A Novel Approach for Mapping the OSI-SM / TMN Model to ODP / OMG CORBA

G. Pavlou
Center for Communication Systems Research
Dept. of Electronic Engineering , Univ. of Surrey
Guildford, SurreyGU2 5XH,
UK
G.Pavlou@ee.surrey.ac.uk

Abstract

Over the last years, OMG CORBA has been emerging as the ubiquitous technology for building distributed systems. Since the TMN is a distributed system, a lot of research has addressed the issue of using CORBA as its base technology. Most solutions have concentrated mainly on interoperability aspects between OSI-SM and CORBA, assuming that OSI-SM will be used at the TMN element and element management level. In this paper we examine the issues behind a native CORBA-based TMN, trying to achieve a solution that maintains the full OSI-SM expressive power while it is at the same time scaleable and performant. While elements of other solutions are used, e.g. the JIDM GDMO to IDL mappings, a number of aspects in this proposal are novel. In particular, CMIS-like access aspects through scoping and filtering are kept separate from managed objects. In addition, managed objects are not required to have distinct IDL interfaces but agents may become computational entities with CMIS-like IDL interfaces. The approach has been partly validated through implementation while relevant CORBA-based applications were used in field trials.

Keywords

OSI-SM, TMN, ODP, CORBA, CORBA-based TMN

1. Introduction

Since the early inception of Open Distributed Processing (ODP), a number of related technologies tried to provide a uniform and ubiquitous framework for building distributed applications. The latest and most powerful of those technologies is the OMG Common Object Request Broker Architecture (CORBA) [1], which can be thought as the pragmatic counterpart of ODP. Given the fact that the Telecommunication Management Network (TMN) [13] is a large scale distributed system, it is valid to consider its mapping to CORBA, considering the latter as the representative distributed object technology. This implies replacing OSI Systems Management (OSI-SM) [15] and the OSI Directory [14] with OMG CORBA as the base technology for the TMN. The relationship between OSI-SM and OMG CORBA has attracted considerable attention from the research community in recent years. In this paper we propose a solution that maintains the expressive power of OSI-SM and provides a smooth migration path towards a CORBA-based TMN.

A key motivation for using CORBA in TMN environments is the following. OSI-SM was conceived as an object-oriented management technology in the absence of a general purpose distributed object-oriented framework. OMG CORBA provides exactly such a framework and it is likely that it will be used in the future for supporting advanced telecommunications services. Using OSI-SM to manage components those services will result in a discrepancy of technologies: one technology for the service control, i.e. OMG CORBA, and another technology for service management, i.e. OSI-SM. OMG CORBA may provide the unifying framework, resulting in economies of scale.

Additional motivations for using OMG CORBA in TMN systems are the following. CORBA provides a superior distribution paradigm, in which every object could be potentially distributed. In OSI-SM only whole agent applications can be distributed. CORBA performance could be potentially better than OSI-SM due to a more lightweight protocol stack. Finally, CORBA exhibits multiple O-O programming language mappings while most OSI-SM/TMN platforms support mainly C++ APIs. On the other hand, OSI-SM exhibits a more expressive object model and superior object access and event dissemination mechanisms, so the mapping between the two poses a difficult technical challenge.

The solution proposed in this paper relies on the initial NMF / Open Group Joint Inter-Domain Management (JIDM) work for the static mapping [6] of the OSI-SM Guidelines for the Definition of Managed Objects (GDMO) [19] to the CORBA Interface Definition Language (IDL) [1]. The issues behind this mapping and its implications are discussed in section 2.2. An initial mapping of the OSI-SM model to CORBA is presented in section 3. This is enhanced to a complete mapping in section 4 which retains the full OSI-SM / TMN expressive power. Section 5 discusses design, implementation and OSI-SM to CORBA migration aspects. Finally, section 6 presents a summary. It should be finally noted that this paper assumes that reader has a relatively good understanding of both the OSI-SM / TMN model and of ODP / OMG CORBA.

2. Mapping OSI-SM to ODP and GDMO to CORBA IDL

2.1 Mapping OSI-SM to ODP

Since the early days of ODP, there have been various attempts to describe OSI-SM in ODP terms. [11] was the first attempt, considering managed objects and managing objects as ODP objects, communicating via ODP platform mechanisms. This implies that the functionality of OSI-SM agents, including name resolution, scoping, filtering, access control and event dissemination, is not explicitly present. Such functionality needs to be supported via ODP mechanisms i.e. through special servers.

A similar approach has been more recently standardized in the Open Distributed Management Architecture (ODMA) [16], which is the ISO/ITU-T approach to describe OSI-SM in ODP terms. ODMA tries to provide a generic management framework that can be mapped to either ODP-based object technologies, such as OMG CORBA, or to OSI-SM communication protocols and relevant engineering concepts. OSI managed and managing objects map onto ODP objects and interfaces while the ODP trader is used for discovering interface references, according to desired object properties. Object creation is supported through factory interfaces, which can be also discovered through the trader.

In the case of an ODP-based supporting platform, managing and managed objects communicate directly with each other. When the underlying platform is OSI-SM-based, the OSI agent becomes an “operation dispatcher” in the engineering viewpoint that performs operations to managed objects. It also becomes a “notification dispatcher” that forwards notifications to managing systems. One or more notification dispatchers may be also necessary within a managing system in order to deliver the notifications to the requesting managing objects.

