Managing ANSA Objects with OSI Network Management Tools

Guy Genilloud

Computer Engineering Department
EPFL-DI-LIT
Swiss Federal Institute of Technology
CH-1015 Lausanne, SWITZERLAND

Marc Polizzi

Computer Engineering Department
EPFL-DI-LIT
Swiss Federal Institute of Technology
CH-1015 Lausanne, SWITZERLAND

OSI Network Management provides a general framework for the management of OSI systems, and by extension of any distributed system. However, as this model is not well-adapted for the management of software components, distributed programming environments (e.g. DCE, CORBA, ANSAware) essentially ignore the OSI Network Management model. We assume nevertheless that OSI Network managers will want to have some control of a distributed infrastructure and application. We examine how access to some of the ANSA (distributed programming environment) objects can be given to OSI Network managers. An implementation of an ANSA-OSI adapter is then presented.

1 Introduction

The Open Systems Interconnection (OSI) network management model [3] provides a general framework for the management of OSI systems, and by extension of any distributed system. It deals with the fundamental management concepts, functional areas, and structure.

However, this framework has been developed for the management of resources within protocol entities. It is not well adapted for the management of software components which are distributed or which migrate frequently from one system to another. Also, this framework does not make use of the tools provided in distributed programming environments (IDL, stub generator, object or interface references) and is therefore ill-adapted for the rapid development of (specific) management tools within those environments. For these reasons , distributed programming environments will provide their own management solutions, and will essentially ignore the OSI Network Management framework and the tools that are provided for it.

Nevertheless, there will probably be circumstances where network managers will need to have some view and possibly some control of distributed systems and applications. We examine how access to management interfaces within the distributed programming environments can be given to OSI Network Management tools, and how this provision of access can be automated. We consider the case of ANSAware[1], a distributed system which has had a considerable impact on the joint ISO-ITU 'Reference Model for Open Distributed Processing' [2] document.

We are interested in providing OSI Network Management tools with access to ANSAware management interfaces, and conversely in providing ANSAware applications with access to OSI Network Management objects. To achieve these goals, we need to map the concepts of one model into those of the other model [2]. 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. This paper focuses on how access to ANSA objects can be given to OSI managers.

The realisation of the mapping of ANSA objects to OSI network management objects is divided into two tasks:

- Specification Translation - This task translates IDL interface specifications into GDMO [1] and ASN.1 [6] and provides specific information to be used during interaction translation. It is done by a tool that we call an IDL-to-GDMO translator.
- Interaction Translation - This task maps CMIS [4] requests into ANSAware operation invocations and ANSAware operation terminations into CMIS responses. It is decomposed into two subtasks:
  - Conversion of CMIP [5] messages into ANSA-CMIS operation invocations or terminations; these are ANSA operations that together implement the CMIS service. This sub-task is performed by an ANSAware application that we call an OSI-CMIS adapter.
  - Mapping of ANSA-CMIS operations to corresponding ANSAware operations as expected by the ANSA interfaces supporting or invoking these operations. This mapping is provided by an ANSAware application that we call an ANSA-CMIS adapter or a pseudo-agent. The terms 'ANSA-CMIS adapter'

[1] ANSAware is a simple realisation of the ANSA model. The terms 'ANSA' and 'ANSAware' are used interchangeably below.

[2] 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.

and 'pseudo-agent' are used interchangeably in this document [3].

This paper is organised as follows. First, we briefly recall the main results of the ANSA and the OSI models comparison (presented at DSOM'94 [9]) and our approach to the mapping. Next, we explain how an 'interface specification database' can be used for that purpose. We then discuss how the specification translation and the interaction translation can be implemented.

1.1 Related Work

X/Open and the Network Management Forum are working on a software architecture that will allow a CORBA-based application [10] to play the manager or the agent role in a manager/agent model of interconnection [8]. The mapping of GDMO specification to CORBA-IDL is well described whereas the reverse mapping (in which we are interested in this paper) needs yet more work.

Their work on the 'agent role' is similar in many respects to ours since ANSAware can be considered as an implementation of CORBA, albeit with a different IDL and object adaptation mechanisms. It differs in that ANSAware does not offer the equivalents of the naming and event services that have recently been adopted by the Object Management Group [11].

