"Integrating the OSI and the CORBA Architectures for Systems Management"

Guy Genilloud
Swiss Federal Institute of Technology of Lausanne (EPFL)
EPFL-DI-LIT, CH-1015 Lausanne, Switzerland
guy.genilloud@di.epfl.ch

Abstract

In the ESPRIT II project SysMan, the EPFL was in charge of integrating the OSI and the distributed objects technologies for systems management. We worked on specification translation between IDL and GDMO (in both directions), and in the provision of gateways. *Adapter-agents* allow distributed applications to be managed in OSI. Conversely, *proxy-agents* enable OSI managed objects to be accessed as ordinary distributed objects.

Our work in Sysman allowed us to uncover some of the weaknesses of the OSI systems management architecture. When integrating the worlds of distributed objects and OSI, we need to revisit the features of OSI and to adapt them. We should not blindly adopt them.

1 Introduction

In the coming years, both the OSI and the distributed objects architectures will be used for systems management, yielding situations where they must coexist. In the ESPRIT III project SysMan[1], which ended in December 1995, our laboratory was in charge of developing tools for supporting that coexistence. More specifically, we have implemented gateways and specification translators, which enable either:

- an OSI management platform to manage ANSAware application objects, using the OSI CMIS service;
- or ANSAware management programs to manage OSI managed objects by invoking ANSA operations on them (i.e., an OSI managed object can be associated with a dedicated ANSA interface).

Our work in SysMan is relevant to the topic of CORBA management in the sense that ANSAware is similar in many respects to CORBA — ANSAware is a distributed objects programming environment that is semantically but not syntactically compliant with the CORBA specification [CORBA-2][ANSAware]. The main differences are the IDLs, the language bindings, the object services and the fact that ANSAware offers specific support for building objects with multiple interfaces (independently of inheritance).

Related Work

The Open Group and the Network Management Forum have set up a working group called JIDM (for Joint Inter-Domain Management), for defining a software architecture that will allow a CORBA-based application to play the manager or the agent role in a manager/agent model of interconnection.

JIDM has started its work by defining two algorithms for translating specifications: one from GDMO to CORBA IDL, and the other from CORBA IDL to GDMO. This work, called *specification translation*, has recently become an Open Group preliminary specification [JIDM97]. JIDM is currently working on *interaction translation*, which is the actual integration of the OSI and the CORBA architectures.

Apparently, JIDM shares the same broad goals that we had in the ESPRIT project SysMan. In fact, JIDM is interested in the realization of the OSI architecture in a CORBA environment. Accordingly, they consider the interworking of CORBA and CMIP technology environments within the OSI SMA. To the contrary, we assumed in SysMan that CORBA would be used in a new architecture based on distributed objects (of which we know little yet), and we favoured an evolution towards that new architecture [Genil96b].

---

1. By "SysMan", we mean here the ESPRIT Project 7026 on "Open Distributed Systems Management," also known as SysMan. We never refer to the SysMan series of standards developed by the Open Group.

Organization of this Paper

This paper is organised in three parts. We first present the OSI systems management architecture (SMA) [X.700] [X.701] and the distributed objects architecture from a common modelling perspective. We then present the SysMan approach to interaction translation, both regarding the provision of CMIS access to CORBA managed objects (Section 3), and the association of CORBA interfaces with OSI managed objects (Section 4).

For simplicity, we ignore a number of issues. In particular, we do not discuss event notifications. We also omit details that are specific to ANSAware or to the SysMan distributed environment.

2 An ODP View of the OSI Systems Management Architecture

For integrating different architectures, it is necessary to have a common representation for them. In Sysman, we tried to map the concepts of ANSAware to those of OSI, and vice-versa. For example, we mapped an OSI managed object to two ANSAware interfaces, one for receiving operations, the other for emitting notifications. We obtained results which were sufficient for our purpose, but which were not fully satisfactory from a modelling point-of-view. For example, we could not explain that an action on one managed object could affect another managed object.

After the project SysMan, we knew that OSI managed objects are more than interfaces: they are objects, but of a more conceptual nature than distributed objects; they are units of specification rather than units of implementation and distribution. We revisited the problem of a common representation using the information and the computational viewpoints of the RM-ODP (Reference-Model of Open Distributed Processing) [Genil96a][Genil96b]. We did not use the ODP enterprise viewpoint because the OSI SMA does not have its equivalent. We did not use the ODP engineering viewpoint because the ODP and the OSI engineering models are quite different and difficult to reconcile, and because we did not need them for defining the specification and interaction translations.

