An Analysis of the OSI Systems Management Architecture from an ODP Perspective

Guy Genilloud

Swiss Federal Institute of Technology, Lausanne

Computer Engineering Department

EPFL-DI-LIT

CH-1015 Lausanne, Switzerland

email: guy.genilloud@di.epfl.ch

Abstract

This paper analyses the OSI Systems Management Architecture (SMA) in terms of the RM-ODP concepts and architecture. It explains why ISO and ITU are considering new modelling techniques for implementing distributed systems management. In the information viewpoint, these new techniques might be inspired from GDMO. The paper also examines the use of automatic translation tools (GDMO to CORBA IDL translators) to integrate existing management agents within the future Open Distributed Management Architecture (ODMA).

1 Introduction

Until recently, network or systems management applications were typically centralized; they were dealing essentially with the management of individual network elements or other non-distributed resources within networked systems. Today, management applications are more and more concerned with the whole network and with distributed applications. Thus, new management applications tend to be distributed and they manage distributed systems. For that reason, telecommunications operators want to make their management systems and models evolve towards the architecture defined by the ODP-RM (Open Distributed Processing Reference Model) [3, 8, 9]. The TINA (Telecommunications Information Network Architecture) [1], developed by the TINA consortium, and more specifically the ODMA (Open Distributed Management Architecture) [21], developed jointly by ISO and ITU-T, are examples of this evolution.

Telecommunications operators may also be encouraged to embrace the ODP architecture because of their interest for the CORBA technology [13, 12]. CORBA standards (e.g. the interface definition language and the definition of interface references) are indeed much better accepted than OSI standards by the IT community. Their application to implement a system conformant to the ODP architecture is also more straightforward than OSI standards.

However, significant investments have already been made on the basis of the OSI SM (OSI Systems Management) standards. Such investments should be preserved; current implementations, or at least current specifications, should be reused, even though they may need to be enhanced or modified. This will be possible if the SMA (OSI Systems Management Architecture) [14, 15, 16, 17, 18, 19, 20] is well integrated within the ODMA. In this context, the joint work of X/Open and the Network Management Forum on inter-domain management specifications is of great relevance [6, 7].

To integrate the SMA within the ODMA, it is essential to have a clear understanding of the SMA and of its relation with the ODP-RM. In the first part of this paper, we analyse the SMA in terms of the ODP concepts and architecture. We conclude that the main deficiency of the SMA for implementing distributed systems management lies in its modelling technique. We also show that MOs (Managed Objects) can be considered as information viewpoint objects. In the second part of the paper, we look at new designs for management agents so that they can be accessed through ODP/CORBA "RPC" protocols. More specifically, we compare the redesign of an agent produced by human designers to that obtained automatically by GDMO to CORBA IDL translators. We then draw some conclusions.

We specifically address an audience interested in distributed systems and in network and systems management. We assume that the reader is at least vaguely familiar with the ODP-RM and the OSI SMA.

2   Interpretation and Use of ODP Viewpoints

The ODP-RM introduces the concept of viewpoint, which is an abstraction, i.e., a model, that focuses on particular concerns within a system [3, clause 8.1.1] [8, clause 3.2.7]. The ODP-RM specifies that five viewpoints are necessary to specify a system conformant to the ODP architecture: the *enterprise, information, computational, engineering* and *technology* viewpoints [9]. Today, there is a wide agreement within the ODP community both on the interest of using viewpoints and on the specific five that have been selected by the ODP architecture. However, there is still debate regarding the precise scope of the viewpoints (see for example the discussion in [4]). This section presents our views on that topic and explains the main reasons why our study is essentially made in the information and the computational viewpoints.

The enterprise viewpoint is "a viewpoint on an ODP system and its environment that focuses on the purpose, scope and policies for that system". We consider an enterprise model to be a way to record all the requirements on a system or a component within that system; an enterprise model may thus be used to record the purpose and rationale of an MO definition. Unfortunately, the OSI SM standards generally do not include the requirements or the rationale that lead to their development. Thus, we consider that the OSI SM standards did not make use of the enterprise viewpoint and do not address it further in this paper.

The information viewpoint is "a viewpoint on an ODP system and its environment that focuses on the semantics of information and information processing". We consider an information model to be the result of the analysis of a system: ideally the model describes everything the system does without mentioning anything about how it does it (so as not to constrain implementations). Perhaps more restrictively, we also consider that an information model provides its user (e.g. a designer of a larger system) with all the necessary information to use the system. This implies that all the assumptions made about the environment of the system need to be specified explicitly.