This ODP view of OSI-SM implies that the resulting framework does not support scoping, filtering and multiple operations to managed objects. In addition, if CORBA is used as the underlying platform, it is mentioned that notifications should be disseminated using relevant mechanisms i.e. OMG event servers and channels. When OSI-SM based platforms are used, the relevant protocols and supporting engineering concepts such as agents and notification dispatchers should be hidden behind the ODP platform APIs. The intention is to allow for the specification of management systems from an information and computational perspective in an implementation neutral fashion. The use of an OSI-SM or ODP-based technology is considered an engineering viewpoint decision that does not affect the system specification.

We could characterize the above approach as a “least common denominator” one, in which the OSI-SM framework is “pruned” to fit the ODP model. Despite its ODP orientation, [16] recognizes the fact that multiple object access through scoping and filtering and event dissemination through filtering and event forwarding discriminators may need to be exposed in the computational viewpoint. This leaves open the possibility for other potential mappings between OSI-SM and ODP. We present such an approach and explain in detail the relevant issues in the rest of this paper.

2.2 Mapping OSI-SM GDMO to CORBA IDL

Describing OSI-SM in ODP terms assumes that it is possible to map a managed object specified in GDMO from an information viewpoint to an ODP computational interface. Assuming that CORBA IDL will be the basis for the ODP computational viewpoint language, we will consider the issues of mapping GDMO specifications to CORBA IDL specifications. This is in accordance with the fact that OMG CORBA is considered as the pragmatic counterpart of ODP and the fact that OMG specifications may evolve into ODP recommendations.

Mappings between GDMO and CORBA IDL have been addressed by JIDM. Though the main intention behind this work was to result in the specification of generic gateways between different management technologies, the same principles and mappings can be used to support pure CORBA-based management systems. The JIDM work started in 1993 and the first important outcome was the comparison of the Internet SNMP, OSI-SM and OMG CORBA object models described in [12]. The *specification translation* aspects followed [6], including the generic mapping of GDMO to CORBA IDL. We discuss the main aspects of the specification translation below.

While IDL interfaces have attributes in a similar fashion to GDMO objects, it is not possible to map GDMO to IDL attributes directly. This is because IDL attributes have only *get* and *set* properties, while GDMO attributes have additional *setToDefault*, *add* and *remove* properties. In addition, it is not possible to define specific exceptions associated with access to attributes in IDL, while this is possible in GDMO. As such, GDMO attributes should map to access methods in accordance with the relevant properties e.g. *<attr>_get*, *<attr>_set*, *<attr>_setToDefault*, *<attr>_add* and *<attr>_remove*.

While this approach solves partly the mismatch problem between GDMO and IDL attributes, it creates other problems. For example, it is not possible to interrogate the CORBA interface repository about the attributes an interface has in order to access those attributes through the dynamic invocation interface; the latter is useful for generic applications such as MIB browsers. GDMO attributes also exhibit "MATCHES FOR" properties which are used for filtering. There is no equivalent in IDL which means that OSI-SM-like filtering cannot be easily supported.

GDMO actions can be naturally mapped onto IDL methods with input argument the action information and output argument the action result. Action parameters, which signify action-specific errors, are mapped onto IDL exceptions. GDMO notifications can be mapped onto separate interfaces that should be supported by managing objects and event channels. Two separate interfaces should be generated for the notifications of a managed object class, one for the "push" and one for the "pull" model.

An additional key difference between GDMO and CORBA IDL concerns the dynamic binding of functionality to managed object instances through conditional packages. This is a key feature of GDMO, used very often by information model designers, while it is not supported in IDL. The only solution is to make the functionality of GDMO conditional packages "mandatory" from a specification point of view. This approach though has a number of problems. First of all, the functionality of assigning packages dynamically to object instances at creation time is lost. A more important problem is that conditional packages are sometimes used by information model designers in a fashion that cannot be supported through "hardwired" implementations. A solution to this problem would be for ISO/ITU-T to "correct" any existing specifications that present this problem and instruct information modeling groups on the proper use of GDMO conditional packages, in an IDL-compatible fashion.

Given the rules for IDL to GDMO translation, it is possible to map OSI-SM GDMO managed objects to CORBA IDL interfaces and preserve all the work that has gone into the relevant OSI-SM / TMN specifications. The relevant translation may support gateways between CORBA and OSI-SM / TMN applications. It may be also used to support the *native* operation of management systems entirely in CORBA, as it is investigated in this paper. The equivalent IDL interfaces follow exactly the same inheritance lattice as in GDMO, while the *i_top* interface is equivalent to the OSI-SM *top* class [18]. The i_top interface inherits from the *i_ManagedObject* one, which in turn inherits from CORBA's Object, as do all the IDL interfaces.

The i_ManagedObject interface may support functionality common to all the managed objects, such as getting an object's name, accessing a number of attributes with one operation, evaluating a filter and returning the interface references of its superior or of its immediately subordinate objects in the containment hierarchy.

3. An Initial Mapping of the OSI-SM Model to CORBA

Having discussed the mapping of GDMO to CORBA IDL, we will consider how CORBA can be used instead of OSI-SM as the basis for TMN systems. This approach will support initially only basic functionality but it will be extended it to support the full OSI-SM functionality in the next section.