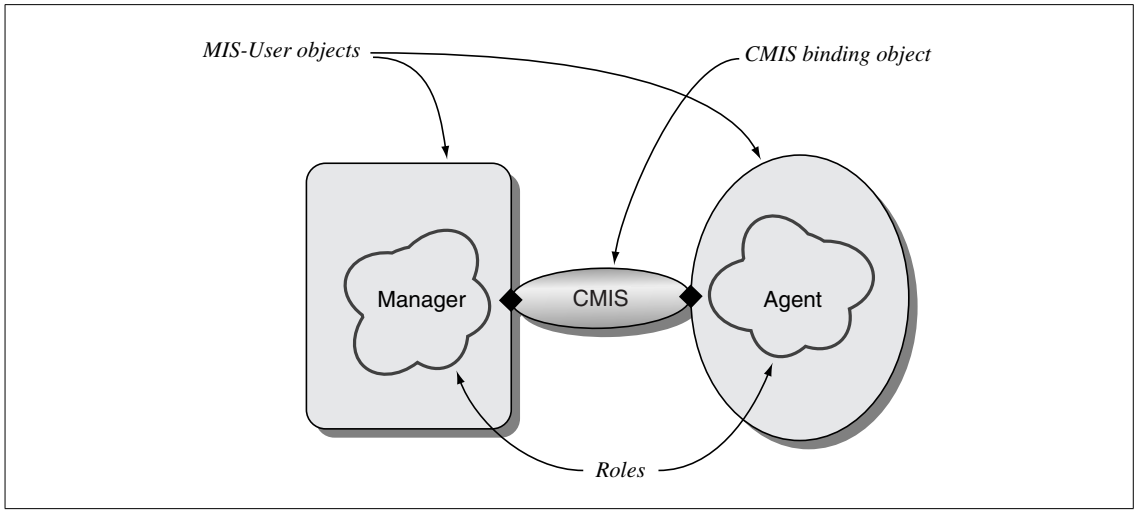

Figure 1 Two MIS-Users bound by one CMIS binding object

2.1 An ODP Computational View of the OSI SMA

The ODP computational viewpoint is inherently suitable for modelling the distributable components of a distributed application and their interactions. Objects in the computational viewpoint are either units of distribution (ordinary computational objects), or objects supporting remote interactions (binding objects). With this in mind, the OSI management architecture is easily summarized as illustrated in Figure 1:

- Distributed management systems are modelled with only two kinds of computational objects: *MIS-Users* and *CMIS binding objects*.
- MIS-Users are computational objects that perform management functions, and interact with each other. They represent the unit of distribution.
- CMIS binding objects provide the communication between MIS-Users.

The central OSI SMA concepts of *manager* and *agent* are *roles* that are attributed to the MIS-Users engaged in a CMIS communication. The manager is the MIS-User with a management mission. It requests services from the agent, the MIS-User that controls the managed resources.

Figure 2 shows that two MIS-Users, bound by two CMIS binding objects, can assume both a manager and an agent role with respect to each other.