The computational viewpoint is "a viewpoint on an ODP system and its environment which enables distribution through functional decomposition of the system into objects which interact at interfaces". We consider a computational model of a system (excluding its environment) to be a configuration of objects, all with computational interfaces (i.e., either signal, stream or operation interfaces); this configuration may be seen as a refinement of the system in the informa-

tion model, and is behaviorally compatible with it. The behaviour of each object in the computational model is unconstrained, and it may be specified by any means deemed appropriate [11]. One possibility amongst others is to consider each computational object as a system and to apply the viewpoints recursively to it; its behaviour is then specified in an information viewpoint model.

The engineering viewpoint is "a viewpoint on an ODP system and its environment that focuses on the mechanisms and functions required to support distributed interactions between objects in the system". Clearly, the OSI communications infrastructure can fulfill a large part of this. However, because of the differences in modelling used by ODP and OSI, it is very difficult to relate precisely the OSI-RM concepts to those of the ODP engineering viewpoint. We avoid the difficulty by encapsulating OSI associations within computational binding objects: since we know how to create such objects (OSI associations), we do not need to model them in the engineering viewpoint.

The technology viewpoint is "a viewpoint on an ODP system and its environment that focuses on the choice of technology in that system". We are in principle not much concerned by this viewpoint because the ODP-RM imposes nothing regarding technology. However, we assume that the systems management community will adopt the CORBA IDL, which is a future ODP standard.

3   Analysis of the SMA

To apprehend the OSI SM standards, it is necessary to first consider their purpose and scope, namely the specification of the communication protocols to be used for the management of OSI networks, and the specification of the management information carried by those protocols. Since only interworking reference points may be imposed by OSI standards, total freedom must be left as to how real open systems are implemented. Moreover, the algorithms to be used for performing management are in principle not standardized. Thus, the OSI SM standards focus on a "core management capability " which is centered around *the control or monitoring of resource usage.*

The SMA specifies a few rules for the construction of distributed management systems [15]. Because these rules pertain to the distribution of the management system, they are best explained in the computational viewpoint.

3.1   The SMA in the ODP Computational Viewpoint

Basically, we see the SMA as an architecture which rules that any computational object be of one of two

object types: an MIS (Management Information Service) type or an MIS-User type. An *MIS object* models the communication support provided to two MIS-User objects; it is therefore a binding object. An *MIS-User object* models a complete real open system (except the communication software and hardware which are abstracted within MIS binding objects); it is an ordinary computational object[1]. Figure 1 shows the computational model of a very simple system conformant to the SMA: it comprises only two MIS-User objects bound by an MIS binding object. Figure 2 gives a more complex example of a management organization with two "top peer managers", each managing other systems in a hierarchical way.
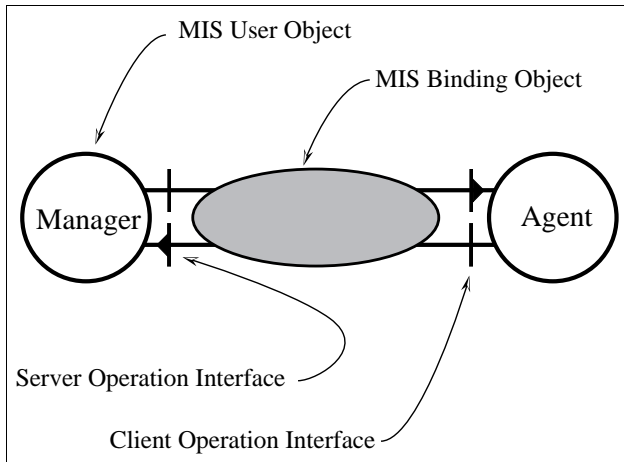

Figure 1: A manager bound to an agent

The SMA is very prescriptive regarding the behaviour of MIS binding objects: their behaviour is fully specified in the CMIS (Common Management Information Service) definition [16]. It is important to note that this behaviour specification implies that two MIS-User objects bound by an MIS binding object must assume two distinct *roles* with respect to each other: one must assume a manager role, the other an agent role. *Manager roles* have a management mission; they issue management operation requests and they receive notifications. *Agent roles* perform the management operations that they receive, and emit notifications of significant events towards managers.

