Intégration du modèle de l'information de gestion CIM dans l'approche OSI

Olivier Festor, Nizar Ben Youssef

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Intégration du modèle de l'information de gestion CIM dans l'approche OSI

Olivier Festor, Nizar Ben Youssef

No 3647

24 mars 1999

THÈME 1

ISSN 0249-6399

Intégration du modèle de l'information de gestion CIM dans l'approche OSI

Olivier Festor, Nizar Ben Youssef

Thème 1 — Réseaux et systèmes
Projet RÉSÈDAS

Rapport de recherche n 3647 — 24 mars 1999 — 35 pages

Résumé : Dans le domaine de la gestion des systèmes, réseaux et services, l'intégration représente toujours l'un des principaux sujets de préoccupation. La normalisation et la standardisation, entreprises il y a plus de dix ans, ont permis de réduire fortement le nombre d'approches sans pour autant atteindre le nirvana, c'est à dire une approche unique reconnue et utilisée par tous les intervenants du domaine de la gestion et déployée pour tous les systèmes et fonctions de supervision.

Partant de ce constat, de nombreux efforts ont été déployés pour construire des passerelles de supervision permettant l'intégration faible entre les différentes approches et fournissant un moyen pragmatique et efficace de construire des solutions de supervision intégrées. Dans ce rapport, nous proposons des règles de traduction de l'approche WBEM implantant le modèle CIM vers une approche OSI. Ce travail étend les approches existantes au travers de trois contributions originales qui sont le support dans un agent OSI du méta-modèle CIM, la prise en compte des relations et leur traduction en spécifications GRM et l'implantation en Java de l'intégrateur.

Mots-clé : CIM, CMIS, Passerelles, GDMO, GRM, Gestion Java, Intégration, OSI, TMN, WBEM

(Abstract: pto)

Ce travail a été partiellement réalisé dans le cadre de l'action ANTARES (Architectures et Nouvelles Technologies pour l'Administration des Réseaux et Service) du GIE DYADE.

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY (France)
Téléphone : 03 83 59 30 30 - International : +33 3 3 83 59 30 30
Télécopie : 03 83 27 83 19 - International : +33 3 83 27 83 19
Antenne de Metz, technopôle de Metz 2000, 4 rue Marconi, 55070 METZ

Integration of the CIM Management Information Model in the OSI Framework

Abstract: In the domain of Systems, Network and Service Management, integration still represents one of the major issues. Standardization, undertaken more than a decade ago, has led to reduce drastically the number of approaches without reaching its nirvana, i.e. one approach accepted by all players in the field of management and deployed for all management purposes.

Based on this fact many efforts have been deployed to build management gateways among the different approaches, providing a pragmatic and efficient way to reach integration. In this report, we propose a set of mappings allowing WBEM-based agents implementing a CIM information model to be managed by OSI-based management platforms and applications. Extending existing integration approaches, this report provides three original items which are the support of the CIM meta model in an OSI agent, the mapping of relationships onto GRM specifications as well as a full Java-based implementation of the integration environment.

Key-words: CIM, CMIS, Gateway, GDMO, GRM, Java Management, Management Integration, OSI, TMN, WBEM

1 Introduction

Malgré de nombreuses années de normalisation/standardisation, la supervision des systèmes, réseaux, services et applications, plus communément appelée gestion, est plus que jamais confrontée à des problèmes d'intégration d'approches hétérogènes. La normalisation est-elle entièrement responsable de ce fait ? Doit-elle être portée au pilori pour n'avoir pas atteint son but ? Certainement pas. Tout d'abord, force est de constater que les efforts de normalisation ont permis de réduire grandement le nombre d'approches propriétaires disponibles il y a quelques années. De plus, la normalisation a permis la reconnaissance à grande échelle des principaux paradigmes de gestion (Gestionnaire/Agent, modèles de l'information génériques, ...) qui forment aujourd'hui la base des architectures de gestion.

Les raisons qui motivent aujourd'hui la survie de nombreuses approches de supervision sont multiples. D'une part, il faut bien admettre que certaines approches sont mieux adaptées à certains niveaux de gestion que d'autres en terme de complexité, puissance d'expression, disponibilité d'environnements de développement, coût et modèles de l'information standards (ex. SNMP pour la gestion des équipements et des piles de communication de l'Internet, OSI pour la supervision des réseaux de télécommunication et CORBA pour l'intégration au niveau services). D'autre part, les investissements financiers énormes réalisés sur toutes ces approches en font les *legacy systems* de demain. Finalement, la croissance rapide des approches et l'explosion des technologies d'objets distribués ouvre des perspectives intéressantes à la communauté de supervision. Tout particulièrement l'utilisation combinée des technologies CORBA et Java semble appropriée au déploiement des systèmes de gestion de services de demain et apporterons de nouveaux paradigmes au domaine.

La normalisation des principes d'architecture pour le réseau de gestion des télécommunications (RGT) [CCITT.M.3010 92a] qui prend en compte les problèmes d'intégration au travers des concepts d'agents de médiation et d'agents d'adaptation (Q.Adaptation) a suscité de nombreux efforts dans la communauté pour la définition et la réalisation d'agents d'intégration pour la plupart des approches de supervision existantes.

L'une des approches sujette à l'intégration est l'approche WBEM (Web-Based Enterprise Management (WBEM) [Jander 96, Todd 97]. Initiée en 1996 au sein du DMTF (Desktop Management Task Force), cette approche gagne chaque mois une acceptation de plus en plus forte dans la communauté et des agents implantant cette approche apparaissent de plus en plus nombreux sur le marché. En effet, initialement *limitée* à la supervision des systèmes NT et Windows 98, WBEM est annoncée comme la solution de supervision pour de nombreux systèmes UNIX, notamment Compaq/DEC et plus récemment SUN pour ses plates-formes SOLARIS. Son développement implique que son intégration avec le monde du RGT [UIT-T.M.3000 92, CCITT.M.3010 92b] est aujourd'hui incontournable pour maintenir des objectifs de gestion de systèmes, réseaux et services de bout-en-bout.

A ce jour, peu d'efforts ont été portés sur l'intégration de ces deux approches et les principales propositions concernent l'accès au monde RGT depuis un superviseur basé sur l'approche WBEM. Ceci illustre une vision orientée services dans laquelle un réseau local est supervisé par une solution WBEM et sur lequel les informations du réseau externe et des services (ligne louée, réseaux privés virtuels) sont fournies via une interface RGT par un fournisseur de services.

Dans notre approche, nous considérons le besoin d'accéder à des agents WBEM depuis un gestionnaire OSI. Ce scénario se présente dans deux situations. La première se présente lorsqu'un usager délègue la gestion de son réseau à un fournisseur de service et/ou opérateur qui lui dispose d'une plate-forme RGT. La seconde se présente dès qu'un fournisseur de services et/ou opérateur et/ou entreprise autre ayant une plate-forme RGT déploie sur son réseau interne des systèmes gérés par des agents WBEM.

Dans ce contexte, l'objectif de notre contribution est de fournir d'une part des algorithmes de traduction des modèles de l'information CIM vers des spécifications GDMO/ASN.1 [ISO-10165.4 92, ISO-8824 90] d'autre part un environnement logiciel réalisant cette traduction et permettant la construction automatique d'un agent d'adaptation (Q.Adapteur dans la terminologie RGT).

Le document est organisé de la manière suivante. La section 2 résume les travaux existants dans le domaine de l'intégration de modèles de l'information et dans celui de la conception d'agents d'adaptation. La section 3 comporte une brève description comparative des deux approches (WBEM et OSI). La section 4 donne une vision globale du résultat recherché dans notre approche. Les sections 5 et 6 présentent de manière détaillée nos algorithmes de traduction. La section 5 est consacrée aux composants GDMO génériques (indépendants des modèles de l'information CIM spécifiques à traduire) utilisés dans notre approche. La section 6 détaille l'algorithme de traduction d'une classe MOF et de ses différents composants en une classe GDMO. La section 7 donne un exemple complet de traduction. La section 8 décrit l'implémentation de la version initiale de l'algorithme de traduction. La section 9 présente les extensions futures ou en cours apportées à l'algorithme de traduction. Finalement la section 10 conclut la présentation. Les spécifications génériques sont donnés en annexe.

2 Les travaux relatifs

Ces dernières années, de nombreux travaux ont été consacrés à l'analyse de méthodes d'intégration et au développement de multiples passerelles d'intégration de gestion.

Un approche générique de l'intégration des modèles de l'information décrivant notamment toutes les stratégies possibles est donnée dans [Rivière 98].

Les travaux les plus nombreux ont porté sur l'intégration SNMP/OSI. Dans [Abeck 93], une passerelle SNMP/OSI est définie. Cette proposition fournit une traduction des spécifications de modèles de l'information SNMP [Case 93] vers GDMO et propose un modèle réseau générique basé sur l'approche OSI. Pour la réalisation d'un agent d'intégration, les concepts d'agent *stateless*[1] et *stateful*[2] sont détaillés. Ils sont repris dans notre proposition.

Une approche générique d'intégration des modèles SNMP dans l'OSI est également proposée dans [Kalyanasundaram 93].

[Mazumdar 93] définit un mécanisme d'intégration au niveau de l'agent de gestion permettant à ce dernier de répondre à la fois à des requêtes CMIS et SNMP. Dans cette approche, une Base d'Informations de Gestion (MIB) indépendante de tout protocole d'accès est proposée en conjonction avec une interface d'accès générique. Au travers de cette interface, deux processeurs de protocoles sont responsables de l'accès aux objets et de la traduction au format de l'approche demandée des informations.

[McCarthy 95] détaille une implantation d'un agent d'intégration basé sur l'approche IIMC (ISO/Internet Management Coexistence) [(Editor) 96a, Chang 96, (Editor) 96b]. Cette approche définit des algorithmes de traduction des modèles de l'information ainsi qu'une architecture d'agent couplée à un algorithme de traduction des requêtes de communication.

Un agent d'intégration inverse permettant à des agents OSI d'être supervisés depuis des plates-formes SNMP est proposé dans [Koerner 97].

Dans le domaine de l'intégration des approches propriétaires, de nombreux travaux ont également été menés. Citons simplement les travaux de [Clemm 97] sur la supervision par SNMP d'équipements gérés par une approche TL.1.

Un travail considérable a été mené par le JIDM (Joint Inter-Domain Management Group) commun à l'X.Open et au NMF [Soukouti 97]. Dans ce travail, l'approche visée est l'intégration des approches OSI, SNMP et CORBA. Plusieurs documents sont issus de ces travaux. L'un fournit une traduction des spécifications GDMO/ASN.1 vers IDL et inversement. Un second fournit des algorithmes de traduction des modèles de l'information SNMP vers IDL et inversement. D'autres document traitent de l'intégration des interactions (CMIS,SNMP,CORBA) et de l'intégration de l'ensemble dans une architecture CORBA (utilisation des services CORBA). Un excellente présentation de ces algorithmes est donnée dans [Mazumbar 98]. Dans notre approche, nous avons réutilisé certains de principes issus de ces travaux, notamment certaines conventions de nommage.

Dans le domaine de la gestion WBEM et de son intégration, les propositions sont aujourd'hui rares. Une seule a été publiée à ce jour sous la forme d'un *white paper*. Il s'agit de la contribution de Vertel sur l'intégration WBEM OSI [Vertel 97]. Dans cette proposition, une intégration à double sens est envisagée mais l'accent est mis sur l'itégration OSI dans WBEM. L'étude détaillée de ce document mène à la conclusion que la proposition pose plus de problèmes qu'elle n'en résoud et se limite à des déclarations de bonnes intentions. Tout reste donc à faire.

1. Un agent qui ne maintient aucune valeur d'objet géré des agents qu'il adapte et ne réalise que des traductions de services.
2. Par opposition, un agent qui maintient un cache de tous les objets des agents qu'il adapte et qui se charge d'actualiser lui-même les valeurs.

