A staged approach for TMN to TINA migration

Daniel Ranc, Alcatel Alsthom Recherche, France
George Pavlou, University College London, UK
David Griffin, University College London, UK
Alex Galis, University College London, UK

Abstract

The TINA architecture is often perceived as a service layer specialized framework ; however a detailed examination of the architectural context and ongoing work in the field demonstrates that it is not so, and that the TINA architecture can be envisioned as a Network Management framework as well if care is taken about some TMN specificities. Having acquired that point, the paper proposes a conceptual architecture for interworking TMN and TINA based systems. It then describes a component architecture for a migrating OSI MIB that demonstrates concretely the feasibility of building a TMN-able agent running in a TINA-based environment.

1. Architectural Context

This section is devoted to an examination of background material regarding CORBA and TINA architectures, specifically from a Network Management architecture and requirements point of view.

CORBA Overview

The OMG (Object Management Group) aims at designing an architectural framework supporting detailed interface specifications that will drive the industry towards inter-operable, reusable, portable distributed software components based on standard object-oriented interfaces. This resulted in the definition of an abstract reference model called the *Object Management Architecture* (OMA), and the specification of a concrete platform called the *Common Object Request Broker Architecture* (CORBA).

The Common Object Request Broker Architecture has been designed by the Object Management Group (OMG) with the goal to provide a truly open object distribution architecture; thus moving one step further both:

- regarding proprietary architectures such as OLE;
- regarding non-object oriented environments such as DCE.

The CORBA architecture (in a snapshot) is composed of a *software bus*, the Object Request Broker (ORB), on which sit CORBA *clients* (referencing distant objects) and CORBA *servers* (holding objects). The clients and servers share one unique definition of their interface, which is specified in a particular language: the *Interface Definition Language* (IDL). IDL is independent from implementation languages, thus the developer has a choice of *IDL bindings* (to C++, to SmallTalk, to ADA.) from which to select.

In addition, the CORBA architecture offers a number of services which are, from bottom to top:

- core object services which are generic: naming, persistence, events etc. which can be thought of as augmenting the functionality of the ORB;
- Common Facilities which define application-level frameworks: Information Management, Task Management etc.
- Business Objects, which are at application level and perform business-specific functions.

The TINA-C consortium integrated much of the OMG concepts into its telecom architecture. For instance its specification language, ODL, is by essence an extension to IDL. Furthermore, although TINA does not specify the communication media between software entities, existing implementations of the Distributed Processing Environment all are based on CORBA in various flavours.

CORBA represents a certain semantic shift from the state-of-the-art middle-ware approach. Two reasons appear:

- The manipulation of the objects changes:
    - Classic middle-ware (XOM/XMP is a good example) provides to the programmer an API allowing to manipulate distant objects, however these objects are not in the user's name-space; instead, they are in the framework's name-space.
    - CORBA, on the other hand, brings these objects within the scope of the implementation language directly as generated code (through the IDL compiler). Operations defined in IDL are translated into target language operations. This way, the distant object is manipulated exactly as if it were local: a transparent access is possible.
- The place of the protocol changes. Whereas in known middle-ware the protocol is part of the API, in CORBA the protocol is largely hidden to the user.

These characteristics give a high level of abstraction to distributed programming. However, it has still to be studied if this abstraction does not introduce performance drawbacks; scalability is another topic to be studied as well.

The interface language definition: IDL

The way in which CORBA envisions the distribution of applications through its architecture is largely different from the OSI approach. Within the CORBA architecture, interoperability is based on the use of a software bus, the ORB (Object Request Broker), instead of stacks as in the OSI paradigm.

Definition of interfaces of software components (clients and servers in the CORBA paradigm) are made using the IDL specification language.

IDL allows to describe operations signatures. It is self-contained and does not need further specs for value transportation (the ORB insures value interoperability and marshalling).

