Implementing TMN using CORBA Object Distribution

O. Potonniée, L.H. Hauw, D. Ranc, Y. Bardout, Z. Canela
Alcatel Telecom RD
E-mails : {potonnie,hauw,ranc,bardout,canela}@aar.alcatel-alsthom.fr

Abstract

Telecommunication Management Networks can be considered as a set of distributed objects interacting to perform the network monitoring and control functions.

In this paper, we explore the way to a full object distribution based on the OMG CORBA principles. Some studies and the early results of the X/Open-NM Forum XoJIDM working group indicate that this is possible and promising for the future.

We analysed, designed and implemented a full TMN Operation System based on CORBA object distribution. One of our main priority has been to preserve the interoperability with existing TMN components based on CMIP, taking advantage of the existing assets of information models defined with GDMO, that we kept as the specification language for interfaces between components. Specialized CORBA services offering TMN specific services have been defined using CORBA Common Object Services whenever that was possible.

Keywords : TMN, OSI, CMIP, CORBA, TINA

1. INTRODUCTION

Existing TMN platforms are built on top of proprietary infrastructures (e.g. HP OpenView, Bull ISM, SUN Solstice...), that reduce the system openness, and entail an undesired dependence to a single supplier. Furthermore, future TMN will have to interact. Thus the trend in TMN evolution is the use of open standards : ODP, CORBA, and future TINA architecture. A proprietary system will either have to perform a costly adaptation, or lower its level of connectivity with external telecommunication actors, leading to poor competitiveness.

The introduction of CORBA object distribution mechanism [CORBA] in the TMN has been discussed for a couple of years. Early feasibility prototypes made by Alcatel Telecom RD and others have shown the technical possibility of introducing CORBA in TMN. X/Open and the NM Forum joint committee, XoJIDM, defined the translation [JIDM] from GDMO/ASN.1 to IDL, enabling the implementation of the Managed Object concepts upon CORBA/IDL. Alcatel Telecom RD launched a research case aiming at developing a prototype of a TMN OS using CORBA distribution to validate this approach.

Section 2 of this paper recalls basic concepts of TMN systems. Section 3 details how CORBA meets our requirements, while section 4 presents our global

architecture. This includes translation of existing specifications to IDL, and the use of three services reproducing CMISE services. Section 5 details how an existing Alcatel TMN component can be extended to ensure CORBA support.

2. THE EXISTING TMN ARCHITECTURE

The TMN is made of a number of building blocks, realizing management functions [M3010] . TMN is structured in management layers : Element Management Layer (EML) for managing Network Elements, Network Management Layer (NML), Service and Business Management Layers (SML and BML) that both offer higher level services related to customers.

2.1 TMN structure

Interactions in the TMN are structured in a threefold entity : The *network resource* is the physical or computational element managed, the *agent*, which builds a standardized view of the resource and the *manager*, which represents the user.
The set of information managed by an agent is called the Management Information Base (MIB), it is defined by a set of GDMO classes [X701] and ASN.1 types that are implemented in Managed Objects (MO). To act on a MO, the manager issues CMISE requests. In this document, we will use the shortcut *Managed Object* (MO) to denote a Managed Object Instance.

2.2 OSI management principles

The fundamental function within OSI systems management is the exchange of management information between two entities. It is referred to as the Common Management Information Service Element (CMISE) [X710], it includes :

- The interface to the user, specifying the service provided (M-GET, M-SET, M-ACTION, M-CREATE, M-DELETE, M-CANCEL-GET and M-EVENT-REPORT). This is the Common Management Information Service (CMIS) ;
- The protocol, specifying the protocol data unit (PDU) format and associated procedures. This is the Common Management Information Protocol (CMIP).

A management operation can be applied on multiple MOs. The selection involves two phases : scoping that defines a set of objects in the MIB, and filtering that specifies an assertion the objects must verify.

3. REQUIREMENTS FOR A CORBA-BASED TMN

TMN systems have three major requirements : extensibility, flexibility and openness. These requirements led to the M.3010 standard [M3010], which focused on Object Technology, Distribution of applications, and Standardized interface. Since CORBA is indeed a standard for object oriented distribution, it was considered as a valid candidate for middleware in TMN area.