3 Approche comparative des modèles et langages de spécification

Le but de cette section n'est pas de présenter en détails chacun des formalismes de description de modèles de l'information ni de fournir un tutoriel sur les deux approches de gestion considérées dans notre étude. Le lecteur trouvera les descriptions détaillées de ces approches dans [ISO-10165.4 92, DMTF 98]. Afin de présenter plus clairement nos algorithmes de traduction, nous résumons simplement les caractéristiques principales des deux approches de gestion utiles à la traduction.

Caractéristique	**OSI**	**CIM**
Langage de spécification	GDMO (Guidelines for the Definition of Managed Objects [ISO-10165.4 92], GRM (General Relationship Model)[CCITT.X.725 95], ASN.1 (Abstract Syntax Notation Number 1)[ISO-8824 90]	MOF (Managed Object Format) [DMTF 98]
Unité de spécification	Module (avec nommage global OSI, i.e. toute spécification a un identificateur permettant de la distinguer de manière unique et globale)	Schéma (avec nommage global de schémas et des finesses sur les noms des objets permettant une identification globale de ces derniers (i.e. concaténation du nom de schéma et du nom d'objet pour l'identifier de manière unique ...)
Héritage	Multiple avec résolution d'héritage d'attributs multiples règles de contraintes d'héritage strictes	Simple aucune règle de contrainte d'héritage clairement définie A priori pas de surcharge ni de redéfinition d'attribut autorisée
Portée de la définition des attributs	Globale (non liée à un objet), regroupement logique dans des paquetages	Locale à une classe
Types de donnès (attributs, paramètres, réponses)	Infini (toutes les possibilités offertes par ASN.1)	Ensemble limité de types de base (entiers, booléens, chaînes de caractères, tableaux)
Architecture de MIB	Arhitecture hérarchique basée sur les corrélations de noms	Architecture hiérarchique basée sur l'inclusion d'espaces de nommage
Nommage des instances	Nom distinctif (DN): séquence de couples <attribut,valeur>dans la hiérarchie de contenance d'objets gérés Un seul attribut de nommage par instance	Nommage hiérarchique suivant espace de nommages, noms de classes pour les classes et multiples attributs clefs pour les instances
Contenu des MIBs	Instances d'objets gérés	Classes et instances d'objets gérés + espaces de nommage + spécifications (qualifieurs, ...)
Relations entre objets gérés	– attributs pointeurs – classes et instances GRM – action et/ou opérations de gestion	Objets gérés spécifiques contenant des attributs de référence vers d'autres objets (*associations*)
Repository de spécifications	Fichiers de définition GDMO/ASN.1 ou via des fonctions de gestion spécifiques [ISO-10164.16 97], dans l'agent	Définitions présentes dans la MIB

Service de communication	CMIS/CMIP [ISO-9595 91, ISO-9596 91] Service de base pour la manipulation d'instances d'objets gérés: accès, modification de valeurs d'attributs, création, destruction La plupart des services peuvent s'appliquer à des groupes d'objets via les facilités de portée et de filtrage offertes par CMIS et l'architecture de MIB OSI.	HMMP [3] Service de base pour la manipulation, la création, destruction et modification de classes et d'instances. Définition d'un langage de requêtes orienté objets (HMQL)
Evènements	Template de spécification GDMO. Générés par les objets gérés qui les déclarent. Fonction de distribution des événements dans les MIB. Service de délivrance fourni dans CMIS	Objets gérés comme les autres. Notion de consommateur (idem que fonction de distribution de l'OSI Service de rapport existant

Les deux approches sont basées sur un paradigme objet pour la modélisation des informations de gestion. Les concepts de maintient dans les MIB d'agents des informations et l'accès à ces informations par un service de communication spécifique sont également proches. Le tableau 3 résume les principales caractéristiques et différences entre les approches. Ces caractéristiques sont simples à comprendre. Cependant pour mieux expliquer notre approche d'intégration, il nous faut revenir sur les aspects liés au nommage des objets gérés dans la MIB.

Les deux points suivants sont importants pour notre traduction de services :

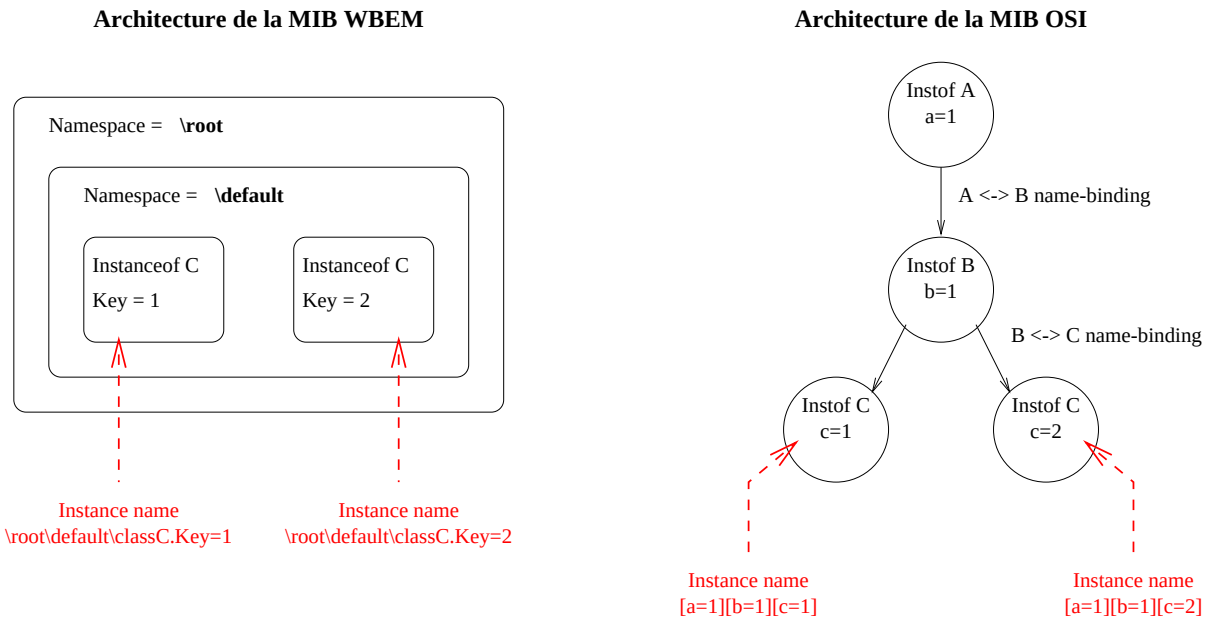

FIG. 1 – *Architecture des MIB OSI et WBEM*

1. L'architecture de la MIB : Bien que les terminologies différent pour chacune des approches, la MIB est dans les deux cas hiérarchisée sous forme d'un arbre appelé dans le modèle OSI *arbre de nommage*.

   Pour l'approche CIM, la hiérarchie de la MIB est définie par un concept d'espace de nommage comparable aux répertoires dans le système UNIX. Ainsi, dans l'arbre de la MIB, les espaces de nommage forment les noeuds et les objets gérés les feuilles.

3. Ce service est en train de disparaître sur NT au profit d'un accès COM/DCOM. Sur les autres systèmes le service n'est pas encore clairement défini...

Dans le contexte OSI, les objets sont hiérachisés par un concept de contenance. Les relations de corrélations de noms (*Name-Binding*) entre les classes premettent de spécifier les liens entre les objets de la MIB formant ainsi ces branches de l'arbre de nommage.

2. L'identification d'une instance : les instances d'objets sont identifiées par la liste des noeuds parcourus depuis la racine jusqu'à l'objet ainsi que par un identificateur local relatif au dernier noeud.

   Dans le modèle CIM, ceci se traduit par le nom absolu de l'espace de nommage, ainsi que par les valeurs prises par certains attributs discriminants dits attributs clefs (*Key properties*).

   Le modèle OSI comprend à peu près le même principe, sauf que seul un attribut peut être discriminant (*naming attribute*) et que la contenance se fait par rapport à d'autres objets de la MIB et non des objets uniquement destinés au nommage (i.e. les namespaces de WBEM). Ainsi, le nom absolu d'un objet de la MIB OSI sera composé par une liste des valeurs des attributs nommage de tous les objets formant le chemin, depuis la racine, jusqu'à l'instance recherchée.

La figure 1 illustre cette différence qui est importante dans la définition des algorithmes de traduction et les translations de service comme nous le verrons par la suite.

4 Vue globale de l'agent d'adaptation

La figure 2 illustre le résultat final recherché pour notre agent d'adaptation. Ce résultat comprend un ensemble d'outils logiciels permettant à un agent d'intégration d'offrir une vue OSI d'un ensemble d'agents WBEM.

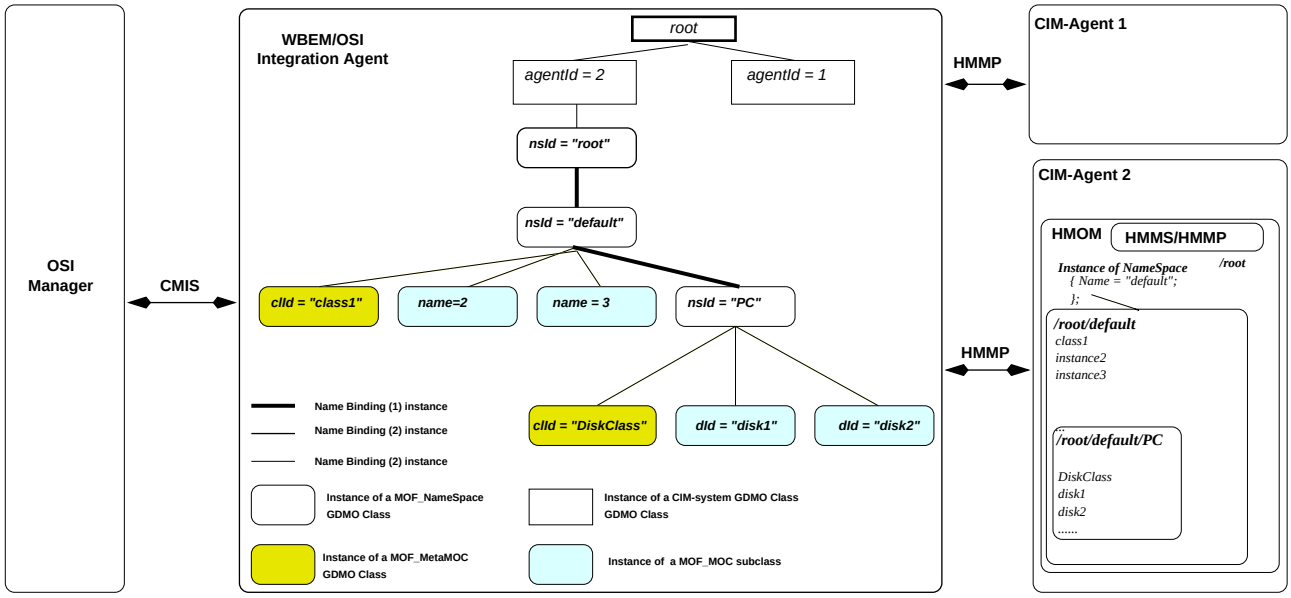

FIG. 2 – *Vue globale de l'adaptation*

L'agent d'intégration doit pouvoir adapter plusieurs agents WBEM simultanément et pour chaque agent, des correspondances entre objets gérés doivent être établies pour permettre à la MIB OSI visible d'offrir une vue hiérarchique des objets gérés qui soit similaire à la hiérarchie d'espaces de nommage tels qu'ils apparaissent dans les agents WBEM.

Dans la figure 2 nous avons illsutré cette hiérarchie en donnatnt la correspondance entre la MIB OSI générée et la mib WBEM fournie dans un agent (**CIM-Agent2**). Cet agent comprend la hiérarchie d'espaces de nommage suivante :

- `root`
- `root/default` : cet espace de nommage comporte également une spécification de classe *class1* ainsi qu'une spécification de deux instances de cette classe *instance1* et *instance2* ;
- `root/default/PC`: qui comporte également une spécification de classe *DiskClass* et deux instances de cette classe *disk1* et *disk2*.

