Distributing Public Network Management Systems Using CORBA

Brian Kinane
Ericsson Applied Research Laboratory - Network Management
Broadcom Eireann Research Ltd.,
Kestrel House, Clanwilliam Pl., Dublin 2., Ireland.
Email: bkinane@broadcom.ie

Abstract

In the search for competitive advantage, network management systems are becoming increasingly important to public network operators. In an attempt to reduce the cost of management systems, the telecommunications community have, for many years, been working on the standardisation of interfaces between systems to encourage a multi-vendor environment. As the computing and telecommunications domains converge, there is now a possibility to use distributed object technology to increase the cost-effectiveness of management systems. This paper discusses experiences of using the Common Object Request Broker Architecture (CORBA) as a basis for a public telecommunications network management system platform.

Keyword Codes: C.2.4; D.1.5; D.1.1
Keywords: Distributed Systems; Object-oriented Programming; Applicative (Functional) Programming

1. INTRODUCTION

The International Telecommunications Union (ITU) addresses the management of telecommunications networks through their Telecommunications Management Network (TMN) M.3000 series of recommendations [1]. Although these address the interoperability of management systems, they do not directly provide methodologies or guidelines for actual implementation of systems. Issues such as the infrastructural requirements of management systems including management service implementation and deployment are not addressed by TMN standards.

As the telecommunication and computing domains converge, technologies based on concepts from Open Distributed Processing (ODP) [2] are being applied to the design and implementation of telecommunication management systems. These technologies support the implementation of management applications as distributed heterogeneous systems.

At present, an on-going study, RACE-II project PRISM R2041 [3], is using TMN methodologies in conjunction with the ODP enterprise, information and computational viewpoints for the specification of management systems. The next logical stage is implementation of these specifications using distributed object technologies.

This paper discusses experiences associated with constructing TMN management systems using the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) [4]. CORBA is a core component of the Object Management Architecture (OMA) - an architecture which reflects the computational and engineering viewpoints of ODP. The suitability of CORBA as the basis for a telecommunications management platform is evaluated through the implementation of a TMN-based security management application using Orbix [5] - a full implementation of CORBA.

CORBA is targeted at the implementation of systems using heterogeneous components but at present there are only C and C++ standardised language bindings available. As a result, use of CORBA implies use of C/C++. To explore use of other language paradigms with CORBA, Erlang [6], a declarative concurrent functional programming language (FPL) was integrated with CORBA through a partial language binding. It was used, in addition to C++, because of its high-level nature (i.e. it is more formal and closer to specification). The construction of management systems using C++ and Erlang components, integrated via an ORB, is discussed in this paper.

The paper consists of three sections. Firstly, a framework for integration of CORBA within TMN is identified. Based on the resulting framework, a case-study which uses the Orbix CORBA-implementation is described. Finally, the implementation is analysed and conclusions presented.

2. DEVELOPING MANAGEMENT SYSTEMS

While TMN guidelines are effective for specification of management systems and identification of points of distribution and interoperability between management applications, it does not provide a path which leads to implementation of the specification. It is suggested here that the CORBA may be used, in conjunction with TMN, as an implementation platform for TMN specifications.

2.1. TMN approach

A Telecommunications Management Network (TMN) provides monitoring and control of another network. The TMN may be separate or share facilities of the network it manages. According to the ITU-T recommendation X.700 [7] a management network should perform five functions - configuration, fault, security, accounting and performance management.

The TMN recommendations standardise some of the functionality and many of the interfaces of management software. This is intended to enable software from different vendors to interoperate within a TMN and to enable the exchange of management information between TMNs of different organisations. The reference architecture of a TMN is defined in the ITU base recommendation M.3010 [8] "Principles for a Telecommunications Management Network".

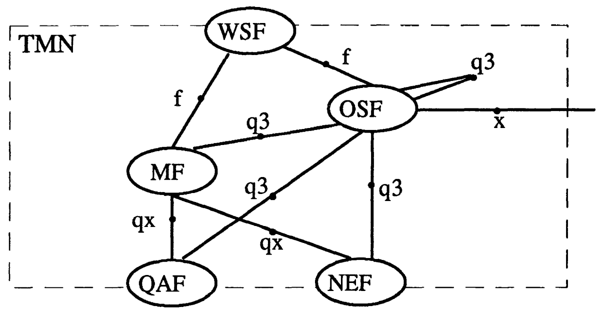

Figure 1 - TMN Functional Architecture showing Operations System, Mediation , Q-Adapter, Network Element, Work Station Function blocks and reference points.