2 Models Comparison and Mapping

The following comparison and proposed reconciliation of the two models are investigated solely for the purpose of accessing ANSA objects from the OSI world. Therefore, several points, which may seem important to the reader, are not addressed here.

This comparison is mainly based on the ANSA computational [7] and engineering viewpoints, for it is at these levels that an OSI-ANSA mapping occurs.

The two models differ in their primary intended use. The OSI Network Management model is used to describe the management of an OSI system whereas the ANSA model aims to specify a complete object-based distributed system from design to implementation. These different uses must not be seen as conflicting but rather as complementary. An ANSA-OSI adapter should allow to get the best of both worlds.

2.1 Objects, Interfaces and Roles

The OSI model assumes a separation of tasks between a 'manager', which is trying to manage something, and an 'agent', which contains the objects to be managed. These objects are referred to as the 'Managed Objects' or MOs.

In ANSA all program entities are objects. These objects can perform any kind of task (e.g. evaluate a function, manage a resource, display information on the screen) or assume any kind of role (client, server, client and server, manager, producer, consumer, etc).

Encapsulation in ANSA is absolute in the sense that all interactions between objects are explicitly modelled as operations invoked by one object on another. These operations, or more precisely their signatures, are defined statically within interfaces using a notation called Interface Definition Language, or IDL [4]. ANSA objects can have more than one interface, and they can create or delete interfaces on themselves dynamically. Whereas objects can assume both client and server (or producer and consumer) roles, interfaces are always restricted to one of those roles.

In OSI management, encapsulation is not absolute. Objects always exist within the context of an agent, which can see their implementation. Also, MOs model resources solely for the purpose of management. An MO may be thought of as a ANSA management interface specified with the GDMO and ASN.1 notations, and which is used to manage a resource or an entity. Contrarily to IDL, it is possible to specify both supported operations (actions and attributes) and invoked operations (notifications) within a single interface specification. Therefore, an MO corresponds in general to two ANSA interfaces.

Since MOs are effectively interfaces to managed entities, we will use the word 'interface' whenever we mean MOs or ANSA interfaces.

2.2 Naming

The two models have very different strategies for naming interfaces: MOs have a globally unique name based on their position in the containment tree of a particular agent. This name may be reused once the object is deleted [5]. ANSA interfaces do not have a programmer-visible name; they are referred to indirectly via interface references (the programmer names interface references). Interface references can be passed as parameters or as results of operations. Interface references are reliable in the sense that they will always produce bindings to the same interface, or they will fail if such a binding cannot be produced by the architecture. On the other hand, there may be more than one reference for a given interface, so it is not possible to compare them for equality to determine whether or not they refer to the same interface.

The fact that interface references may have aliases seems to prevent mapping interface references to global distinguished names, but it is nevertheless the mapping that we have implemented as there are no other reasonable solutions. We hope that OSI managers will not be confused by this situation. They will have to think of ANSA objects as having an arbitrary number of identical management interfaces, each of which having a global distinguished name.

The pseudo-agent will need to implement a containment tree for naming purposes. Theoretically, it is possible to make use of a configurable and complex tree, but the advantages of building such a tree are not

[3] The ANSA-OSI adapter or more simply the 'adapter' refers to the overall application responsible for converting CMIS operations to ANSAware operations.

[4] An IDL specification does not contain any indication of role; it can therefore be interpreted as specifying (very partially) a *server interface*, i.e., a set of operations that are supported by an object, but it can just as well be interpreted as specifying a *client interface*, i.e., a set of operations that are invoked by an object.

[5] Specifiers of MOs need to take care that doing this will not confuse the managers of that object.

clear. As ANSA interfaces are named in a flat naming context, and as they are unaware of any containment tree, we chose to support only a flat naming tree [9].

2.3 Creation and Deletion of Objects

The OSI management services of object creation and object deletion are hard to map in a general way. They imply creating or deleting objects (not just interfaces), but this requires information (which template, which capsule, etc.) that the pseudo-agent cannot obtain or deduce in the general case. Moreover, creating a new object solely for the purpose of management does not make much sense; the object must be linked with other ANSA objects or with real resources to be of any use. For these reasons, we do not support the OSI create and delete operations (M-CREATE, M-DELETE). Note that OSI managers may still be capable of creating and deleting some objects or interfaces through specific operations on some ANSA interface (mapped to actions on some MO).