---

[1]As for any computational object, it is possible to refine an MIS-User object into a composition of computational objects. The SMA does not impose anything regarding this point (so an agent may be distributed). As for MOs, we will show in the next section that they are information objects, not computational objects.
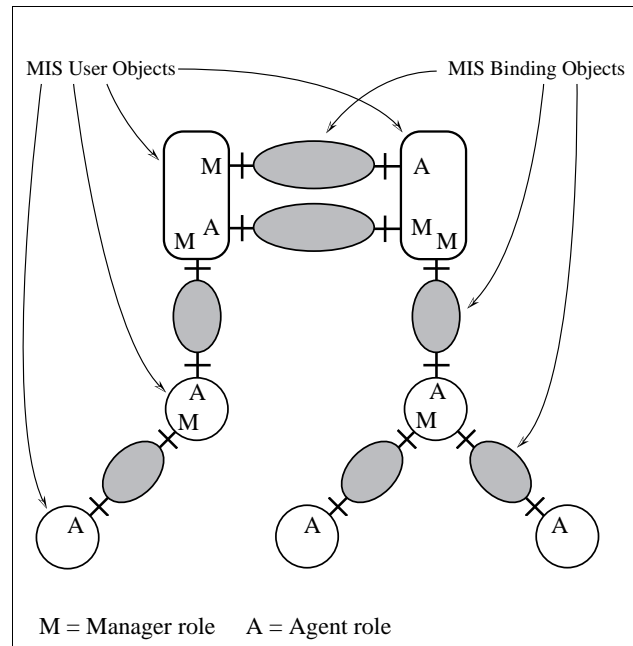

Figure 2: Example of a management organization

An MIS object is therefore an asymmetrical binding object.

However, being not concerned with portability, the SMA does not prescribe in detail the interfaces that are to be supported by an MIS binding object: any set of interfaces that is compatible with its behaviour is acceptable. Figure 1 illustrates a possible way to model a MIS binding object using exclusively computational operation interfaces; tables 1 and 2 give the signatures of those interfaces (we do not fully specify the arguments of these announcements, but those can easily be deduced from the CMIS or the CMIP specifications). Using such interfaces, the behaviour of an MIS binding object may be roughly summarized as *forwarding announcements and maintaining their order of delivery*. Note that we do not use interrogation operations because the SMA allows operations to have zero, one or more replies, whilst interrogations have exactly one. Note also that we ignore all actions related to the instantiation and the deletion of MIS binding objects; for the sake of simplicity, we do not model the set-up and control of associations.

In the way we presented them, the interfaces of an MIS binding object are apparently very simple: they contain very few operations. However, the arguments of those operations may be very complex and their values and types can differ a lot from one invocation to

```
M-Get ANNOUNCEMENT(InvokeId, BOC, BOI, ..., AttributeList)
M-Cancel-Get ANNOUNCEMENT(InvokeId, GetInvokeId)
M-Set ANNOUNCEMENT(InvokeId, Mode, BOC, ..., ModificationList)
M-Action ANNOUNCEMENT(InvokeId, Mode, BOC, ..., ActionType, ActionInfo)
M-Create ANNOUNCEMENT(InvokeId, MOC, MOI, ..., AttributeValueList)
M-Delete ANNOUNCEMENT(InvokeId, BOC, BOI, ..., Synchronization)
M-Event-Report-rsp ANNOUNCEMENT(InvokeId, MOC, ..., EventReply, Errors)
```

Table 1: Announcements invoked by a manager on an agent.

```
M-Get-rsp ANNOUNCEMENT(InvokeId, ..., CTime, AttributeValueList, Errors)
M-Cancel-Get-rsp ANNOUNCEMENT(InvokeId, Errors)
M-Set-rsp ANNOUNCEMENT(InvokeId, ..., AttributeValueList, CTime, Errors)
M-Action-rsp ANNOUNCEMENT(InvokeId, ..., CTime, ActionReply, Errors)
M-Create-rsp ANNOUNCEMENT(InvokeId, ..., AttributeValueList, CTime, Errors)
M-Delete-rsp ANNOUNCEMENT(InvokeId, ..., MOC, MOI, CTime, Errors)
M-Event-Report ANNOUNCEMENT(InvokeId, Mode, ..., EventTime, EventInfo)
```

Table 2: Announcements invoked by an agent on a manager.