3.1 Conform to standards

In order to enable interoperability with existing TMN components, and their integration, we kept OSI principles (agent-manager) and specifications language (GDMO - ASN.1). An important input that allows a mapping between CORBA and OSI comes from the JIDM Task Group, from X/Open and NM Forum [JIDM]. This group develops specifications defining an algorithm to translate managed object definitions in GDMO/ASN.1 into IDL. This enables the inter-working of management systems based on OSI with those based on OMG technology which should also facilitate integration between systems and network management.

The TINA-Consortium applies the principles of ODP [X901] and OMG standards to the needs of the telecommunication industry [TINA]. Our work can be seen as a migration step from existing OSI TMN to future TINA TMN.

3.2 CORBA object oriented distribution

The use of CORBA provides :

- Flexibility. Agent/manager communication through CMIP makes applications more flexible than monolithic construction; but this association is still rigid. CORBA uplifts modularity with the possibility of grouping objects in and out the processes, without modification to the source code. This allows a fine grained modularity.
- Language independence. CORBA IDL supports bindings for C, C++, Smalltalk and Java. IDL acts as a language bridge, allowing the choice of language for a component without impacting others.
- Independence from supplier. CORBA is an open technology with several suppliers, available on many platforms.
- Openness. Its generic standardized interfaces enable inter-operability with other software components from the same TMN domain or from different domains.
- Access from commonplace terminals (i.e. PCs, Macintosh, X terminal, or upcoming “Network Terminal”) is a major opening in TMN. It lowers costs and allows reuse of existing development, for instance through the use of the Java mapping that allows to have web-enabled CORBA components.

4. ARCHITECTURAL PRINCIPLES FOR A CORBA-BASED TMN

This section first discusses the manager/agent paradigm mapping into distributed objects, then we present the main interfaces used to provide the CMISE services.

4.1 Mapping Manager/Agent on distributed objects

Porting the execution model and CMISE naming schemes to the CORBA world induces several questions, that we address in this section.

4.1.1 Are agents objects ?

OSI management model relies on the notion of agent containing a set MOs. Agents perform operations on MOs to satisfy managers requests. Thus, MOs are considered as non-autonomous objects.

CORBA execution scheme relies on the notion of servers, containing a set of objects. Those objects are passive, but they are stimulated by incoming invocations which generates a temporary activity which terminates when the invocation completes. Depending on CORBA implementations, there can be several activities inside a server. CORBA takes charge of the agent execution role with this mechanism. The scope handling role can be handled by the managed objects themselves, as we describe it later on in this article. This leads to a new architecture where the agent functionality is spread among several actors of an interaction, and where the only entities to consider are managers and managed objects.

The interaction between a manager and a managed object uses two interfaces : management operations, and notifications, described in Figure 1.

This simplified view does not detail the complexity of communications. When more than two entities are involved in an interaction, it is necessary to use a communication object to control the communications. This issue arises for notifications. We present in 4.3.2 a special binding object (*Notification Channel).*

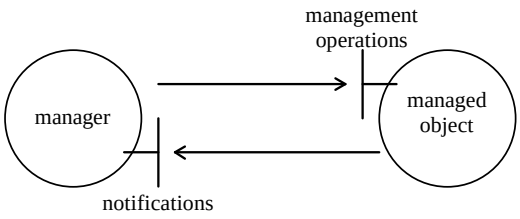

Figure 1 - Manager - MO interfaces

Managers and MOs can naturally be implemented in CORBA objects. Those objects can be configured differently depending on the needs : The more two TMN objects have to communicate, the more they should be collocated in the same address space. It is possible to implement in the same CORBA server a manager and all its MOs. It is also possible to allocate a server for each MO, but this would be very costly in memory and invocation time. However, each MO can be developed independently. Comparing to OSI management architecture, where the unit of development is the agent, the CORBA approach allows more flexibility in the application configuration.