Compared to GDMO, which conceptually lies in a similar sphere in the OSI paradigm, IDL differs from GDMO in following ways:

- GDMO has been specifically designed for TMN purposes, whereas IDL intends to be generic. This explains the constructs and entities that are aimed at the specific requirements of TMN that are found in GDMO and which are, naturally, absent from IDL. An example of such features is the Managed Object Class, which is conceptually on a different level than the class notion of IDL.
- GDMO bears features such as the containment concept (`NAME BINDING` template), Attribute Value Change specifier, packages, etc. that are not found in IDL.
- Some of the features of GDMO are tightly coupled to the TMN architecture. These features are naturally not found in IDL. An example is the containment relation, which is linked to the concept of MIB. Such features do not exist in IDL (although they can, of course, be provided by external services e.g. a relationship service as proposed by the OMG).
- GDMO, due to a historical semantic evolution, shifted slightly from a strict interface definition language, towards a system definition language: the GDMO compilers are indeed directly used to build agents and managers out of the GDMO interface specification, as if GDMO did a conceptual intrusion *within* the Operations Systems instead of staying at their interface. IDL is not, and will not, although the ODL language, an extension of IDL used by TINA-C, can be considered as going in a similar direction.
- GDMO has been extended with a powerful modelling tool, the Generic Relationship Model (GRM) which is a formalism intending to model any kind of relationship between GDMO entities, along with consistency modelling templates. IDL will clearly never need such a model.

The grammar of IDL is largely inspired from C++, however it introduces some particular constructs dedicated to interface specification; on the other hand it naturally does not feature any procedural elements. An IDL specification looks somewhat like a C++ header file: declaration of a class, its attributes, methods.

The mapping between GDMO and IDL that is chosen within the study is based on the standardisation work done by XoJIDM in its static translation proposal. The XoJIDM mapping can be thought of as a way to allow information transfer between the two types of models - however it does not intend to map the spirit behind both formalisms.

Naming

In CORBA the address space unit is the server. A server could handle one or several MOs, depending on the implementation strategy and scalability. OSI naming, on the other hand, makes use of the universality of the X.500 scheme to denote each and every entity throughout

the system.

In a CORBA environment, naming and addressing are different issues. The *naming* consists mainly of the host name and port number that allows to reach the CORBA server while the *addressing* concerns the object reference which doesn't depend on the implementation at all.

And since an object reference depends on the name of the server, the IP address and the port, it is not location transparent and the object can not be moved in another location transparently. The role of a Naming Service consists of keeping the association between logical names and object references.

In OSI, the naming tree is responsible for retrieving the MO and, through scoping and filtering it allows multiple object selection. CORBA is optimised for single object selection. To provide this functionality the CORBA Naming Service [COSS] is used and the root object of the OSI naming tree should be registered in the CORBA Naming Service.

In a nutshell*:*

Compared to previous technologies or designs, CORBA realises three shifts:

- *from proprietary to open, inter-operating*
- *from an API-based approach to an object-oriented approach*
- *from explicit remote objects to implicit remote objects.*

Joint interdomain management (JIDM)

XoJIDM produces two documents*:*

1- The Specification Translation [JIDMs] relates to the static/compile-time environment. It defines a mapping of ASN.1 to IDL types, using exclusively 'typedef'. Complex constant values cannot be represented in IDL, they are defined as operations returning the constant value. The generated IDL types ignore some ASN.1 specifications*:* defaults values, tagged types, constrained types and subtypes appear only in IDL comments. Those features do not need to appear in the IDL, which is concerned only by the way the data is coded and transferred, not the constraints applied on their use. The same document presents a translation from GDMO to IDL. A managed object class maps to an IDL interface with the same name, plus two additional interfaces which support multiple replies and notifications. We will extensively use this work in our architecture, for it carries out a mandatory step for the migration from OSI to CORBA TMN.

2- The Interaction Translation [JIDMi] relates to the dynamic/run-time environment. It describes how the CMISE services can be performed by CORBA entities. However, this JIDM work is still in a draft status at the time of delivery of this article.

TINA architecture from a TMN point of view

The TINA architecture as seen from a TMN perspective, delivers a number of powerful features to network management:

- thorough object orientation

The OSI framework, while defining a genuine object model for the information modelling itself, is not object oriented in its component architecture nor in its handling of inter-component communication. In contrast, the TINA framework is object oriented in all of its aspects, including the computational architecture and the way in which the DPE handles communication (since it is based on CORBA features in this context).