another. Moreover, different agents may respond to a same announcement (by issuing further announcements) in quite different ways. Indeed, CMIS does not specify the semantics of most of the arguments that are exchanged by managers and agents. Some of these arguments (e.g. AttributeValueList, ActionInfo, EventInfo) change so much between invocations that the only practical way to declare them to the MIS binding object is as sequences of bytes. Encoding and decoding must then be performed within the MIS-Users, possibly by a special layer. Event though an application may not need to perform encoding or decoding itself, it may have difficulties to specify or to interpret an argument. Moreover, encoding arguments amounts to escaping the computational type checking mechanisms; so these mechanisms would be useless to catch most errors. For these reasons, APIs (application programmatic interfaces) for CMIS are difficult to use.

The SMA imposes very little regarding the behaviour of MIS-User objects: the main requirement is that they use MIS binding objects to exchange management information among themselves. Fundamentally, this is not an overly restrictive requirement because two MIS-User objects may be bound by two MIS binding objects in opposite directions (see figure 2). In this way, both MIS-User objects assume both a manager and an agent role w.r.t. each other; both objects

are thus capable of invoking any operation on each other (the SMA imposes little semantic constraints on M-Actions), just like they would be able to do if MIS binding objects were not imposed! In other words, what can be done with CORBA may also be done with CMIS.

3.2 Agents are Modelled in the Information Viewpoint

The SMA is independent of the missions attributed to management systems and of the resources to be managed. Thus, it does not define the behaviour of the MIS-User objects nor even the universe of discourse between them; this is the role of OSI SM standards. However, the SMA proposes a modelling technique, to be used by OSI SM standards, for specifying the universes of discourse of MIS connections. This technique includes a notation called GDMO (Guidelines for the Definition of Managed Objects)[20].

From an ODP perspective, the *SMA modelling technique* consists in specifying an agent role: the universe of discourse of the MIS connection is then the definition of all the messages that can be received or emitted by the agent on that connection. The SMA modelling technique thus consists in a partial specification of an MIS-User (only the part of the behaviour that is relevant to management, and specifically to the agent role, needs to be specified). Moreover, it imposes that only the pieces of specification that are

directly relevant to the universe of discourse of the MIS association be formally specified; the rest may be specified informally in a natural language. Since the SMA intends not to constrain the implementation of an MIS-User, it only requires an information viewpoint specification of its agent role.

The SMA defines a modular approach to specify the behaviour of an agent according to the managed resources that it contains; this approach is based on special information objects known as *managed objects* or *MOs*. For specifying the behaviour of these MOs, the SMA suggests a method that is very compatible with the information viewpoint: it uses terms such as "invariants", "preconditions", "postconditions", "consistency constraints" and "relationships" [18, clause 5.1.2.4].

The information model of an agent behaviour is given by a configuration of MOs, and by an extra object that is only specified implicitly by the SMA. We call this extra object the "dispatcher"[2]. The function of the dispatcher object is to handle MIS communications, to resolve object names, and to dispatch operations towards MOs. The dispatcher also sends replies and notifications to managers which expect them. Importantly, it is allowed to discard notifications when it is overloaded. All other information objects, i.e., the MOs, are specified by OSI SM standards.

An ODP viewpoint model requires that the environment of a system be specified. GDMO allows to model the environment of an agent in an implicit way: the environment is any system that can invoke operations on the agent, and receive its replies and its notifications. The signatures of the agent operations and notifications are therefore an integral part of its information specification. It is important to note that these signatures are only partially specified in GDMO; CMIS specifies a few additional arguments (e.g. current time, access control) that can be very important.

3.2.1 Limitations of the SMA Modelling Technique

The SMA modelling technique and its associated notation, GDMO, are adequate for their purpose, i.e., specifying the universe of discourse of CMIS associations. However, they are not suited for specifying systems in general. This is perhaps not obvious because one might choose to use GDMO for specifying a distributed system and still produce a good design.

---

[2]The SMA, in particular [18], uses the term "agent" to refer to the entity (it would not call it an object) which interacts directly with the MOs. We prefer to use the term "dispatcher" to avoid confusion with the agent role of an MIS-User.

But that is because GDMO allows to use natural language, and everything can be said in natural language! The problem with GDMO is that its formal part is biased towards specifying agent behaviours; this leads to biased or unbalanced specifications.