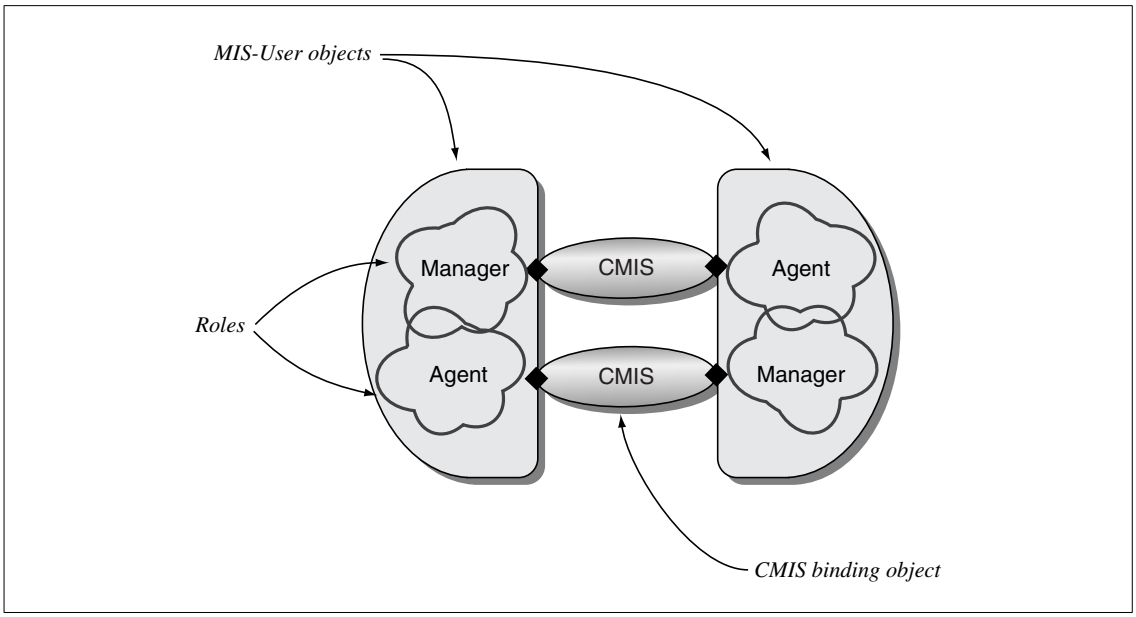

Figure 2 Two real open systems bound by two CMIS connections

2.2 An ODP Information View of the OSI SMA

The ODP information viewpoint allows to express the application-level semantics of the interactions of a system or a component, using units of specification called information objects. The OSI management information model (MIM) and the GDMO notation provide a mechanism for specifying the semantics of interactions between a manager and an agent — they introduce units of specification called managed objects. OSI managed objects are ODP information objects [X.720][X.722].

The OSI MIM defines the overall structure of *an agent's information model*:

- Agents are expressed as a composition of information objects: there is one *Dispatcher object* and a combination of *Managed Objects* (MOs).
- Managed objects provide an abstraction of the managed resources.
- The Dispatcher object provides an "information gateway" between the managed objects and the manager, handling the CMIS communication, selecting managed objects, dispatching the operations, filtering and forwarding notifications, creating new managed objects, et cetera.

Note that the information object that we call "Dispatcher" is called "agent" in the OSI literature. We avoid the OSI terminology because it is confusing to use an agent object as a part of the definition of an agent role.

The OSI SMA does not address the specification of managers. OSI is concerned with the specification of the universe of discourse of a CMIS connection — specifying the agent is enough for this purpose.

2.3 Fundamental Weaknesses of the OSI SMA

The use of CMIS for management communications is sometimes presented as being incompatible with collaborative management. In fact, there is no computational limitation since an MIS-User can assume both an agent and a manager role with respect to another MIS-User. A drawback of CMIS is its interface, which is very general by necessity, and thus difficult to use.

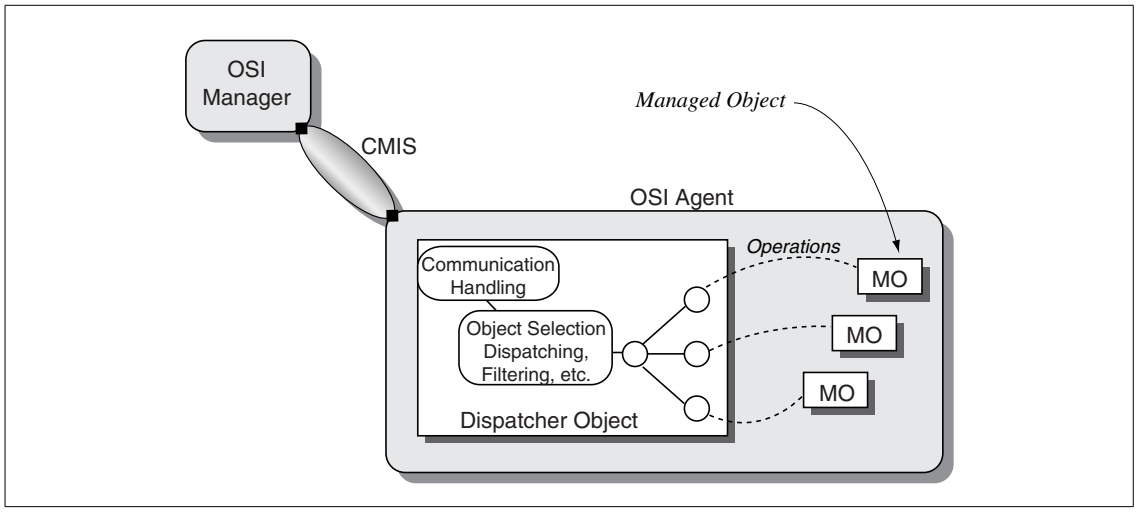

Figure 3 Information model of an agent

The OSI SMA is biased against collaborative management because of limitations in its information modelling technique. As we explained, this technique was conceived essentially for specifying the universe of discourse of a CMIS connection — as a result, it is just adequate for specifying subsystems which have only an agent role.