2.4 Interactions

Interactions are also different. In the OSI world, they use a message based communication between manager and agent. They may apply to multiple objects selected from the containment tree, in which case the whole interaction may be made atomic and produces multiple results. In ANSA, an interaction is either an interrogation, i.e., an operation invocation followed by a single response (the operation termination), or an announcement operation which yields no response. These differences do not create problems for the mapping since both interrogations and announcements map to actions, and since our pseudo-agent can map interactions with multiple objects onto a series of ANSA operations on the appropriate interfaces. However, the pseudo-agent cannot support atomic execution of a multiple interaction since most ANSA interfaces are not transactional. This is not really a problem since atomic execution is optional and is almost never supported anyway.

In the OSI model, an agent can emit notifications to which managers can subscribe. ANSAware does not provide an equivalent mechanism in support of notifications, but nothing prevents application objects to invoke interrogations or announcements on a manager's interface. Announcements are very similar to unconfirmed notifications, but interrogations are quite different from confirmed notifications since they always require a single response (termination). Supporting "notification interrogations" would require mapping zero or more notification confirmations to a single termination, which is problematic. Supporting "notification announcements" does not create such problems, but announcements are very rarely used in ANSAware [6]. Since our goal it to support "existing" ANSAware applications with only a few minor extensions, we decided not to support notifications in a first version of the 'adapter'.

[6] Since there is no support in ANSAware for explicit binding of interfaces, there are a number of problems associated with using announcements. For example, announcements will be lost silently every time a binding cannot be established.

3 Adaptation Tools

This section briefly describes the different ANSAware applications used for the implementation of the ANSA-OSI adapter. The latter is not implemented as one ANSA application but rather with a series of tools which may be used independently for different purposes in different contexts.

3.1 Interface Specification Database

The interface specification database, or interface repository, stores information on network management interface specifications. Three kinds of specifications are of interest: GDMO standards, ASN.1 modules and IDL interfaces.

3.2 IDL-to-GDMO Translator

This tool translates the interface specification written in IDL to GDMO and ASN.1 specifications. It is implemented as a client of the interface specification database.

3.3 OSI-CMIS Adapter

This low-level protocol adapter [7] provides effective access to the CMIS/CMIP protocol stack within an ANSAware platform. It converts CMIP messages into corresponding operations that support the CMIS service at a rather low-level, most parameters being passed encoded in BER within a SEQUENCE OF OCTETS. We call these operations ANSA-CMIS operations.

3.4 ANSA-CMIS Adapter or Pseudo-Agent

Together with the OSI-CMIS adapter, this tool implements the interaction translation. It converts ANSA-CMIS operations into operations that are supported by the ANSA application objects. The OSI model requiring an 'agent' for all object use, the ANSA-CMIS adapter models explicitly one or more 'pseudo-agents'.

Figure 1 shows how CMIS/CMIP requests initiated by a manager are converted to ANSAware operation invocations using a 2-stage architecture (the interface specification database and the specification translator are not represented).

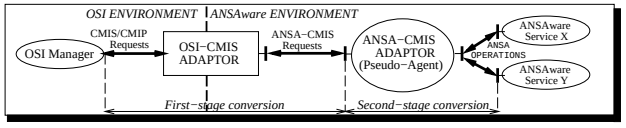

Figure 1: 2-Stage request conversion

4 Interface Specification Database

This database stores information on network management interface specifications written either in GDMO/ASN.1 or in IDL.

[7] We did not implement this adapter ourselves; this is the task of one of our partners in the the ESPRIT III project SysMan.

4.1 Use of the Database

Contrarily to CORBA, ANSAware does not provide a dynamic invocation interface. It only provides static interfaces, i.e., all the interfaces used by a program must be known at compile-time[8].

For the purpose of interaction translation, we must be able to 'dynamically' invoke an ANSA operation within an interface, whatever this operation and interface are. Using the specification database is a way to achieve such a requirement : on the reception of a CMIS request, a pseudo-agent will query the database to find out its signature and determine how it can be converted into an ANSA operation.

