12

Joint Inter Domain Management:

CORBA, CMIP and SNMP

By the JIDM taskforce and as editors:

Dr. N. Soukouti
SMILE, Sarl.
E-mail: soukouti@smile.fr
Postal: 29, rue Viala
75015 Paris
France
Tel: :+33 1 40 59 09 00
Fax: +33 1 40 59 09 01

Dr. U. Hollberg
IBM European Networking Centre
E-Mail: hollberg@heidelbg.ibm.com
Postal: Vangerowstr. 18
D-69115 Heidelberg
Germany
Tel: ++49 (6221) 59-4223
Fax: ++49 (6221) 59-3300

Abstract

In this paper, we present an overview on the work of a joint activity of the X/Open and the Network Management Forum known as *"Joint Inter Domain Management (JIDM)"*. The activity of this group has been split into two phases, called *"Specification Translation"* and *"Interaction Translation"*. In this paper, we present the basic concepts of the *Specification Translation*, which has been defined to allow for the migration of information models between different domains of management technology. We will also present in short the problems to be solved during the ongoing *Interaction Translation* phase, which aims at enabling the inter-working between management software across the different domains: CMIP, CORBA and

A. A. Lazar et al. (eds.), *Integrated Network Management V*

SNMP. The Specification Translation document is currently in final review as X/Open preliminary specification. The Interaction Translation document is expected to be released in the second Quarter of 1997.
The intention of the paper is to make the JIDM activity known and to raise interest in its results; the interested reader is referred to the X/Open preliminary specification.

Keywords

CORBA, OMG, COSS, OSI, CMIP, GDMO. SNMP. ASN.1, SMI, MIB

1 INTRODUCTION

The *Joint Inter-Domain Management (JIDM)* group is jointly sponsored by X/Open and the Network Management Forum (NMF). The group was initiated in response to the perceived need to provide tools that can enable interworking between management systems based on different technologies. The JIDM group has focused on the three key technologies: CMIP, SNMP, and CORBA, and is seeking to enable inter-operability between them both within a single organisation and between organisations. SNMP has a large established base in the general purpose computing market, CMIP is mandated in the telecommunications arena by the TMN standard and CORBA is recognised as the emerging standard covering distributed object oriented programming. Each technology has its strength; thus full inter-operability will enable designers to select the most appropriate technology to apply to any given problem. The *ISO-Internet Management Coexistence (IIMC)* group of the Network Management Forum has addressed SNMP/CMIP interoperability. Thus, JIDM has chosen to concentrate on CMIP/CORBA and SNMP/CORBA inter-working.

To enable inter-working it is necessary to be enable the mapping between the relevant information models and to provide a mechanism to handle protocol conversion on the domain boundaries. This allows objects in one domain to be represented in the other domain and the interactions can be governed by the domain of choice rather than by the domain in which the target object is implemented. For example, an object in the CORBA domain should be able to interact with a GDMO object as if it were in the CORBA domain, ideally without having to know that the target object is in a different domain. Naturally the reverse is also desirable, that an OSI Manager should be able to manage CORBA objects as if they were defined in GDMO (this requires the reverse mapping).

A major advantage of the JIDM approach is that the strength of CORBA (object oriented system with well-defined APIs which are aimed at standardising and simplifying the task of creating distributed applications) can be combined with the strength of CMIP (powerful protocol with strong wire compatibility allowing integration of multi-vendor hardware) to give the best of both worlds. The implementor would have an effective environment in which to implement manager or agent functionality and yet be able to easily integrate components from multiple vendors. Figure 1 on the next page illustrates the main interoperability scenarios identified for OSI, SNMP and CORBA domains.

Section two gives an overview of JIDM's activities, the *Specifications Translation* and the *Interaction Translation.* Section three deals with *Specifications Translation,* it explains in four subsections the principles of the translation processes from ASN.1 and GDMO to CORBA IDL and vice versa, and from SNMP MIBs to CORBA IDL. The paper closes with hints on related work, conclusions and references.