3.1 Object Discovery Through Hierarchical Naming and Containment Relationships

This approach assumes the same hierarchical naming scheme as in OSI-SM / TMN systems, based on the GDMO name bindings in "agent" domains and on the OSI Directory global name schema specified in [21]. For example, the name of the root object in a CORBA managed object cluster that constitutes a TMN OS could be:

{ c=GB, o=UoS, ou=CCSR, cn=NM-OS, systemId=NM-OS }

This is now an instance of the CORBA CosNaming::Name IDL type as specified by the OMG Name Service [2]. Both OMG and OSI-SM names are ordered lists of *type=value* components, so there can be a direct mapping between the two. The key difference is that the OMG name space is generally a graph instead of a hierarchical tree. Since we are adopting the TMN hierarchical naming principles, the OMG *management* name space becomes a hierarchical tree.

The first four components of the above example name denote *naming contexts*, i.e. they are not associated to CORBA objects. The fifth component, i.e. systemId=NM-OS, is a name bound to the compound context defined by the previous four. These contexts and the relevant name will be registered with the CORBA naming service [2]. A client or manager object will be able to resolve the object's name to an interface reference through the naming server. In addition, the client will be also able to discover all the management applications running in the UoS CCSR domain by performing a *list* operation on the naming context *{ c=GB, o=UoS, ou=CCSR }*. It will subsequently be able to obtain the subordinate object names of those contexts, e.g. *systemId=NM-OS* for the *cn=NM-OS* context, and resolve them to interface references. This architecture provides discovery functionality similar to that of the OSI Directory in OSI-SM / TMN environments [21] but it is supported through the use of naming services [2]. The latter may be federated in order to cope with large name spaces and different administrative domains.

```
interface i_ManagedObject
{
    CosNaming::Name      getName ();
    CosNaming::NameComponent
                         getRelativeName ();
    i_ManagedObject      resolve (in CosNaming::Name name)
                                raises (NotFound);

    i_ManagedObject      getSuperior ();
    ManagedObjectList    getSubordinates ();

    void                 addSubordinate (in i_ManagedObject subord)
                                raises (InvalidObject);
    void                 removeSubordinate (in i_ManagedObject
                                                           subord)
                                raises (InvalidObject);

    void                 destroy ()
                                raises (NotDeletable,
                                       DeleteContainedObjects);
};
```

Code 1 The i_ManagedObject IDL Interface

Having presented the system discovery aspects, we also need to address discovery facilities within an MIT cluster. Every managed object is aware of its name, which will be passed to it at by its "factory" at creation time. In addition, every managed object is aware of its superior and subordinate objects. Those object references form now a "virtual" MIT, since the relevant managed objects may be physically distributed across different network nodes. The superior reference is passed to an object at creation time. The subordinate references are passed to an object by the subordinate objects themselves, which update their superior at creation and deletion and maintain the "referential integrity" of the MIT. The Code 1 caption shows a potential IDL specification of the i_ManagedObject interface that supports this functionality. Every managed object provides access to the references of its superior and immediate subordinate objects in the MIT. It can *resolve* a subordinate name to a reference by using recursively the *getSubordinates* and *getRelativeName* methods.

Manager objects may discover the exact structure of the MIT, starting from the root object and using those discovery facilities. This approach is in fact similar to the OSI-SM / TMN one apart from scoping and filtering. It should be noted that every object acts essentially as a name server for objects in its subtree. The key advantage of the approach is that managed objects other than the MIT root do not have to register with the name service; this results in fewer interactions across the network and faster operation. If the name service was used instead, subordinate names would have to be retrieved from the name server through a "list" operation and subsequently mapped to interfaces references through a "resolve" operation. In this approach, the CORBA name service is only used for getting access to the root MIT object.

We could have added scoping at least to the i_ManagedObject interface, since it can be easily supported by traversing the "contains" relationship through the getSubordinates method. The problem though is one of "culture": scoping is a facility related to the OSI-SM Common Management Information Service (CMIS) [17] while the i_ManagedObject interface is totally unrelated to CMIS as an access method. Adding scoping, filtering and the full OSI-SM access functionality over CORBA is the next step. We will address the relevant issues in section 4.

3.2 Object Lifecycle and Event Dissemination Aspects

Two other important aspects of a management framework are object lifecycle, i.e. managed object creation and deletion, and event dissemination. We will address those here, taking a realistic approach which uses existing CORBA facilities.

In every "agent" domain, there will exist a *factory finder* object, bound to the domain naming context e.g. cn=NM-OS. A client will be able to obtain its name from the name server through a "list" operation and resolve it subsequently to an interface reference. A further optimization can be achieved by agreeing in advance on the relative name of the factory finders e.g. *factoryFinderId=NULL*, since there will always exist a single instance in a domain. The factory finder will provide access to specific factories for a particular type of interface as advocated by the CORBA lifecycle services [2]. Specific factory interfaces will exist for every GDMO class that has *create* properties. A factory interface will take parameters according to the GDMO class specification, which may include the name of the object to create, its superior's name, a "reference" object and initial values for attributes. A factory interface bears similarities to the CMIS *m-create* primitive but initial attribute values can be strongly typed. Managed object deletion is supported through the *destroy* method of the i_ManagedObject interface. Derived implementation classes will have to redefine the relevant method behavior according to the GDMO name binding properties i.e. deny deletion, instruct the caller to delete contained objects first or delete the whole subtree. This code could be automatically produced by “JIDM-aware” IDL compilers.