The TMN functional architecture consists of a number of **function blocks** - the Operations System (OSF), Network Element (NEF), Q-Adaptor (QAF), Mediation (MF) and Work Station (WSF) function blocks (TMN functional architecture in figure 1) . Each of these blocks performs a particular function and cooperate to provide management services. Between each pair of communicating function blocks there exists a TMN **reference point.** The reference point (q3, qx, x, f in Figure 1) defines the 'service boundary between two management function blocks' and is a potential point of physical separation between systems (provided by different vendors or existing in different organisations).

Building blocks are actual systems which implement the above function blocks. Interconnection between these TMN building blocks is facilitated by a set of standard inter-operable **interfaces** . These interfaces are defined with respect to reference points and specify what information a building block should present and what means of communication it should use.

2.2. Distributing TMN systems

TMNs are physically distributed information processing software systems. Such systems are realised using heterogeneous technologies from different vendors and hence present substantial inter-operability problems. Specifying a management system using the TMN guidelines identifies function blocks which cooperate to provide particular services. According to TMN these blocks can physically exist separately. Support is provided to define standardised interfaces to these functions blocks in order to enable different function blocks to interoperate.

Building blocks can be further decomposed into functional components which are a set of simpler and more generic Management Software Components (MSCs). Little attention is given to defining interfaces, implementation and distribution of these components. It is contended, in this paper, that building blocks will be implemented as heterogeneous MSCs distributed over heterogeneous architectures. ODP addresses such distribution issues. By mapping these components to a model such as the ODP computational viewpoint, these logically distinct functional components can be defined and implemented, using ODP 'support environments' (e.g. CORBA, ANSAware), as physically distributed management software objects with well defined IDL interfaces. This enables a commodity environment where MSCs are re-used and combined to rapidly construct and customise management systems.

The Object Management Architecture (OMA) and specifically the Common Object Request Broker Architecture (CORBA) has the potential to be an implementation medium for TMN applications since it offers a location/technology transparent object distribution environment which overlaps with the computational, engineering and technology viewpoints.

2.3. CORBA

The goal of the Object Management Group is 'to develop a set of standard interfaces for inter-operable software components'. This is being accomplished through the OMG's Object Management Architecture (OMA) Reference Model which is a model for object management. In this model, application objects communicate with objects that provide common facilities and with low level object services through a communications infrastructure called the Object Request Broker (ORB). The ORB is a location transparent, distributed object platform and the Common Object Request Broker Architecture (CORBA) standard is a specification defining standard interfaces to the ORB.

2.3.1. Orbix

Orbix is a full implementation of the Object Management Group's Common Object Request Broker Architecture standard, developed by Iona Technologies Ltd. Orbix provides a C++ language binding for CORBA and is supported on SunSoft SunOS, Silicon Graphics IRIX, HP/UX and Windows NT. Orbix is being ported to Windows 3.1 and other Unix platforms. Orbix interoperates across all supported platforms.

2.4. Role of CORBA within TMN

In the TMN model, the distribution of management systems is at building block granularity. Each block provides particular services such as configuration or accounting. These services are accessed through an information model over the Common Management Information Service/Protocol (CMIS/P) [9]. The building block is considered a monolithic system which presents a TMN management interface.

CORBA extends this above concept of a building block. Instead of being monolithic, it is a set of cooperating objects distributed via CORBA. The TMN management interface is provided by the 'gatekeeper' object through which services offered by the building block are accessed (see Figure 2). Consequently, two layers of distribution exist;

- TMN distribution between systems belonging to different organisational domains or built by different vendors;
- CORBA within the building block inter-connecting both generic and specific MSCs which cooperate to provide the service offered by that building block.

The building block is a unit of vendor and domain inter-operability whereas CORBA provides component distribution and integration within the building block.