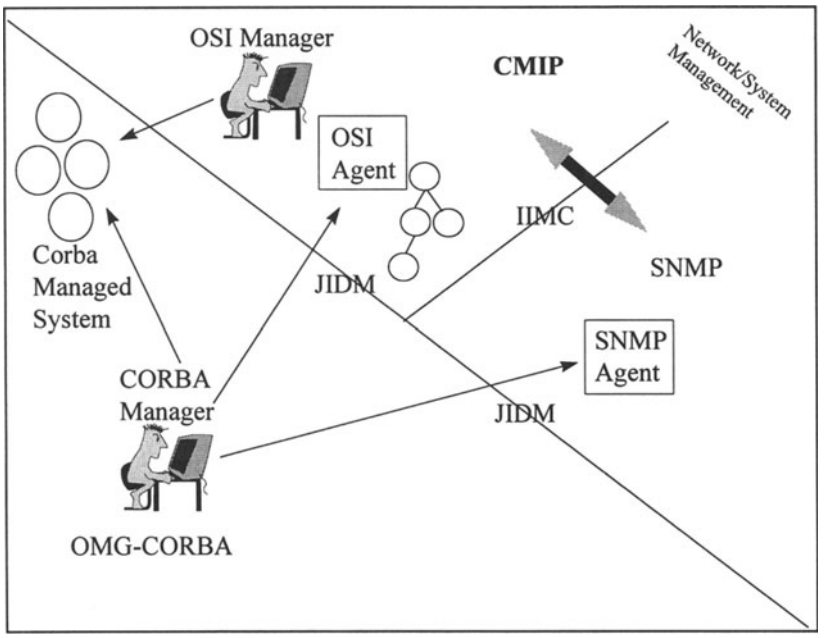

Figure 1: Inter-Working between CORBA, CMIP and SNMP

2 JOINT INTER DOMAIN MANAGEMENT

2.1 Specification Translation

The *Specification Translation* covers the process by which specifications are translated from one management domain to another. Algorithms for the static translation between GDMO/ASN.1 specifications and CORBA IDL interfaces, and for the static translation from SNMP MIBs to CORBA IDL have been defined. These algorithms may need to be extended later on to generate additional material for use in the Interaction Translation. When translating from GDMO/ASN.1 to IDL, trade-offs are encountered between enabling access to the full power of CMIP and generating simple IDL representations which simplify the application programmer's task. Wherever possible, these have been resolved according to the principle of keeping it simple in the most number of cases at the expense of making lesser used constructs more complex.

2.2 Interaction Translation

The *Interaction Translation* covers the process by which interactions in one management domain are transformed into corresponding interactions in the other domain. A *gateway*, the entity responsible for transforming interactions, may receive a CMIP PDU and must map this into one or more requests or replies on CORBA IDL interfaces. For example, if a scoped and filtered CMIP GET request is received, the gateway would have to identify the set of objects matching the filter within the scope and invoke the appropriate operation on each of those objects. The results would be collated and formatted into one or more CMIP PDUs in reply. Interaction Translation must cover initialisation identifying how the gateway is initialised and populated, and how it identifies the existing object population and what other service instances it may need to use. The gateway will probably make use of existing standard services in the CORBA domain, e.g. OMG Name Service to resolve Distinguished Names, OMG Lifecycle Service to create new object instances and the use of OMG Event Channels for event distribution.
Interaction Translation requires that the interactions be captured by a gateway which converts them in accordance with the mapping rules. Thus, in the OSI/CORBA scenarios, the gateway must:

- receive any incoming CMIP SET/GET/ACTION request and translate it into one or more invocations of methods supported by the addressed object(s),
- receive any event generated by an application object and translate it into a CMIP EVENT-REPORT request to be forwarded to those remote systems which had

registered their interest in receiving that kind of events,
- receive incoming method invocations and forward them as CMIP SET/GET/ACTION requests to the appropriate OSI agent,
- receive CMIP EVENT-REPORT requests and forward them as CORBA events to interested parties in the CORBA domain,
- receive any incoming CMIP CREATE/DELETE request and translate it into an invocation to a method being supported by an object (e.g. a factory object),
- receive method invocations for creating or deleting objects in a remote system and forward a CMIP CREATE/DELETE request to the addressed OSI agent.

This protocol conversion is complicated by such things as the need to map identifiers due to the differences between GDMO and IDL scoping and case-sensitivity, to map between GDMO Distinguished Names and CORBA Object References and to handle CMIP scoping and filtering requests which may require one CMIP request to be mapped to multiple sequences of IDL operations.