The final important point for a complete CORBA-based architecture is event dissemination. This can be based on the existing OMG *event services* [2]. In every “agent” domain, there will exist a *channel finder* object, in a similar fashion to the factory finder one. This will provide access to event channels that serve specific event types. Managed objects that generate notifications will register with the corresponding event channels as “event suppliers”. Manager objects will get access first to the channel finder through the naming service and then to the particular event-specific channels, registering as “event consumers”. Generated notifications will be sent to the corresponding channels and will be subsequently passed to the manager objects using the push or pull model. The fact that event channels correspond to specific event types can support strongly typed event dissemination. Event type specific push and pull interfaces will be produced for every GDMO notification and will be supported by the relevant channels. In summary, this event reporting approach is based on the CORBA event services.

3.3 The Proposed Architecture

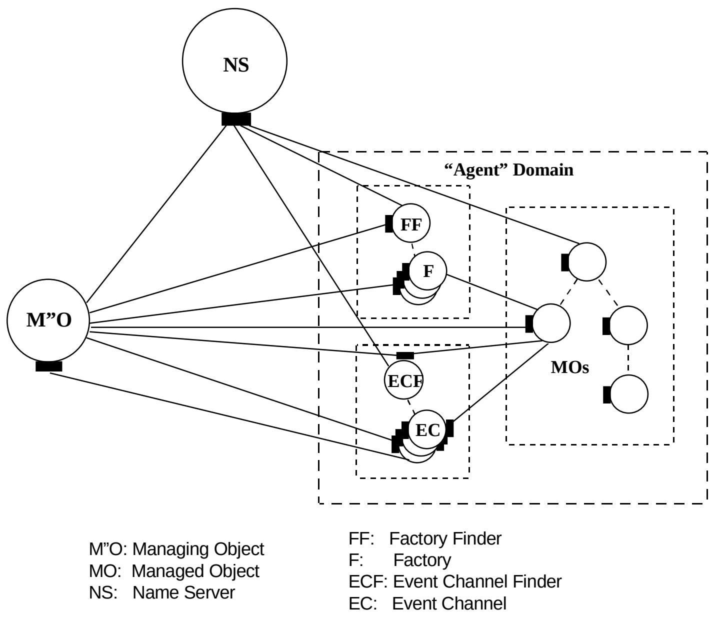

Figure 1 A Basic Architecture for OSI-SM to CORBA Mapping

The relevant architecture is depicted in Figure 1, showing the various interactions as described before. The key advantage of using CORBA is that the managed objects that constitute a logical “agent” cluster may be distributed across different “capsules”, i.e. operating system processes, which may in turn be distributed across different network nodes. The event channel finder and event channels will be located in the same capsule. The managed object factories will be located in the capsules where the relevant interfaces will be created. The factory finder will be statically configured to know the references of the relevant factories in that domain.

The disadvantages of this approach in comparison to OSI-SM are the following:

- there is no support for multiple attribute access;
- there is no support for multiple object access through one management operation;
- object discovery facilities do not support scoping and filtering;
- events are disseminated based on the event type i.e. there is no support for filtering; and
- there is no support for logging.

4. A Complete Mapping of the OSI-SM Model to CORBA

In this section we extend further the approach presented in the previous section to include TMN aspects such as scoping, filtering, EFD-based event dissemination and logging, addressing the disadvantages of the previous approach and migrating effectively the OSI-SM / TMN framework over CORBA.

Since the JIDM work on the mapping between GDMO and CORBA IDL was first published in 1994, a number of researchers started investigating the issues behind a CORBA-based management architecture. We will examine first the most important related work by other researchers, presenting it in chronological order.

4.1 Related Research Work

[3] discusses the application of the TINA ODP-based architecture to management services. It presents the view that management applications should be modelled by OSI-SM-like agents, which are computational objects with IDL interfaces in the TINA management architecture. Managed objects do not have their own computational interface but are specified as information objects in Quasi-GDMO and mapped to engineering objects within the agent.

The GDMO to CORBA IDL mapping presented in [6] addresses the static translation aspects. The architecture of a management environment based on the resulting CORBA specifications is another issue. [7] presents the first research work on such an architecture as a proposal to the JIDM group. The first version of this work appeared in 1995 and tries to re-use as much as possible the existing OMG services. The key element of this approach is that it establishes a “shared management knowledge” repository in CORBA, which recaptures aspects of the GDMO specification lost in the translation e.g. the MATCHES FOR properties of attributes.

The author’s initial approach (circa 1995) was to model OSI-SM agents as computational entities with CMIS-like IDL interfaces, based on the initial ideas in [3] but taking those to completion as presented here. The first version of the relevant architecture and specification was produced in the ACTS VITAL project which validated the TINA framework. As a result, the author was pointed to a TINA-C group working in a very similar direction and proposing a similar architecture [4].

This proposes that managed objects are grouped together in “agent” clusters and named using TMN-based hierarchical naming principles. In addition, it proposes those to be administered by a Management Broker (MB), which is a computational entity similar to an OSI-SM agent. The latter offers a CMIS-like interface which supports multiple attribute access and multiple object access through scoping and filtering. Event reporting and logging are supported through Event Forwarding Discriminator (EFD) and log managed objects. This approach was only a paper exercise that never went into considerations behind the potential realization of such a framework. As such, it was never taken any further within TINA-C.