The OSI SMA specifies services (naming, multiple selection, filtering, et cetera) only with respect to a single agent system. As a result, these services are not always well adapted to the needs of management programs, nor to a distributed implementation of management. For example, the OSI architecture imposes that notifications be filtered and dispatched twice, once within the agent (for deciding whether they should be emitted on a CMIS connection), and once within the manager (for deciding which actions will be triggered, and with what priority). Management applications are concerned at two levels, when one level should be sufficient. This problem becomes particularly noticeable when a managed resource changes its location, or when two different managed objects provide access to a same resource.

The OSI SMA pays a lot of attention to information within agents, and it provides powerful mechanisms for reading and updating that information. On the other hand, it pays little attention to the tasks to be performed for management. In other words, it focuses on “data operations” rather than on “functional operations”. There are both advantages and disadvantages with that approach. On the positive side, an agent is likely to offer the required “data operations” in support of a particular management task, even though it was designed independently of that management task. On the negative side, agent services rarely provide management operations that support a task in an optimal way (and if they do, these operations are hard to identify and use).

For example, CMIS allows to access multiple attributes with a single operation. This means that each managed object supports a considerable number of operations for updating its state. These operations are difficult to use, because it is impossible to provide a specific procedure or method for each of them, and because the possible errors are numerous (there can be several errors per attribute). In contrast, a CORBA designer tends to provide a few operations for updating an object state, and he tries to minimize the number of errors that can be returned by those operations. Note that GDMO allows to specify object operations as well (called actions), but specifying actions is discouraged rather than encouraged in the OSI SMA.

As another example, CMIS supports a powerful query service (scoping and filtering), that allows to access multiple resources with a single message. This very general mechanism is useful for performance, and for applying a same function to several resources at once. However, it can be difficult or even risky to use, because there can be more objects in scope than the manager thinks. The problem here is that managed objects cannot be grouped by design (or in fact, that groupings cannot be enforced and documented). John Day says that scoping and filtering is only useful for monitoring purposes [Day92].

For the above reasons, we are convinced that systems management should evolve towards a distributed objects architecture: manager objects, managed objects, and distributed services. When integrating the worlds of distributed objects and OSI, we need to take into account this evolution. We should revisit the OSI SMA design decisions rather than to blindly adopt them.

3 OSI Management of CORBA Distributed Objects

In the SysMan project, we decided to work on the following problem: how can an OSI management platform manage applications (and systems) in a distributed objects world? We assumed that the applications already implemented management interfaces defined in IDL. We aimed at minimizing the changes to be made to applications, rather than forcing them to implement unnecessary features of the OSI SMA.

Our solution was to use a smart gateway, called an *adapter-agent*. The main function of this adapter-agent is to translate CMIS requests into CORBA operation invocations, and CORBA operation terminations into CMIS responses [BGG94]. This solution is based on the observation that applications are designed to be managed with operations, and that most IDL operations are easily translated into CMIS M-actions (specified in GDMO).

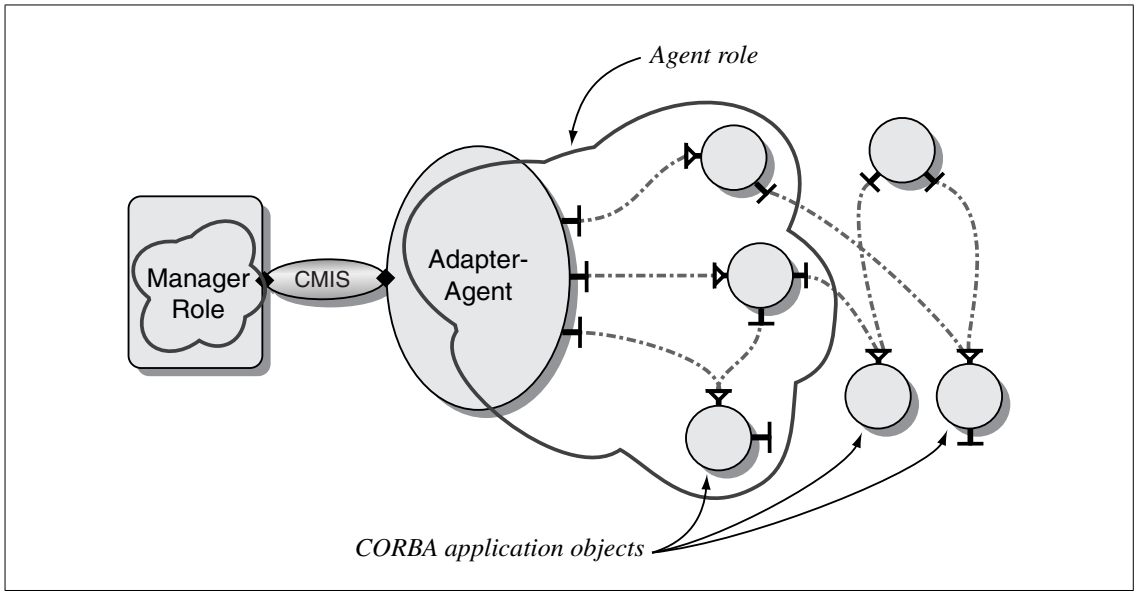

Figure 4 An adapter-agent together with a CORBA application implement an OSI agent role.