As we noted already, the SMA modelling technique has very limited power to specify the environment of a system. It is thus inadequate for specifying systems which have non-trivial contracts with their environment. Since a system designed in this way cannot rely on its environment for its proper functioning, one might argue that the SMA modelling technique is biased towards good design. Still, there are cases when this is an unacceptable limitation.

A more important flaw of the SMA modelling technique is that it is only adequate for specifying the behaviour of an interface (i.e., an abstraction of an object behaviour [8]) corresponding to an agent role. The problem is that the behaviour of an interface may be of little use to determine the complete behaviour of an object. Indeed, it is not sufficient to specify the behaviour of the other interfaces since inferring a behaviour from several of its abstractions can be a very difficult task. On the other hand, the opposite (i.e., deducing the behaviour of an interface from that of the object) is relatively easy. For that reason, the unit of specification should be the object, not the interface.

The above argument is very abstract. The following example illustrates the problem more precisely and more concretely. We consider the specification of a management mission to be performed in collaboration by two MIS-Users, say A and B; the management mission can be any kind of mission that needs A and B to collaborate, e.g. the performance of a series of tests on the communication links between them; we are more specifically interested in the information specification of A.

The mission may be specified by putting an MO, say MO_A, within A; a manager (it may be A itself, or B, or yet another MIS-User) would initiate the mission by invoking an action on MO_A, say Act. Since this mission is collaborative, A needs, on several occasions, to send intermediate results to B, and to request it to perform some parts of the mission. Using the replies to Act for doing this is not possible because B is not necessarily the invoker of the action. Using notifications is not adequate because there is no guarantee that they will be delivered to B, and because B cannot reply to MO_A directly. The best solution is to specify other MOs within B, and to let MO_A invoke operations on these MOs. Unfortunately, this can only be specified within MO_A in natural language: there

is no formal way in GDMO to specify or to reference the signature of the operations that are invoked by an MO. It is thus likely that the information specification of A will not stand by itself, and that the information specification of B will be needed for understanding it. So, a designer will be tempted to misuse notifications to avoid this problem.

In summary, the operations invoked by an MO (on other agents) are as important as the operations performed by that MO; both kinds should thus be treated equally, or not treated at all (GDIO [4], an information specification notation proposed by ITU SG15, consists in GDMO with all its "operation aspects" removed). As for notifications, it is important to understand their semantics. The designer of a subsystem should never assume that they will be received by anyone, even though it might be desirable that they are received and acted upon. For example, a printer may send notifications that it is out of paper (there is nothing it can do about it), but it should invoke operations on a font server if it needs to load fonts when printing a file.

4   The SMA in the ODMA

4.1   The ODMA

The ODMA suite of standards is at an early stage of development, and it is difficult to predict in detail what it will be. However, it is already clear that the ODMA standards will extend and modify the SMA modelling techniques and notations, so as to fully cover the needs of management. They will propose notations for expressing models in the enterprise and the computational viewpoints. An example of the latter is the TINA-ODL notation [10]. Regarding the information viewpoint, it is likely that the ODMA standards will extend or modify the GDMO and GRM notations so that they can be used for specifying the behaviour of any system, and not just agent roles. A rationale for this choice is that it should facilitate the reuse of existing MO specifications. Examples of adapted GDMO notations are GDIO [4] and Quasi-GDMO+GRM [2].

The ODMA will also recognize the existence of managing objects, in addition to MOs. That is, some ODM standards will explain how to structure and distribute managing applications.

The ODMA will probably recognize all the classes of ODP reference points, in particular the class of *programmatic reference points*. This is important because considering the functionality of an application programmatic interface can shade a new light on some issues such as naming and allomorphism (see our discussion of naming and typing issues below). Also, the

bindings of the computational notation to programming languages become an important issue.

4.2   Evolution of Computational Models

The ODMA standards will probably standardize the way existing management systems will be integrated within the more general ODMA framework. Indeed, ISO and ITU are already planning to recognize and use the standards being developed by JIDM for that purpose: the specification translation between GDMO and CORBA IDL, and the interaction translation between CMIP and the CORBA protocol. Automatic translation and related tools (e.g. gateways) will ease the transition. However, an alternative for ISO and ITU would be to refine manually their "MO standards" and to standardize the operation interfaces [9] to be supported by managed systems.

