | 5.01 | 8.53 |
| Stand. Deviation | 1.09 | 1.15 | 3.39 |

## 6 RELATED WORK

The ODP management approach is also presented in different reported researches (ITU-T/ISO, 1995b; Farooqui, 1995). Friedrich et all (1995a) described an architecture and a prototype for distributed application performance monitoring, based on Distributed Computing Environment (DCE). Although it provides information that helps to understand the operating behaviour of DCE, such data is also useful for management purposes. (Schade, 1996) proposed the design of a general method to integrate a distributed application into a management environment, based on a Management IDL, an extension to CORBA IDL, and an instrumentation library to support the development of manageable application components. (Uslander, 1996) also proposed an architecture for enabling management of CORBA-based applications and integrating them with the standardized approaches of System and Network Management. Our work has presented a modular way to instrument CORBA objects and to monitor them in the context of the integrated management architecture for an ODP platform.

## 7 CONCLUSION

This paper presented an ODP monitoring system over CORBA, using the available mechanisms, that allows a relatively simple and straightforward way to instrument CORBA applications components. It described a generalized monitoring object model, discussed some implementation issues and showed some results. The results observed from applications over a heterogeneous environment, like the Multiware, lead us to the need of a full integrated distributed system management. Such data provides management information for performance and accounting functional areas and the prototype implements a Management Information Base defined in IDL and accessible by CORBA objects. The experience gained from the prototype implementation over CORBA showed that standardization efforts must be also concentrated in management. The majority of required operations and mechanisms are just available in Orbix. They are not part of the CORBA specification, but X/Open SysMan has submitted an OMG RFC (X/Open, 1995) to provide an approach for developing distributed system management applications.

The goal of the distributed system management architecture described here is to provide a framework and a set of services useful to the management of applications based on CORBA, in order to achieve a truly integration of the network, system and application aspects. The components of the architecture are ordinary CORBA objects and can be accessible through ORB operation requests as management protocol. RM-ODP provides a generic framework for

distributed applications, and it is becoming capable of providing the basis for their management, a starting point for an integrated distributed management. Nevertheless, the standardization currently is more focused more on distributed system design and implementation than on integrated management. The experience gained from OSI/ISO management must be used for management in ODP architecture; nevertheless its modelling technique is not so adequate for implementing distributed system management. ODMA standardization efforts are trying to fulfil this gap.

Acknowledgements: The authors wish to thank FAPESP and CNPq that have partially supported this work.

## 8 REFERENCES

Ban, B. (1995) Towards an Object-Oriented Framework for Multi-Domain Management. *IBM Zurich Research Laboratory*, december.

Bauer, M.A. et al (1994) Reference architecture for distributed systems management. *IBM Systems Journal vol 33, no 3*, pp 426-444.

Case J., Fedor M. and Davin J. (1990) A Simple Network Management Protocol (SNMP). *RFC 1157 Networking Group*, may.

Costa, F.M. and Madeira, E.R.M. (1996) An object group model and its implementation to support cooperative applications on CORBA. *Distributed Platforms (*Ed. A. Schill, C. Mistach, O. Spaniol and C. Popien), Chapman & Hall, pp 213-227.

Farooqui, K. (1995) OSI Management in the ODP Architectural Framework. *Proc. IFIP/IEEE DSOM'95*- pp 1-13, october.

Friedrich, R.; Martinika, J.; Sienknecht, T. and Saunders, S. (1995a) Integration of Performance and Modelling for Open Distributed Processing. *Proc. of ICODP'95,* pp 341-352, february.

Friedrich, R.; Saunders, S.; Zaidenweber, G.; Bachman, D. and Blumson, S. (1995b) Standardized Performance Instrumentation and Interface Specification for Monitoring DCE Based Applications. *OSF DCE RFC33.0,* may.

Hegering, H.G. and Abeck, S. (1994) Integrated Network and System Management. *Data Communication and Network Series.*Addison-Wesley.

ITU-T Rec X901/2/3 | ISO/IEC 10746-1/2/3 (1995a) ODP Reference Model. Part 1. Overview and Guide to use; Part 2. Foundations; Part 3. Architecture.

ITU-T | ISO/IEC (1995b) Open Distributed Management Architecture, Working Draft 3, november.

Iona Technologies Ltd. (1995) Orbix 1.3 advanced programmer's guide, release 1.3.1.

Lima, L.A.P. and Madeira, E.R.M. (1995) A model for a Federative Trader. *In Proc. of the ICODP'95*, pp 155-166, february.

Loyolla, W.; Madeira, E.R.M.; Cardozo, E.; Magalhães, M.F. and Mendes, M.J. (1994) Multiware Platform: An open distributed environment for multimedia cooperative applications. *IEEE COMPSAC'94*, november.

OMG (1995a) CORBAfacilities: Common Facilities Architecture, rev. 4.0, january.

OMG (1995b) CORBAservices: Common Object Services Specification, rev. march.

OMG (1995c) Common Object Request Broker: Architecture and Specification, rev 2.0, july.

Rolia, J.A. (1993) Distributed Application Performance, Metrics and Management. *Proc. of ICODP'93*, pp 205-216.

Schade, A.; Trommler, P. and Kaiserswerth, M. (1996) Object Instrumentation for Distributed Applications Management. *Distributed Platforms* (Ed. A. Schill, C. Mistach, O. Spaniol and C. Popien), Chapman & Hall, pp 173-185.

Sloman, M.; Magee, J.; Twidle, K. and Krammer, J. (1993) An Architecture for Managing Distributed Systems. *Proc. 4th IEEE Workshop on Future Trends of Distributed Computing Systems*, pp 40-46.

Sloman, M. (1995) Management Issues for Distributed Services. *Proc. IEEE SDNE'95*, pp 52-59, june.

Usländer, T. and Brunne, H. (1996) Management View upon CORBA Clients and Servers. *Proc. of ICDP'96*, Industrial and Poster Session, pp 165-169.

X/Open Company (1995) System Management: Common Management Facilities, Volume 1, Version 2 - OMG 95-12-05, december.

Yemini, Y. (1993) The OSI Network Management Model.*IEEE Communications Magazine*, pp 20-29, 31(5), may.

## 9 BIOGRAPHY

Edmundo R. M. Madeira is an assistant professor in the Institute of Computing at University of Campinas - UNICAMP, in Brazil. He received his PhD in Electrical Engineering at UNICAMP in 1991. He currently is a coordinator member of the Multiware Project.

João Augusto G. de Queiroz is pursuing his Master in Computer Science at UNICAMP. He works at Bureau of Brazilian Navy Telecommunication.

## A IDL INTERFACE

```
interface Server_Managed_Object {
    enum state {idle, unmarshalling_request, processing, marshalling_response, committing,};
    readonly attribute string operation_name;
    readonly attribute string implementation_name;
    readonly attribute string marker_name;
    readonly attribute string activation_mode;
    readonly attribute string hostname;
    readonly attribute string principal;
    readonly attribute long number_of_request;
    readonly attribute double unmarshalling_time;
    readonly attribute long number_of_response;
    readonly attribute double marshalling_time;
    readonly attribute double throughput;
    readonly attribute double utilization;
    readonly attribute double residence_time;}
```