- platform

The DPE realizes a qualitative step beyond existing management platforms of the ISO/ITU-T world. Its CORBA-based distribution infrastructure hides physical implementations, technologies and DCCE details (e.g. Kernel Transport Network).

- NRIM information model

The NRIM information model brings a technology-independant view of the network that can be mapped onto existing ISO information models. The corresponding management level in ISO terms is the Network Management Layer (NML).

- Business Model

TINA proposes a sophisticated Business Model filling the relative weakness of OSI's specifications regarding the Service Layer. This model is largely in line (although not equal to) with the NMF definitions at this level. The Business Model based applications and components represent as well a streamlined opportunity to integrate the latter with existing OSI based systems in a hierarchical applicative layout (see below).

2. Migration perspectives and proposed approach

2.1 Possible perspectives of TMN-TINA co-existence

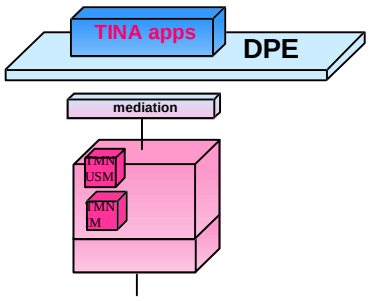
When examining the different scenarios of TMN systems migration toward the TINA architecture, different co-existence options come to mind. These perspectives, or migration paths, differ regarding the relative topology of OSI and TINA systems one to another on one hand, and regarding the way in which information is really exchanged between both worlds on the other.

As will be perceptible through the following descriptions, the applicability of one particular option depends on the applicative context, and on the degree of coupling between OSI and TINA/CORBA entities.

Applicative vs. peer-to-peer

The exchange of information between OSI/TMN-based components and TINA-based components may involve two different semantics :

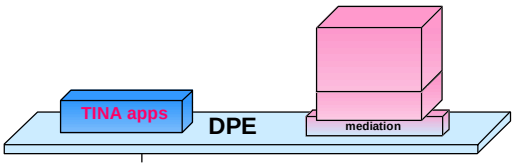
- In the *applicative* relation, a TMN-based component offers a specific, application-oriented access to a TINA component. This topology makes interoperation possible through a particular (ad hoc) interface for a specific application.



applicative relation.

The applicative relationship between TINA and OSI components represents a realistic migration scenario from TMN to TINA, because it inherently preserves existing OSI installations and allows to provide TINA service layer applications to the customer at the cost of a mediation device (CMIP-CORBA gateway).

- In the *peer-to-peer* relation, the TMN-based components and the TINA-based ones behave exactly in the same manner, with respect to the DPE. In this case, all applications have access to the managed objects disregarding whether they reside in TMN agents or in TINA components.



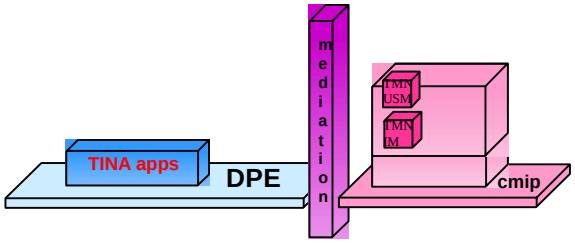
peer-to-peer relation.

The peer-to-peer relation projects mixed TMN systems bearing both ISO components and TINA components on the same level of responsibility regarding ressource management. The DPE insures communication between all components. A mediation couples the DPE with the OSI components.

loose vs. tight coupling

Depending on the tightness of the relation between TMN and TINA components, two layouts can be distinguished :

- In the *loose coupling* layout, the two types of components reside in distinct protocol domains, like « pure » entities (they communicate each in their own protocol). The interoperation requires therefore an additional component, the mediation. Loose coupling enables out-of-the-box or existing applications to communicate easily without any modification. This allows to envision CORBA interoperability in the context of deployed TMN software at a minimum redesign cost.

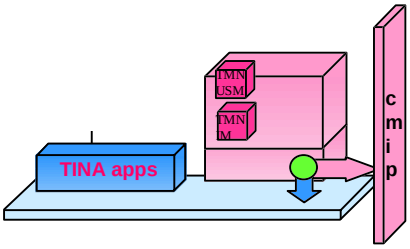


loose coupling layout.