In the reminder of this paper, we are interested in evaluating the pertinence of using translation tools; we want to compare the CORBA IDL specifications they can produce with those that may be produced manually by designers. For this mini-study, we consider the refinement of an agent so that it can be accessed through ODP/CORBA protocols rather than or in addition to CMIP. We assume that an ODP/CORBA system is available on the agent; thus, gateways are not an issue.

Figure 3 illustrates the transformation that we investigate. Considering a system which has a single CMIS interface, we change it so that it offers operation interfaces in addition to or in replacement of that CMIS interface. Since we are interested in the capabilities of automatic tools, we only consider the signatures of the new interfaces exhibited by the agent; we do not consider the internal design of the agent. We limit our study to operations which are invoked on a single MO.

4.2.1   Interfaces and Operations

MOs define operations that can be invoked on an agent. Considering the function of the dispatcher object, an agent should logically offer a server interface for each MO; this interface would contain all the attribute and action operations that may be invoked on the MO, plus optionally other operations (e.g. subscription to the notifications of the MO [5]).

GDMO and CORBA IDL allow operations to be defined as attributes. Because of differences in the inheritance rules, automatic translation maps all GDMO attributes to IDL operations [6]. A designer would probably use some IDL attributes, but this does not really matter. The translation of attributes proposed in [6] is therefore a good design.
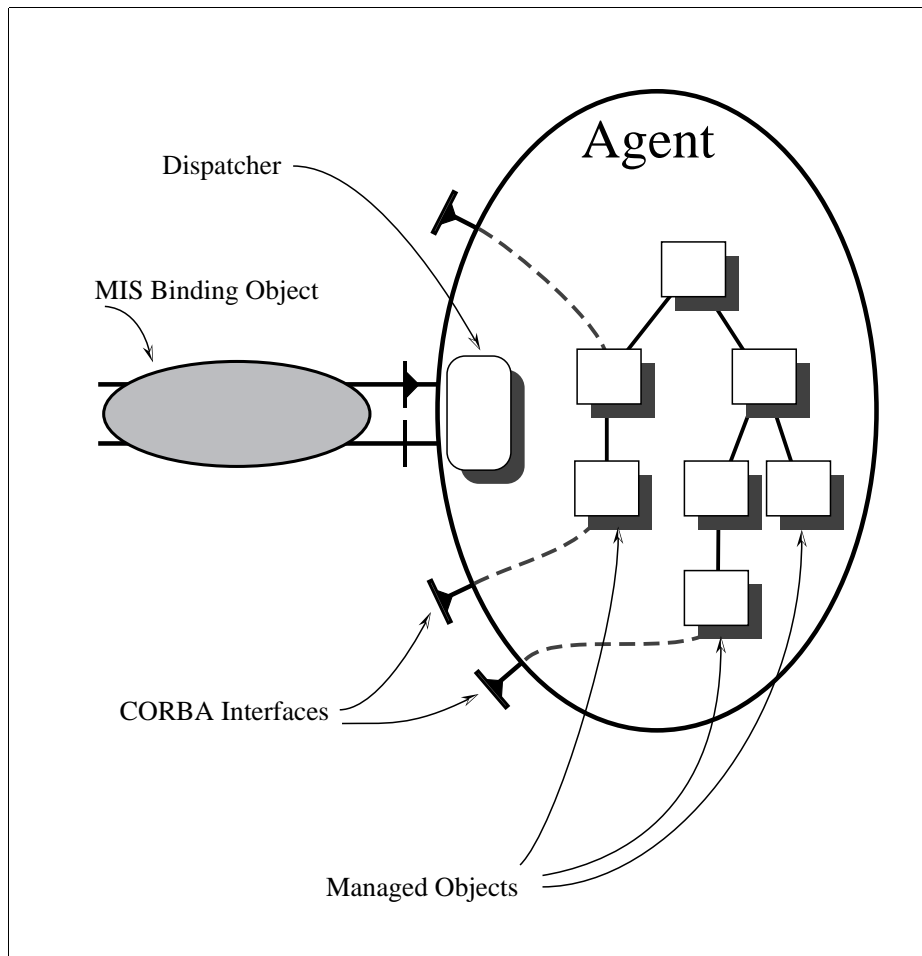
Figure 3: Transformation of an Agent for using CORBA

Many actions have a single reply; they are thus interrogations and they should appear as such in the interface. Other actions do not return any information and may be invoked in confirmed mode or unconfirmed mode. Considering the use of an "RPC protocol", automatic translation can map them all to interrogations. Indeed, an interrogation termination is the only way for a client to find out about communication problems since it knows nothing about connections.