L'on désire pouvoir retrouver cette hiérarchie dans la MIB OSI correspondante (*WBEM/OSI Intgration Agent*). Celle-ci est illustrée sous forme d'objets gérés et de corrélations de noms dans l'agent d'intégration. On y retrouve la hiérarchie correspondant aux espaces de nommages (ici sous forme d'objets gérés contenus) ainsi que les instances d'objets gérés. Une catégorie supplémentaire d'objets gérés est insérée dans l'agent d'adaptation : les objets de description de classes MOF. Ils seront détaillés dans les sections suivantes.

Permettre la construction de ce type d'agent d'intégration nécessite :

- une définition de classes génériques utilisées dans l'agent d'intégration ,
- un algorithme de traduction générique générant pour chaque classe MOF, une classe GDMO associée ,
- une approche pour la traduction des services.

Ces différents points sont présentés dans les sections suivantes

5 L'intégration des modèles d'information

Cette section a pour objectif de présenter notre algorithme de traduction de modèles de l'information issus de l'approche CIM vers des modèles OSI exprimés en GDMO et maintenus dans une MIB OSI.

Pour la traduction des modèles de l'information, nous avons opté pour une technique de traduction de *recast* permettant de définir une traduction automatique entre modèles. Une telle approche nécessite la définition des correspondances entre les différents éléments de chacun des méta-modèles. Les sections suivantes détaillent ces correspondances.

5.1 Classes, attributs et corrélations génériques

Notre approche propose de définir en GDMO un certain nombre de classes, d'attributs et de corrélations génériques qui serviront de base pour la traduction des spécifications MOF vers des formulaires GDMO, ainsi que pour la construction d'une MIB OSI équivalente à une MIB WBEM. Ces classes et corrélations sont données dans la figure 3.

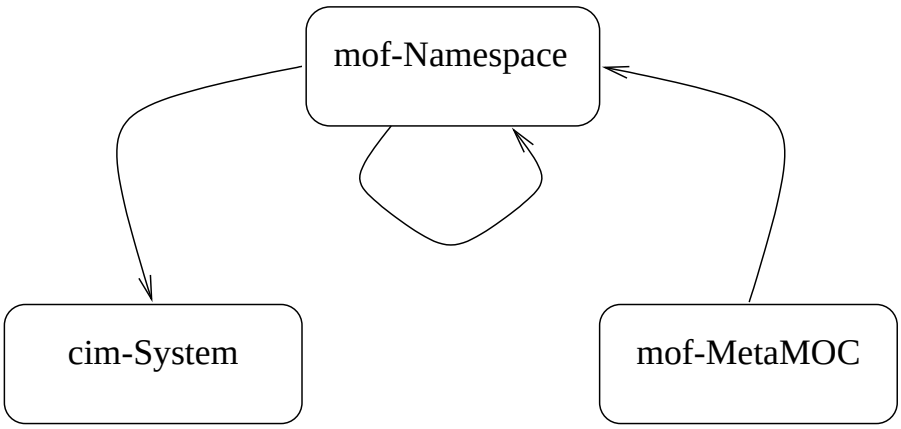

FIG. 3 – *Les classes et corrélations génériques*

Les classes génériques sont les suivantes :

- **mof-MOC**: définit une classe GDMO de base de laquelle hériterons toutes les classes GDMO correspondant à une spécification de classe dans un modèle WBEM ;
- **mof-NameSpace**[4]: définit une classe de description d'un composant d'espace de nommage de l'approche WBEM. À chaque espace de nommage dans un schéma CIM, on établira une instance de la classe mof-NameSpace dans la MIB OSI correspondante ;

4. Il n'est pas possible de construire une corrélation de noms générique entre une classe mof-MOC et une classe mof-Namespace pour deux raisons. D'une part, l'attribut de nommage référencé dans toute corrélation de noms pour l'attribut subordonné change d'une classe à l'autre. D'autre part, il existe des classes CIM qui ne sont pas instanciables et pour lesquelles on ne peut donc pas définir de corrélation de noms.

- **mof-MetaMOC**: définit une classe GDMO de base qui modélise toute définition de classe CIM. À toute définition de classe dans un schéma CIM, on associera une instance de la classe mof-MetaMOC de la MIB OSI correspondante ;
- **mof-Qualifier**: définit une classe GDMO qui modélise toute définition de qualifieur MOF. À toute définition de qualifieur dans un fichier MOF on associera une instance de mof-Qualifier.

A ces classes s'ajoutent deux définitions de corrélations de noms qui serviront à automatiser la projection d'une MIB WBEM dans le modèle OSI. Ces corrélations sont les suivantes :

- **mof-NS-NS-NameBinding** : elle permet de traduire les relations de contenance entre les espaces de nommage supportés par le modèle CIM ;
- **mof-NS-Meta-NameBinding** : elle permet d'inclure des définitions de classes CIM dans la MIB OSI (à travers les instances de la classe mof-MetaMOC).

Ces spécifications, que nous détaillons dans les paragraphes suivants sont inclues dans un module GDMO générique baptisé *BasicModule* qui sera chargé systématiquement par le traducteur.

De plus, nous avons défini dans ce module générique un ensemble d'attributs "types" sur lesquels s'appuyera la syntaxe de la plupart des attributs générés par le traducteur. L'utilisation de ces attributs "types" permet essentiellement de donner les règles de correspondance (matching rules) pour les types MOF simples. La spécification de ces attributs est donnée en annexe.

5.1.1 La classe mof-MOC

Elle ne comporte aucun attribut. Elle sert juste à donner une racine à l'arbre d'héritage des classes GDMO issues des traductions des spécifications MOF. Le formulaire GDMO spécifiant cette classe dans le module générique *BasicModule* est donné dans la figure 4.

```
"BasicModule":mof-MOC  MANAGED OBJECT CLASS
 DERIVED FROM "Rec. X.721 | ISO/IEC 10165-2 : 1992":top;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) classes(3) 1 };
```

FIG. 4 – *La classe mof-MOC*

5.1.2 La classe mof-Namespace

Elle correspond à une traduction directe de la classe __Namespace définie dans le schéma *Core* du modèle CIM. Elle comporte un seul attribut de type chaîne de caractères qui donne le nom de l'espace de nommage. Les formulaires GDMO spécifiant cette classe dans le module générique *BasicModule* sont donnés dans la figure 5.

```
"BasicModule":mof-Namespace  MANAGED OBJECT CLASS
 DERIVED FROM "BasicModule":mof-MOC ;
 CHARACTERIZED BY "BasicModule":mof-NamespacePackage ;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) classes(3) 2 };

"BasicModule":mof-NamespacePackage  PACKAGE
   ATTRIBUTES
    "BasicModule":mof-NamespaceName GET ;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) packages(4) 1 } ;

"BasicModule":Name  ATTRIBUTE
 DERIVED FROM "BasicModule":GraphicStringAttribute ;
 REGISTRED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 1 } ;
```

FIG. 5 – *La classe mof-Namespace*

5.1.3 La classe mof-MetaMOC

Elle correspond à une description du méta-modèle d'une classe CIM par le formalisme GDMO. Elle comprend alors 5 attributs:

- className : qui donne le nom de la classe MOF ;
- superClassName : donne le nom de la super-classe MOF ;
- propertyBufferList : donne la liste des spécifications des attributs de la classe MOF ;
- methodBufferList : donne la liste des spécifications des méthodes de la classe MOF ;
- qualifierList : donne la list des qualifieurs attachés à la classe.

Ainsi chaque instance de cette classe donnera une description d'une classe MOF. Les formulaires GDMO spécifiant cette classe dans le module générique *BasicModule* sont donné dans la figure 6.

```
"BasicModule":mof-MetaMOC  MANAGED OBJECT CLASS
 DERIVED FROM "BasicModule":mof-MOC ;
 CHARACTERIZED BY "BasicModule":mof-MetaMOCPackage ;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) classes(3) 2 };

"BasicModule":mof-MetaMOCPackage  PACKAGE
    ATTRIBUTES
     "BasicModule":mof-MetaClassName GET_REPLACE,
     "BasicModule":mof-MetaSuperClassName GET-REPLACE,
     "BasicModule":mof-MetaPropertyBufferList GET-REPLACE,
     "BasicModule":mof-MetaMethodBufferList GET-REPLACE,
     "BasicModule":mof-MetaQualifierList GET-REPLACE;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) packages(4) 2 };

"BasicModule":className  ATTRIBUTE
    DERIVED FROM "BasicModule":GraphicStringAttribute;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 1 };

"BasicModule":superClassName  ATTRIBUTE
    DERIVED FROM "BasicModule":GraphicStringAttribute;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 2 };

"BasicModule":propertyBufferList  ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.PropertyBufferList;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 3 };

"BasicModule":methodBufferList  ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.MethodBufferList;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 4 };

"BasicModule":qualfierList  ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.QualifierList;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 5 };
```

FIG. 6 – *La classe mof-Meta MOC*

5.1.4 La classe mof-Qualifier

Elle correspond à la description du modèle de spécification d'un qualifieur MOF à l'aide du formalisme GDMO. Elle comprend alors 4 attributs :

- QualifierName : donne le nom du qualifieur.
- QualifierType : donne le type du qualifieur MOF.
- QualifierScopeList : donne la liste des éléments du méta-modèle sur lesquels le qualifieur peut agir.

– QualifierFlavorList : donne une liste de caractéristiques du qualifieur.

Chaque instance de cette classe donnera ainsi une description d'un qualifieur MOF. Les instances de mof-Qualifieur seront inclues sous une racine particulière de la MIB OSI qui leur sera exclusivement réservée.

```
"BasicModule":mof-QualifierDef  MANAGED OBJECT CLASS
 DERIVED FROM "BasicModule":mof-MOC ;
 CHARACTERIZED BY "BasicModule":mof-QualifierDefPackage ;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) classes(3) 3 };

"BasicModule":mof-QualifierDefPackage  PACKAGE
     ATTRIBUTES
      "BasicModule":mof-QualifierDefName GET ,
      "BasicModule":mof-QualifierDefType GET ,
      "BasicModule":mof-QualifierDefScope GET ,
      "BasicModule":mof-QualifierDefFlavor GET ;
  REGISTERED AS { autoMOFtoGDMO BasicModule(0) packages(4) 3 };

 "BasicModule":MOf-QualifierDefName  ATTRIBUTE
     DERIVED FROM "BasicModule":GraphicStringAttribute ;
  REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 5 };

 "BasicModule":mof-QualifierDefType  ATTRIBUTE
     DERIVED FROM "BasicModule":GraphicStringAttribute ;
  REGISTERED AS { BasicModule attributes xx };

 "BasicModule":mof-QualifierDefScope  ATTRIBUTE
     WITH ATTRIBUTE SYNTAX ASNBasicModule.ScopeList;
  REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 8 };

 "BasicModule":mof-QualifierDefFlavor  ATTRIBUTE
     WITH ATTRIBUTE SYNTAX ASNBasicModule.FlavorList;
  REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 9 };
```

FIG. 7 – *La classe mof-Qualifier*

5.1.5 La corrélation mof-NS-NS-NameBinding

Cette corrélation permet de définir un lien de contenance entre deux objets de la classe mof-Namespace. Ainsi, l'imbrication des espaces de nommage définissant l'hiérarchie d'une MIB WBEM aura une représentation directe au niveau de la MIB OSI image (voir sec. 5.3).

Pour cette corrélation, nous utiliserons le nom de l'espace de nommage comme attribut de nommage (clause **WITH ATTRIBUTE**).

```
"BasicModule":mof-NS-NS-NameBinding  NAME-BINDING
 SUBORDINATE OBJECT CLASS "BasicModule":mof-Namespace ;
 NAMED BY SUPERIOR OBJECT CLASS "BasicModule":mof-Namespace ;
 WITH ATTRIBUTE "BasicModule":mof-NamespaceNameAttribute ;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) namebindings(6) 1 };
```

FIG. 8 – *La corrélation de noms mof-NS-NS-NameBinding*

5.1.6 La corrélation mof-NS-Meta-NameBinding

Cette corrélation permet d'instancier des objets de la classe mof-MetaMOC dans la MIB OSI. Ainsi, le modèle de l'information (i.e. les définitions de classes) pourra être accessible aux services de gestion OSI.