The loose coupling layout attempts to keep either interoperating system completely unaffected by the respective other protocol, concentrating all the translation mechanisms into the mediation component. Loose coupling enables to reuse existing OSI components directly.

At the opposite, in the *tight coupling* layout some bi-domain entities are able to communicate both through an ORB and through CMIP. In a sense, these entities encapsulate their own mediation. In this situation, a CORBA manager is able to communicate directly with a

CMIP agent. Tight coupling, when combined with the ability to recompile/relink existing application code, makes existing applications access transparently new CORBA entities. Indeed, as this layout proposes really an extension of the communication classes of the TMN framework towards CORBA, tight coupling brings the advantage of reuseability of existing TMN software.



tight coupling layout.

Tight coupling introduces a new kind of components not any more bound to one distribution technology. More precisely, the ability to communicate with respects to these protocols is a property of the object instances themselves, instead of a centralized scheme where some dedicated service would provide communication capabilities. Implementations make use of the inheritance and polymorphism properties of OO languages in order to confer to the objects the new behavior in a transparent manner.

2.2 A migrating OSI MIB

In OSI Systems Management (OSI-SM), managed elements or management applications that assume an agent role provide management interfaces consisting of the formal specification of management information and of an access service/protocol that is mapped onto a well defined protocol stack. While the management information specification provides the MIB *schema*, object discovery and multiple object access facilities allow applications in manager roles to dynamically discover existing object instances. Operations to objects are always addressed through the supporting agent, which provides query facilities in a database-like fashion.

The CORBA operational paradigm is different to that of OSI and Internet management, as it originates from the distributed system world. CORBA objects are specified and accessed separately, in contrast to the managed object cluster administered by an agent. CORBA objects are most commonly addressed by *type* and not by *name*. This is due to the nature of distributed systems where, typically, instances of the same type offer exactly the same service e.g. printer servers, statistical calculation servers, etc. Of course this does not mean that there are no support mechanisms to distinguish between instances of the same type (name servers, traders). It means though that the whole framework is optimised towards a "*single object access, address by type*" style of operation, in contrast to the manager-agent model which is optimised for "*multiple object access, address by name*" style of operation.

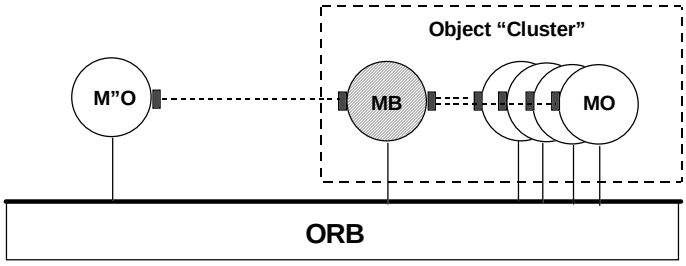
If CORBA is used as the underlying access and distribution mechanism for a TMN-based management system, managed objects could be mapped one-by-one onto CORBA objects, accessed by client objects in managing roles. The key difference is that, in CORBA, clusters of managed objects logically bound together are not seen collectively through an agent. As such, an important issue is to provide object discovery and selection facilities similar to OSI scoping and filtering. Such facilities are very important in management environments where many instances of the same object type typically exist, and names are not always known in advance e.g. objects representing current connections.

The use of the ODP / OMG CORBA model for network and service management presents a number of difficulties to overcome due to the dynamic nature of management information and the number of managed objects typically present in managed elements. On the other hand, the OSI-SM model scales much better and has already been used for managing large telecommunications infrastructures e.g. SDH/SONET, ATM, etc. Based on this observation, an ideal framework would combine the expressive power of OSI-SM and the programmability, portability and distribution aspects of ODP / OMG CORBA. In order to specify such a framework it is important to be able to map management information specifications in GDMO to computational interfaces in IDL. The information modelling aspects of the two frameworks are similar and the X/Open Joint Inter-Domain Management task force (XoJIDM) has defined rules for this mapping [JIDM].