The author architected a very similar approach which could be realized based on the OSIMIS environment [8] and its reusable software components. A first implementation of a generic gateway between CORBA and OSI-SM was produced in the summer of 1996. A second implementation followed, with native CORBA-based management agents [9] as described here.

Finally, [5] is the official JIDM proposal. While different from the other initial JIDM proposal [7], it combines elements of the other approaches. Managed objects are organized in “agent” domains and are named hierarchically. Event dissemination is handled through a specialization of the OMG event service, using event channels in both manager and agent domains. Multiple attribute and multiple object access is supported through the JIDM i_ManagedObject interface which is CMIS-like. This means that every managed object acts as an agent or management broker for its subtree. This approach is different to both [9] and [4].

4.2 Multiple Attribute Access and Filtering

With the current GDMO to IDL mapping, every attribute is mapped to one or more access methods. As a consequence, manager objects have to access attributes on a one-by-one basis, which creates unnecessary management traffic. Accessing multiple attributes is an important management requirement. In addition, many applications use the CMIS *“get all attributes”* facility, which should also be supported. The obvious place to put this functionality is the i_ManagedObject interface.

The key problem is knowing what the attributes of a managed object instance are. The i_ManagedObject part of a MO instance could interrogate the CORBA interface repository for the attributes of every derived interface and access them locally, through the DII. Unfortunately, this approach will not work. The problem is that as a result of the GDMO to IDL translation, the notion of attributes is lost which means that the CORBA interface repository cannot be used. An alternative approach would be to provide “shared management knowledge” about the information of a GDMO-derived IDL interface. For example, this information is stored in a *discovery* managed object in OSI-SM / TMN environments [21]; [7] proposes such an approach.

A third and most efficient approach would be similar to that of most TMN platforms: every derived implementation class should pass the names of its attributes to the i_ManagedObject part of an instance at creation time. The only problem with this approach is that this code will need to be hand-written, which is both tedious and error prone, while in TMN platforms it is automatically produced by GDMO compilers. A way around this problem would be the existence of special “JIDM-aware” IDL compilers which could produce this code automatically.

In summary, it is possible to support multiple attribute access. The example method signatures for getting multiple attributes are shown in the Code 2 caption. A similar *setAttributes* method could also be provided. It should be finally noted that the resulting methods are weakly-typed because the IDL *any* type is used for attribute values.

```
struct Attribute {
    AttributeId_t attrId;
    any           attrVal;
};

enum GetListError_t {
    noError,
    noSuchAttribute
};

struct GetAttribute_t {
    GetListError_t error;
    AttributeId_t  attrId;
    any            attrVal;   // "empty" in errors
};
typedef sequence<GetAttribute_t> GetAttributeList_t;


interface i_ManagedObject
{
    // . . .

    void       getAttributes    (in  AttributeIdList_t  attrIdList,
                                    out GetAttributeList_t attrList);
    void       getAllAttributes (out AttributeList_t attrList);


    boolean    evaluateFilter (in Filter_t filter);

};
```

Code 2 Multiple Attribute Access and Filtering

The next aspect to consider is filtering, which is a much more difficult proposition. [7] proposes to use the OMG *property* service, together with "shared management knowledge" which provides access to the GDMO MATCHES FOR properties of attributes, the solution being very complex. [5] specifies filtering as part of the CMIS-like access methods of the i_ManagedObject interface but does not discuss at all how it is going to be provided. It should be noted that supporting filtering in CORBA to OSI-SM gateways is easy since the filter will be actually evaluated in the target OSI-SM agent; this is not the case in native CORBA environments.

Let's revisit first how filtering is supported in OSI-SM environments. Filter assertions on a particular attribute are evaluated by the attribute itself while the ASN.1 compiler produces comparison methods. The problem with CORBA is that attributes are not "first class citizens" of the framework. Defining an attribute in an IDL interface results in nothing more than access methods being produced, without any special support for the relevant data type. As such, there is no automatic support for equality comparison and subsequently for the evaluation of filter assertions. One solution to this problem would be for OMG to consider providing such support through a special extension to IDL. Types preceded by some special keyword, e.g. `attribute`, could be treated specially, deriving from a generic attribute class and supporting equality and other comparisons. This requires though the modification of both the IDL and the relevant programming language mappings. [7] mentions that the comparison methods required for filtering could be either provided by hand, which is obviously not desirable, or produced by modified IDL compilers which understand comment lines with special significance.

In summary, it is not easy to provide filtering in native CORBA environments. In general, the mapping of IDL types to object classes is not rich enough, lacking support for comparison, pretty-printing and other generic functionality. As such, it is problematic to deal with instances of the *any* type. This is an area that needs special attention by the OMG.

Given the support for filtering and the fact that containment relationships can be navigated through the *getSuperior* and *getSubordinates* methods, multiple access to managed objects and sophisticated discovery facilities can be provided. The relevant functionality is similar to that provided by OSI-SM agents and the question is where it should be placed. [5], the official JIDM proposal, places it as part of the i_ManagedObject interface. This essentially means that every managed object behaves as an agent for its subtree. In contrast, the author [9] and [4] propose to separate this functionality from the managed objects, so that different sophisticated access styles can be provided.