L'attribut de nommage utilisé pour cette corrélation est le nom de la classe MOF (i.e. l'attribut ClassName de la classe mof-MetaMOC).

```
"BasicModule":mof-NS-Meta-NameBinding  NAME-BINDING
 SUBORDINATE OBJECT CLASS "BasicModule":mof-MetaMOC ;
 NAMED BY SUPERIOR OBJECT CLASS "BasicModule":mof-Namespace ;
 WITH ATTRIBUTE "BasicModule":mof-MetaClassNameAttribute ;
 REGISTERED AS { autoMOFtoGDMO BasicModule(0) namebindings(6) 2 }
```

FIG. 9 – *La corrélation de noms NS-Meta-Binding*

5.2 L'enregistrement des formulaires

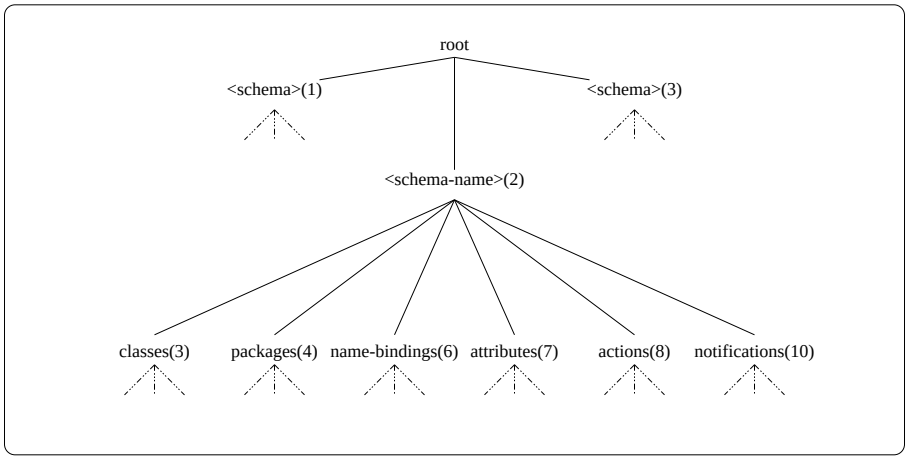

FIG. 10 – *L'enregistrement des formulaires*

Le modèle CIM ne supporte pas la notion d'enregistrement des objets comme dans le modèle OSI. Nous construisons alors un arbre d'enregistrement pour identifier chacune des spécifications GDMO issue des traductions. Pour la génération de l'arbre de nommage GDMO, nous appliquons les règles suivantes (illustrée dans le schéma 10) :

- on propose pour racine d'enregistrement des spécifications traduites *autoMOFtoGDMO* l'identificateur **{ 1 2 3 4 5 6 69 }**[5] ;
- à chaque schéma correspond une entrée dans l'arbre de nommage ;
- à chaque type de formulaire correspond une entrée dans l'arbre de nommage sous la racine du schéma contenant la spécification ;
- à chaque nouveau formulaire d'un type donné est attribué un numéro d'enregistrement. Ce numéro correspond à une incrémentation séquentielle dans l'ordre d'apparition parmis les spécifications du même type. Ce numéro commence par 1 pour chaque schéma.

5.3 Traduction des espaces de nommage

Toute MIB d'un gestionnaire d'objets CIM est structurée en espaces de nommages qui peuvent être imbriqués. Afin de retrouver cette structuration dans l'agent de Q.Adaptation, notre algorithme traite une instance d'espace de nommage comme un objet géré et lui associe une instance dans la MIB OSI correspondante. Pour modéliser cette imbrication dans le contexte OSI, nous nous appuyons sur la corrélation mof-NS-NS-NameBinding définie dans le module générique *BasicModule*.

La figure 11 illustre le mécanisme de traduction des espaces de nommage.

5. Une démarche est actuellement en cours pour l'obtention d'un numéro d'enregistrement INRIA.

Type de formulaire	Identificateur d'enregistrement
ASN.1 Module	asn(2)
Managed object class	classes(3)
Package	packages(4)
Name binding	namebindings(6)
Attribute	attributes(7)
Action	actions(9)

TAB. 2 – *Identificateurs d'enregistrement des formulaires*

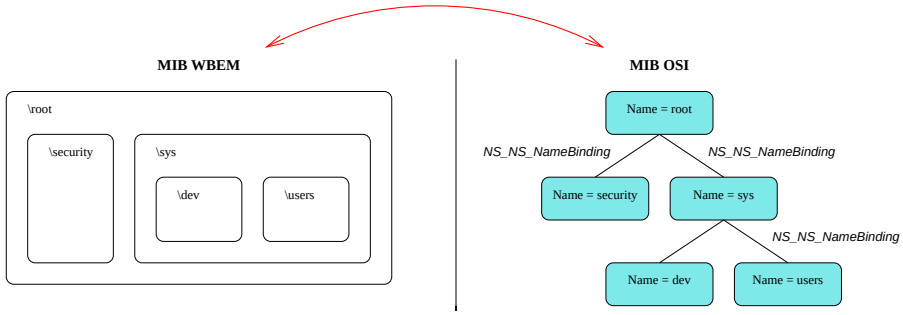

FIG. 11 – *Traduction des espaces de nommage*

6 La traduction d'une spécification de classe

La traduction d'une classe s'effectue à deux niveaux différents: au niveau de la base d'objets gérés permettant de manipuler les définitions des classes MOF, ainsi qu'au niveau du *repository* contenant le modèle de l'information.

1. instanciation d'un objet géré de la classe mof-MetaMOC qui va contenir la spécification proprement dite ;
2. génération de la spécification de classe GDMO correspondante à la classe MOF.

Dans les paragraphes qui suivent nous détaillons ce dernier point, c'est à dire les correspondances entre les modèles.

6.1 Le schéma global de correspondance de classe

La figure 12 illustre le schéma de traduction d'une spécification de classe MOF vers la spécification de classe GDMO correspondante.

Une spécification de classe dans un schéma CIM se traduit par une spécification de classe dans le module correspondant au schéma. Le label d'une classe MOF est traduit par un label identique étendu par les préfixes "mof-" et le nom du schéma dans lequel la classe est déclarée.

Si la classe à traduire hérite d'une autre classe (MOF: ligne 4), la classe GDMO correspondante comporte ce lien d'héritage dans sa clause **DERIVED FROM** (GDMO : ligne 4). Si la classe MOF à traduire n'hérite d'aucune super-classe, la classe GDMO correspondante héritera de la classe générique *mof-MOC* (GDMO: ligne 3).

À chaque formulaire de classe d'objet géré issu de la traduction sont associés deux autres formulaires GDMO :

- un formulaire de module obligatoire (clause **CHARACTERIZED BY**) déclarant les attributs et les actions de la classe MOF (GDMO : lignes 5,6) ;
- un formulaire de comportement (**BEHAVIOUR**) décrivant sous forme de commentaires les qualifieurs qui sont associés à la classe MOF [6] (GDMO : lignes 7,8).

6. Ceci n'est supporté que dans MODERES. Pour respecter exactement la syntaxe GDMO, il faut générer ce comportement dans le paquetage obligatoire de la classe.

MOF

```
[ qualifiers]
 CLASS < class-label>
   [ AS < alias-identifier>]
   [ : < super-class-label>]
 {
...
 }
 ;
```

GDMO

```
 mof-<schema_name>-<class-label>Class
     MANAGED OBJECT CLASS
( DERIVED FROM  "BasicModule":mof-MOC ;
|  CHARACTERIZED BY  mof-<schemaName>-<super-class-label>; )
 PACKAGES
   mof-<schema-name>-<class-label>Package;
[ BEHAVIOUR
   mof-<schema-name>-<class-label>BehaviourPackage; ]
 REGISTERED AS
   { autoMOFtoGDMO <schema-Ind> classes(3) xx }  ;
```

FIG. 12 – *La traduction d'une classe MOF*

6.2 Le contenu du package de classe

Le formulaire de module (**PACKAGE**) permet dans le formalisme GDMO de spécifier les différents composants des classes (attributs et actions). Le traducteur génère pour chaque spécification de classe MOF un formulaire de module (figure 13). Le label du formulaire de module est identique à celui de la classe MOF étendu par les préfixes "mof-" et le nom du schéma dans lequel la classe MOF est déclarée, ainsi que par le suffixe clef **Package**.

```
 mof-<schema-name>-<class-label>Package  PACKAGE
[  ATTRIBUTES
     mof-<schema-name>-<class-label>-<property-label>Attribute  <ACCESS>
    [, mof-<schema-name>-<class-label>-<property-label>Attribute  <ACCESS>]*;]
[  ACTIONS
     mof-<schema-name>-<class-label>-<method-label>Action
    [, mof-<schema-name>-<class-label>-<method-label>Action]*;]
 REGISTERED AS { autoMOFtoGDMO <schema-Ind> packages(4) xx }  ;
```

FIG. 13 – *Le package de classe*

Ce formulaire déclare les attributs et les actions de la classe GDMO issus de la traduction des propriétés et des méthodes de la classe MOF correspondante :

- 1 propriété MOF ⇒ 1 attribut GDMO : les attributs GDMO sont déclarés dans la clause **ATTRIBUTES**. Le label de propriété MOF est traduit par un label identique étendu par les préfixes "mof-", le nom du schéma et le nom de la classe MOF , ainsi que par le suffixe clef **Attribute**.
- 1 méthode MOF ⇒ 1 action GDMO : les actions GDMO sont déclarées dans la clause **ACTIONS**. Le label de méthode MOF est traduit par un label identique étendu par les préfixes "mof-", le nom du schéma et le nom de la classe MOF, ainsi que par le suffixe clef **Action**.

Le formulaire de module est enregistré sous la racine correspondant aux formulaires de module du schéma MOF, avec le même numéro d'identification que celui attribué au formulaire de classe d'objet géré.

6.3 Traitement des qualifieurs de classe

Dans sa version actuelle, notre algorithme traîte de façon identique la majorité des qualifieurs de classe : ils sont inclus sous forme d'un commentaire dans un formulaire de comportement associé à la classe (section 6.10).

Seul le qualifieur `Abstract`, défini dans le shéma *Meta* du modèle CIM pour signifier que la classe est non instanciable, échappe à cette règle. Ainsi, nous tenons compte de la présence de ce qualifieur pour la génération d'une corrélation de noms entre la classe traduite et la classe mof-Namespace : seules les classes instanciables pourrons être corrélées (voir génération des corrélations de noms, sec. 6.8).

6.4 Traduction des attributs

La propriété dans les spécifications MOF est caractérisée par son nom, son type, sa valeur par défaut, et par la liste des qualifieurs qui peuvent lui être attachés. Ainsi, La traduction d'une propriété MOF se fait en deux étapes.

- Définition de l'attribut GDMO : elle se fait à partir des caractéristiques de base de la propriété MOF (nom, type, valeur par défaut) dans un formulaire d'attribut GDMO (figure 14);
- Le formulaire d'attribut ainsi que le formulaire de module, sont complétés ensuite par l'analyse de la liste des qualifieurs attachés à la propriété MOF.

Pour définir la syntaxe d'un attribut, le formalisme GDMO offre deux possibilités : l'héritage de la syntaxe d'un autre attribut, ou la référence vers une définition ASN.1. Dans notre approche nous utilisons l'héritage de syntaxe pour les types MOF simples (clause **DERIVED FROM** ligne 2) et dans ce cas, nous faisons références aux attributs "types" définis dans le module générique *BasicModule*, ou la référence vers une définition de type ASN.1 généré lors des traductions pour les types composés (clause **WITH ATTRIBUTE SYNTAX** ligne 3).