3 A SCENARIO

A common scenario is that a set of objects are defined in GDMO. The GDMO document is statically translated into IDL interfaces by applying the Specification Translation algorithm as explained in this text. A manager implemented as a set of CORBA objects, would manage objects supported by an OSI agent as if they were CORBA objects (i.e. via the generated IDL interfaces). These interfaces would be supported by a gateway supporting the Interaction Translation algorithm. This would dynamically translate IDL requests into CMIP PDUs based on the original GDMO specification. The Interaction Translation needs to bi-directionally translate between CMIP PDUs originating from the agent and the appropriate IDL requests and replies. If the Manager is an OSI manager, it generates CMIP PDUs exactly as if the objects were supported by an OSI agent. The CMIP protocol is transformed by the gateway into IDL interactions on the CORBA implemented object; the results of the IDL interactions are transformed back into CMIP PDUs.

For the rest of this paper, we will focus on the specification translation and implementation issues.

4 JIDM SPECIFICATION TRANSLATION

4.1 Assumptions and Principles

The algorithms have been designed using a number of guiding principles:

- **Completeness**. The aim was to provide a complete mapping as so far as it was possible. Rules have been provided for all cases regardless of their frequency.

- **Simplicity**: ASN.1 allows many constructs that are difficult to map into IDL. Many of these are not frequently used. In the light of the completeness principle, it was decided to select the simplest mapping for 80% of the cases allowing the remaining, more obscure cases, to be more complicated if necessary.

- **Reuse of OMG services**: The CORBA domain does not have a network management architecture; it provides a distributed processing environment. It is populated by an increasing number of Common Object Services such as naming and events which are useful building blocks. The JIDM group is trying to exploit these services where possible e.g. event services are used for OSI notifications.

- **Freedom of Implementation:** The JIDM group refrains from defining or constraining implementation unless it is absolutely necessary.

In order to translate between GDMO and CORBA IDL, (hereinafter referred to simply as IDL), or between SNMPv2 and IDL, it is necessary to be able to map the basic type definitions (i.e. mapping between ASN.1 types and IDL type definitions). This section describes the translation by addressing the mapping of the primitive types and of the constructed types.

There are two versions of ASN.1 defined, X.208 and X.680. Since GDMO explicitly builds on X.208 and all new GDMO will provide X.208 versions (at least in the short term), JIDM focused on that version. However, translation is also provided for all the basic types (e.g. BMPString and UniversalString) from X680 as a step towards migration to X.680.

ASN.1 has a much more complex type system than IDL. As a result, the translation necessarily risks to lose some information; e.g. tag values, value range constraints or compound type constants. Capturing this information for subsequent use in the run-time system is a key issue. Several schemes have been proposed including the use of string constants and #pragma directives, but there is no published solution yet. In some cases, the complexity of some data types makes it desirable to define operations for manipulating their values. In CORBA, the standard technique is to define pseudo

IDL (PIDL), which allows a fairly tight definition of the operations but with the implication that these operations have library implementations and can only be invoked locally. For example, this is used to provide access methods to support manipulation of the BitString data type.

4.2 ASN.1 to IDL translation

This section provides the translation of ASN.1 types in an ASN.1 module to IDL types in an IDL module.
Translate each ASN.1 module in the GDMO document (input file) to an IDL module.
Ignore ASN.1 macros and the export clause.
At the beginning of the module, translate each imported type in the ASN.1 module into a typedef of the corresponding imported IDL type and an #include statement for the module to be imported from.
Map each value assignment as a const declaration or method of the *ConstValues* interface of the generated module according to the rules.
Generate named IDL types for anonymous types and use the named type in the rest of the IDL. Repeat this recursively until all types are either named constructed types or one of the base types. This includes applying the rules for "COMPONENTS OF Type" and OPTIONAL and DEFAULT for sequences and sets.
Rearrange the order of generated types such that there are no forward references.
While generating the final output, resolve the conflict in ASN.1 identifiers due to case-insensitive nature of IDL identifiers and different rules for name scopes.

Note that the mapping is not direct in the sense that one ASN.1 type may be translated into several IDL types and constants. Some ASN.1 constants are not translated into IDL constants but into IDL operation.

4.2 GDMO to IDL Translation

GDMO to IDL translation is important from several points of view, e.g.

- It allows to implement GDMO MIBS as CORBA objects and thus implementing distributed OSI agents.
- It allows to give a CORBA manager an IDL view of the GDMO managed objects.

In this section, the Specification Translation process is described in terms of inputs and outputs and a rough outline of the process is given. The process can be implemented by a compiler which operates on a set of input files and produces some output files. Since IDL definitions are processed in terms of files which determine their granularity and reusability, it is necessary to specify which definitions are generated in which files.