The specification database is also a common place to store any additional information related to interface specifications. Currently, the database contains a link between both the initial and the converted version of a specification.

As the interfaces to the object definitions maintained in the specification database are public, client applications may use this information. Examples of such applications are:

- a Management Control Monitor: allows a manager to browse and manipulate management information; displays incoming event notifications;
- GDMO, ASN.1, IDL specification browsers;
- translation tools: our IDL-to-GDMO translator (as well as our GDMO-to-IDL translator) uses the database in order to avoid the tedious management of several files; all the cross-referenced components are easily accessible.

4.2 Implementation

The database consists of a persistent ANSAware object which stores information about all the specifications, and of several custom programs which are used to update the database. These are a GDMO parser, an ASN.1 parser, an IDL parser and a database manager. The architecture of the specification database is illustrated in figure 2.

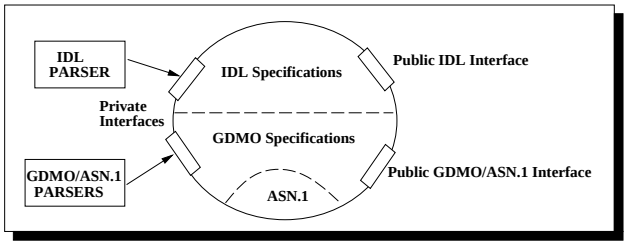

Figure 2: Database architecture

4.2.1 Data Representation and Organisation

The database stores GDMO/ASN.1 and IDL specifications separately (i.e., it does not convert them into a common format), as this simplifies the job of using the information from the database.

To simplify implementation and maintenance of the database, cross-references between the various specifications stored in the database are not explicitly represented in the database. Instead, references are checked and updated when information is read from the database. Thus, the database does not provide any consistency or completeness guarantees, and all read requests may return a list of errors. This allows specifications to be arbitrarily added or removed from the database.

While running, the database maintains two copies of its contents, one in memory and one in files:

- *In memory representation* - This representation is chosen to allow easy access and update, so that the database operations will be simple and reasonably efficient. The sets of information stored in memory are kept in simple doubly linked lists, and information is retrieved by searching these lists using several hash-tables.
- *In files representation* - This representation is chosen so that all the information can be kept in a few sequential files. Given the complex structure of the information, defining such a representation may be quite complicated. However, as our types are specified in IDL, they already have a defined representation as a stream of bytes, that used for transmission of values over the network. We use this same representation for our disk files. Storing a value can then be done by calling a marshalling function produced by the ANSAware stub generator and saving the results.
  A simple strategy is used to recover correctly in case of a crash during an update.

4.2.2 Updating the Database

The database is maintained by adding or removing GDMO, ASN.1 and IDL specifications from it. Removal of specifications is done with a simple interactive program. Addition of specifications is done by a GDMO, an ASN.1 and an IDL parser. These three parsers, which we built using the standard UNIX *yacc* and *lex* tools, do the same operations:

- Parse a specified file and build up a data structure representing the specifications.
- Do some partial consistency checks on these specifications. Basically, these are some checks that can be performed on the compilation module considered in isolation. The parsers primary aim is not to check the validity of their inputs (a task more properly left to a compiler) but to prepare data for the database. It is assumed that the specifications are essentially correct.
- Transform the results of the parse stage into the form accepted by the database.
- Send the parsed specifications to the database.

4.2.3 Read Access to the Database

Read access to the database is done through two public (i.e., registered to the ANSAware Trader [9])

[8] In fact, interfaces that are actually used by a program may be of types conformant to interface types known by the program.

[9] The Trader is a dedicated ANSAware application used for dynamic binding. It is used by servers to advertise services and by clients to locate interesting service offers.

ANSA interfaces on the database server. Therefore, it can be done remotely. An interface is dedicated to the GDMO/ASN.1 information while the other is dedicated to the IDL information.

The *GDMO/ASN.1 interface* includes operations to get the information associated with each component of a GDMO standard: classes, packages, parameters, name bindings, attributes, attribute groups, behavior, actions, notifications. An operation is also provided that does all the complex inheritance processing and returns the complete definition for a GDMO class. Another operation returns links to converted IDL interface specifications. Finally, several operations allow to explore the contents of the database; these are useful for browser-style applications.