4.3 Fine-grain Event Dissemination and Multiple Object Access Through the Management Broker

Given the support for filtering, fine-grain event reporting and logging can be provided by migrating the relevant OSI-SM models over CORBA. In every "agent" domain, there will exist a Event Processing (EP) object. Managed objects will get access it through local means, e.g. the factories may pass its reference to MOs at creation time. MOs will subsequently "push" their notifications to it. The EP object will create the "potential event report / log record" through the relevant object factory, evaluate the filters of EFDs and logs and may instruct the latter to send the event or log it as a log record accordingly. This is exactly the behavior prescribed in [20]. Note that the existence of the EP object is totally transparent to manager objects that are interested to receive event reports.

Manager objects will request the forwarding of events by creating EFDs and setting the *destination* attribute to contain either their name or object reference. This implies that the syntax of the *destination* and *backupDestinationList* EFD attributes [20] will have to be slightly modified. Destinations are currently specified as OSI-SM distinguished names which can be mapped to CORBA names, but CORBA object references should be added.

The last aspect of the OSI-SM / TMN framework that needs to be provided is support for multiple object discovery and access facilities based on scoping and filtering. Such access facilities are certainly "OSI-SM / TMN specific" and should be provided in an incremental fashion, without being an integral aspect of the rest of the framework. A key reason for considering those separately is they are do not represent the only way of providing this type of functionality. For example, in the CMIS/P access model containment relationships are navigated first through scoping with filtering applied at the end of the selection process. [10] proposes an enhanced model in which any relationship could be navigated, with filtering possibly applied at various stages of the selection process. Other mechanisms may be invented in the future that suit best the needs of particular management environments.

This is the why the author proposes to separate the CMIS-based access aspects from the rest of the management framework. As such, CMIS-based access should *not* be part of the i_ManagedObject interface but should be supported by a separate *Management Broker* (MB) object. Given the fact that CMIS is the current access mechanism in TMN environments, a MB should always exist in an “agent” domain with its name bound to the domain naming context e.g. *{ c=GB, o=UoS, ou=CCSR, cn=NM-OS, brokerId=CMIS }*. Managed objects could be accessed either directly or through the MB. The advantage of MB-based access is object discovery and multiple object access through scoping and filtering. The disadvantage is that the relevant access paradigm is weakly-typed: attribute and action values are of the IDL *any* type. The architecture of the proposed framework is depicted in Figure 2, including the event dissemination through EFDs. This updates the architecture that was presented in Figure 1.

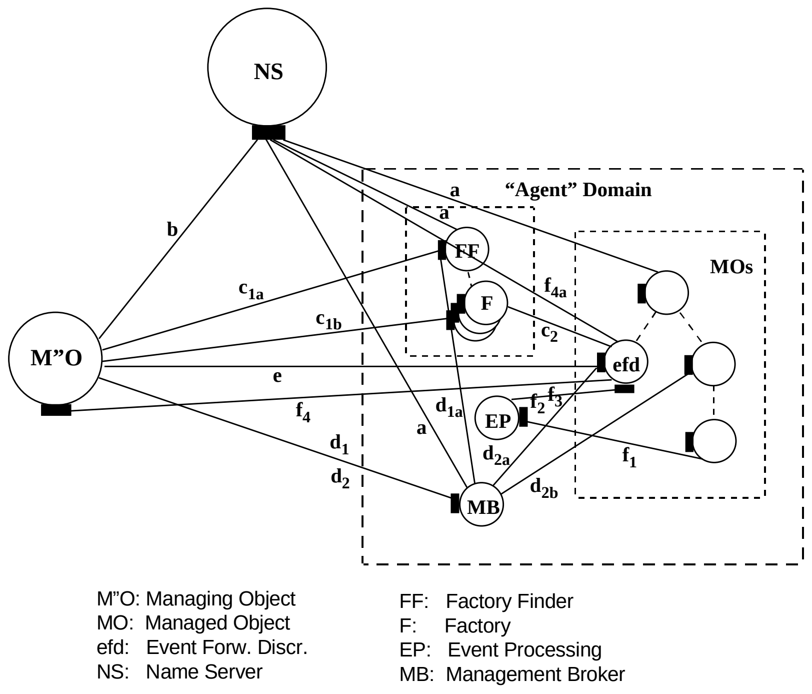

Figure 2 A Complete Architecture for OSI-SM to CORBA Mapping

When an “agent” domain is instantiated, the root MIT MO, the factory finder and the management broker register themselves with the name service (interactions *a* in the figure). Manager objects need to know in advance the domain name, e.g. *{ c=GB, o=UoS, ou=CCSR, cn=NM-OS }*. They may invoke a list operation on the name service and discover the names and subsequently the references of the MIT root, FF and MB objects (interaction *b*). A MO may be created either in a strongly-typed fashion through the relevant factory (interactions c_{1a} and c_{1b}) or in a generic, weakly-typed fashion through the MB (interactions d_1 and d_{1a}). The manager may subsequently access the object either directly (interaction e) or through the MB. A MO emits a notification by “pushing” it to the event processing object (interaction f_1). The latter will create first a “potential event report”, retrieve an EFD’s filter (interaction f_2) and evaluate it. The potential event report is not shown since it is manipulated locally by the EP i.e. can be thought as encapsulated by it. If the filter evaluates to true, it will instruct the EFD to send the event report (interaction f_3). The EFD may need to resolve the name of the manager to an interface reference through the name service (interaction f_{4a}) and “push” the event to the manager (interaction f_4).