In addition, GDMO adds some complexities since GDMO specifications use full text names e.g. "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2 : 1992". Since such names are used to import parts of other specifications, there must be a way for the translation process to access the files containing these specifications. In addition, it is desirable to be able to associate the resulting IDL files with the original GDMO to facilitate browsing and reuse. Both will be achieved by providing a *nickname database* which maps from the unique registered name of the GDMO document (or relevant Object Identifier) to a short nickname suitable for use as a filename base. This nickname will be used to find imported files and to control the names of the generated IDL files.

4.2.1 Outline of Translation Algorithm

- Translate ASN.1 types and constants according to the ASN.1 to IDL translation algorithm.
- Translate each GDMO managed object classes template into an IDL interface as follows:
    1. Retain the inheritance relationship of the class for the interface,
    2. Unwind the packages of the managed object class:
    3. Translate each attribute to operations on the interface in accordance to its associated property lists.
    4. Translate each action to an operation of the interface; in addition, generate interfaces for the support of multiple replies to be used with OMG event transmission mechanisms.
    5. Translate each notification to an operation to be used by OMG event services; in addition, generate interfaces for the handling of notifications in both, the *Push* and *Pull* model.
- Translate Parameter templates into IDL type definitions.
- Translate Behaviour templates into comments.
- Ignore Name Bindings templates.

Figure 2 summarises the translation process of GDMO documents.

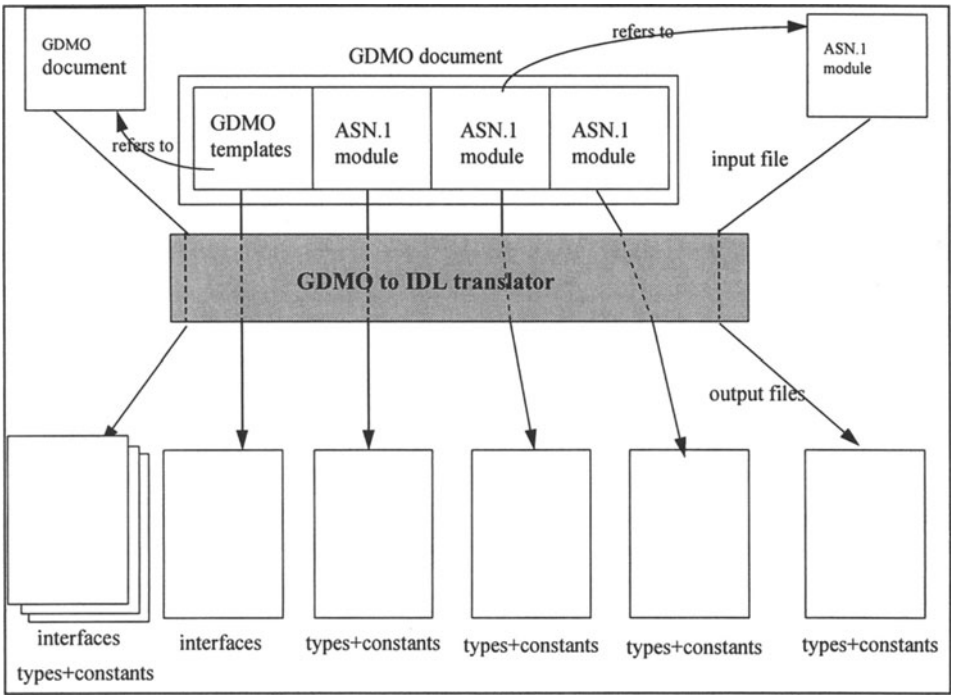

Figure 2: GDMO to IDL translation process

GDMO managed object classes are all derived directly or indirectly from the class *top,* as defined in X.722. In turn, *top* will be derived from a CORBA interface called *ManagedObject* which will hold generic information. For instance, operations on groups of attributes will be supported by the inherited interface *ManagedObject*; it will also offer operations for the support of multiple attribute access.

4.3 IDL to GDMO Translation

The CORBA-IDL to GDMO translation is important from several points of view, e.g.:

- It allows to manage CORBA applications from an OSI manager once the IDL interfaces are translated into GDMO/ASN.1.

- It allows to develop applications based on a CORBA IDL information model and to translate the model later on into GDMO/ASN.1, to be used to build OSI agents.