The *IDL interface* is shorter, reflecting the greater simplicity of IDL interfaces. For a given specification, it provides the following operations: return its structure, its text and its link to the equivalent GDMO class. As for GDMO, an operation is provided to do all the inheritance processing and return the results. An operation allows to get the list of all interface specifications contained in the database.

The database does not support replication nor distribution of data. This simplifies its design and implementation, but may create some problems in the future regarding load-balancing and availability. However, nothing prevents us from running several independent copies of the database. Indeed, we may use the relocation mechanisms provided by ANSAware to allow clients of a crashed or heavily overloaded database to rebind transparently to an alternate database.

5 Specification Translation

The IDL-to-GDMO specification translator is implemented as a client of the specification database.

5.1 Overall Translation Algorithm

The specification translator systematically converts all the native [10] IDL interface specifications present in the database into GDMO and ASN.1 specifications. These specifications are then parsed and stored into the database using the GDMO and ASN.1 parsers. A link between both the initial and the converted versions of a specification is kept.

The different steps of the translation are as follows:

- Create an ASN.1 file for collecting all the data types used within the IDL interfaces specifications. Store the required common and basic ASN.1 definitions.
- Create a GDMO file for collecting all the GDMO templates corresponding to the IDL interface specifications. Store the required common and basic GDMO definitions.
- Create a file for keeping the links between the GDMO classes and the IDL interfaces.
- Get the list of the IDL interfaces definitions to be translated from the specification database.
- For each IDL interface - Generate the corresponding GDMO and ASN.1 specifications: one managed object class template, one package template, some operation templates, ASN.1 types.
- Add the new ASN.1 module and GDMO standard to the database using the ASN.1 and GDMO parsers.

GDMO and ASN.1 have specific rules for naming types, identifiers and constants which do not exist in IDL. Moreover, the ASN.1 character set does not contain the IDL underscore ('_'). Therefore, a lexical translation is needed to convert IDL identifiers into valid GDMO or ASN.1 identifiers.

Before describing how the IDL specifications are translated into GDMO/ASN.1, we will present several common and basic GDMO/ASN.1 definitions used by the converted IDL interfaces.

5.1.1 ANSAware Super-Class

All converted IDL classes must inherit from the 'ansaClass-Top' class which inherits directly from the GDMO class 'top'. 'ansaClass-Top' includes an extra-attribute ('ansaName') for naming the interfaces within the containment tree. We chose to define this attribute as a STRING for simplicity:

```
ansaClass-Top MANAGED OBJECT CLASS
 DERIVED FROM "ISO/IEC 10165-2: 1992":top;
 CHARACTERIZED BY ansaPackage PACKAGE
               ATTRIBUTES ansaName GET;;;
REGISTERED AS {ANSA-ROOT-OID xxx};

ansaName ATTRIBUTE
 WITH ATTRIBUTE SYNTAX Asn1Module.AnsaString;
 MATCHES FOR EQUALITY;
REGISTERED AS {ANSA-ROOT-OID  xxx};
```

5.1.2 Name Binding

A managed object class definition has associated with it one or more name bindings which define how instances of the class are named and the rules for their creation and deletion. We define a name binding that will apply to all the managed object classes that are derived from 'ansaClass-Top'. This binding reflects the very simple binding structure (a flat tree) that we decided for the pseudo-agent.

```
ansaNameBinding NAME BINDING
  SUBORDINATE OBJECT CLASS ansaClass-Top
                     AND SUBCLASSES;
  NAMED BY SUPERIOR OBJECT CLASS
          "ISO/IEC 10165-2: 1992":system
                     AND SUBCLASSES;
  WITH ATTRIBUTE ansaName;
  DELETE DELETES-CONTAINED-OBJECTS;
REGISTERED AS {ANSA-ROOT-OID xxx};
```

5.1.3 ANSAware Errors

The ANSAware standard errors are reported to the OSI manager as a specific GDMO error parameter. The error parameter template is defined as follows:

```
ansaStdError PARAMETER
  CONTEXT     SPECIFIC-ERRORS
  WITH SYNTAX Asn1Module.ANSAstdERRORS
REGISTERED AS {ANSA-ROOT-OID xxx}
```

'ANSAstdERRORS' is an ASN.1 ENUMERATED type which collects the ANSAware errors.