4.1.2 Reference vs. Name

In OSI, each Managed Object Instance is identified uniquely throughout the system. The MIB structure defines a Containment tree of object instances. Each object class has attributes which are used in instance naming. The Distinguished Name (DN) is both a name, used at a logical level and handled by developers, and a reference used at a communication level to locate the entities involved in an interaction. Every communication packet contains a DN to denote its destination.

In CORBA, naming and addressing are two separate topics. To access a server, a client must first acquire its reference. Both clients and objects implementations have an opaque notion of object references and are insulated from their actual representation. Two ORB implementations may differ in their objects reference representations. However, they all include information about the configuration of the object (e.g. the name of the server containing the object, the host IP address, the port). Thus, this reference is not location transparent.

CORBA defines a Common Object Service, the Naming Service [COS], that associates a logical name with a reference. Given a logical name, a client can get the reference of a service through this Naming Service. The name being purely logical, it is the only location independent reference provided in CORBA.

This separation between names and references in CORBA implies to maintain a structure storing the corresponding reference of each logical name. This is the Naming Service role. Section 4.3.1 details how this service fits our needs for TMN.

4.1.3 Dynamic vs. Static routing

The previous distinction entails a different addressing scheme between the existing TMN systems and CORBA. The first one uses a dynamic routing scheme to find the target object, whereas CORBA uses static addressing.

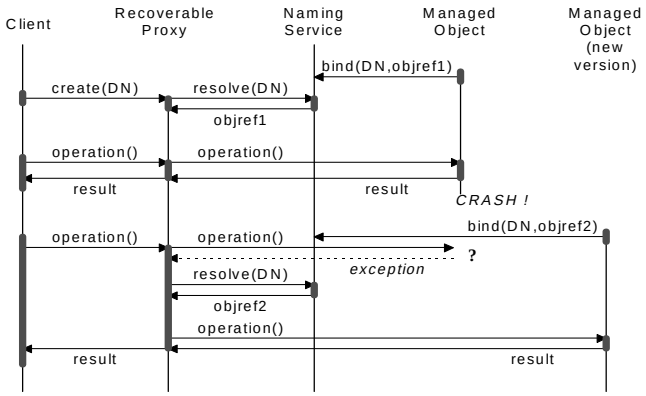

Figure 2 - Reference Recovery Service : Interaction diagram

CMIP communication infrastructure dynamically locates the destination of each packet using its contained DN. A modification in the configuration of the system does not impact the identifier, as far as the containment tree remains unchanged. This is an important feature for TMN systems, where components have long life cycles and are subject to sporadic restarts for maintenance reasons (crash, update, reconfiguration...).

CORBA separates the communication procedure in two steps : first acquire the reference, then use it. While this can enhance performance, there is an important drawback : the reference has a limited validity period. Any modification in the object configuration invalidates the reference. CORBA object references should therefore

be considered as access hints. In our architecture we have designed a mechanism that transparently provides reconstruction of this reference when it occurs to be invalid. We have extended basic CORBA stubs by implementing a *Reference Recovery Mechanism*. A *smart stub* has been designed by encapsulating the original one, and enriching it. The use of these stubs is completely transparent to the client. These new stubs are created with a DN, which is the only reliable identifier of the remote object. Those smart stubs provide both the speed of static CORBA routing, and the OSI identifiers (DNs) independent of the configuration.

This leads to define a translation of an ASN.1 ObjectInstance in IDL that reflects both address and name.

```
struct ObjectInstanceType{
                        ManagedObject         objReference;
                        DistinguishedNameType objName;};
```

Actually, this structure might be more complex, due to the various forms allowed for names and references. This structure avoids to the manager a lookup process that would be necessary if only the name was given, and the name allows a recovery process if the reference becomes invalid.

4.2 System interaction scheme overview

Figure 3 gives a picture of our system elements interaction.