The basic guidelines of the translation algorithm are the following:

- Translate each IDL module into one GDMO document.
- Translate each IDL interface into one GDMO managed object class with one inlined package template.
- Translate exceptions into error parameters.
- Translate IDL attributes into GDMO attributes.
- Translate IDL operations into GDMO actions.
- Translate IDL types and constants into ASN.1 types constants.

4.4 SNMP-MIB to IDL Translation

The SNMP-MIB to IDL translation is important from several point of view, e.g.:

- It allows to implement an SNMP MIB as CORBA objects, thus opening up access to such an SNMP agents MIB to any CORBA client. Besides, such a kind of implementation will be very flexible and allow for easy extensibility.

- It allows to manage SNMP MIBs from a CORBA manager as soon as the SNMP MIB definition has been translated into CORBA-IDL.

The translation algorithm can be summarised as follows:

- Translate each SNMP information module into an IDL module:
- Process any ASN.1 type and value as described earlier in the ASN.1 to IDL section.
- Translate each Table entry into an interface where the entry components are attributes in this interface.
- Translate single-instance variable of a group into one interface where each variable is mapped into one attribute in this interface.
- Translate notifications (NOTIFICATION TYPE macro) into operations on a separate interface which will be used with OMG event channels.

This algorithm is compatible with the IIMC recommendations of the NMF for the SNMP to GDMO translation.

5 RELATED WORK

A taskforce called *telecom* sponsored by the Object Management Group is defining currently a TMN based on CORBA and going to use JIDM work. The ISO-ODMA(Open Distributed Management Architecture) group would use JIDM work to provide support for OSI management on top of ISO-ODP (Open Distributed Processing). A working group of the NMF and X/Open, called "*High Level CMIS API* " is very actively working towards a C++ embedding of GDMO and ASN.1 defined information models, as well as an embedding of ACSE and CMISE services. Their work will become especially important during the interaction translation phase, it has been of some impact on the specification translation phase. It should be noted, that the translation from GDMO/ASN.1 to C++ is less constraint than that of GDMO/ASN.1 to CORBA IDL.

6 CONCLUSION

The JIDM *specification translation algorithms* make it possible to bridge across the gaps between the different management domains, CORBA, CMIP and SNMP. Available information model in GDMO and ASN.1 can be translated into CORBA IDL specifications, and vice versa. Also SNMP MIB definitions can be translated into CORBA IDL. This allows to *reuse specifications* in a different domain, to *build gateways* between different domains, e.g. between CMIP and CORBA. The latter potential will be explored during the second, the *interaction translation* phase of JIDM.

- JIDM allows CORBA programmers to write OSI managers and OSI agents without needing to know GDMO, ASN.1, CMIP and its programming interfaces.
- JIDM allows GDMO and CMIS programmers to access CORBA implemented resources, services or applications without knowing CORBA.
- Implementing manager applications using CORBA provides a uniform view of the resources independent of whether they are basically managed via GDMO, SNMP, or even proprietary resources when the latter are encapsulated into CORBA objects. This is an important step toward integration of management systems.
- Once the JIDM interaction translation is finished, complete management platforms

can be based entirely on CORBA and access the non-CORBA world through gateways.

7 ACKNOWLEDGEMENT

The work described in this text has been done jointly by the JIDM team during the last three years. The authors wishes to thank the other JIDM task force members, mainly Subrata Mazumdar and Tom Rutt (both Lucent Technologies), Tim Roberts (BNR), Juan Hierro (Telefonica), and Martin Kirk (X/Open) for their ideas, work and co-operation.

8 REFERENCES

X/Open Preliminary Specification: "Inter Domain Management. Specification Translation". Final Draft, November 1996.

X/Open Preliminary Specification: "TMN/C++ Application Programming Interface", May 1996, with three parts: ASN/C++, May 1996, CMIS/C++, Aug. 1996 (planned) and GDMO/C++, Dec. 1996 (planned).

9 BIOGRAPHY

Nader Soukouti is senior engineer at Smile. He received his Ph.D. from the university Paris VI in 1995 and the computer engineer degree from HISSAT, Damascus, in 1988. Nader has been member of JIDM task force since its creation in 1993. His main domain interest is Network Management and CORBA.

Ulf Hollberg is senior consultant at the IBM European Networking Center in Heidelberg with architectural responsibility for IBM's TMN product components. Ulf received his Ph.D. from the University of Bonn. Ulf Hollberg is member of the JIDM team since 1993.