Our gateways can be referred to as *adapter-agents,* because they enable OSI managers to see applications as (ordinary) OSI agents — adapter-agents, together with the application, effectively implement an agent role. The essential burden of supporting OSI management is implemented by the adapter-agent, for example all the OSI support managed objects (e.g., Event Filter Discriminators), logging, filtering, scoping, allomorphism, et cetera. The other OSI managed objects, those that directly refer to the application, are partially implemented by the management interfaces of the application — the rest is implemented by the adapter-agent (e.g., the attributes used to name the managed objects in the OSI world).

Figure 4 shows that an adapter-agent is more than a simple gateway. Together with the CORBA application, it implements the agent role required by OSI managers and CMIS connections. In our approach, the adapter-agent assumes the largest part of the OSI burden, and the application objects remain essentially unaware of OSI — the adapter-agent is the only component object aware of OSI concepts such as OSI managed objects[1] and containment, and of related services such as scoping and filtering. It implements all by itself the OSI support managed objects (e.g., Event Filter Discriminators), logging, filtering, scoping, allomorphism, et cetera. The application participates in the implementation of the “OSI application managed objects”, by providing management interfaces — but it needs not know this (in Figure 4, “AMOF” stands for “application managed object fragment”).

In SysMan, we used a specification translation from IDL to GDMO. This has two advantages: changes to existing applications are minimized, and an undesirable bias towards OSI is avoided. We concluded from our work in SysMan that the best approach is to use an automatic translation tool, but to allow a translation designer to provide complements and improvements, and to select options. We call this approach *flexible translation*.

1. The CORBA objects represented on Figure 4 do not correspond to OSI managed objects — for example, the OSI attributes (specified in GDMO) are implemented by the adapter-agent.

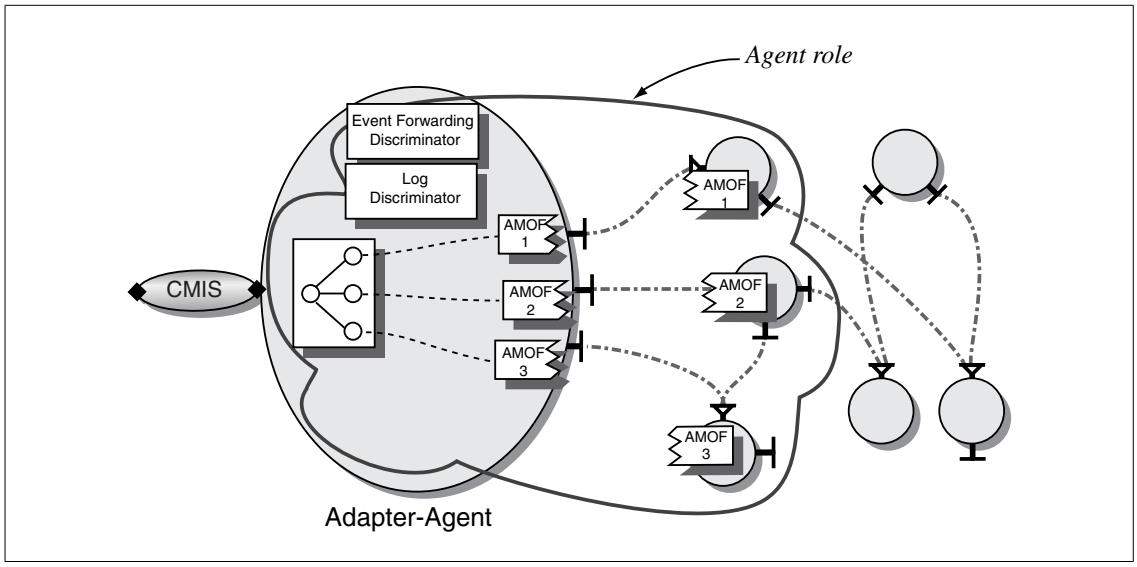

Figure 5 The adapter-agent implements the largest share of the agent role

JIDM have goals that are slightly different from ours — they are interested in using CORBA technology for implementing CMIS agents. Their initial working document requires all CORBA objects to participate in the provision of scoping and filtering, and they assume a specification translation from GDMO to IDL [HieGon96]. As a result, existing CORBA applications need to be modified in depth for being managed from OSI. Moreover, these modifications are difficult to implement, particularly for CORBA application programmers. On the other hand, all the usual features of OSI management are available, in particular the invocation of operations on multiple objects, with filtering.