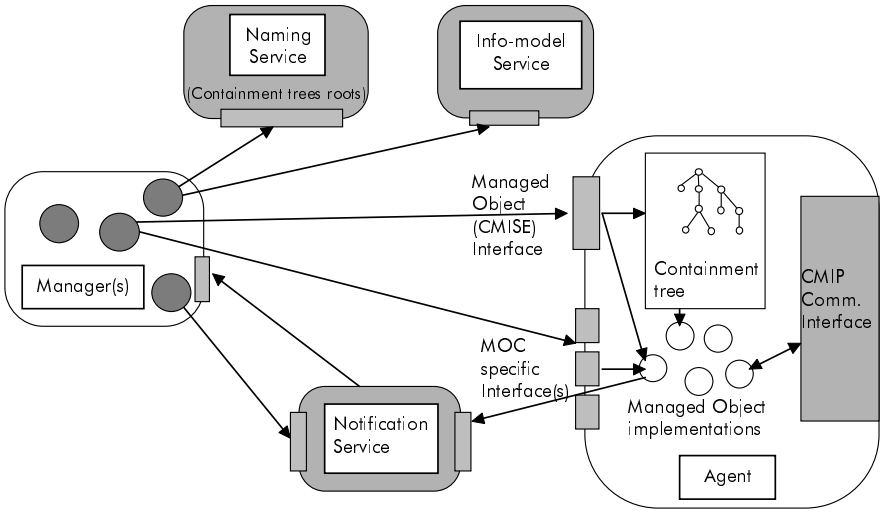

Figure 3 - The global picture

The Manager accesses MOs either via a generic CMISE interface (on an agent), or via a specialized MOC interface generated from its GDMO specification. Some CMISE fonctionality require the presence of external (i.e. neither in the manager, nor in the agent) components, that we call CMISE Services. There are three identified

CMISE Services : the *Naming Service*, that maintains the relation between DN and Object References. The *Notification Service*, that transfers notifications, spontaneously generated by MO, to managers that subscribed. An *Info-Model Service* maintains a complete and consistent representation of the OSI information model.
The objective of our system is to offer a management compliant with the OSI standards. To reach this goal, we needed an automatic translation of OSI specifications into CORBA interfaces description in IDL. We extensively used the work of the XoJIDM group [JIDM], that specified the translation of ASN.1 and GDMO to IDL.

4.2.1 ASN.1 to IDL translation

It is important to keep in mind the difference between ASN.1 and IDL concerning values description. Whereas IDL is an interface language, that describes the format of the information transferred during invocations, ASN.1 is a complete specification language, that provides information about the use of the data : allowed/default values, allowed operations... From the ASN.1 description, the translation will only keep the data structure necessary to transport the possible values. Some unauthorized values may however be stored in the same structure.

We adopted XoJIDM specifications regarding this translation. It defines a mapping for all ASN.1 definitions. It takes care of specific scoping and rules of each language, and of all the syntactic differences (hyphen replaced by underscores, case conflicts...)

4.2.2 Managed Object Interface

Management operations on a managed object can be generic, through an interface providing the standard CMISE operations. We called this interface *CMISE/IDL*. It offers the full CMISE richness, including scopes and filtering. Alternatively, they can be typed, through an interface generated from the GDMO specification of the MOC. We called this interface *GDMO/IDL*. It is much simpler to use, the name of its operations being those defined in the GDMO MOC. To be simple, scoping/ filtering and operations on multiple attributes are excluded from this interface.

Our architecture is MO centric, and does not rely on any separate agent entity. MOs have to maintain and manage their relationship (superior/descendants). Thus each MO inherits from the standard Naming Context interface, its descendants in the containment tree are entries in the Naming Context. The full description of this feature is given in section 4.3.1. To ensure that a single CORBA object reference points to those three interfaces, we used inheritance, as defined in the figure below.

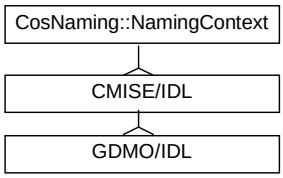

Figure 4 - MO inheritance tree

In legacy TMN platform, it is the agent that handles creation and deletion of MOs. In CORBA objects creators are called *factories*. In our architecture, each MO is the factory of its immediate descendants in the MIB. A creation can either be done through the CMISE/IDL interface (the operation is not typed) with M-CREATE or through the GDMO/IDL interface, using operations generated from the GDMO NAME BINDING.

