Accessing OSI Managed Objects from ANSAware

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

David Gay
Computer Engineering Department
EPFL-DI-LIT
Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland
dgay@cs.berkeley.edu

Abstract

This paper presents a mechanism allowing an ODP compliant distributed system, ANSA, to access OSI network management objects as if they were ANSA objects. It defines a mapping from the OSI object model to the ANSA object model, and it specifies how an adapter implements this mapping.

1 Introduction

Management of networks and management of distributed systems are interrelated; there is often a need for a network manager to manage part of a distributed system or application, and conversely, for a distributed system manager to manage the underlying network.

Both the OSI network management framework [16] and the ANSA[1] distributed system [8, 2] are object oriented. Unfortunately, the management objects of both systems live in different worlds, with no way of communicating.

Our contribution in the Esprit III project SysMan is to provide bridges between these worlds, one to allow management of OSI managed objects (*MOs*) from ANSA applications, the other to make ANSA objects appear in the OSI world [3, 5]. Two separate mappings are necessary because of the significant differences in approach taken by both systems, an attempt at finding a common subset would exclude most, if not all, existing objects. A special gateway, called an *adapter*, sits on each bridge and provides translations between the worlds.

We do not wish merely to provide access from each system to the other, we want objects from each world to appear as transparently as possible on the other side of the bridge. For instance, OSI management objects could be managed from ANSA simply by providing access to the standard OSI network management protocol, CMIP [19]. But this would not allow MOs to be handled like ANSA objects.

In related work, a group called JIDM (Joint Inter-Domain Management Working Group), composed of experts from X/Open and the Network Management Forum is aiming at providing access to MOs from a CORBA environment and the reverse [9, 6, 7]. JIDM divides its work in two parts: *specification translation* and *interaction translation*. Specification translation covers the static translation of GDMO specifications in CORBA IDL, and the reverse. Interaction translation covers the dynamic translation of the actual messages exchanged for interaction. Currently, JIDM's specification translation is nearly completed while interaction translation is at the early stages of development.

We already presented our work on the mapping from ANSAware to OSI network management in [3] and [5]. We present here our work on the reverse mapping, from OSI network management to ANSA. We specifically address an audience interested in network and systems management. Thus, we assume that the reader is familiar with the concepts of OSI network management and we do not introduce those concepts here. We do not assume that the reader is familiar with either ODP or ANSA.

This paper is organised as follows: first, we briefly introduce the ANSA model; then, we present our design goals and we introduce our approach for making MOs appear as ANSA interfaces; we then discuss the provision within

[1] ANSA stands for 'Advanced Networked Systems Architecture'; it represents an architecture for distributed systems that is very close to the 'Open Distributed Processing Reference Model' developed jointly by ISO and ITU-T (formerly CCITT) [22, 23, 24]. ANSAware is a simple realisation of that architecture. The terms 'ANSA' and 'ANSAware' are used interchangeably in this paper.

ANSA of agent-level and object-level operations in more detail; we terminate with a very short discussion comparing our specification translation with the specification translation proposed by the JIDM group.

2 The ANSA model

In this section, we introduce the main concepts of ODP [24] and ANSA [11]. Since most readers are probably more familiar with CORBA, we compare briefly the ANSA/ODP concepts with those of CORBA. We discuss our mapping between ANSA and OSI network management in the remaining sections of this paper.

Object: a unit of (distributed) program modularity having state, and *operations* for initializing , accessing and updating that state. Objects may contain *references* to the *interfaces* of both the object itself and other objects.

Interface: a view of an object as an abstract service. An interface is specified as a set of *operations* to be invoked by a client object. An object may have multiple interfaces.

Operation: part of an interface. An operation has a *signature* and a body which defines the outcome from an invocation of the operation.

Operation signature: a specification of the name of an operation, the number and (data or interface) types of the argument parameters and, optionally, a set of *terminations* which specify the possible outcomes for the operation[2].

Termination signature: a specification of a possible outcome from an operation invocation. It consists of the name for the termination and the number and types of its parameters.

Interface type: a set of *operation signatures*. An interface satisfies an interface type if it supports all the operations listed in that type.

Server: in the context of an interaction, the object which provides the interface containing the operation being invoked.