4 CORBA Access to OSI Managed Objects

In the SysMan project, we decided to work on the following problem: to let CORBA management programs access OSI managed objects using CORBA interfaces, i.e, in exactly the same way they access resources and services in a distributed objects environment. This would enable the creation of application management programs which are unaware of the OSI SMA or CMIS. Thus, OSI technology could be phased out gracefully and replaced with CORBA technology.

There are two solutions to this problem: the first is to upgrade existing OSI agents; the second is to use gateways and to reuse OSI agents without modifying them. The first solution is more powerful than the second, but is excessively costly. For that reason, we attached ourselves to the use of gateways. Nevertheless, we proposed a solution which is compatible with upgrading agents, for avoiding to specify an implementation.

As shown in Figure 6, an OSI agent can be “virtually upgraded” by using a smart gateway, called a *proxy-agent* [GeGa95]. The main function of this proxy-agent is to translate CORBA operations invoked on a resource into CMIS requests, and forward these to the OSI agent. The key idea behind this approach is that the IDL interfaces proposed by a proxy-agent are derived from the GDMO specifications of the OSI managed objects in the agent — there is therefore a CMIS request for each operation

4.1 GDMO to IDL Translation

A first issue in our solution is how to derive IDL interfaces from GDMO specifications. The temptation is great to use an automatic translation. However, this is not ideal because GDMO is biased towards using CMIS for communications. For example, CMIS proposes services for reading or writing multiple attributes at once, while CORBA attribute operations are limited to one attribute. On the other hand, a manual translation would imply high specification and implementation costs.

An interesting compromise is to use a flexible translation, i.e., to allow a translation designer to make controlled improvements upon automatic translation. For example, he could specify several *multi-attribute operations*

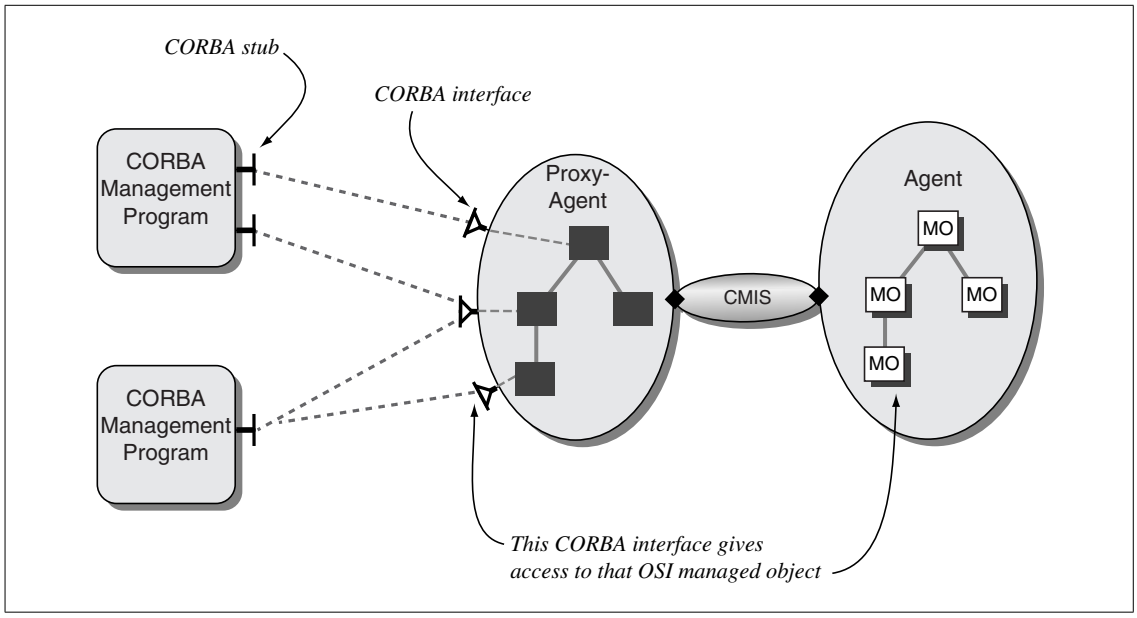

Figure 6 A proxy-agent can be used to virtually upgrade an OSI agent

(CORBA operations which access a specific group of attributes). Flexible translation is compatible with a generic implementation of the proxy-agents.