La clause **MATCHES FOR** définit les règles de correspondances qui peuvent être associées à l'attribut GDMO. Ces règles sont générés de manière automatique en fonction du type de l'attribut (voir table 3). Pour les attributs spécifiés par héritage, les règles de correspondances ne sont pas redéfinies.

La clause **BEHAVIOUR** optionnelle déclare le formulaire de comportement de l'attribut GDMO. Ce formulaire contient, une description sous forme de commentaires des qualifieurs MOF qui n'ont pas été retenus par l'algorithme de traduction des qualifieurs d'attributs(cf 6.5). Pour une vision facilitée des qualifieurs, une extension de l'algorithme permet de les inclure tous dans la clause de comportement.

Le formulaire d'attribut est enregistré (clause **REGISTERED AS**) sous la racine réservée aux attributs du schéma. A chaque formulaire est attribué un numéro correspondant à une incrémentation séquentielle dans l'ordre d'apparition des propriétés MOF.

```
1  mof-<schema-name>-<class-label>-<property-label>Attribute  ATTRIBUTE
2  ( DERIVED FROM "BasicModule":<attribute-type>;
3  |
2     WITH ATTRIBUTE SYNTAX  <reference-type>; )
3  [ MATCHES FOR  qualifier [,  qualifier]*;
3  [ BEHAVIOUR
4       mof-<schema-name>-<class-label>-<property-label>AttributeBehaviour;]
5  REGISTERED AS { autoMOFtoGDMO <schema-Ind> attributes(7) yy }  ;
```

FIG. 14 – *La spécification d'attribut*

6.5 Traitement des qualifieurs d'attributs

D'une manière générale, les qualifieurs d'attributs sont traduits sous forme de commentaires dans un formulaire de comportement associé à l'attribut GDMO.

Dans notre traducteur, nous n'avons tenu compte de manière plus spécifique que de certains qualifieurs définis dans le schéma *Meta* du modèle CIM. Seuls ces qualifieurs sont traduits par des éléments de spécification GDMO directement exploitables par des applications de gestion.

6.5.1 Les droits d'accès

Dans le formalisme GDMO, les droits d'accès à l'attribut sont donnés à la déclaration de l'attribut dans le formulaire de module par l'une des propriétés suivantes: GET, REPLACE, GET-REPLACE.

Dans le formalisme MOF, ces droits sont fixés par les attributs Read et Write définis dans le schéma *Core* (voir table 3).

```
Qualifier     read : bool = true, Scope( property );
Qualifier     write : bool = true, Scope ( property );
```

Par défaut ces qualifieurs ont la valeur TRUE. Ainsi la propriété d'accès associée à l'attribut GDMO est fixée à GET-REPLACE si aucun de ces qualifieurs n'est attaché à la propriété MOF. Sinon, la valeur de ces qualifieurs est prise en compte pour déterminer les droits d'accès aux instances de la classe (voir table 3).

Qualifieur **Read**	Qualifieur **Write**	Droit d'accés GDMO
TRUE	TRUE	GET-REPLACE
TRUE	FALSE | NOT PRESENT	GET
FALSE | NOT PRESENT	TRUE	REPLACE
FALSE	FALSE	GET + Warning: wrong access specification.

TAB. 3 – *Traduction des qualifieurs read et write*

La sémantique associée à l'absence d'un qualifieur Read ou Write lorsque l'autre est présent est définie comme la volonté du spécifieur de ne pas autoriser l'opération associée. Donc en cas d'absence de l'un des 2 qualifieurs, la lecture, resp. l'affecation ne sera pas autorisée dans la spécification GDMO correspondante.

6.5.2 Les clefs de nommage

Dans le formalisme MOF, l'identification d'une instance d'une classe donnée se fait à l'aide des valeurs des attributs de nommage. Ces attributs sont alors appelés attributs clefs, et sont spécifiés grâce au qualifieur **Key** du schéma *Core* (cf 3).

```
Qualifier     Key : bool = false, Scope( property, reference ), Flavor( DisableOverride );
```

La traduction du qualifieur **Key** se fait lors de la génération du formulaire de corrélation de nom **NAME-BINDING** (voir 6.8) à travers la déclaration de l'attribut de nommage. Elle a des implications éventuelles sur les attributs de la classe subordonnée.

6.6 Traduction des actions

A chaque méthode définie dans une classe, on associe une spécification d'action GDMO. Cette spécification est également référencée dans le module (*package*) de la classe correspondante.

La méthode MOF est décrite par son nom, la liste des paramètres de son invocation ainsi que par le type des données qu'elle retourne. Dans le formalisme GDMO, ces paramètres sont décrits dans le formulaire **ACTION** (cf 15).

Toutes les méthodes MOF sont déclarées par défaut en mode confirmé par la clause **MODE CONFIRMED** (ligne 2).

La clause **WITH INFORMATION SYNTAX** fait référence au type du paramètre en entrée de l'action. Si dans la spécification MOF la méthode comporte plusieurs paramètres d'entrée, le type du paramètre d'entrée referencé dans l'action GDMO sera une séquence ASN.1 (SEQUENCE) composée des types des différents paramètres MOF. Cette séquence ASN.1 sera déclarée avec le nom suivant :
mof-<schema-name>-<class-label>-<method-name>MethodInformation

La clause **WITH REPLY SYNTAX** fait référence au type du paramètre de retour de l'action GDMO. Ce type est directement obtenu à partir de la table de traduction des types MOF vers des types ASN.1.

Le formulaire d'action est enregistré sous la racine réservée aux actions du module. A chaque formulaire est attribué un numéro correspondant à une incrémentation séquentielle dans l'ordre d'apparition des méthodes MOF.

```
1 mof-<schema-name>-<class-name>-<method_label>Action ACTION
2   MODE CONFIRMED
3   WITH INFORMATION SYNTAX <reference-type>;
4   WITH REPLY SYNTAX <reference-type>;
5 REGISTERED AS { autoMOFtoGDMO <schema-Ind> actions(9) xx } ;
```

FIG. 15 – *La spécification d'une action*

6.7 Traduction des associations

Une association CIM, est définie dans le formalisme MOF comme une classe contenant le méta-qualifieur `Association` et pouvant contenir des attributs de types références.

Dans la version actuelle du traducteur, nous avons gardé le même algorithme de traduction que celui utilisé pour la traduction d'une classe. L'attribut référence vers un objet étant traduit comme un attribut GDMO ayant la syntaxe ASN.1 `ObjectInstance`.

Une meilleur prise en compte des associations par utilisation des concepts du GRM reste envisagée pour les futures versions du traducteur.

6.8 Le nommage des instances

Chaque modèle utilise une stratégie différente pour le nommage des instances : le modèle CIM s'appuie sur un nommage par un (ou plusieurs) attribut(s) clé(s) dans un contexte d'espace de nommage (*namespace*) donné, alors que le modèle OSI s'appuie sur des relations de contenances entre classes spécifiées par des formulaires de corrélations de noms.

Ainsi, pour garder la même hiérarchie de nommage définie dans le modèle WBEM, nous générons pour chaque classe instantiable une corrélation de nom avec la classe mof-Namespace.

De plus, CIM fournit un mécanisme pour "émuler" la relation de contenance, en spécifiant un type particulier d'associations dites "faibles" (*weak associations*) à l'aide du qualifieur **weak**. Pour traduire cette relation, nous proposons de générer une autre corrélation de noms entre les classes participantes à ces associations.

6.8.1 Le formulaire de corrélation de noms générique

Il s'agit des corrélations que nous générons entre toute classe instanciable et la classe mof-Namespace pour traduire l'hiérarchie de la MIB WBEM structurée par les espaces de nommage.

Pour générer ces corrélations, nous nous sommes appuyés sur la statégie d'identification d'une instance dans le modèle CIM basé sur la spécification des attributs clefs. Ces derniers sont déclarés par les qualifieurs `Key` du schéma *Meta*. Mais le modèle CIM dans sa version 2.0, interdit la déclaration d'attributs clefs dans une classe si celle-ci hérite d'une classe qui en déclare au moins un.

Nous avons défini les trois règles suivantes pour la génération des corrélations de noms :

1. Le formulaire de corrélation de noms ne peut être généré que pour les classes instanciables (ne comportant pas le qualifieur `Abstract` avec la valeur TRUE) ;
2. Le formulaire de corrélation de noms est généré pour toute classe instanciable définissant des attributs clefs ;
3. Le formulaire est généré pour toute classe instanciable ne définissant pas d'attributs clefs, mais sous-classe directe d'une classe abstraite (qui peut déclarer ou hériter des attributs clefs).

Pour les classes satisfaisant les règles citées ci-dessus, le formulaire de corrélation de nom génére une relation de contenance entre la classe mof-Namespace (**SUPERIOR**) et la classe GDMO (**SUBORDINATE**), étendue à toute sous-classe de cette dernière (fig. 16).

Pour la déclaration de l'attribut de nommage nous avons distingué quatre cas de figures :

- Si la classe MOF déclare un seul attribut clef, celui-ci sera pris comme attribut de nommage (ligne 7) ;
- Si la classe hérite d'un unique attribut clef, celui-ci sera pris comme attribut de nommage (ligne 8) ;
- Si la classe MOF déclare plusieurs attributs clefs, nous rajoutons un nouvel attribut à celle-ci. Il aura pour syntaxe la séquence ASN.1 (SEQUENCE) composée des types de chacun des attributs de nommage. Le label que nous donnons à cet attribut, ainsi qu'à la déclaration de la séquence ASN.1 aura la forme générique suivante : mof-<schema-name>-<class-label>-*GDMONaming*Attribute (ligne 9) ;

```
1  mof-<schema-name>-<class-label>NameBinding  NAME BINDING
2   SUBORDINATE OBJECT CLASS
3      mof-<schema-name>-<class-label>  AND SUBCLASSES ;
4   NAMED BY SUPERIOR OBJECT CLASS
5      "BasicModule":mof-Namespace ;
6   ( WITH ATTRIBUTE
7      mof-<schema-name>-<class-label>-<key-property>Attribute ;
8    | mof-<schema-name>-<super-class-label>-<key-property>Attribute ;
9    | mof-<schema-name>-<class-label>-GDMONamingAttribute ;
10   | mof-<schema-name>-<super-class-label>-GDMONamingAttribute ;
11  )
12   REGISTERED AS { autoMOFtoGDMO <schema-Ind> namebindings(6) xx }  ;
```

FIG. 16 – *Le formulaire de corrélation de noms généré pour une classe GDMO*

– Si la classe hérite d'un attribut de nommage (séquence ASN.1 des attributs clés déclarés dans une super-classe abstraite), celui-ci sera pris dans la relation de corrélation pour le nommage (ligne 10).

Le formulaire de corrélation est enregistré sous la racine réservée aux formulaires de corrélations du schéma. A chaque formulaire est attribué un numéro correspondant à une incrémentation séquentielle dans l'ordre de leur création.

6.8.2 Les corrélation de noms spécifiques aux associations "faibles"

En plus de ces corrélations génériques, et indispensables pour le nommage des instances, nous générons des corrélations de noms plus spécifiques aux classes instanciables participantes aux associations dites "faibles".

Ces classes sont identifiées par la présence d'attributs clés marqués par le qualifieur `propagated` signifiants qu'ils ont été propagés de la classe "forte" de l'association vers la classe "faible".

La corrélation de nom générée spécifie alors la classe forte comme supérieure, la faible comme subordonnée, et garde le même attribut de nommage utilisé dans la corrélation générique avec mof-Namespace.

La forme générique du formulaire est donnée dans la figure 17. Un exemple de génération de cette corrélation de noms est donnée dans la section 7.

```
1  mof-<weak-class>-<strong-class>NameBinding  NAME BINDING
2   SUBORDINATE OBJECT CLASS
3      mof-<weak-class>  AND SUBCLASSES ;
4   NAMED BY SUPERIOR OBJECT CLASS
5      mof-<strong-class>  AND SUBCLASSES ;
6   WITH ATTRIBUTE <weak-class-naming-attribute> ;
7   REGISTERED AS { autoMOFtoGDMO <schema-Ind> namebindings(6) xx } ;
```

FIG. 17 – *La traduction des associations* `weak`

6.9 Correspondances des types

Le formalisme MOF offre un certain nombre de types de données qui peuvent être associés à des attributs ou à des paramètres. Dans le cadre de la transformation d'une spécification MOF en une spécification GDMO, nous proposons une correpondance entre ces types et des types ASN.1.