[10] We call an IDL interface specification 'native' if it is not the result of the translation of a GDMO class.

5.1.4 Generation of Object Identifiers

An object identifier (which is a sequence of numbers) is used to identify any statically GDMO defined element[11] such as class template, attribute template, etc. Object identifiers are mandatory for all the components which must be referred to at run-time.

All the mapped classes inherit from 'ansaClass-Top'. We define an object identifier ('ANSA-ROOT-OID') for this class. Then, a mechanism generates an object identifier for each GDMO component resulting from the IDL interfaces translation.

5.2 Specification Translation Rules

A specification written in IDL has the following structure:

```
InterfaceNAME : INTERFACE =
  IS COMPATIBLE WITH InterfaceNAME1
  NEEDS InterfaceNAME2
  BEGIN
    -- constructed data types...
    -- operation signatures...
  END.
```

According to our object model mapping, each IDL interface definition is converted into:

- one GDMO managed object class template,
- one GDMO package template,
- one GDMO action template for each IDL operation,
- all the IDL types defined in the interface are converted to ASN.1 types and collected into one ASN.1 module.

5.2.1 Managed Object Class Template

The label of the template is the concatenation of 'ansaClass-' and the name of the IDL interface name (InterfaceNAME). We only need to define three components:

- **DERIVED FROM:** lists all the classes mapped from the interfaces types names that appear after the 'IS COMPATIBLE' in the IDL specification. The 'ansaClass-Top' class must also be mentioned here.
- **CHARACTERIZED BY:** indicates the name (based on the name of the IDL interface) of the GDMO package template.
- **REGISTERED AS:** gives an object identifier to this class.

5.2.2 Package Template

This template collects the actions corresponding to the IDL operations of the IDL interface being translated. Its label is defined as the concatenation of 'ansaPackage-' and InterfaceNAME. Two components are required:

- **ACTIONS:** lists the names of all the action templates derived from the IDL specification.
- **REGISTERED AS:** gives an object identifier to this template.

[11] The term 'Object Identifier' can create some confusion. 'Object Identifiers' are not used for identifying OSI Managed Object instances.

5.2.3 Action Template

This template is used to define the syntax associated with a particular action (corresponding to an IDL operation) type. Its label is defined as the concatenation of 'ansaOperation-', InterfaceNAME and OperationNAME. Five elements must be defined:

- **WITH INFORMATION SYNTAX:** refers to an ASN.1 SEQUENCE type which will hold the arguments of the ANSA operation invocation.
- **WITH REPLY SYNTAX:** refers to an ASN.1 SEQUENCE type which will hold the arguments of the returned ANSA operation termination.
- **MODE CONFIRMED:** if present, the action shall operate in confirmed mode. Used with ANSA 'interrogations', not used with ANSA 'announcements'.
- **PARAMETERS:** is used for reporting the standard ANSAware errors (which may occur during the request conversion or ANSA operation invocations) to the OSI manager.
- **REGISTERED AS:** gives an object identifier to this template.

5.2.4 IDL Types Conversion

The ASN.1 module collects all the types defined in the IDL interface specifications being translated. The specification of ASN.1 types equivalent to IDL basic and constructed types is straightforward as ASN.1 provides a more general notation than that of IDL.

The **interface reference** type is an IDL abstract type that has no equivalent in ASN.1. But interface references are essentially names of interfaces, the corresponding concept is therefore the 'Global Distinguished Name' of an object. Therefore, the 'ansa_InterfaceRef' type can be converted into an ASN.1 type that represents such names (and which also includes a reference to the type of the interface).