In OSI Management, managed objects can be accessed collectively through the CMIS/P scoping and filtering facilities. These may be used for discovery services and they minimise the management traffic incurred on the managed network. In addition, the same operation may be performed on many managed objects which is not only an engineering-level optimisation but also allows a higher level of abstraction to be provided to managing functions. Discovery facilities may be provided through naming servers and traders in CORBA but the efficiency of such mechanisms, with potentially thousands of transient managed objects in network elements, needs to be evaluated. In addition, the CMIS/P operational

paradigm with potentially multiple operations expressed through a single request is lost, unless similar facilities are provided over CORBA.

Our proposal is that the operational framework of OSI management is retained over CORBA through Management Brokers (MBs). A logically bound cluster of managed objects similar to an OSI/TMN agent application is administered by a management broker providing multiple object access facilities similar to CMIS. Of course, managed objects may be also accessed directly in the standard CORBA fashion. Event management is provided by event discriminators and logs through filtering, in order to overcome the relevant CORBA limitations. Finally, the rest of the OSI Systems Management Functions [SMF] are maintained as generic CORBA objects that may be instantiated within a cluster. This approach essentially maintains the OSI operational model over CORBA but replaces the access (i.e. CMIS/P) and distribution (OSI directory) mechanisms. As such, it retains the OSI management expressive power, event model and generic management facilities while it benefits from the distribution, portability and easy programmability of CORBA. The approach is depicted in the figure below.



M"O: Managing Object
MO: Managed Object
MB: Management Broker

The OSI-SM Operational Model Over CORBA

A Management Broker does not only have a CMIS-like interface but also interfaces to be "informed" about new objects it needs to administer and to receive notifications from the managed objects it administers. In addition, every MO inherits from a *ManagedObject* interface through which the MB notifies it that it administers it. A MB and the objects it administers may be physically distributed but they logically form a "cluster" which can be collectively accessed. A managed object may belong to more than one MB domain. In the case of managed network elements, at least one MB needs to be physically located together with the local managed objects so that it provides optimised access and event dissemination facilities with minimal management traffic. In general, it is not necessary to provide separate IDL interfaces for every managed object as this may not be technologically feasible with the current state of CORBA implementations. In this case, the managed objects and the broker may interact through a local mechanism e.g. internal C++/Smalltalk interfaces if they share a common address space (an engineering "capsule" in ODP terms).

3. Long-term perspective

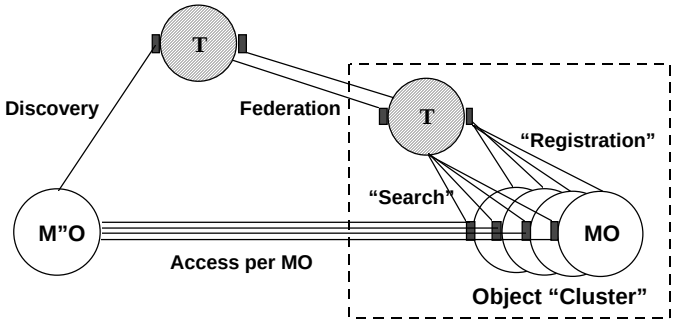
Rather than using dedicated OSI-SM-like Management Brokers, it is possible to provide similar clustering, searching, notification, etc. facilities through the use of CORBA services. A combination of the services provided by name servers, traders and event/notification servers (with some extensions) could reproduce the necessary features of OSI-SM. One problem with the use of CORBA is that federation is a key aspect in order to achieve scaleable management systems. It would be necessary to have dedicated name servers, traders and event/notification servers for every logical cluster of managed objects, e.g. in every managed element, in order to reduce traffic and increase real-time response. These "low-level" servers will be unified by "higher-level" servers in a hierarchical fashion but federation issues have not yet been worked out and are not simple. In addition, even with such facilities in place, the generated management traffic will be at least twice that of OSI management. With CORBA, matching object references will be returned to the client object and the operations will be performed on an object-by-object basis. In OSI management, the multiple object access request will be sent in one packet while the results will be returned in linked replies, one for each object accessed.

An example of the use of hierarchical traders to provide global filtering features is given next.

A trader [X9xx] supports sophisticated queries, matching sought properties of the target object(s). Objects can export their interfaces to the trader together with a list of *attributes* and a list of *properties*. Clients may request the object references of a particular type that match assertions on attributes and properties. The difference between the latter is that attributes may change dynamically while properties are fixed during the lifetime of an object instance. As such, the trader needs to evaluate assertions on attributes by retrieving them from all the instances of the type associated with the query. The function of the trader is very similar to filtering in OSI management. A key difference is that only interfaces of a particular type can be found through the trader. An additional difference is that filtering is

tightly coupled with OSI managed objects through the supporting agent while the ODP/OMG trader is a separate server.