6.9.1 Les types simples

La correspondance entre les types simples du langage MOF avec les types ASN.1 est donnée dans le tableau 4. La dernière colonne du tableau (Matching rules) donne les règles de correspondance associées à l'attribut GDMO défini dans le module générique *BasicModule*.

Type MOF	Référence ASN.1	Définition ASN.1	Matching rules
uint8	UInteger8	INTEGER(0..255)	EQUALITY, ORDERING
sint8	SInteger8	INTEGER(-128..127)	EQUALITY, ORDERING
uint16	UInteger16	INTEGER(0..65535)	EQUALITY, ORDERING
sint16	SInteger16	INTEGER(-32768..32767)	EQUALITY, ORDERING
uint32	UInteger32	INTEGER	EQUALITY, ORDERING
sint32	SInteger32	INTEGER	EQUALITY, ORDERING
uint64	UInteger64	INTEGER	EQUALITY, ORDERING
sint64	SInteger64	INTEGER	EQUALITY, ORDERING
real32	Real32	REAL	EQUALITY, ORDERING
real64	Real64	REAL	EQUALITY, ORDERING
boolean	Boolean	BOOLEAN	EQUALITY
char16	Char16	GraphicString	EQUALITY, ORDERING
string	String	GraphicString	EQULITY, ORDERING, SUBSTRINGS
datetime	DateTime	GeneralizedTime	EQUALITY, ORDERING

TAB. 4 – *Définitions des types simples*

6.9.2 Types structurés

Le seul type structuré utilisé par le langage MOF est le tableau. Nous traduisons le type tableau par le constructeur ASN.1 **SEQUENCE OF**. Si la taille du tableau est donnée par la spécification MOF, il en sera de même pour la séquence ASN.1 (voir exemples du tableau 5).

Type MOF	Type ASN-1
string arrayOfStr[]	<prefix>arrayOfStr<suffix>::= SEQUENCE OF GraphicString
uint64 arrayOfInt[2]	<prefix>arrayOfInt<suffix>::= SEQUENCE SIZE(2) OF UInteger64

TAB. 5 – *Exemple de traduction d'un vecteur MOF*

Les mentions `prefix` et `suffix` dans la déclaration ASN.1 suivent les régles de traduction du nommage des attributs ou des paramètres de méthodes MOF.

De plus, pour les besoins de traduction cités dans les sections 6.6 et 6.8, nous définissons de nouveaux types ASN.1 à l'aide du constructeur **SEQUENCE** (voir example 6).

Déclarations MOF	Type ASN.1

[key] string Name;
[key] uint64 Identifier;
	<prefix>GDMONamingAttribute::= SEQUENCE {
	Name GraphicString,
	Identifier UInteger64
	}

TAB. 6 – *Exemple de traduction d'un ensemble de déclarations d'attributs clef MOF en une séquence ASN.1*

Ces déclarations de types sont générées automatiquement dans l'un des deux modules ASN.1 suivants selon la nature de la déclaration du tableau MOF :

- AttributesModule : si la déclaration concerne le type d'une propriété MOF, ou l'attribut de nommage ;
- ActionsModule : si la déclaration concerne le type d'un paramètre de méthode MOF, ou la séquence de paramêtres d'une méthode.

6.10 Le formulaire de comportement

Un formulaire de comportement **BEHAVIOUR** peut être associé à tout autre formulaire GDMO généré par le traducteur. Dans notre approche cette génération est liée à la présence de qualifieurs dans la spécification des éléments MOF.

Le formulaire de comportement que nous générons possède la forme donnée par la figure 18.

```
1  <behaviour-label>  BEHAVIOUR
2   DEFINED AS
3      !BEGINPARSE
4      [ QUALIFIERS
5        <qualifier-name> ::= <qualifier-value> ;
6      [<qualifier-name> ::= <qualifier-value> ;]*
7      ]
8      [ DESCRIPTION
9      ! <qualifier-Description-value> !;
10     ]
11     ENDPARSE!;;
```

FIG. 18 – *Le formulaire de comportement*

7 Un exemple de traduction

L'exemple donné ici est tiré des spécifications MOF données dans le *Core* modèle de CIM. Nous avons cependant modifié la description des classes, par rapport à celles données par le DMTF, pour souligner les points les plus significatifs illustrants les traductions que nous proposons.

7.0.1 Les spécifications MOF

Nous donnons ici 3 classes :

- **Computer System** : classe représentant, dans un contexte de gestion, un système informatique.
- **System Device** : association permettant d'aggréger un système informatique à une représentation de l'un de ses dispositifs logiques (`Logical Device`).
- **Logical Device** : classe représentant un dispositif logique i.e une abstraction d'un périphérique ou tout autre composant physique du système.

```
      [ Abstract,  Description (
       "A class derived from System that is a special collection of "
       "ManagedSystemElements ..."),  Schema ("CIM")]
 class CIM_ComputerSystem:CIM_System
{
        [ Override ("CIM_System:SystemName"),  Key]
   string SystemName;

};

      [ Association,  Aggregation,  Description (
       "LogicalDevices may be aggregated by a System ... "),
        Schema ("CIM")]
 class CIM_SystemDevice:CIM_SystemComponent
{
      [ Override ("CIM_SystemComponent:Group") ]
    CIM_ComputerSystem REF Group;
      [ Override ("CIM_SystemComponent:Part"),
        Description ("The LogicalDevice which is a part of the System"),
        Weak]
    CIM_LogicalDevice REF Part;
};

      [ Description (
       "An abstraction or emulation of a hardware entity ..."),
        Schema ("CIM") ]
 class CIM_LogicalDevice:CIM_LogicalElement
{
      [ Key]
```

```
    string CreationClassName;
      [ Description (
       "An address or other identifying information ... "),
        Key]
    string DeviceID;
      [ Propagated ("CIM_ComputerSystem:Name") ,  Key]
    string SystemName;
};
```

7.0.2 Les spécifications GDMO issues des traductions

La génération de spécifications GDMO issues de la traduction des spécifications MOF précédantes comprend les formulaires suivants :

- trois formulaires de classe d'objet géré corrspondant chacun à une des trois classes MOF ;
- trois formulaires de modules obligatoires (PACKAGE) correspondant chacun à une des trois classes MOF ;
- six formulaires d'attributs correspondant chacun à un des attributs (y compris les références) MOF ;
- un formulaire d'attribut correspondant à l'attribut de nommage de la classe `LogicalDevice`, défini comme une séquence ASN.1 de chacune des propriétés clefs dans la spécification MOF.
- un formulaire de corrélation de noms `mof-CIM-LogicalDeviceNameBinding` correspondant à la corrélation générique avec la classe `mof-Namespace` ;
- un formulaire de corrélation de noms `mof-CIM-LogicalDevice-ComputerSystemNameBinding` correspondant à la corrélation de nom spécifique aux classes *faibles* dans les associations.
- plusieurs formulaires de comportement correspondants à la description des qualidieurs associés aux définitions MOF. Dans les spécifications qui suivent, seul le formulaire de comportement de la classe `ComputerSystem` est donné.
- une spécification ASN.1 donnant le type SEQUENCE de l'attribut de nommage de la classe `LogicalDevice`.

```
MODULE "CIM";

"CIM":mof-CIM-ComputerSystem MANAGED OBJECT CLASS
   DERIVED FROM "BasicModule":mof-CIM-System;
   CHARACTERIZED BY "CIM":mof-CIM-ComputerSystemPackage;
 REGISTERED AS { autoMOFtoGDMO 1 classes(3) 1 };

"CIM":mof-CIM-SystemDevice MANAGED OBJECT CLASS
   DERIVED FROM "CIM":mof-CIM-SystemComponent;
   CHARACTERIZED BY "CIM":mof-CIM-SystemDevicePackage;
 REGISTERED AS { autoMOFtoGDMO 1 classes(3) 2 };

"CIM":mof-CIM-LogicalDevice MANAGED OBJECT CLASS
   DERIVED FROM "CIM":mof-CIM-LogicalElement;
   CHARACTERIZED BY "CIM":mof-CIM-LogicalDevicePackage;
 REGISTERED AS { autoMOFtoGDMO 1 classes(3) 3 };

"CIM":mof-CIM-ComputerSystemPackage PACKAGE
   BEHAVIOUR
      "CIM":mof-CIM-ComputerSystemBehaviourPackage;
   ATTRIBUTES
      "CIM":mof-CIM-ComputerSystem-SystemNameAttribute GET,
 REGISTERED AS { autoMOFtoGDMO 1 packages(4) 1 };

"CIM":mof-CIM-SystemDevicePackage PACKAGE
   BEHAVIOUR
      "CIM":mof-CIM-ComputerSystemBehaviourPackage;
   ATTRIBUTES
      "CIM":mof-CIM-SystemDevice-GroupAttribute GET-REPLACE;
      "CIM":mof-CIM-SystemDevice-PartAttribute GET-REPLACE;
 REGISTERED AS { autoMOFtoGDMO 1 packages(4) 2 };
```

```
"CIM":mof-CIM-LogicalDevicePackage PACKAGE
   BEHAVIOUR
      "CIM":mof-CIM-ComputerSystemBehaviourPackage;
   ATTRIBUTES
      "CIM":mof-CIM-LogicalDevice-CreationClassNameAttribute GET,
      "CIM":mof-CIM-LogicalDevice-DeviceIDAttribute GET,
      "CIM":mof-CIM-LogicalDevice-SystemNameAttribute GET,
      "CIM":mof-CIM-LogicalDevice-GDMONamingAttribute GET;
 REGISTERED AS { autoMOFtoGDMO 1 packages(4) 3 };

"CIM":mof-CIM-ComputerSystem-SystemNameAttribute ATTRIBUTE
   DERIVED FROM "BasicModule":GraphicStringAttribute;
   BEHAVIOUR "CIM":mof-CIM-ComputerSystem-SystemNameAttributeBehaviourPackage;
 REGISTERED AS { autoMOFtoGDMO 1 attributes(7) 1 };

"CIM":mof-CIM-SystemDevice-GroupAttribute ATTRIBUTE
   WITH ATTRIBUTE SYNTAX ASNBasicModule.Reference;
   BEHAVIOUR "CIM":mof-CIM-SystemDevice-GroupAttributeBehaviourPackage;
 REGISTERED AS { autoMOFtoGDMO 1 attributes(7) 2 };

"CIM":mof-CIM-SystemDevice-PartAttribute ATTRIBUTE
   WITH ATTRIBUTE SYNTAX ASNBasicModule.Reference;
   BEHAVIOUR "CIM":mof-CIM-SystemDevice-PartAttributeBehaviourPackage;
 REGISTERED AS { autoMOFtoGDMO 1 attributes(7) 3 };

"CIM":mof-CIM-LogicalDevice-ClassCreationNameAttribute ATTRIBUTE
   DERIVED FROM "BasicModule":GrapgicStringAttribute;
   BEHAVIOUR "CIM":mof-CIM-LogicalDevice-ClassCreationNameAttributeBehaviourPackage;
 REGISTERED AS { autoMOFtoGDMO 1 attributes(7) 4 };

"CIM":mof-CIM-LogicalDevice-DeviceIDAttribute ATTRIBUTE
   DERIVED FROM "BasicModule":GrapgicStringAttribute;
   BEHAVIOUR "CIM":mof-CIM-LogicalDevice-DeviceIDAttributeBehaviourPackage;
 REGISTERED AS { autoMOFtoGDMO 1 attributes(7) 5 };

"CIM":mof-CIM-LogicalDevice-SystemNameAttribute ATTRIBUTE
   DERIVED FROM "BasicModule":GrapgicStringAttribute;
   BEHAVIOUR "CIM":mof-CIM-LogicalDevice-SystemNameAttributeBehaviourPackage;
 REGISTERED AS { autoMOFtoGDMO 1 attributes(7) 6 };

"CIM":mof-CIM-LogicalDevice-GDMONamingAttribute ATTRIBUTE
   WITH ATTRIBUTE SYNTAX AttributesModule.Mof-CIM-LogicalDevice-NamingSequence;
   BEHAVIOUR "CIM":mof-CIM-LogicalDevice-GDMONamingAttributeBehaviour;
 REGISTERED AS { autoMOFtoGDMO 1 asn(2) 1 };

"CIM":mof-CIM-LogicalDeviceNameBinding Name Binding
   SUBORDINATE OBJECT CLASS
       "CIM":-mof-CIM-LogicalDevice AND SUBCLASSES;
   NAMED BY SUPERIOR OBJECT CLASS
       "BasicModule":mof-Namespace;
   WITH ATTRIBUTE
       "CIM":mof-CIM-LogicalDevice-GDMONamingAttribute;
 REGISTERED AS { autoMOFtoGDMO 1 name-binding(6) 2 }

"CIM":mof-CIM-LogicalDevice-ComputerSystemNameBinding Name Binding
   SUBORDINATE OBJECT CLASS
       "CIM":mof-CIM-LogicalDevice AND SUBCLASSES;
   NAMED BY SUPERIOR OBJECT CLASS
       "CIM":mof-CIM-ComputerSystem AND SUBCLASSES;
   WITH ATTRIBUTE
       "CIM":mof-CIM-LogicalDevice-GDMONamingAttribute;
```

```
 REGISTERED AS { autoMOFtoGDMO 1 name-binding(6) 2 }

"CIM":mof-CIM-ComputerSystemBehaviourPackage  BEHAVIOUR
   DEFINED AS
   !BEGINPARSE
  QUALIFIERS
    Abstract = TRUE ;
    Schema = "CIM" ;
  DESCRIPTION
    !! A class derived from System that is a special collection of
       ManagedSystemElements ... !!;
   ENDPARSE! ;

 .....

-- generated ASN.1 module for SEQUENCE / SEQUENCE OF types

AttributesModule
{ autoMOFtoGDMO 1 asn(2) 1 }
 DEFINITIONS ::= BEGIN

Mof-CIM-LogicalDevice-GDMONamingSequence ::= SEQUENCE {
       mof-CIM-LogicalDevice-CreationClassName  GraphicString ,
       mof-CIM-LogicalDevice-DeviceID           GraphicString ,
       mof-CIM-LogicalDevice-SystemName         GraphicString
}

 END
```