Client: in the context of an interaction, the object which invokes the operation.

Interface reference: a name that unambiguously identifies an *interface*. Interface references are never reused to identify another interface.

Trading service: A "yellow page" name service [12] provided by one or several objects.

The trading service maps a description of a service (an *interface type*, and named properties associated with the service, e.g. "location" and "quality" for a printer) to *interfaces* providing that service.

An *ANSA interface* corresponds to a *CORBA object*. There is no direct equivalent of an *ANSA object* in CORBA. The reason is that encapsulation is absolute in ANSA: all interactions with an ANSA object must occur through one of the interfaces of that object.

An *ANSA interface type* corresponds both to a *CORBA interface* and to a *CORBA interface type*. So the statement "a CORBA object may have multiple interfaces" only means that a CORBA object satisfies an interface type and its supertypes as well. Similarly, an ANSA interface may satisfy more than one interface type.

An ANSA object may have multiple interfaces independently of typing considerations: two interfaces on a same object may be instances of the same interface type or of different interface types. Each interface on an object has operations to access or update the state of that object, but not necessarily all its state. Indeed, an interface may have some state associated with it; this state, a part of the state of the object, is only accessed through operations on that interface.

An ANSA operation corresponds to a CORBA operation. An important difference is that *INOUT* parameters do not exist in ANSA. An ANSA operation may have several terminations, but these correspond to the normal results and to the exceptions of a CORBA operation. A CORBA*oneway operation* is called an *announcement* in ANSA; normal operations are called *interrogations*. Note that an interrogation has exactly one response which is one of its terminations; multiple replies are not allowed in ANSA nor in CORBA.

Some operations are defined as *attributes* in CORBA. The concept of attribute does not exist in ANSA nor in ODP.

[2] In ANSAware 4.1, there can be only one user defined termination per operation.

As CORBA operations, ANSA operations can have interfaces as parameters — these interfaces are passed by reference. This is achieved in practice by passing (by copy) names called *interface references* in ANSA and *object references* in CORBA; these names are invisible (in principle) to application programmers[3].

The ANSA trading service is a "yellow page" name service that allows a client to find servers given the identification of a trading context and a description of the characteristics of that service. The trading service is in several respects similar to the CORBA naming service [10].

An important difference between ANSA and CORBA is that ANSAware provides no *dynamic invocation interface*. This implies that an ANSA object can only invoke an operation of which it knows the signature at compile-time.

3 Design Goals

Our aim is to provide access to OSI network management [17] objects from the ANSA distributed system. The simplest method is to provide an access to the CMIS [18] service which is the standard way of accessing such objects. However this approach has several disadvantages:

- It requires the programmer to learn a new and complex object model [17, 20], with a new object notation, GDMO [21].
- He or she must also learn to use CMIS [18], the service that gives access to MOs. Being defined independently of the object classes, CMIS is complex to use[4].
- The MOs are not accessed via ANSA interfaces, so no standard ANSA services and mechanisms may be used with them (for instance they may not be placed in a trader).

We propose an approach that attempts to circumvent most of these problems. The differences in approach and modelling taken by the two systems make this quite difficult, so any solution is imperfect. But at least the programmer's job can be made easier.

Our design has been guided by several conflicting criteria:

1. Simplicity: complex solutions are difficult to understand, take a long time to implement and have subtle bugs.
2. Ease of use: we want to simplify the access to OSI network management. If our proposal is more complicated than using CMIS directly, nobody will use it.
3. Completeness: within the bounds imposed by the previous two constraints, we want to provide complete access to the facilities offered by an OSI agent.

Our approach is not without disadvantages. In particular, it is not possible to hide all the specificities of OSI management, so the programmer will still have to learn about some of its aspects. It is not possible to provide full features and simple use, so we needed to make trade-offs. For example, we chose not to support operations on multiple OSI objects.

Our approach and the tools we provide will be very useful for building specialised managers that know before runtime the types (or the supertypes) of the MOs they will be dealing with. However, they will be of little use for building very general managers, such as a "network object browser". In that case, it is better to use the CMIS service directly.