The Managed Object contains all functionality it needs to perform the complete set of CMISE services. Our architecture offers more flexibility than the existing one : a MO can be independently developed and executed in any configuration.

This architecture provides a way to avoid the scalability problem of current ORBs, that may not handle enormous (millions) number of objects. All CORBA MOs do not have to be registered to be managed, since they are always reachable by the CMISE/IDL interface of its superior object.

Following sections give further details on CMISE/IDL and GDMO/IDL interfaces.

4.2.2.1 CMISE/IDL : the generic CMISE interface

We have defined a generic interface offering the complete set of CMISE services. This interface will actually be made of several IDL interfaces. It can be seen as an agent access point. This interface allows managers to ask directly for CMISE management operations (Get, Set, Action, Create, Delete), with parameters corresponding to the CMIP PDU parameters. The agent will be in charge of the management of its subtree : locating its contained objects (given their DN), scoping, filtering, creation/deletion of managed objects... This corresponds to a direct integration of agents from current TMN platforms.

On the manager side, there is an IDL interface allowing to receive asynchronous responses to previous requests. Events awaited by this interface are multiple replies (from a single MO, or from a scoped range of MOs) and notifications.

4.2.2.2 GDMO/IDL : specific MOC interfaces

In addition to the generic CMISE interface, we needed specific interfaces, describing in IDL the attributes and actions available on a specific MO. For that purpose, we used a GDMO to IDL translation algorithm, derived from XoJIDM.

A GDMO `MANAGED OBJECT CLASS` is translated to two interfaces :

- An agent IDL interface: it has a set of IDL operations per `ATTRIBUTE`, depending on its properties (`GET`, `REPLACE`...), and one IDL operation per `ACTION`. When the action has a `REPLY SYNTAX`, the corresponding IDL operation has a `UsingMR` exception to inform the manager of the use of a multiple replies mechanism. We extended XoJIDM specifications by adding to the interface a creation operation per possible descendant type, to fulfil its factory role (based on the `NAME BINDING` templates).
- A manager IDL interface: it allows to receive notifications.

4.3 CMISE services

As stated in the previous section, it is necessary to develop specific services to provide full CMISE functionality in CORBA. Two are mandatory (Naming and Notifications), and one is provided to implement a dynamic Shared Management Knowledge (SMK). We give some information on the realization of these services.

XoJIDM early work proposed to realize CMISE services reusing as much as possible the CORBA Common Object Services [MAZU]. We adopted a more cautious position considering their actual availability and their functional limitations. We used them when already available, and fully fit our requirements.

4.3.1 Naming

Our system is heterogeneous, it comprises:

- Component accessible by CMIP only. Their naming is managed by their CMIP Communication Infrastructure. CORBA objects access them through a gateway.
- Integrated components. Those were developed to work on CMIP, but our generic CMISE interface enabled them to work on CORBA. An agent is a CORBA server. It manages the naming of its MIB.
- Newly developed components. They offer a set of Managed Object interfaces. Each MO, being a Naming Context, manages the naming of its direct children, conforming to the CORBA Naming Service specification.