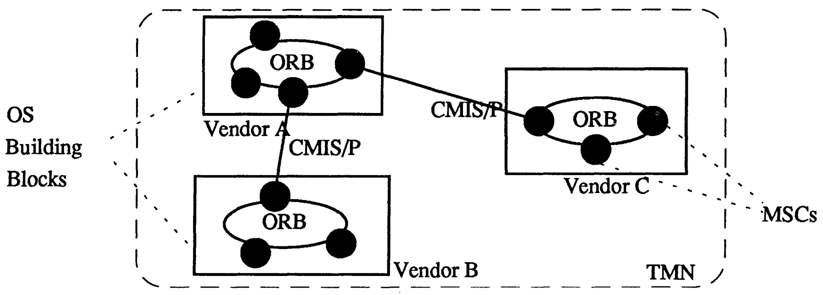

Figure 2 - TMN and CORBA Distribution

This approach raises a number of issues:

- Why not replace TMN distribution with CORBA?
- How does the TMN and the CORBA model interwork?
- How is the building block structured in terms of the OMA model?

TMN distribution is required as it is the most accepted mechanism for achieving shared management knowledge (SMK) between different systems. CMIP is standardised for the Q3 interface and is being used today. In addition, a large amount of work has been done in areas such as OSI security. CORBA, can however, support TMN distribution by providing a distribution and component integration platform for building blocks. It is likely that much of the new computing technology will be accessible over ORBs [10]. Using the ORB as a building block platform will enable the use of this technology in management applications.

In a CORBA-based system, OMA application objects interact in a client server model invoking methods from the operational interface of other application objects. Common Facilities (CF) objects and object services provide platform support to these management oriented application objects. For CORBA to interact with TMN, a mapping from CMIS requests on the management information model (MIM) to requests on server objects is required. This request initiates an activity (chain of operations on the MSC application objects) which upon completion will manipulate the TMN interface to cause a response to be sent to the manager application.

TMN describes the functional decomposition of building blocks in the form of functional components which include:

- **Message Communication Function** (MCF) which provides communication over a CMIS/P interface in either a manager or an agent role;
- **Management Information Base** (MIB) which contains the available management information;
- **Management Application Function** (MAF) which performs processing capability and implements the management services;

- **Presentation Function** (PF) and **Human Machine Adaptor** (HMA) which combine to transform and present management information in a human comprehensible format.

These functional components are implemented as application objects on the ORB platform. Building blocks should have the structure shown in Figure 3.

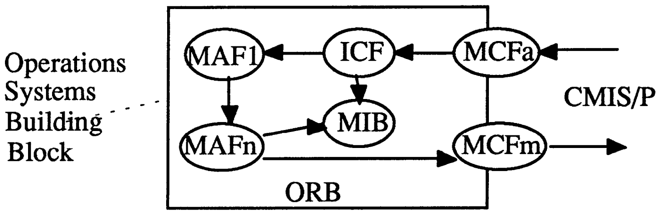

Figure 3 - Building Block as a set of objects

In this structure, an ORB object acts as an *MCF* (or gatekeeper) and in conjunction with the *MIB* object provides a TMN agent interface to the building block. An *MCFm* object is required for *MAF* objects to use services from other building blocks (manager role). When a request is received via the *MCFa* (agent role), it is transformed into an operation on a *MAF* object by the Information Conversion Function (ICF). This causes a chain of further requests (activity) on objects within the building block which upon completion will cause a response to propagate back to the *ICF* and be transformed to a TMN response. Both *MAF* objects and the *ICF* object can query and update the *MIB*. A specification process such as the PRISM methodology could be used to define the object interfaces and MIB specification which could also be used as the basis for the TMN Management Information Model.

In this model, the building block becomes a unit for deployment of objects which interact to provide a service. Certain attributes such as location, security and throughput are associated with the building block. Attributes such as throughput are requirements that must be met by the deployment environment (e.g. hardware, software). Others, such as security, relate to the interface of a building block. Since the building block is a closed set (objects are not directly accessible outside the building block), these attributes are determined by the TMN interface of the *MCFa* (gatekeeper) object.

2.5. Implementation with CORBA

As CORBA matures, standardised facilities such as Object Oriented (OO) database object adaptors will become available. These can be used to support MIB implementation. More language bindings such as Smalltalk will be standardised enabling a multi-language in addition to the present multi-hardware environment. At present, however, CORBA implementations offer only C and/or C++ standard bindings. C++ is a low-level language. As a result, programmers have to cope with low-level issues such as memory management instead of the logic of their application. While C++ can be used to develop robust applications, it is not, in the opinion of this author, an ideal language for implementation of management application functionality. This is partly due to its lack of built-in primitives for concurrency and fault-tolerance.