4 Overview of the Mapping

ANSA objects access services or information provided by other objects by invoking operations at their interfaces. This is not the case in OSI, where all access to MOs is mediated by an agent. We chose to hide the participation of the agent by representing an MO as an interface in the ANSA world; the actions and attributes provided by an MO become operations of its corresponding ANSA interface.

[3] Interface reference should not be confused with *interface identifiers*, which are variables (or constants) which a programmer uses to denote interfaces. Interface identifiers are names chosen by a programmer, and are therefore simple unstructured names (e.g. $myServer$).

[4] A usability study of XMP, the standard application programming interface for CMIS, concluded that XMP was extremely difficult to use [1]. We think that this can be attributed both to XMP and to CMIS.

However, we cannot completely hide the role of the agent in ANSA; there are agent-level operations, e.g. creation of objects and subscription to notifications, that we need to make available in the ANSA world. These operations are explicitly provided by an adapter which is thus acting as a proxy agent in the ANSA world. We chose to have an adapter per agent.

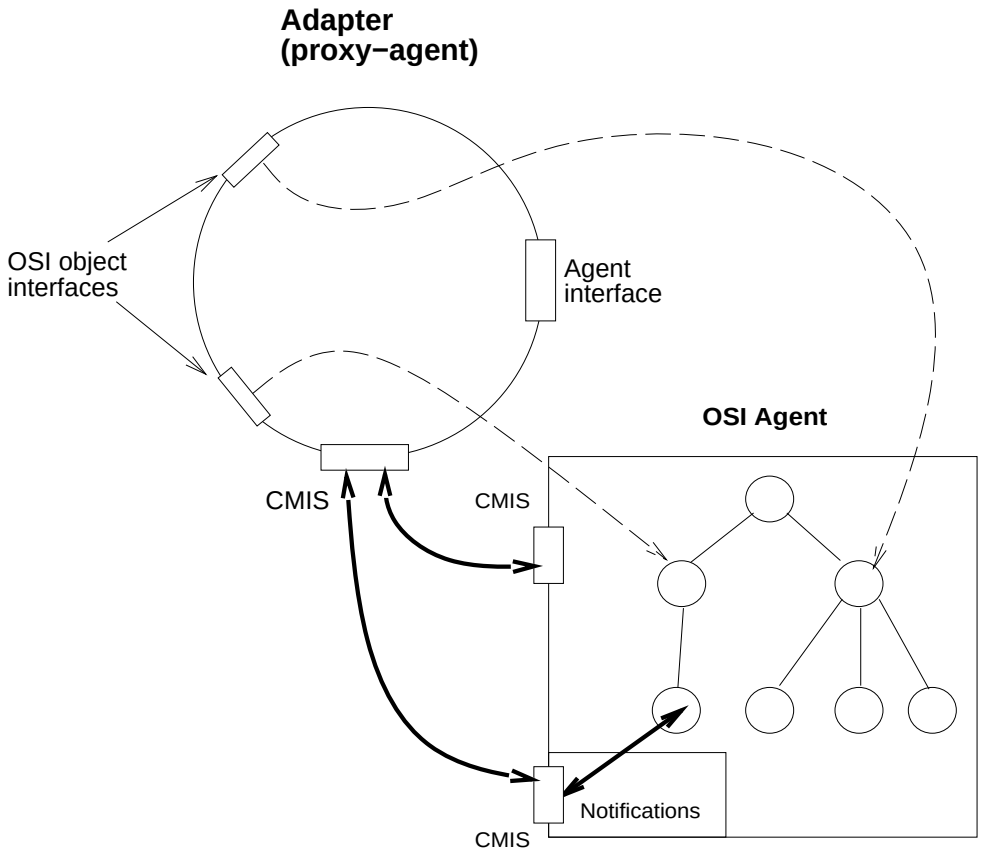

Figure 1: Adaptation architecture

An adapter is also the ANSA object which owns the interfaces that correspond to the MOs in its associated agent. This is of no concern to ANSA managers; for them, everything happens as if these interfaces were on the MOs themselves. All the ANSA managers need to know is that it is the adapter which gives out the references to those interfaces.