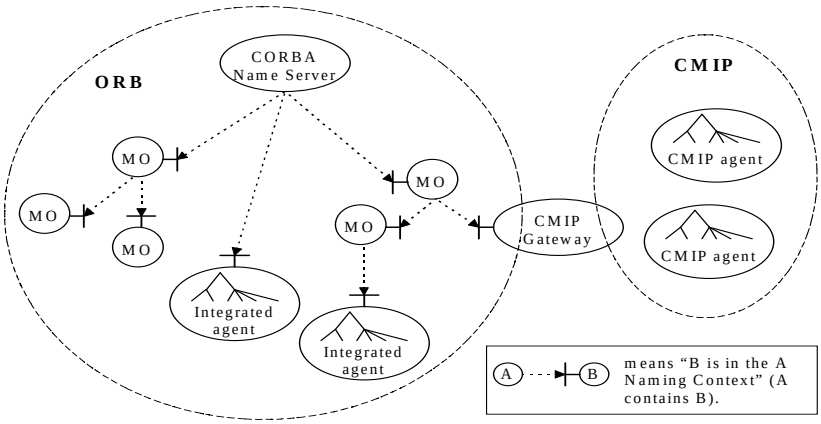

Figure 5 - Global naming scheme

Thus we have three kinds of naming in the same system. However, all naming islands use the same notion of containment tree and DN. We decided to provide a unified method to access these islands (without getting wet). For this purpose, we used the CORBA Naming Service. Each root of a naming island is a Naming Context, and inherits from the standard related interface. Those contexts are registered in one (or several connected) CORBA Name Server(s).

4.3.2 Notifications

CORBA Event Service lacks filtering capabilities to fulfil TMN event Forwarding Discriminator role. We adopted TINA specifications of a Notification Service [TINA], which is a specialization of CORBA Event Service, and used an implementation prototyping these specifications. In this service, filtering of events is performed inside communication Channels.

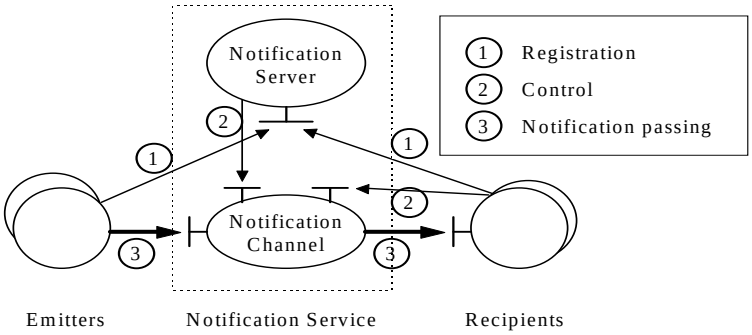

Figure 6 - Notification Service architecture

The Notification Server is responsible for the management of the Notification Channel. It receives registrations from emitters and recipients of notifications. It returns them the reference of the Notification Channel. The recipient gives a filter expression to this channel. When the emitter, wants to propagate a notification, it invokes a push operation on the channel. This channel processes the list of recipients which requested that notification, and forwards them the notification if it matches their filter.

4.3.3 Info-model service

GDMO is a specification language while IDL is an interface language. Hence, IDL doesn't reflect behaviour, required or permitted values, optionality of features. An optional external component was added to our framework : the Meta Management Information Base (MMIB) which is handled by the Info-model service. It mirrors all GDMO concepts, as defined in X.722 [X722] (e.g. type, name, OID, contained objects of any MO).

5. IMPLEMENTATION OVERVIEW

Two ways of implementing our concepts were possible, either by using a dynamic approach or by using a dedicated static approach. In the dynamic one, all requests are interpreted and no compilation is required. This approach uses the dynamic mechanisms of CORBA such as DII and DSI. But since this approach is complex and could lead to performance problems. In the static approach, the software is dedicated to a given information model. The following gives a rough description of our implementation scheme and the changes we made in order to obtain a full CORBA-interoperable component.

Our architecture provides facilities on the agent part to receive indications (with error-checking) and send back the responses (possibly many), to emit event reports and to retrieve Managed Object implementations concerned by an incoming request. While on the manager part, mechanisms allow to send requests, receive possible corresponding confirmation, handle the responses and receive event reports. We have

introduced in our existing component all the features for the support of CORBA-based interactions. Thus, communication between managers and agents can be either CORBA or CMIP based.

Actual TMN component implementation consists of three main software [ALCA] parts:

- some generic support software that insures all basic functions of a CMISE agent (naming tree management, communication infrastructure, stacks, CORBA libraries). This software forms a generic agent/manager with containment tree management support;
- some software which is generated by specific tools of the development environment associated to the framework. Classes generated from GDMO, ASN.1, and IDL specifications are classified in this category. This set of classes is therefore information model dependant;
- some software which supports the application-specific behaviour of the GDMO-generated classes. This software is implemented by the application designer.