It was decided to address these issues by using C++ in conjunction with another language technology for implementation. Erlang, a functional programming language (FPL), was chosen because it provides comprehensive support for real-time concurrency and fault tolerance and because the declarative nature of FPLs reduces the gap between specification and implementation. CORBA purports to offer the capability of task driven language selection. This was investigated through the ORB-supported integration of Erlang and C++ components.

2.5.1. Erlang

Erlang is a concurrent, real-time, declarative, and functional language intended for the implementation of real-time industrial control systems. It was initially developed by Ericsson for the implementation of telecommunication systems.

Erlang provides a number of primitives which support real-time, fault tolerance, code modularisation and hot replacement of code. In addition, since Erlang is a functional programming language it has the property of variable single assignment. This has important ramifications for the application of formal methods to Erlang specifications and also code reliability. Function selection is made by pattern matching which leads to highly succinct code.

Erlang has a process based model of concurrency. Concurrency is an explicit and natural part of the language and encourages design of systems as numerous lightweight processes. Message passing between processes is asynchronous and is based on Communicating Sequential Processes. The use of processes overcomes a number of the problems associated with using functional languages.

Erlang provides a distribution mechanism based on TCP/IP which supports the location/ technology transparent distribution of processes. Application processes can be distributed over a heterogeneous network without affecting the semantics of the process interaction.

3. APPLICATION CASE STUDY

3.1. Scope

In order to evaluate the suitability of the above framework (i.e. ORB distribution within TMN and the resulting building block structure) a detailed case-study was conducted. This case-study consists of a TMN conformant security management application which provides management services that configure and monitor the security controls of a service layer management system. The demonstrator consists of the following:

- A Value Added Service Provider (VASP) Operations Systems (OS) building block providing end-to-end virtual leased lines to Customer Network Management (CNM) building blocks over an X interface;

- A Security Manager OS which provides security management services for the VASP OS's security mechanisms using Management Application Functions (MAFs). The Security Manager is implemented in Erlang and C++;
- A Security Manager WorkStation (WS) building block (implemented in Erlang) which provides a Graphical User Interface (GUI) to a human operator.

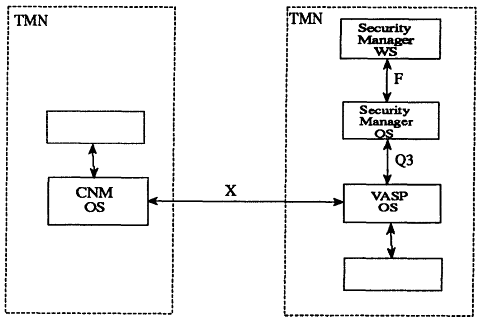

Figure 4 - Case-study scenario

Communication between the Security Manager OS and VASP OS is via the Q3 interface. The F interface enables communication between the Security Manager OS and WS.

3.2. Specification of the management application

Before any implementation could begin it was necessary to specify the requirements on the security manager. As TMN offers a framework for defining and specifying management functionality and their relationships, its guidelines were followed:

The services provided by the application are defined as follows:

- Visualisation and configuration of the access controls of the OS;
- Monitoring and reporting of access violation notifications emitted from the OS.

Once the services have been defined, the building blocks and the constituent functional components (which are needed to provide these services) are identified. The building block and functional component inter-relationships must also be specified. After this analysis phase, a reference configuration for this scenario was formed.

The chosen reference configuration for the Security Manager consists of two building blocks - an Operations Systems building block which provides the actual management services and a Work Station building block which enables the operator to interact with the provided services. These building blocks are composed of a number of functional components such as Message Communication Function (MCFs), Management Application Function (MAF) and Management Information Base (MIB) components.

Mapping functional components into objects

In this demonstrator the functional components were mapped directly into OMA application objects. These objects are grouped into packages based on the function block to which they belong. The relationship between the objects is client-server and the specific relationships are derived from the manager-agent roles used by TMN.