The architecture of our adapter is illustrated in figure 1, while our basic mapping is summarised in the following table:

OSI	ANSA
agent	adapter
managed object	interface
managed object class	interface type

A typical interaction between an ANSA manager and an MO would thus proceed as follows:

1. The manager finds the desired proxy-agent using the trader.
2. It requests a reference for the interface associated with the desired object. The type of this interface is defined with ANSA concepts but it provides access to the functionality of the object, as specified in its GDMO class description.
3. The manager makes a request on this interface.
4. The proxy-agent converts the request to CMIS, sends it to the real agent and waits for the result. It converts this and returns it to the ANSA manager.
5. The manager uses the returned result.

4.1 Restrictions Imposed by the Mapping

To keep the complexity of the mapping within manageable bounds, the following restrictions were selected:

- All interactions are with single MOs. There is no scoping or filtering.
- No action may have multiple replies.
- No access control.
- Some options present in CMIS are ignored, and some of the results (such as the current time) are not returned by the adapter.
- Only a subset of ASN.1 can be translated to IDL. The restrictions are mainly due to the nature of the Interface Definition Language (IDL) of ANSAware, but some were selected to keep things simple. For example, we do not support circular (recursive) type definitions[5], the type REAL and value definitions.

These restrictions are somewhat arbitrary, and any of them could be removed without fundamentally changing the mapping. But some changes would make it much more complicated.

5 Agent-level Operations

Agent-level operations are provided by a special interface on the adapter, whose type name is *Agent*. This interface provides operations for creating and deleting MOs, for discovering the interfaces of the MOs, and for subscribing to notifications.

When associated with an agent, each adapter registers this interface with the trader, with an attribute containing the agent's OSI address. It can therefore be found easily by all the ANSA objects that wish to access this agent.

5.1 Creation and Deletion of Objects

There is no general object creation facility in the ANSA model to which the CMIS create and delete services may be mapped. There is a standardised service, the 'Factory' [2], but its model of object creation is not sufficiently close to warrant creation of MOs as requests on a pseudo-factory service that would be provided by the adapter. Instead, the adapter's *Agent* interface proposes operations *Create* and *Delete* which are directly mapped from the corresponding CMIS services. As a consequence, the attribute values must be encoded in BER [15], i.e., as sequences of bytes.

5.2 Discovery of Interfaces Associated to MOs

An adapter can provide an interface to any MO in its associated agent. When an MO supports allomorphic classes (i.e., the MO has several different interfaces), the adapter may have several interfaces of different types that refer to that same MO.

ANSA managers can discover these interfaces by invoking an operation of the adapter's *Agent* interface called *GetInterface*. Given the OSI name and the GDMO class of an object, this operation will return its associated interface.

```
GetInterface : OPERATION [ type :      ObjectClass;
                           name :      OSIName;
                           permanent : BOOLEAN     ]
               RETURNS [ interface : ansa_InterfaceRef ];
```

The name of the MO that we use is its local distinguished name. The MO class is indicated by its ASN.1 object identifier. The indication of the class is necessary because an ANSA manager may specify the true class of the object or one of its allomorphic classes. The interface that is returned is the one that corresponds to the class requested: it provides explicit ANSA operations generated by translation from the GDMO specification of the MO. If that class is not supported by the MO, or if it has not been translated in IDL, an error is returned to the manager.

[5] It is possible to convert ASN.1 recursive types into IDL types, as shown in [4]. But this process is quite complex and does not always produce very clear types. As most management applications do not contain very complex types, the inclusion of this transformation does not seem necessary.

5.2.1 Naming Considerations

When an interface is passed (by reference) as a parameter of an operation in ANSA, an interface reference is passed (by copy) instead. Interface reference are therefore quite similar to names used for denoting MOs in network management (we call those names *OSI names*). However, 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[6] (probably because it was deleted). OSI names are not reliable because they may be reused when an object is deleted.

The adapter's interfaces contain OSI names; the adapter uses these names to forward operation invocations to its associated agent. Since OSI names may be reused, there is a danger that the reliability of interface references be violated. There is no perfect solution to this problem unless the agent participates, but this would also require an extension of the CMIP protocol [19]. We must therefore content ourselves with an approximate solution: the proxy-agent associates an interface reference with an OSI name only if its corresponding MO exists; it invalidates an interface reference when it is notified that its associated MO has been deleted. This allows the adapter to discover most of the reuses of OSI names on time, but not all of them (because a notification is only sent after an object is deleted, or because no notification may even be sent if the agent is overloaded). Nevertheless, we think that this solution is acceptable because OSI names are typically reused in situations where it is reasonably safe to do so.