In any case, we can draw several conclusions from our observations:

- A systematic translation of GDMO specifications in IDL can result in poor designs when using CORBA operations for management.
- Companies and organisations who want to use CORBA operations for management should use IDL for specifying their management interfaces, not GDMO.
- ISO, ITU and the NMF should consider standardizing IDL interfaces corresponding to their standard GDMO managed object specifications.

4.2 Naming

Naming is a difficult issue when mapping CORBA interfaces to OSI managed objects. The naming policies and mechanisms of CORBA and OSI are quite different (even though this is not always explicitly acknowledged).

OSI relies on a global and hierarchical naming scheme, which can solve many identification problems (global names have the same meaning everywhere), but which is difficult to set up and enforce completely. CORBA proposes a federated naming service, which is more flexible but also more difficult to use (names in different contexts can mean different things). The CORBA naming service can be used as a front-end to the OSI naming service, but does OMG want to enforce a global naming scheme?

Another problem is that the OSI SMA lacks a concept of object reference, or reliable name: in general, an OSI name may refer to one OSI managed object at some time, and to another OSI managed object at a later time. This means that proxy-agents cannot just rely on OSI names for identifying OSI managed objects— if they do, they can violate the referential integrity of their clients (in other words, the encapsulation of the objects that hold these references).

Fortunately, a proxy-agent can be configured so that it implements object references correctly, or with a very high probability (i.e., so that it truly associates a CORBA interface with one and the same OSI managed object). For example, it can use the *unique identification attribute* of some OSI managed objects, or it can get notified when an object is deleted in the agent. However, the required configuration information cannot be automatically deduced from the GDMO specifications — it must be provided by a translation designer. In some sense, a flexible translation is necessary.

4.3 Common Services

The main benefit of proxy-agents is that they isolate CORBA management programs from the OSI architecture. However, as illustrated in Figure 7, common services such as naming, trading, factory, property, query, and notification will probably need to be aware of OSI or proxy-agents, or even both. A distributed objects architecture that encompasses OSI managed objects must therefore specify common services in a careful way.

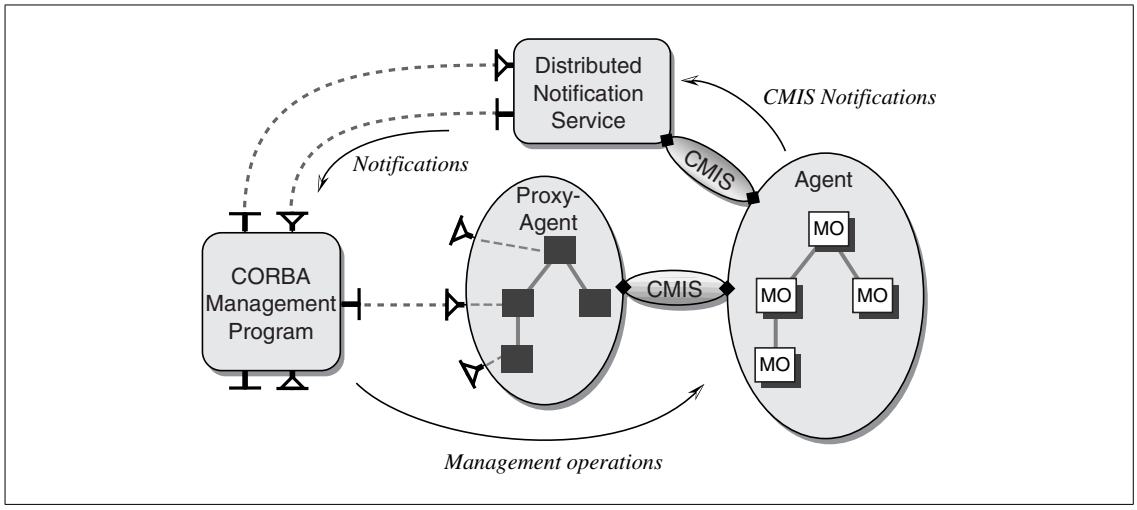

Figure 7 Notifications may be handled by a dedicated management service object

For example, a distributed notification service needs to know in which OSI agent an OSI managed object resides. It also needs to participate in the provision of location transparency regarding that managed object.

4.4 Scoping, Filtering, and Multiple Selection

As already mentioned, the OSI SMA supports a general mechanism (scoping and filtering), that allows to invoke a management operation of several OSI managed objects at once. It is possible to provide this service to CORBA management programs by using generic operations that can return multiple results. This can be valuable of course, but these operations are difficult to use, and they expose the OSI architecture (and one of its drawbacks). If management programs overuse them, the evolution towards a distributed objects architecture might be compromised.