8 Réalisation

L'ensemble des traductions définies dans les sections précédentes sont implémentées dans l'environnement MODERES Java [Festor 96, Festor 97a, Festor 97b]. Au sein de cet environnement nous fournissons actuellement un analyseur de spécifications au format MOF [Festor 98], ainsi qu'un analyseur sémantique des spécifications MOF. Au dessus de ces deux analyseurs nous générons l'arbre abstrait contenant les spécifications GDMO.

Dans la version actuelle de l'environnement, les analyseurs syntaxique et sémantique MOF, et le traducteur MOFtoGDMO supportent la version CIM 2.0.

Le traducteur MOFtoGDMO est inclu dans l'interface graphique de l'environnement MODERES (figure 19). Il permet de traduire les spécifications MOF préalablement chargées dans l'environnement, et de générer les trois fichiers suivants :

- un fichier des spécifications GDMO issues des traductions ;
- un fichier des défintions ASN.1 des types SEQUENCE et SEQUENCE OF définies par le tarducteur ;
- un fichier de correspondances entre les noms MOF et les noms/OID GDMO correspondants : Ce fichier présente des facilités pour la réalisation d'un agent intégrateur OSI/WBEM. Il donne une table de coorespondances des noms MOF et GDMO qui simplifie considérablement l'intégration entre les services de gestion CMIS et HMMP.

  Chaque ligne du fichier contient le nom de l'élément MOF suivi du nom de l'élément GDMO qui lui correspond, lui-même suivi de son OID (Object IDentifier). Dans le cas d'un attribut la ligne est complétée par l'identificateur de type défini dans l'interface CMISoverJava.

  Pour illustrer la forme du contenu de ce fichier, nous en donnons 2 lignes issues des traductions des spécifications de l'exemple présenté dans la section 7 :

```
CIM_SystemDevice > mof-CIM-SystemDevice = 0.0.1.3.5

CIM_ComputerSystem:SystemName > mof-CIM-ComputerSystem-SystemNameAttribute = 0.0.1.7.1
                                (VALUE_GRAPHIC_STRING)
```

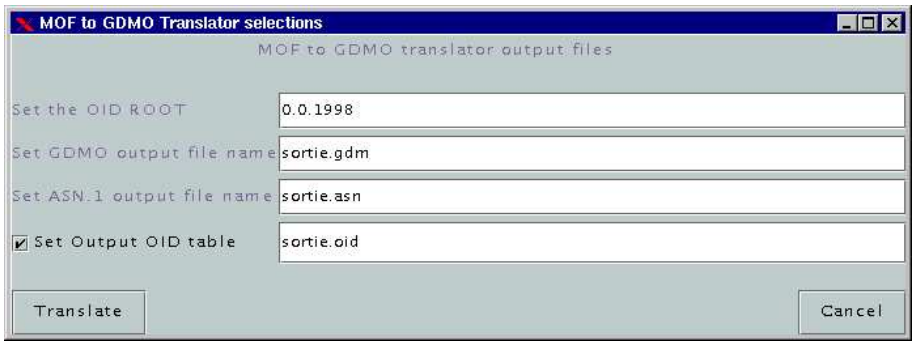

FIG. 19 – *Utilisation du traducteur dans l'interface fraphique de MODERES*

Pour exécuter le traducteur MOFtoGDMO sur un fichier de spécifications MOF, deux possibilités sont offertes par MODERES :

- une exécution "commande en ligne" qui lance automatiquement l'analyseur syntaxique, l'analyseur sémantique avant de lancer le traducteur ;
- une exécution "graphique" qui requiert une exécution des trois applications séparémment. L'exécution de l'analyseur sémantique avant le traducteur est recommandée pour une meilleure considération des caractéristiques du modèle CIM et notamment pour la distinction entre classes et associations qui n'apparaît pas à l'issu d'une passe d'analyse syntaxique.

9 Extensions futures des algorithmes de traduction

9.1 Meilleure considération des qualifieurs

Dans la version initiale de notre algorithme de correspondance, seul les qualifieurs d'attributs décrivant les droits d'accès à ces derniers (lecture, écriture) ainsi que les qualifieurs Abstract, weak et propagated sont pris en compte dans la traduction. Les autres sont insérés dans la spécification GDMO sous forme de commentaires dans le comportement des composants.

De ce fait ces informations ne sont plus accessibles ni exploitables automatiquement dans le gestionnaire OSI ou l'agent de proximité interfacant les deux mondes. Il serait intéressant de pouvoir intégrer les qualifieurs de manière plus complète dans les spécifications GDMO. Une étude sur ce point est en cours dans notre groupe de recherche. Différentes solutions sont possibles mais aucune ne se dégage clairement à ce jour.

9.2 Traduction des associations vers des spécifications GRM

Les spécifications de relations entre objet gérés issues des spécifications MOF sont actuellement traitées comme n'importe quel autre description de classe. Cette information peut être précieuse au gestionnaire OSI sous forme de spécification GRM [CCITT.X.725 95]. Sur ce point également une étude est en cours et des propositions de traduction devraient être prochainement faites.

9.3 Traduction des notifications et déclencheurs de notifications

Une étude est actuellement en cours sur l'intégration des notifications dans le traducteur. Ceci n'est pas abouti pour le moment.

10 Conclusion et travaux futurs

Dans ce rapport, nous avons présenté les résultats de nos travaux sur l'intégration WBEM/OSI. Ces travaux comprennent la définiton d'une architecture d'un agent d'intégration, des algorithmes de traduction automatique de spécifications MOF en spécifications GDMO/ASN.1, des règles de transcription de services et un agent intégrateur pilote.

L'ensembles des algorithmes sont implantés dans l'environnement MODERES et quelques extensions ont été ajoutées à la demande de BULL (génération de tables de correspondances principalement) dans le cadre de l'action ANTARES.

Si la plupart des fonctionnalités et correspondances sont aujourd'hui fixées, certains points peuvent faire l'objet d'évolutions dans les versions futures de l'environnement MODERES. Nous travaillons notamment actuellement à une meilleure prise en compte des qualifieurs du langage MOF dans les algorithmes de traduction. Un support pour l'ensemble des qualifieurs standards est envisagé.

Remerciements

Ce travail a été partiellement réalisé dans le cadre de l'action ANTARES, action de recherche et développement entre le LORIA et le groupe BULL sous le patronage du GIE DYADE. Si les outils MOF et les algorithmes de base existaient au sein du projet RESEDAS, cette coopération nous a permis de réaliser un agent d'intégration sur l'interface CMISoverJava et d'affiner en coopération avec nos partenaires certains points critiques des algorithmes. Aussi tenons particulièrement à remercier, Didier Zhang et Laurent Andrey pour leur contribution faramineuse sur l'interface CMISoverJava, Bruno Farcy pour sa contribution à l'affinage du traitement des dépendances dans nos algorithmes de traduction, Serge Lassabe et Alain Brunet pour leurs commentaires judicieux sur nos développements.

Références

[Abeck 93] S. Abeck, A. Clemm et U. Hollberg. *Simply Open Network Management: An Approach for the Integration of SNMP into OSI Management Concepts.* pages 361–375, 1993. in [ISINM'93 93].

[Case 93] J. Case, K. McCloghrie, M. Rose et S. Waldbusser, *Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)*, RFC1442, SNMP Research Inc., Hughes LAN Systems Inc., Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.

[CCITT.M.3010 92a] Comité Consultatif International Télégraphique et Téléphonique (CCITT), *Principles for a Telecommunications management network*, International Standard, CCITT.M.3010, Janvier 1992.

[CCITT.M.3010 92b] Comité Consultatif International Télégraphique et Téléphonique (CCITT), *Maintenance: Réseau de Gestion des Télécommunications: Principes pour un Réseau de Gestion des Télécommunications*, International Standard, CCITT.M.3010, Janvier 1992.

[CCITT.X.725 95] Comité Consultatif International Télégraphique et Téléphonique (CCITT), *Information Technology - Open Systems Interconnection - Structure of Management Information - Part 7: General Relationship Model*, International Standard, CCITT.X.725, November 1995, [ISO-10165.7 97].

[Chang 96] April (Editor) Chang. Forum028 - ISO/CCITT to Internet Managemet Proxy, Issue 1.0, October 1996.

[Clemm 97] A. Clemm. *SNMP and TL-1: Simply integrating management of legacy systems.* pages 192–203, 1997. in [IM'97 97].

[DMTF 98] DMTF. Common Information Model (CIM) version 2.0. Rapport, Desktop Management Task Force, November 1998.

[(Editor) 96a] LaBarre Lee (Editor). Forum026 - Translation of Internet MIBs to ISO/CCITT GDMO MIBs, Issue 1.0, October 1996.

[(Editor) 96b] LaBarre Lee (Editor). Forum029 - Translation of Internet MIB-II to ISO/CCITT GDMO MIBs, Issue 1.0, October 1996.

[Festor 96] O. Festor, E. Nataf et L. Andrey. MODE-FE: A GRM/GDMO Parser and its API -Release 1.0-
Reference Manual. Technical Report no. 0190, INRIA Lorraine, 1996.

[Festor 97a] O. Festor. The gdmo and grm modules semantic checker of the moderes java toolkit. Rapport no. RT-0208, INRIA, July 1997.

[Festor 97b] O. Festor. MODERES Java: Architecture and Core Packages. Rapport no. RT-0205, INRIA, May 1997.

[Festor 98] O. Festor. The Managed Object Format specification parser of the MODERES Java Toolkit. Rapport technique no. RT-???, INRIA, Février 1998.