A few actions may generate one or several replies. In this case, a good idea is to translate the operation into an interrogation with a normal termination to be used when there is a single reply, and an alternate termination (exception) to be used otherwise: the alternate termination indicates that there will be multiple replies; call-backs are then used to carry those replies [6]. An input argument in the action may indicate to the agent which call-back interface to use.

Interrogations which may send call-backs are much more complicated to invoke than other interrogations, so only those actions which may send multiple replies should be translated in this way. In GDMO, the use of multiple replies can only be indicated in the behavioural description of the action, i.e., in natural language. Thus, an automatic tool cannot translate actions in an optimal way.

4.2.2 CMIS Arguments

All operations on MOs invoked through CMIP inherit some CMIS arguments which are not always useful, but which can sometimes be necessary.

The *current time* argument might be useful in several circumstances, for example when receiving multiple replies. The *filter* argument might be necessary

to allow a manager to invoke an operation while specifying a precondition on the state of the MO. More importantly, security should be considered with great care. One cannot be sure now that the access control argument will never be needed, e.g. for deciding precisely whether an operation should be authorized or not.

A designer may decide which of the CMIS arguments are useful for a given operation. An automatic tool needs to always include these arguments, thus cluttering many operations unnecessarily. The translation proposed in [6] currently does include CMIS arguments for notifications, but it does not for operations.

4.2.3 Typing Issues

The OSI SMA uses a complicated mechanism, called allomorphim, to allow the substitution of an MO of a given class for an MO of a superclass of that class (independently of inheritance considerations). In ODP, an operation interface supports automatically all the supertypes of its type. Thus, there is no need to use the OSI mechanisms on top of the ODP/CORBA mechanisms.

In ODP and CORBA, an interface generally does not provide operations returning the name of its actual class, and even less the names of its superclasses. This is not necessary and is even considered highly undesirable by ODP. Thus, designers should carefully consider whether they should put operations to read the values of the attributes *ManagedObjectClass* and *Allomorphs*. They should only include these operations if they have a requirement to do so. Of course, automatic translation will either always include those operations or never.

MOs can pass names of other MOs as arguments of operations. A translated argument can be a name too, but it can also be an interface; in both cases, the argument carries a reference to an interface. Regarding typing, the difference is that the type of an interface argument is the type of the interface (the actual interface – the principal interface in CORBA – has to be a subtype of that type) while the type of a name argument is just the datatype of that name. This difference matters because the types of interface arguments can be used for signature-based type checking.

Designers of an MO certainly know whether they want to pass a name or an interface as the argument of an operation. Usually, they also know the types of the MOs that can be passed as arguments of an operation. Thus, they would often decide to use interface arguments rather than names when using CORBA-

IDL. Automatic translation cannot make this kind of decision; moreover, it cannot determine the type of interface arguments.

4.2.4 Naming Issues

In ODP, when an interface is passed by reference as an operation argument, an interface reference is passed by copy instead. Interface references are *reliable*, i.e., they always refer to the same interface or to no interface at all if that interface is not available (probably because it was deleted). This reliability property – called *referential integrity* in CORBA – is an essential property of object-based systems because interface references are part of the state of an object.

MO names, as defined in the OSI SMA, are not reliable because they may be reused when an object is deleted. They are nevertheless very useful because they can be read by humans and because they can be compared for equality (i.e., it is possible to determine whether two MO names refer to a same MO or to two different MOs). For example, an MO name rather than an interface argument should be used to indicate the origin of a notification.

ODMA will therefore need to define a naming framework comprising both kinds of names. This framework should in particular explain which naming servers (e.g. ODP traders, the X.500 directory, or the agents themselves) can be used to obtain interface references given an MO name.

Regarding automatic tools, the lack of reliability of OSI names is a problem for building gateways; it is indeed impossible in general to build an interface reference by encapsulating an OSI name within it [5].

5 Summary and Conclusions

In the first part of this paper, we analysed the OSI SMA in terms of the ODP concepts and architecture. We concluded that the main deficiency of the SMA for implementing distributed systems management lies in its modelling technique. New modelling techniques, as outlined in the ODP-RM, are therefore needed urgently. In the information viewpoint, these new techniques might be inspired from GDMO and GRM. Indeed, MOs can be considered as information viewpoint objects.