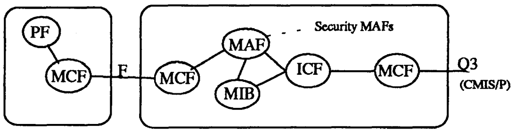

Figure 5 - Object specification for case-study management application

Applying this approach to the reference configuration the MIB, MCF and MAF functional components map to objects (see Figure 5). Where a function block is realised as a building block, the MCF application object provides the TMN management interface.

Defining the object interfaces

Once the application objects have been defined, interfaces for them can be specified. These interfaces are used by other objects to access the services offered by an object acting in a server role. The Object Management Group's Interface Definition Language (IDL) was used because it offers an object oriented abstract specification language and because all CORBA implementations support IDL. An IDL interface class was specified for each object - its methods defining services offered by that object.

3.3. Implementation of the object model

The transformation of the TMN management application specification into an object model (where each component has a clearly defined interface) enables the use of distributed object techniques to be applied to the implementation.

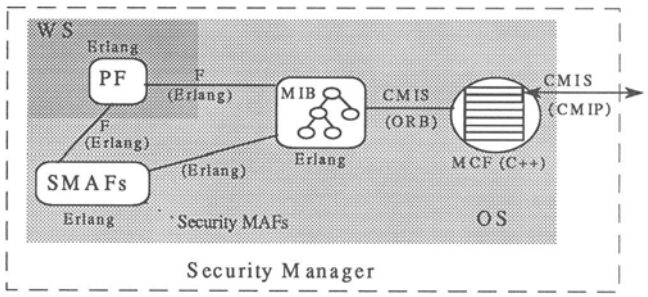

Figure 6 - Implementation Specification

In the implementation, which is shown in figure 6, each Management Software Component was implemented as either an Orbix object or an Erlang process interacting via a combination of Orbix and the Erlang distributed environments

The Presentation Function, Management Application Function and Management Information Base Functional Components were implemented as Erlang processes and the CMIS Message Communication Functional Component was implemented as a C++ object encapsulating a sourced C++ based ISODE CMIS/P stack [11] (This is further discussed in section 4).

3.4. Integration of the Erlang distributed process environment & CORBA

In order for Erlang to be used as part of the CORBA environment, a partial binding from Orbix/IDL to the Erlang environment had to be developed. This was possible due to the similarity of the models:

- Both models consist of active objects (processes in Erlang can be viewed as active objects);
- Communication between these active objects is via message passing (method invocation is a form of message passing) which can be either asynchronous or synchronous;
- Active objects present clearly defined interfaces.

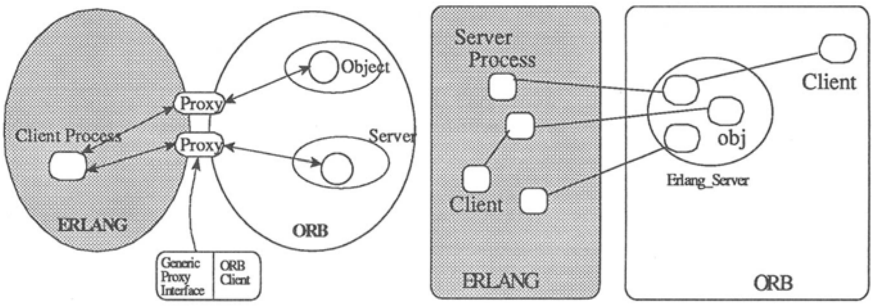

Figure 7- Representation (a) ORB objects to Erlang (b) Erlang processes to ORB

Representing resources in the ORB as native Erlang resources requires the mapping of objects which exist within the ORB environment into processes which exist within the Erlang environment. These processes act as Erlang proxies (i.e. a local entity which acts on behalf of a remote entity) for the remote objects (as shown in Figure 7(a)). Within the Erlang domain a proxy appears as a normal Erlang process. Simultaneously it appears as an ORB client in the ORB domain. When a process, acting as a proxy for a remote object within the ORB's domain, is created in Erlang, any messages sent to the process will be transformed into methods invoked upon the remote object. If the method returns a result then this result is mapped into an Erlang *term* (i.e. data structure) and is sent to the client process. The only ORB specific information an Erlang client needs is the name of the remote object which is required when spawning a proxy.