We show below (a part of) the broker’s CMIS-like interface. This supports single object operations (get), object discovery operations (objectSelection) and multiple object operations (multipleObjectGet), in a similar fashion to CMIS [17].

```
// Scope_t, Filter_t and Sync_t map exactly to the
// X.711 Scope, CMISFilter and CMISSync ASN.1 types

typedef struct ObjectSelection_t {
    Scope_t    scope;
    Filter_t   filter;
};

typedef struct ObjectNameList_t {
    CosNaming::Name    name;
    i_ManagedObject    objectRef;
};

interface i_CMISBroker
{
     CosNaming::Name   getName ();    // the broker’s name

    void       get (
               in  CosNaming::Name    objectName,
               in  string             objectClass,   // allomorphism
               in  AttributeIdList_t  attrIdList,    // empty -> “all”
               out GetAttributeList_t attrList
               )
               raises (GET_ERRORS);

    void       objectSelection (
                      in  CosNaming::Name   baseObjectName,
                      in  ObjectSelection_t objectSelection,
                      out ObjectNameList_t  objectNameList
               ) raises (OBJECT_SELECTION_ERRORS);

    void       multipleObjectGet (
                      in  CosNaming::Name   baseObjectName,
                      in  ObjectSelection_t objectSelection,
                      in  Sync_t            sync,
                      in   AttributeIdList_t  attrIdList,  //  empty  ->
“all”
                      out GetResultList_t   resultList
               ) raises (MULTIPLE_OBJ_OPER_ERRORS);

    // . . .
};
```

Code 3 Generic CMIS-like Managed Object Access in IDL

5. OSI-SM to CORBA Migration Approach

The first step for migrating towards the target framework is to support only agent discovery and CMIS interactions through CORBA, without individual IDL interfaces for managed objects. This essentially means that the management broker will act as an agent that provides access to managed objects implemented by existing TMN platform infrastructure. The MB may be used in conjunction to the existing Q_3 agent object within an OSI-SM agent application. In this case, the TMN application in agent role will have two interfaces: the existing Q_3 interface and the CORBA version of the “Q_3” interface as specified by the i_CMISBroker and i_CMISManager IDL interfaces. This minimal approach is depicted in Figure 3. Existing OSI-SM-based manager applications will continue to function while new CORBA-based management applications may be developed. This architecture provides a “dual-agent” access paradigm.

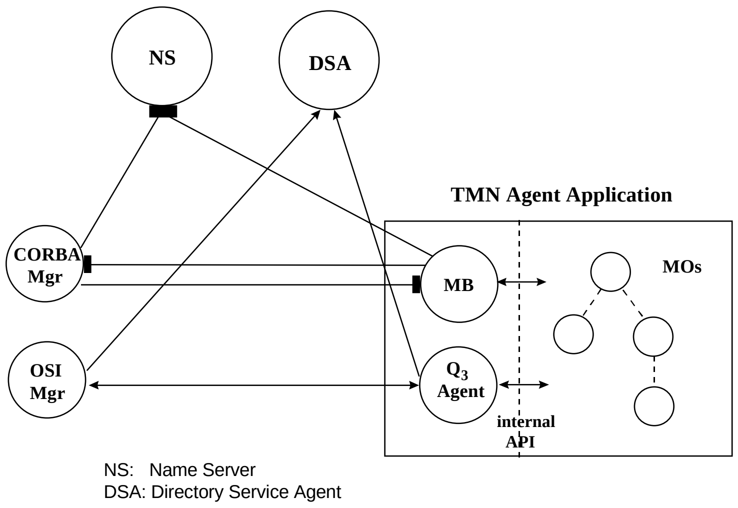

Figure 3 Dual Q_3 and CORBA Agent

A variation of this approach is the gateway approach, in which the management broker becomes a separate application which accesses one or more management agents in the "back-end" through their Q_3 interfaces. The gateway approach is useful to provide adaptation for TMN applications that are already deployed, in which case it is not possible to add to them the management broker in a tightly-coupled fashion. A gateway version of the MB was first produced in 1996. The tightly-coupled dual agent approach was subsequently implemented in 1997.

The target architecture is one in which managed objects will be native CORBA objects administered by Management Brokers, as presented previously in detail.

6. Summary

Given the emergence of CORBA as the ubiquitous object technology for open distributed systems, in this paper we examined in detail how it can be used as the base technology for the TMN. We presented first a minimal approach which retains the TMN hierarchical naming and containment relationships but does not support scoping, filtering, multiple object access and fine-grain event reporting and logging. A key aspect of this approach is that only few objects in each "agent" domain need to export their names to the name server. This avoids scalability problems in TMN environments where network elements and operations systems may contain 10's of thousands of objects.

We then added multiple attribute access and filtering to the managed objects and explained how CMIS-like multiple object access can be supported through the management broker. The proposed architecture retains the advantages of OSI-SM with the drawback that support for filtering and knowledge about the attributes of a particular object need to be hand-coded i.e. they cannot be automatically supported by IDL compilers. An advantage of the presented approach is that managed objects are not required to have separate IDL interfaces, which helps scalability for nodes with a large number of managed objects.