In the second part of the paper, we looked at new designs for OSI management agents so that they can be accessed through ODP/CORBA "RPC" protocols. We compared (aspects of) a good redesign of an agent with that produced automatically by a GDMO to CORBA IDL translator. We showed that automatic translation lacks sometimes information which is im-

portant to produce a good design in the spirit of ODP or CORBA.

In the light of these results, ISO and ITU should consider whether to use automatic translation tools for producing standards for operation interfaces, or whether to ask working groups to specify those interfaces manually using CORBA IDL. A combined approach is possible; it might also be desirable. More urgently, ISO and ITU should strongly encourage all working groups currently involved in the definition of new management standards to specify operation interfaces in CORBA IDL (at least in a non-normative annex of their standard).

Acknowledgments

References

[1] Martin Chapman and Stefano Montesi. Overall concepts and principles of TINA. TINA Baseline TB_MDC.018_1.0_94, Telecommunications Information Networking Architecture Consortium, feb 1995. Version 1.0 – Publicly released.

[2] H. Christensen and E. Colban. Information modelling concepts. TINA Baseline TB_EAC.001_1.2_94, Telecommunications Information Networking Architecture Consortium, apr 1995. Version 2.0 – Publicly released.

[3] *Draft ITU-T Recommendation X.901 (1995) | Draft ISO/IEC International Standard 10746-1:1995, Open Distributed Processing - Reference Model - Part 1: Overview.*

[4] ETSI. *ETSI, Transmission and Multiplexing (TM) – The Application of ODP to the Management of a Transport Network, Draft,* sep 1995. Work Item No: DTR/TM-2221.

[5] Guy Genilloud and David 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.

[6] Joint Inter-Domain Management Working Group. *Inter-Domain Management Specifications: Specification Translation (Draft).* X/Open and Network Management Forum, apr 1995.

[7] Joint Inter-Domain Management Working Group. *Inter-Domain Management Specifications: Preliminary CORBA/CMISE Interaction Translation Architecture.* X/Open and Network Management Forum, apr 1995.

[8] *ITU-T Recommendation X.902 (1995) | ISO/IEC International Standard 10746-2:1995, Open Distributed Processing - Reference Model - Part 2: Foundations.*

[9] *ITU-T Recommendation X.903 (1995) | ISO/IEC International Standard 10746-3:1995, Open Distributed Processing - Reference Model - Part 3: Architecture.*

[10] B. Kitson, P. Leydekkers, N. Mercouroff, and F. Ruano. TINA object definition language (TINA-ODL) MANUAL. TINA Baseline TR_NM.002_1.3_95, Telecommunications Information Networking Architecture Consortium, jun 1995. Version 1.3 – Publicly released.

[11] Elie Najm and Jean-Bernard Stefani. A formal semantics for the ODP computational model. *Computer Networks and ISDN Systems,* 27(8):1305–1329, 1995.

[12] Object Management Group. *Common Object Services Specification,* mar 1994.

[13] Object Management Group. *CORBA V2.0,* jul 1995.

[14] *ITU-T Recommendation X.700 (1989) | ISO/IEC International Standard 7498-4:1989, Open Systems Interconnection - Basic Reference Model - Part 1: Management Framework.*

[15] *ITU-T Recommendation X.701 (1992) | ISO/IEC International Standard 10040:1992, Open Systems Interconnection - Systems Management Overview.*

[16] *ITU-T Recommendation X.710 (1991) | ISO/IEC International Standard 9595:1991, Open Systems Interconnection - Common Management Information Service Definition.*

[17] *ITU-T Recommendation X.711 (1991) | ISO/IEC International Standard 9596-1:1991, Open Systems Interconnection - Common Management Information Protocol - Part 1: Specification.*

[18] *ITU-T Recommendation X.720 (1992)| ISO/IEC International Standard 10165-1:1993, Open Systems Interconnection - Structure of Management Information: Management Information Model.*

[19] *ITU-T Recommendation X.721 (1992)| ISO/IEC International Standard 10165-2:1992, Open Systems Interconnection - Structure of Management Information: Definition of Management Information.*

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

[21] *ITU-T | ISO/IEC Open Distributed Management Architecture, Workinfg Draft 3*, jul 1995. Output of the Ottawa collaborative meeting.