As part of our toolchain, two compilers generate C++ code from the ASN.1 definition and from the GDMO description. Classes that handle C++ representation of ASN.1 types also handle C++ representation of IDL types. This allows to use either IDL types or ASN.1 C++ types in a transparent way. These classes are the basic support for type interoperability with a CORBA-based communication in a CMIS environment. Each Managed Object Class of the GDMO specification is implemented as a C++ class that implements management operations. We have customized our GDMO compiler in order to generate the required CORBA/GDMO adaptation code. Since each MO is a CORBA Naming context.

The software component has both the behaviour of a CMIS agent and of a CORBA server. Internally it is built with a CMIS agent architecture. Its CORBA server behaviour relies on specifically generated C++ code (mentioned above) from the ASN.1 and the GDMO specification. In the generic CMISE interface context, the CMIP paradigm is partially exported to the CORBA interface. Incoming CORBA requests mimic CMIP requests (SET/GET/CREATE...). The agent responds to the CORBA requests just like a CMIP request, except that the outgoing answer is supported by classes generated from the IDL specification.

6. CONCLUSION

The TMN as defined by M.3010 has been an important step towards open distributed management, but its relative heaviness and its limited inter-operability with the mainstream information technologies have slowed its penetration. Today, the TMN should evolve to meet the challenging requirements of the rapidly changing telecommunication market. The integration of CORBA as the base middleware should launch TMN towards its future while securing existing and on-going developments.

The architecture and the principles described in this document show that the implementation of a TMN Operation System based on CORBA is fully feasible. The proposed technology will also allow the integration and the inter-operability with

CMISE based applications and other CMIP agents or managers. This smooth evolution path will largely increase the acceptance of the introduction of the full object distribution paradigm into telecommunication management. This could be considered as an important step towards TINA [TINA], which aims at being the future telecommunication architecture.

To optimize the overall performance of the system, we propose a limited (but pertinent) usage of the CORBA Common Object Services (COS). Other propositions [MAZU] have proposed, at the opposite, to use them as much as possible. But their current limited availability, and most of all the architectural constraints they imply, have led us to build innovative and adapted services using existing COS interfaces whenever possible.

This document should constitute an important input for the on-going and future work of the XoJIDM on the CMISE to CORBA Interaction Translation.

The architecture presented in this document has been prototyped, demonstrating the interoperability between CMIP and CORBA components. This prototype makes use of the Naming and Notification services. Performance measurements are currently in progress, but early results seem to give a slight advantage for CORBA compared to CMIP. The next step will be to experiment on a large scale the proposed implementation, to validate the scalability of the architecture.

References

[ALCA] The ALCATEL Management Platform : ALMAP, Product description, 1997. http://www.alcatel.com/almap

[AAR1] Introduction of CORBA in TMN systems, AAR internal report, 1995

[CORBA] Common Object Request Broker : Architecture and Specification r2.0, OMG, 1995

[COS] CORBA services : Common Object Services Specification, OMG 95-3-31, 1995

[JIDM] Inter-Domain Management : Specification Translation, X/Open and NMF Preliminary Specification, Ed. Open Group, 1997

[M3010] Principles For A Telecommunications Management Network, ITU-T Recommendation M.3010, 1993

[MAZU] Mapping of Common Management Information Services to OMG Common Object Services Specifications. Subrata Mazumdar, AT&T Bell Lab CSRL, 1996

[TINA] TINA DPE Services Specification, TINA-C, 1994

[X701] Open systems interconnection - System Management Overview, ITU-T Recommendation X.701, 1992

[X710] Common Management Information Service Definition for CCITT Applications, ITU-T Recommendation X.710, 1991

[X722] Structure Management Information : Guidelines for the Definition of Managed Objects, ITU-T Recommendation X.722, 1992

[X901] Reference Model of Open Distributed Processing, ISO/IEC JTC1/SC21/WG7, ITU-T X.901 | ISO/IEC 10746-1, 1995