We should finally answer the question of what is the architectural impact to the TMN if CORBA is adopted. The answer is that there should be no impact at all. The TMN architecture and methodologies will remain the same. Interface specifications will be produced in GDMO, following possibly guidelines which will guarantee that generic translation to IDL is possible. In addition, the Q_3 interface profiles will be modified, including CORBA protocols as valid choices for TMN interfaces. The use of CMIS/P-GDMO or GIOP-IDL will become an engineering issue for implementing the same specifications. An additional benefit of using CORBA is that TMN Operation System components could be distributed across different network nodes.

In summary, CORBA seems a very promising technology for future TMN systems. Its main value compared to OSI-SM is that it tends to become the ubiquitous technology for future heterogeneous distributed systems. Its use for both management and service control will result in economies of scale and allow for the easier integration of new managed resources.

Acknowledgments

This work was undertaken under the ACTS FlowThru, REFORM and VITAL projects which are all partially funded by the Commission of the European Union.

The author would like to thank S. Mazumdar of Lucent Technologies, the editor of the GDMO to IDL mappings document [6] which formed the basis for this work.

References

[1]. Object Management Group, The Common Object Request Broker: Architecture and Specification (CORBA), Version 2.0, 1995.

[2]. Object Management Group, *CORBA Services: Common Object Services Specification (COSS)*, Revised Edition, 1995.

[3]. L.A. de la Fuente, J. Pavon, N. Singer, *Application of TINA-C Architecture to Management Services*, in Towards a Pan-European Telecommunications Service Infrastructure, H.-J. Kugler, A. Mullery, N. Niebert, ed., pp. 273-284, Springer, 1994.

[4]. E. Garcia-Lopez, *Distributed Management Facilities Architecture*, TINA-C baseline document TB_EGL.002_2.1_1996, 1996.

[5]. J. Hierro, J. Gonzalez, *Common Facilities for Systems Management*, Telefonica I+D Submission to the NMF - X/Open Joint Inter-Domain Management task force, 1996.

[6]. NMF - X/Open, Joint Inter-Domain Management (JIDM) Specifications, *Specification Translation of SNMP SMI to CORBA IDL, GDMO/ASN.1 to CORBA IDL and IDL to GDMO/ASN.1*, 1995.

[7]. S. Mazumdar, *Mapping of Common Management Information Services to OMG Common Object Services Specification*, ATT Bell Labs, TM # BL0112540-96.09.30-02, 1996.

[8]. G. Pavlou, G. Knight, K. McCarthy, S. Bhatti, *The OSIMIS Platform: Making OSI Management Simple*, in Integrated Network Management IV, A. Sethi, Y. Raynaud, F. Faure-Vincent, ed., pp. 480-493, Chapman & Hall, 1995.

[9]. G. Pavlou, D. Griffin, *Realizing TMN-like Management Services in TINA*, Journal of Network and System Management (JNSM), Special Issue on TINA, Vol. 5, No. 4, pp. 437-457, Plenum Publishing, 1997.

[10]. G. Pavlou, A. Liotta, P. Abbi, S. Ceri, *CMIS/P++: Extensions to CMIS/P for Increased Expressiveness and Efficiency in the Manipulation of Management Information*, IEEE Network, Vol. 12, No. 5, pp. 10-20, September/October 1998.

[11]. S. Proctor, *An ODP Analysis of OSI Systems Management*, in the Proc. of the TINA'92 Workshop, Narita, Japan, January 1992.

[12]. T. Rutt, ed., *Comparison of the OSI Systems Management, OMG and Internet Management Object Models*, Report of the NMF - X/Open Joint Inter-Domain Management task force, 1994.

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

[14]. ITU-T Rec. X.500, Open Systems Interconnection, *The Directory: Overview of Concepts, Models and Service,* 1993.

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

[16]. ITU-T Rec. X.703, Open Systems Interconnection, *Open Distributed Management Architecture (ODMA)*, 1997.

[17]. ITU-T Rec. X.710, Open Systems Interconnection, *Common Management Information Service Definition (CMIS) - Version 2,* 1991.

[18]. ITU-T Rec. X.720, Open Systems Interconnection, *Structure of Management Information - Management Information Model*, 1991.

[19]. ITU-T Rec. X.722, Open Systems Interconnection, *Structure of Management Information - Guidelines for the Definition of Managed Objects*, 1992.

[20]. ITU-T Rec. X.734/X.735, Open Systems Interconnection, *Systems Management Functions - Event Report and Log Control Functions,* 1992.

[21]. ITU-T Rec. X.750, Open Systems Interconnection, Systems Management Functions - Management Knowledge Management Function, 1995.

Biography

George Pavlou received his Diploma in Electrical and Mechanical Engineering from the National Technical University of Athens, Greece and his MSc and PhD in Computer Science, both from University College London, UK. Over the last 12 years he has been undertaking and directing research in the areas of communications protocols, performance evaluation, distributed systems, broadband network technologies, network management and service engineering. He has architected the OSIMIS TMN platform and has contributed to standardization work in ISO, ITU-T, TMF, OMG and TINA. Since the beginning of 1998 he is a full professor at the University of Surrey, School of Electrical Engineering and Information Technology, where he leads the activities of the networks research group.