An MO may be deleted and later another MO may be created with the same OSI name, but a manager may wish to consider that it is the same MO. When this is the case, the *permanent* parameter of *GetInterface* may be used to tell the adapter to ignore all notifications of deletion of the MO.

5.2.2 Typing Considerations

Passing an interface by reference is different than passing a name and an actual type for that interface because an interface parameter in an operation has a *formal type*[7]. The point is that formal types of parameters allow type checking to be performed at compile time: they ensure (together with sound subtyping rules) that any received interface is a subtype of the parameter formal type[8].

GetInterface can return an interface reference of an arbitrary type, the only valid return type is therefore *ansa_InterfaceRef* (which has no operations and is therefore a supertype for all interface references). The callers of *GetInterface* must cast this back to the correct type.

A similar solution is used for the trading function in ODP [24, 13].

This solution can be considered "type safe" because binding to an ANSA interface denoted by an OSI name is only possible after the adapter has checked that no typing errors will occur as a result of that binding, i.e, that the MO supports the requested class.

5.3 Subscription to Event Notifications

ANSA managers can subscribe to event notifications by using an operation of the adapter's *Agent* interface called *RegisterEvent*. They pass as parameters a valid callback *interface* (the same typing considerations as with *GetInterface* apply here) and the specification of a filter. Callback interfaces are further discussed in section 6.2.

```
RegisterEvent : OPERATION [ event_type : EventType;
                            filter : OSIFilter;
                            callback : ansa_InterfaceRef ]
                RETURNS [ RegisterEventResult ];
```

A filter specification may contain values of any type, so it has to be specified in BER. Since this makes the use of *RegisterEvent* rather cumbersome, we provide an alternate way of requesting events from a particular object. See section 6.1.3.

6 Object-level Interactions

There are two kinds of interactions with objects: operations (operations on attributes and actions) and notifications.

[6] Reliability of interface references, or *referential integrity* as it is called in CORBA, is an essential property of pure object-based systems because interface references may be part of the state of an object.

[7] We call "formal type" the type of a formal parameter, and "actual type" the type of an actual parameter.

[8] We consider a type to be a subtype of itself.

6.1 Object-level Operations

As selection of multiple objects is not supported, an ANSA manager can select an MO by using its ANSA interface reference; it can obtain that reference by using the *GetInterface* operation, but it can also receive it from other ANSA objects.

Operations of the ANSA interface allow an ANSA manager to invoke any actions supported by its associated MO. In addition, they allow the invocation of all the operations that are defined on the MO attributes and the subscription to all the events of the MO.

6.1.1 Actions

In general, actions on MOs may yield multiple replies, but unfortunately, this is only specified in the action's behaviour, i.e., in English text. It is therefore not possible for an automatic tool to know this. The adapter indicates its refusal to handle multiple replies at connection-establishment time, so it can ignore this problem[9].

An action is therefore mapped to an ANSA interrogation. The IDL argument types of the interrogation are obtained by the translation of the *WITH INFORMATION SYNTAX* construct of the action template. The results are mapped similarly, from the *REPLY SYNTAX* construct. If the *INFORMATION SYNTAX* is omitted, the operation has no argument. If the *REPLYSYNTAX* is missing, *Null* is used; an action is never mapped to an announcement so that errors can be returned. These are represented by a *GenericError* type; this keeps the interfaces simple to use (unless the ANSA manager has to handle errors effectively).

The use of GDMO parameters (PARAMETERS templates) for specifying arguments and results is very complex, and it is not recommended except for extensions [14, 14.3.8, pp. 68]. Because of this, we do not handle GDMO parameters in the GDMO/ASN.1 to IDL specification translation. The adapter can cope with the GDMO parameters that are part of the request or reply arguments of an action, but it does not transform them. As a result, ANSA managers must handle them in encoded form (BER). As indicated above, error parameters are treated similarly but are included in the *GenericError* type.