The following tables briefly summarize the type conversion rules:

Basic IDL Type	*ASN.1 Type*
Boolean	Boolean
Short Cardinal	Integer(Size($0..2^{16}-1$))
(Long) Cardinal	Integer(Size($0..2^{32}-1$))
Short Integer	Integer(Size($-2^{15}..2^{15}-1$))
(Long) Integer	Integer(Size($-2^{31}..2^{31}-1$))
(Long) Real	Real
Octet	OctetString(Size(1))
Char	Ia5string(Size(1))
String	OctetString

Constructed IDL Type	*ASN.1 Type*
Array n Of X	Sequence Size(n) Of X
Sequence Of X	Sequence Of X
Record[ ]	Sequence { }
Choice	Choice (with ASN.1 Tags)
Enumerated{ }	Enumerated{ }

6 Interaction Translation

Interaction translation is performed by an ANSAware distributed application whose main components are one or more pseudo-agents. The 'adapter-controller' is an ANSAware object that controls these pseudo-agents. Figure 3 shows the global architecture of the ANSA-OSI adapter (the interface specification database is not represented).

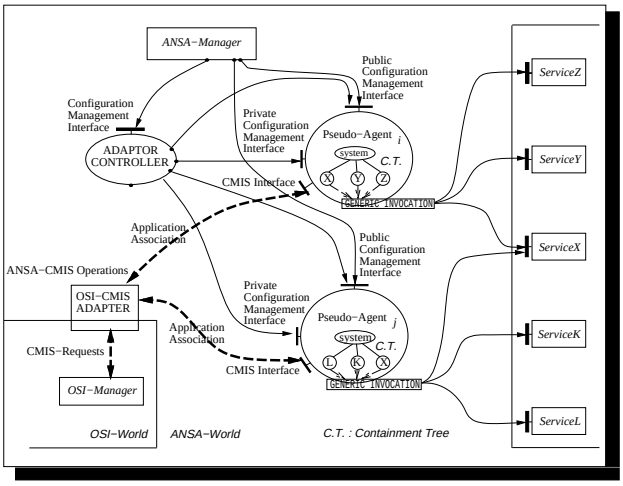

Figure 3: Adapter architecture.

6.1 ANSA-CMIS Adapter - Pseudo-agent

The pseudo-agent is implemented as an ANSAware object which provides one interface to communicate with the OSI-CMIS adapter and two interfaces required for configuration and management (by an ANSA-manager) of all the pseudo-agents.

6.1.1 Containment Tree

As explained earlier, each pseudo-agent contains a flat tree, with a 'system' object at the top and all the mapped ANSA interfaces immediately below it.

As ANSAware interfaces do not support any attribute, the pseudo-agent must explicitly model the attributes related to the ANSAware interfaces it has to manage. These attributes are those derived from the GDMO 'top' and 'system' classes and from the ANSAware super-class 'ansaClass-Top'.

6.1.2 Typical Interaction

Interaction translation consists in mapping an ANSA-CMIS request sent by the OSI-CMIS adapter to an ANSAware operation, in indentifying the ANSA interfaces that correspond to the GDMO objects selected within that request, and in invoking the mapped operation on all those interfaces; in addition, the reverse translation must be done for the operation terminations. More precisely, interaction translation is performed as follows:

1. Reception of an ANSA-CMIS request corresponding to the original (OSI) CMIS request.
2. Depending on the nature of the request, the pseudo-agent performs one of the following actions:
   (a) M-DELETE: This request is only used to remove an ANSA interface from the containment tree. This does not imply that interface is 'physically' suppressed from the system.
   (b) M-GET: This request concerns attributes which are actually stored in the pseudo-agent. Therefore, this request does not require any ANSA operation invocation and can be performed locally.
   (c) M-ACTION: This operation is used to perform an action (other than those related to the attributes) on one or more GDMO objects.
   Each action corresponds to an ANSAware operation in an ANSA interface. Thus the reception of such a request implies one (or possibly several, if more than one object is selected) ANSAware operation invocation(s).
3. Encode a reply depending on the type of the processing above.
4. Send it to the OSI-manager via the CMIS interface.

6.1.3 CMIS Interface

The ANSA-CMIS adapter provides an interface which maps OSI CMIS services to corresponding ANSA operations:

- M_CREATE/M_DELETE,
- M_GET/M_CANCEL_GET/M_SET,
- M_ACTION,
- M_EVENT_REPORT.