Access by non-Erlang clients to Erlang processes via the ORB is of equal importance. To achieve this, it is necessary to represent Erlang processes as objects in the ORB environment. Erlang processes make themselves available to ORB clients by registering as objects within the ORB domain as depicted in the figure 7(b). Any methods invoked on such an object will cause an equivalent message to be sent to the associated Erlang process. The return message, if any, will be sent to the ORB client application via the result of the method invocation.

4. ANALYSIS

Implementation of the demonstrator allowed a number of issues relating to the use of distributed environments in supporting TMN applications to be considered. These issues cover a range from management services development to implementation of management systems.

4.1. Developing application on a CORBA based platform

Any telecommunications platform should support development of management applications. By using CORBA, it was possible to use its support for object oriented software construction in management application development. IDL proved to be a very useful abstract language for defining interfaces to the Management Software Components (MSCs). It is independent of implementation. Once MSCs were associated with the interface, the ORB provided trouble free integration of these components. The ORB interface repository enabled the development of components to proceed without having to see the class hierarchy of the particular interface class being used. The main difficulty in using the ORB was providing a TMN (CMIS/P) interface to the application objects. This required mediation to convert requests on the management information models and requests on the application objects. Mediation was hardwired in the demonstrator (but more automatic mediation is being investigated by a joint Network Management (NM) Forum / OMG committee at present [12]).

Re-use of components is essential to rapid construction of systems and was well supported by the CORBA implementation used here. It is possible to bind to a required type of MSC using its interface name. The ORB locator will find an appropriate active object and will launch one if none are available. One feature lacking in Orbix was an interface repository browser. More tools such as browsers are required.

Flexibility is well supported through mechanisms like the Dynamic Invocation Interface. Objects can learn about new services and make use of them without recompilation. In addition, once an object is registered with the interface repository, it is available to all ORB clients in a black box (binary) format.

Support for new services can be achieved through the facility provided by CORBA of hot addition and replacement of MSCs. In this way a MSC that adds new functionality may be added at run-time. In addition, a new service may be developed by using different combinations of existing MSCs. Orbix allows numerous instances of an interface to exist. Because of its unique naming structure for objects, they can be easily distinguished.

Finally, CORBA enables the re-use of existing management tools and components such as the ISODE CMIS/P stack and manager/agent code through encapsulation with C++ wrapper classes. Re-use is not limited to management software. As new object services and third party Common Facility (CF) objects become available in the computing community, those telecommunications vendors using CORBA will have a more structured method for integration of new computing technologies and products into their platforms.

4.2. Orbix support for management platform requirements

Management applications impose a number of requirements such as distribution and fault tolerance, scalability, technology independence and security. Orbix, as a full implementation of CORBA, was evaluated against these requirements. The beta-version of Orbix, used in the demonstrator, provides the basis for the experiences outlined in the paper. Continued development of Orbix has since addressed a number of issues raised in this section.

Orbix provides a fully *distributed* location/technology independent platform. As it uses a XDR/TCP/IP communication protocol, full distribution via the internet is supported. Orbix also provides binding, brokering and limited trading services. This enables the platform to provide transparent distribution to management applications.

CORBA is clearly targeted at achieving *independence from the computing environment.* Orbix supports transparent access to objects over a variety of operating systems e.g. SunOS, Solaris, HP-UX, Windows NT. This is a very useful feature to management system vendors in reducing the costs of their management systems.

Controlled scalability is supported by the Orbix in the sense that services can be instantiated an arbitrary number of times. Extra support such as load-balancing objects is required to control when and where new instantiations should occur. In the demonstrator this is done manually by registering new object-servers on extra hosts. Vendors can, using a CORBA-based platform, scale the processing power (e.g. number of workstations) for their applications to suit the requirements of the customer's network.

Fault tolerance was not well supported by the ORB used in the demonstrator. Persistence for objects was not easily accomplished. The entrance of companies specialising in fault tolerance into the CORBA market (e.g. ISIS has since been integrated with Orbix) in conjunction with the OMG's efforts with object services should rectify this problem.

An important requirement of management platforms that Orbix lacked was *security* (e.g. authentication at binding). There are, however, other ORBs which provide object security as a value added feature. The OMG is looking at this and it will become a standardised object service. In the demonstrator, OSI-based security control was performed by the Message Communication Function agent object (which acts as a gatekeeper to the building block).