Traders can be federated in order to be able to cope with big object spaces and different administrative domains, as demonstrated in the following figure.



4. Conclusions and remaining issues

The perspective of concretely using TINA as the architecture for deployed TMN systems raises, from the perspective of TMN requirements, a number of issues related to the specificities of TMN applications. Indeed, TMN systems require a degree of availability and scalability seldom encountered in IT.

The scalability issue hides, in turn, a number of architectural questions. Today, deployed systems are able to handle on the magnitude of 1^E6 managed objects while providing a high level of resiliency. The core property enabling this functionality is, fundamentally, the *structure* of the objects through the naming tree, naming scheme (inherited from X.500) and containment relationship.

While being well aware that these notions are partially taken over by the TINA/CORBA architecture (e.g. XoJIDM naming emulates largely X.500 naming, but without the notion of naming tree as such), present support components such as the trader, which take in charge the retrieval of managed objects according to search conditions, are still not validated regarding full scale applications. Other research directions include a methodology to actually group managed objects into scalable *clusters* or *sets* which reproduce to some extent the configuration of an OSI agent within the TINA paradigm, while hiding the vast majority of object references to the DPE.

The migration of the management systems presented retains the operational model of OSI-SM/TMN over a CORBA-based Distributed Processing Environment. This approach is based on the results of the XoJIDM work for mapping GDMO specifications to IDL but it focusing on a native CORBA-based Open Distributed Management Architecture. This approach provides a smooth migration path from current TMN-based management systems to target systems operating over CORBA-based DPEs. While there is plenty of ongoing research regarding OSI-SM/TMN and CORBA migration and interworking, the approach presented in this paper retains the relevant advantages of TMN for network management while it is both compliant and complementary to the JIDM approach.

References

[X701] ITU-T Rec. X.701, *Information Technology - Open Systems Interconnection - Systems Management Overview*, 1992.

[SNMP] J.Case, M.Fedor, M.Schoffstall, J.Davin, *A Simple Network Management Protocol (SNMP)*, RFC 1157, 1990.

[X901] ITU-T Rec. X.901, *Information Technology - Open Distributed Processing - Basic Reference Model of Open Distributed Processing - Part 1: Overview*, 1993

[CORBA] OMG, *The Common Object Request Broker Architecture and Specification (CORBA)*, 1991.

[M3010] ITU-T Rec. M.3010, *Principles for a Telecommunications Management Network (TMN),* SG IV,1996.

[TINA] *An Overview of the Telecommunications Information Networking Architecture (TINA)*, TINA'95 Conference, Melbourne, Australia, 1995.

[X720] ITU-T Rec. X.720, *Information Technology - Open Systems Interconnection - Structure of Management Information -Management Information Model (MIM)*, 1991.

[X722] ITU-T Rec. X.722, *Information Technology - Open Systems Interconnection - Structure of Management Information: Guidelines for the Definition of Managed Objects (GDMO)*, 1992.

[IDL] OMG, *Specification of the Interface Definition Language (IDL)*.

[COSS] OMG, *Common Object Services Specification (COSS) - Event, Life-Cycle, Name, etc.*, 1994.

[X710/11]ITU-T Rec. X.710/711, *Information Technology - Open Systems Interconnection - Common Management Information Service Definition and Protocol Specification (CMIS/P) Version 2*, 1991.

[IIOP] OMG, *CORBA Internet Inter-Operability Protocol*.

[X9xx] ITU-T Rec. X.9xx, *Information Technology - Open Distributed Processing - Trader*.

[SMF] ITU-T Rec. X.730-750, *Information Technology - Open Systems Interconnection - Systems Management Functions*.

[JIDM] X/Open / NMF, *Joint Inter-Domain Management (JIDM) Specifications - SNMP SMI to CORBA IDL, ASN.1/GDMO to CORBA IDL and IDL to GDMO/ASN.1 translations*, 1994.