6.1.2 Attributes and Attribute Groups

Since ANSA does not have a notion of attribute, operations on attributes have to be translated into operations. For example, an attribute called *Colour* will be translated into the following IDL interrogations, provided its definition specifies them: *GetColour*, *ReplaceColour*, *ReplaceWithDefaultColour*, *AddColour*, and *RemoveColour*. We use variant records for carrying the values of attributes specified in GDMO conditional packages; a discriminant indicates whether the value is present or not. To avoid defining a very large number of operations, and for simplicity, we do not define any operation operating on more than one attribute.

Only *Get* and *ReplaceWithDefault* operations are expected on attribute groups, so for an attribute group called *Colours*, we generate at most the two operations *GetColours* and *ReplaceWithDefaultColours*. The result of the *GetColours* operation is a record which contains the values of *Colours*'s member attributes.

In fact, allomorphism makes things a bit more complicated. An MO always applies an attribute group operation to all the attributes which are members of that group, without consideration for the class that has been requested [20]. Thus, a reply to a *Get* operation may include values for attributes which are not specified in the class that applies to the operation (in this case, the class specified by the adapter). Since an ANSA object cannot receive values that it does not expect, the adapter translates only those attribute values that are expected, and discards the others.

The GDMO parameters of an attribute must be error indications. As for actions, they are represented in ANSA by the *GenericError* type.

6.1.3 Subscription to Event Notifications

Since the generic *RegisterEvent* operation is rather cumbersome to use, we provide in each translated interface a *Register*{*Event*} operation specialised to each type of event that can be emitted by the MO. For instance if there is a *Fire* event in an *MO* of type *Premises*, this operation is:

```
RegisterFire : OPERATION [ filter: OSIFilter;
                           callback: Premises_callback]
               RETURNS [ RegisterEventResult ];
```

[9] Actions with multiple replies are very rarely specified in GDMO, probably because a manager may decide not to handle multiple replies.

Calling this operation on an interface that represents an MO named x is equivalent to calling $RegisterEvent(Fire,\ objectInstance = x\ and\ filter,\ callback)$. This requests notification of all events of type $Fire$ that occur on object x and which match the filter (in most cases, the empty filter can be passed, thus requesting forwarding of all $Fire$ events which occur in x).

Note that the type of the callback interface is specified in the operation $RegisterFire$. This allows to check at compile-time that the invoker of $RegisterFire$ passes an appropriate callback interface. The possibilities of errors are therefore reduced when using $RegisterFire$ instead of $RegisterEvent$.

6.2 Notifications

Notifications of events that occur in MOs are sent by agents to interested managers. ANSA does not contain a corresponding mechanism, so it must be modeled explicitly.

ANSA interfaces may only define operations that are invoked by a client on a server, and not notifications emitted by a server. An ANSA manager needs therefore to support one or more *callback interfaces* to receive notifications. A callback interface is specialized to the MO class that emits the notification: its type is obtained by translating the GDMO specification of the MO class; it contains operations which are direct translations of the GDMO event specifications.

The adapter is responsible for forwarding a notification from the OSI world to an ANSA manager. It creates an 'event forwarding' object on its behalf, receives the notification, works out an equivalent ANSA operation, and invokes it on the callback interface of the ANSA manager.

Since event notifications contain no indication of what filter sent them, the adapter uses a different OSI forwarding address on each event discriminator that it creates; this allows it to determine on which callback interface it should invoke the "notification operation".

7 Comparison with the JIDM Specification Translation

Our specification translation from GDMO/ASN.1 to ANSA IDL is very close to the translation defined by JIDM for CORBA IDL [6]. In particular, our handling of attributes, notifications and GDMO parameters is similar to that of JIDM. However, there are a few differences. For example, JIDM does not currently provide a simplified way to subscribe to the notifications of an MO.

A more important difference concerns actions. JIDM considers that it must support actions with multiple replies, so it translates every action as if it has multiple replies (JIDM contemplates using the CORBA event channel to transmit multiple replies to a CORBA manager). The drawback of this approach is that it makes the large majority of actions more complicated to use than they need to be. ANSA does not support event channels, and we favour simplicity, so we simply ignore the problem by refusing to handle multiple replies. The best approach would probably be to let an operator specify which actions need to be translated differently because they may yield multiple replies[10].