This interface is invoked by the OSI-CMIS adapter to transmit the messages from the CMIP protocol stack to the pseudo-agent.

Operation signature does not include all the parameters specified in [4]. Indeed, some parameters (e.g. InvokeId, Synchronization) have no sense in the ANSAware environment, while others refer to features which are not supported by the pseudo-agent (e.g. Filter, CurrentTime).

6.1.4 Configuration of the Adapter

The adapter is configured at two different levels. We distinguish between the overall configuration of the adapter and the configuration of each of its pseudo-agents.

The overall configuration of the adapter is typically done manually through an interactive program which provides an interface to the 'adapter controller' object. This ANSA object provides services to set up, run and stop a pseudo-agent on a particular node. It also saves the adapter's overall configuration on stable storage and deals with crash problems and recoveries.

Pseudo-agents are partially configured by the adapter controller, through their private management interface. But for the largest part, they are configured through their public management interface by the managed applications themselves. Indeed, only the applications know all the interface references to their management interfaces, and therefore only they can completely configure the containment tree of a

pseudo-agent [12]. This requires that ANSA applications be modified so that they can be managed, but this should not be too demanding in general since applications need to register their management interfaces to ANSA managers anyway.

6.1.5 Generic Invocation

As explained earlier, the pseudo-agent must be able to invoke any ANSA operations, whatever their signatures are. For this purpose, we developed a *generic* ANSAware stub, whose main task is to marshal and unmarshal operation parameters.

The main steps for generating an ANSAware operation invocation, from a CMIS M-ACTION are:

1. Extract the names of the selected objects and their GDMO class.
2. Find the associated interface references and their IDL type.
3. Extract the GDMO arguments of the action.
4. For each selected ANSA interface, invoke the generic stub with the interface reference, the name of the operation and its arguments.

7 Results and Conclusion

Our experience has shown that it is possible to implement an adapter allowing OSI network management applications to access management interfaces within the ANSAware environment, with a minimum of effort.

The adapter that we built preserves the essential features of both the ANSA and the OSI worlds. However, it imposes several restrictions either for simplicity or because of the limited capabilities of ANSAware. Effective use of the adapter within the ESPRIT III project SysMan will tell us how useful it really is and how it should be expanded to make it even more useful.

Within the SysMan project, we are also implementing the reverse mapping, allowing ANSA applications to access MOs as if they were ANSA interfaces.

References

[1] ISO/IEC 10165-4: 1992. Information technology - Open Systems Interconnection - Structure of Management Information: Guidelines for the Definition of Managed Objects. ITU-TS X.722.

[2] ISO/IEC 10746-3: 1995, Open Distributed Processing - Basic Reference Model - Part 3: Architecture, ITU-TS Recommendation X.903.

[3] ISO/IEC 7498-4: 1989, Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework. ITU-TS X.700.

[4] ISO/IEC 9595: 1991. International Standard - Information technology - Open Systems Interconnection - Common Management Information Service Definition. ITU-TS X.710.

[5] ISO/IEC 9596-1: 1991. International Standard - Information technology - Open Systems Interconnection - Common Management Information Protocol - Part 1: Specification. ITU-TS X.711.

[6] ISO/IEC 8824: Information Technology - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)., 1988.

[7] The ANSA Computational Model, ANSA Architecture Report AR.001.00, August 1991.

[8] Tom Rutt (Editor) - Comparison of the OSI management, OMG and Internet management objects models - A Report of the Joint XOpen/NMForum Inter-Domain Management Taskforce - Internal Working Document - AT & T Laboratories., March 1994.

[9] G. Genilloud K. Berrah, D. Gay. Accessing ANSA Objects from OSI Network Management - DSOM-94, October 1994.

[10] Object Management Group (OMG). The Common Object Request Broker: Architecture and Specification - Revision 1.1, December 1991.

[11] Object Management Group (OMG). Common Object Services Specification - Volume 1., March 1994.

[12] The pseudo-agent could query the trader or some other location service to discover the references of an application's management interfaces. We decided against implementing such a solution within our pseudo-agents since it is less general than our approach and since it still requires an applications to register all its management interfaces to a location service. Moreover, this solution can easily be implemented outside of the pseudo-agent by a dedicated object.