4.3. CORBA & Erlang

Erlang offers a fully distributed location/technology transparent process (active object) environment. It is not, however, open. There are no standardised interfaces for Erlang components, no standardised object model or standardised interface definition language, although the Erlang environment could potentially be used as a technology basis for a 'ODP support environment'.

From the perspective of this paper, Erlang as an adjunct (via an IDL to Erlang binding) offers a more effective method than C++ for implementing the control functionality of systems. This is, primarily, due to the declarative and functional paradigm to which Erlang conforms.

Using a functional approach, programs are implemented by specifying the relationships between the input set and output set of a function rather than describing imperatively how the function accomplishes its state transformation. With Function Programming Languages, data is not stored explicitly by the programmer. Instead, garbage collection is done by the environment. This form of environment has advantages for implementing MSCs.

- Due to the declarative nature of Erlang, the code required and development time required in this demonstrator to implement MSCs is less than C++ (e.g. Erlang provides automatic memory management);

- As a MSC implemented in Erlang is defined in terms of function clauses, the state of the MSC and the possible transitions are specified explicity and transparently. Consequently, the behaviour of the MSC is defined in a more formalised manner and is more tractable. In the demonstrator, Vienna Design Method (VDM) specifications for Directed Acyclic Graphs (DAG) were used for the Management Information Tree (MIT) operations.

It was found that the transparent integration of C++ and Erlang via CORBA was useful. Existing OSI/TMN software utilities (e.g. ISODE) could be encapsulated within C++ classes and made accessible over the ORB, in a manner analogous to a toolbox. Erlang was used for the complex logic of the Management Application Functions. Erlang was also used for the Management Information Base. In retrospect, C++ or an Object Oriented database would have been a superior solution due to Erlang's lack of support for inheritance (although there is work on-going at present at developing an OO extension to Erlang).

5. CONCLUDING REMARKS

The Object Management Architecture and specifically CORBA provides a viable platform for implementation of TMN conformant management systems. The benefits of this approach are two-fold. By using CORBA, management specification methodologies based on the synergy of ODP and TMN (e.g. PRISM) can be mapped more easily to physical implementation. This requires the convergence of the OMA with the computation and engineering viewpoints of ODP. Secondly, because CORBA is an accepted middleware standard, the use of CORBA as a telecommunications platform will support the incorporation of advanced (and out-sourced) computing technology (e.g. object oriented databases) into management systems. More work is required, however, on the integration of TMN specifications and the OMA object model.

The relationship between GDMO and the OMA object model (IDL), including mediation, is being addressed by a joint Network Management Forum / OMG task force at present.

New language bindings need to be developed for CORBA IDL to take advantage of languages that provide more formalised methods for specifying object behaviour. The use of Erlang in this project has shown the benefit of using declarative FPLs for implementing MSCs. The combination of Erlang, C++ and CORBA is a useful advance in the development of telecommunication management applications. We found that the use of Erlang significantly improved development productivity and code reliability of the Management Application Functions.

REFERENCES

[1] K. Shrewsbury, "TMN in a Nutshell", Network Management Forum, 1994

[2] ISO/IEC JTC 1/SC 21/N 7053 (CCITT X.901) Draft Recommendation, Basic Reference Model for ODP - Part 1: Overview and Guide to Use, December 1993

[3] RACE R2041 Prism Deliverable 8: Reports on Selected Areas of the Service Management Reference Configuration, September 1994

[4] Common Object Request Broker : Architecture and Specification OMG Document Number 91.8.1

[5] "Orbix - A Technical Overview", Iona Technologies Ltd., July 1993

[6] J. Armstrong et al., "Concurrent Programming in Erlang", Prentice Hall 1993

[7] X.700 Recommendation X.700 I ISO/IEC 7498-4 "OSI - Basic Reference Model - Part 4: Management Framework"

[8] ITU-T Recommendation M.3010, "Principles For A Telecommunications Management Network", November 1992

[9] ITU-T Recommendation X.710, "Common Management Information Service For CCITT Applications"

[10] J. Stikeleather, "Why Distributed Object Computing is Inevitable", Object Magazine, March-April, 1994

[11] G. Pavlou, "Implementing OSI Management - A Tutorial", Dept. of Computer Science, University city London

[12] C. Ashford, Comparison of the OMG and ISO/CCITT Object Models, The Report of the Joint NM Forum/OMG Taskforce on Object Modelling, April 1993