An alternate approach is to define higher level services that obviate the need to resort to scoping and filtering. In the ideal case, these services will be standardised and integrated within distributed services. But companies will probably need to define and implement ad hoc services of their own, on top of the CMIS service. A proxy-agent, that is organised as a flexible programming environment, will allow them to do so quite easily.

4.5 Completeness and Evolution Towards a New Architecture

An important difficulty in our integration work was the lack of a reference management architecture in the CORBA domain. In SysMan, we chose to make only a few “safe” assumptions about such an architecture, and we provided only partial solutions (i.e., we provided solutions to be completed while developing a new management architecture). Our view was that management applications should access OSI directly when they need extensive OSI features (they may however be decomposed in an OSI part and a CORBA part).

Facing the same problem, JIDM has decided to work on the realization of the OSI architecture in a CORBA distributed environment. This allows them to make all the features of the OSI architecture available in the CORBA domain: generic operations for accessing multiple attributes, scoping and filtering, global names, et cetera. The drawbacks of their approach are that writing management applications in CORBA requires an important knowledge of OSI, and remains as difficult as in OSI. Moreover, the characteristics of the OSI architecture will probably conflict with those of a truly distributed objects architecture for systems management.

5 Conclusion

The OSI Systems Management Architecture and the distributed objects technologies can be integrated, and they can lead to a powerful and ubiquitous management architecture. Tools such as automatic translators, from GDMO to IDL or vice-versa, and special gateways, can provide invaluable services in support of this integration. However, several issues need to be looked at specifically to avoid undermining the benefits of distributed objects. In particular, solutions are needed to overcome the GDMO bias towards CMIS communications. New management distributed services must replace the location dependent and overly general services of OSI. The naming issue must be truly addressed, and the referential integrity of object references must be enforced.

When integrating the worlds of distributed objects and OSI, we need to revisit the features of OSI and to adapt them. We should not blindly adopt them.

Acknowledgments

The work mentioned in this paper was funded by the ESPRIT III project SysMan through the OFES, and by the Swiss Telecom.

6 Bibliography

[BGG94] K. Berrah, D. Gay and G. Genilloud, "Accessing ANSA Objects from OSI Network Management," in Proceedings of the Fifth IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM'94), Toulouse, 1994.

[ANSAware] ANSA, "ANSAware 4.0 Application Programmer's Manual," Technical Report, March 1992.

[CORBA-2] OMG, "The Common Object Request Broker: Archictecture and Specification (2.0)," Object Management Group. 1995.

[Day92] J. Day, "Comparing SNMP and CMIP or What the World Needs is A Good Five Cent Management Protocol," Orchestra Working Paper #4. unpublished paper, private communication. 1992.

[DJ96] A. Debski and E. Janas, "The SysMan monitoring service and its management environment," Distrib. Syst. Engng, vol. 3, no. 2, pp. 136—147, 1996.

[GeGa95] G. Genilloud and D. Gay, "Accessing OSI Managed Objects from ANSAware," in Proceedings of the Sixth IFIP/ IEEE International Workshop on Distributed Systems: Operations and Management (DSOM'95), Ottawa, 1995.

[Genil96a] G. Genilloud, "An Analysis of the OSI Systems Management Architecture from an ODP Perspective," in Proceedings of the IEEE Second International Workshop on Systems Management, pp. 72--81, Toronto, 1996.

[Genil96b] G. Genilloud, "Towards a Distributed Architecture for Systems Management — PhD. Thesis 1588," Computer Science. Swiss Federal Institute of Technology of Lausanne (EPFL). 1996.

[HieGon96] J.J. Hierro and J.A. González, "Common Facilities for Systems Management," TELEFONICA I+D SUBMISSION — Code : 9542-010-3415. 1996.

[JIDM97] X/Open and NMF, "Inter-domain Management: Specification Translation," Open Group Preliminary Specification P509. March 1997.

[X.700] ISO/IEC and ITU-T, "Open Systems Interconnection - Basic Reference Model - Part 1: Management Framework," Standard 7498-4, Recommendation X.700. 1989

[X.701] ISO/IEC and ITU-T, "Open Systems Interconnection - Systems Management Overview," Standard 10040-2, Recommendation X.701. 1992.

[X.720] ISO/IEC and ITU-T, "Open Systems Interconnection - Structure of Management Information: Management Information Model," Standard 10165-1, Recommendation X.720. 1992.

[X.722] ISO/IEC and ITU-T, "Open Systems Interconnection - Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects," Standard 10165-4, Recommendation X.722. 1992.