JIDM does not translate attribute groups; instead, it contemplates providing a function that could be used for all the attribute groups of an MO. Because it would be generic, this function would not decode the attributes in a group and would shift the work to the programmer. We feel attribute groups deserve better support so we take a different approach: we translate every attribute group into an action with the attributes of the group as its results. There is however a problem with our approach: because of subtyping, an actual attribute group may contain more attributes than are expected by an ANSA manager. Our solution is to have the adapter discard all the attributes that are not specified in the MO class known to the ANSA manager. We think this is acceptable because an ANSA manager would not know what to do with those attributes anyway (remember that we do not intend to satisfy all managers).

JIDM defines complex algorithms to avoid clashes due to name translation. For instance, name clashes may occur when a GDMO attribute and a GDMO action are translated into operations: the translation may yield two operations with the same name! For the sake of simplicity, we decided to let an operator resolve these clashes by modifying the names in the original GDMO specifications. Modifying an action or an attribute name in a GDMO specification has no consequences on interaction since an ASN.1 object identifier is used instead of that name during interaction.

Translation clashes may also result from multiple inheritance. In GDMO, an MO class may inherit an action (or an attribute or a notification) from two superior classes; it is clear that it is one and the same action because of its object identifier — it cannot be an accidental clash of names. Object identifiers are not used in ANSAware and

[10] Ideally, the GDMO notation should be extended to indicate when multiple replies are possible.

CORBA, so their inheritance rules impose that all operations be originally defined in a single class: an operation may be inherited multiple times, but indirectly, it is inherited from at most a single superior class. Clearly, a straightforward translation of GDMO specifications will break the ANSA or the CORBA inheritance rules. JIDM solves this problem by modifying the inheritance tree to avoid the clash. Our strategy is different: we translate every action, attribute and notification into a single operation in a distinct ANSA interface; these "pseudo" interfaces are then inherited by the real ANSA interfaces. In that way, every operation is indirectly inherited from a single class.

8 Conclusion

This paper has shown a simple, but reasonably complete, mapping of MOs into interfaces in the ANSA world. This mapping is not perfect, but its implementation is quite feasible. Indeed, our work in the Esprit III project SysMan demonstrates this. Also, the resulting ANSA interfaces are much more easy to use than any general purpose interface to CMIS.

Most of the restrictions introduced in the name of simplicity can be removed if necessary, at the expense of greater complexity of use and implementation.

References

[1] Wade Allen. Experiences gained from the cmipWorks project. *CMIPRun*, 3(2), 1994.

[2] Architecture Projects Management Ltd., Cambridge (UK). *ANSAware 4.0 Application Programmer's Manual*, mar 1992.

[3] Karrim Berrah, David Gay, and Guy 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.

[4] David E. Gay. Interface Definition Conversions: Recursive Types. In *Proceedings of the ACM Workshop on Interface Definition Languages*, pages 101–110. Carnegie Mellon University, jan 1994.

[5] Guy Genilloud and Marc Polizzi. Managing ANSA objects with OSI network management tools. In *Proceedings of the Second International Workshop on Services in Distributed and Networked Environments (SDNE'95), Whistler, Canada*. IEEE Computer Society Press, 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] Andrew J. Herbert. An ANSA overview. *IEEE Network*, pages 18–23, jan 1994.

[9] Object Management Group. *The Common Object Request Broker: Archictecture and Specification (1.1)*, dec 1991.

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

[11] The ANSA Project. Mapping ANSA concepts to C++. ANSA Technical Report TR.036.00, Architecture Projects Management Ltd., Cambridge (UK), feb 1993.

[12] R.J. van der Linden. The ANSA naming model. ANSA Architecture Report AR.003.01, Architecture Projects Management Ltd., Cambridge (UK), feb 1993.

[13] Andrew J. Watson. Revising the DPL type system. ANSA Request for Comments RC.339.02, Architecture Projects Management Ltd., Cambridge (UK), jun 1992.

[14] John Westgate et al. *Technical Guide for OSI Management*. NCC Blackwell, Manchester - Oxford, 1992. ISBN 1-85554-187-4.

[15] *ITU-T Recommendation X.209 (1990) — ISO/IEC International Standard 8825:1990, Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

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

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

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

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

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

[21] *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.*

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

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

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