[IM'97 97] IM'97. *Integrated Network Management, V.* Chapman & Hall, 1997. Proc. IFIP IEEE 5th Int. Symp. on Integrated Management, San Diego, CA, 12-17 May, 1997.

[ISINM'93 93] ISINM'93. *Integrated Network Management, III (C-12).* Elsiever Science Publishers B.V. (North-Holland), 1993. Proc. IFIP IEEE 3rd Int. Symp. on Integrated Network Management, San Francisco, CA, 18-23 April, 1993.

[ISINM'95 95] ISINM'95. *Integrated Network Management, IV.* Chapman & Hall, 1995. Proc. IFIP IEEE 4th Int. Symp. on Integrated Network Management, Santa Barbara, CA, 1-5 May, 1995.

[ISO-10164.16 97] International Organization for Standardization (ISO), *System Management - Part 16: Management knowledge management function* , International Standard, ISO-10164.16, 1997.

[ISO-10165.4 92] International Organization for Standardization (ISO), *Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects*, International Standard, ISO-10165.4, January 1992.

[ISO-10165.7 97] International Organization for Standardization (ISO), *Structure of Management Information - Part 7: General relationship model* , International Standard, ISO-10165.7, 1997, [CCITT.X.725 95].

[ISO-8824 90] International Organization for Standardization (ISO), *Specification of Abstract Syntax Notation Number One (ASN.1)*, International Standard, ISO-8824, December 1990.

[ISO-9595 91] International Organization for Standardization (ISO), *Common Management Information Service Definition*, International Standard, ISO-9595, June 1991.

[ISO-9596 91] International Organization for Standardization (ISO), *Common Management Information Protocol Specification*, International Standard, ISO-9596, June 1991.

[Jander 96] M. Jander. Web-based Management: Weclome to the revolution. *Data Communications International*, 25(16):38–56, 1996.

[Kalyanasundaram 93] P. Kalyanasundaram et A.S. Sethi. *An Application Gateway Design for OSI-Internet Management.* pages 389–401, 1993. in [ISINM'93 93].

[Koerner 97] E. Koerner. Design of a proxy for managing CMIP agents via SNMP. *Computer Communications*, 20(5):349–360, 1997.

[Mazumbar 98] S. Mazumbar. *Inter-Domain Management: CORBA, OSI, SNMP. Tutorial NOMS'98*, New Orleans, 1998.

[Mazumdar 93] S. Mazumdar, S. Bradly et D.W. Levine. *Design of Protocol Independent Management Agent to Support SNMP and CMIP Queries.* pages 377–388, 1993. in [ISINM'93 93].

[McCarthy 95] K. McCarthy, G. Pavlou, S. Bhatti et J. Neuman De Souza. *Exploiting the power of OSI Management for the control of SNMP-capable resources using generic application level gateways.* pages 440–453, 1995. in [ISINM'95 95].

[Rivière 98] A.I. Rivière et M. Sibilla. Management Information Models Integration : From Existing Approaches to New Unifying Guidelines. *Journal of Network and Systems Management*, 6(1):333–356, November 1998.

[Soukouti 97] N. Soukouti et U. Hollberg. *Joint Inter Domain Management: CORBA, CMIP and SNMP.* pages 153–164, 1997. in [IM'97 97].

[Todd 97] S. Todd, *HMMP Overview*, Not yet assigned, Microsoft Corporation, One Microsoft Way, Redmond, WA 98053, USA, May 1997.

[UIT-T.M.3000 92] Union Internationale des Télécommunications, *Maintenance: Réseau de Gestion des Télécommunications: Vue d'Ensemble des Recommandantions Relatives au Réseau de Gestion des Télécommunication*, International Standard, UIT-T.M.3000, Janvier 1992.

[Vertel 97] Vertel. Accessing TMN Through Web-based Entreprise Management. Rapport, Vertel, February 1997. White Paper.

A Le module GDMO BasicModule

```
MODULE “BasicModule”;

"BasicModule":mof-MOC MANAGED OBJECT CLASS
DERIVED FROM  “Rec. X.721 | ISO/IEC 10165-2 : 1992":top;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) classes(3) 0 };

"BasicModule":mof-Namespace MANAGED OBJECT CLASS
        DERIVED FROM "BasicModule":mof-MOC;
        CHARACTERIZED BY "BasicModule":mof-NamespacePackage;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) classes(3) 1 };

"BasicModule":mof-MetaMOC MANAGED OBJECT CLASS
        DERIVED FROM "BasicModule":mof-MOC;
        CHARACTERIZED BY "BasicModule":mof-MetaMOCPackage;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) classes(3) 2 };

"BasicModule":mof-QualifierDef MANAGED OBJECT CLASS
        DERIVED FROM "BasicModule":mof-MOC;
        CHARACTERIZED BY "BasicModule":mof-QualifierDefPackage;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) classes(3) 3 };

"BasicModule":cim-Agent MANAGED OBJECT CLASS
        DERIVED FROM "BasicModule":mof-MOC;
        CHARACTERIZED BY "BasicModule":cim-AgentPackage;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) classes(3) 4 };

"BasicModule":mof-NamespacePackage PACKAGE
        ATTRIBUTES
                "BasicModule":mof-NamespaceNameAttribute GET;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) packages(4) 1 };

"BasicModule":mof-MetaMOCPackage PACKAGE
        ATTRIBUTES
                "BasicModule":mof-MetaClassNameAttribute GET,
                "BasicModule":mof-MetaSuperClassNameAttribute GET-REPLACE,
                "BasicModule":mof-MetaPropertyBufferListAttribute GET-REPLACE,
                "BasicModule":mof-MetaMethodBufferListAttribute GET-REPLACE,
                "BasicModule":mof-MetaQualifierListAttribute GET-REPLACE;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) packages(4) 2 };

"BasicModule":mof-QualifierDefPackage PACKAGE
        ATTRIBUTES
                "BasicModule":mof-QualifierDefNameAttribute GET,
                "BasicModule":mof-QualifierDefTypeAttribute GET-REPLACE,
                "BasicModule":mof-QualifierDefScopeListAttribute GET-REPLACE,
                "BasicModule":mof-QualifierDefFlavorListAttribute GET-REPLACE;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) packages(4) 3 };

"BasicModule":cim-AgentPackage PACKAGE
        ATTRIBUTES
                "BasicModule":cim-AgentIPAdressAttribute GET,
                "BasicModule":cim-AgentStateAttribute GET-REPLACE;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) packages(4) 4 };

"BasicModule":mof-NamespaceNameAttribute ATTRIBUTE
        DERIVED FROM "BasicModule":graphicStringAttribute;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 111 };

"BasicModule":mof-MetaClassNameAttribute ATTRIBUTE
        DERIVED FROM "BasicModule":graphicStringAttribute;
```

```
REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 121 };

"BasicModule":mof-MetaSuperClassNameAttribute ATTRIBUTE
        DERIVED FROM "BasicModule":graphicStringAttribute;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 122 };

"BasicModule":mof-MetaPropertyBufferListAttribute ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASNBasicModule.PropertyBufferList;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 123 };

"BasicModule":mof-MetaMethodBufferListAttribute ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASNBasicModule.MethodBufferList;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 124 };

"BasicModule":mof-MetaQualifierListAttribute ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASNBasicModule.QualifierList;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 125 };

"BasicModule":mof-QualifierDefNameAttribute ATTRIBUTE
        DERIVED FROM "BasicModule":graphicStringAttribute;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 131 };

"BasicModule":mof-QualifierDefTypeAttribute ATTRIBUTE
        DERIVED FROM "BasicModule":graphicStringAttribute;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 132 };

"BasicModule":mof-QualifierDefScopeListAttribute ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASNBasicModule.ScopeList;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 133 };

"BasicModule":mof-QualifierDefFlavorListAttribute ATTRIBUTE
        WITH ATTRIBUTE SYNTAX ASNBasicModule.FlavorList;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 134 };

"BasicModule":cim-AgentIPAdressAttribute ATTRIBUTE
        DERIVED FROM "BasicModule":graphicStringAttribute;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 141 };

"BasicModule":cim-AgentStateAttribute ATTRIBUTE
        DERIVED FROM "BasicModule":booleanAttribute;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) attributes(7) 142 };

"BasicModule":mof-NS-NS-NameBinding NAME BINDING
        SUBORDINATE OBJECT CLASS "BasicModule":mof-Namespace ;
        NAMED BY SUPERIOR OBJECT CLASS "BasicModule":mof-Namespace ;
        WITH ATTRIBUTE "BasicModule":mof-NameSpaceNameAttribute ;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) namebindings(6) 1 };

"BasicModule":mof-NS-Meta-NameBinding NAME BINDING
        SUBORDINATE OBJECT CLASS "BasicModule":mof-MetaMOC ;
        NAMED BY SUPERIOR OBJECT CLASS "BasicModule":mof-Namespace ;
        WITH ATTRIBUTE "BasicModule":mof-MetaClassNameAttribute ;
REGISTERED AS { autoMOFtoGDMO BasicModule(0) namebindings(6) 2 };

-----------------------------------------------------------------
-- Basic attributes used to define ASN.1 types
-----------------------------------------------------------------

"BasicModule":uInteger8Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.UInteger8 ;
    MATCHES FOR  EQUALITY, ORDERING ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 1 };
```

```
"BasicModule":sInteger8Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.SInteger8 ;
    MATCHES FOR  EQUALITY, ORDERING ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 2 };

"BasicModule":uInteger16Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.UInteger16 ;
    MATCHES FOR  EQUALITY, ORDERING ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 3 };

"BasicModule":sInteger16Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.SInteger16 ;
    MATCHES FOR  EQUALITY, ORDERING ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 4 };

"BasicModule":uInteger32Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.UInteger32 ;
    MATCHES FOR  EQUALITY, ORDERING ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 5 };

"BasicModule":sInteger32Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.SInteger32 ;
    MATCHES FOR  EQUALITY, ORDERING ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 6 };

"BasicModule":uInteger64Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.UInteger64 ;
    MATCHES FOR  EQUALITY, ORDERING ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 7 };

"BasicModule":sInteger64Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.SInteger64 ;
    MATCHES FOR  EQUALITY, ORDERING ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 8 };

"BasicModule":real32Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.Real32 ;
    MATCHES FOR  EQUALITY, ORDERING ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 9 };

"BasicModule":real64Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.Real64 ;
    MATCHES FOR  EQUALITY, ORDERING ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 10 };

"BasicModule":char16Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.GraphicString ;
    MATCHES FOR  EQUALITY, ORDERING, SUBSTRINGS ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 11 };

"BasicModule":graphicStringAttribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.GraphicString ;
    MATCHES FOR  EQUALITY, ORDERING, SUBSTRINGS;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 12 };

"BasicModule":booleanAttribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX ASNBasicModule.Boolean ;
    MATCHES FOR  EQUALITY;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 13 };

"BasicModule":dateTimeAttribute ATTRIBUTE
```

```
    WITH ATTRIBUTE SYNTAX ASNBasicModule.DateTime ;
    MATCHES FOR  EQUALITY, ORDERING ;
REGISTERED AS { autoMOFtoGDMO BaiscModule(0) attributes(7) 14 };
```

B Le module ASN.1 ASNBasicModule

```
--
-- ASN.1 Basic Module
--

ASNBasicModule
{ autoMOFtoGDMO BasicModule(0) asn(2) asnBasicModule(0) }
DEFINITIONS ::=BEGIN

--
-- Basic types directly mapped from MOF types
--

UInteger8 ::= INTEGER(0..255)

SInteger8 ::= INTEGER(-128..127)

UInteger16 ::= INTEGER(0..65535)

SInteger16 ::= INTEGER(-32768..32767)

UInteger32 ::= INTEGER

SInteger32 ::= INTEGER

UInteger64 ::= INTEGER

SInteger64 ::= INTEGER

Real32 ::= Real

Real64 ::= Real

Char16 ::= GraphicString

GraphicString ::= GraphicString

DateTime ::= GeneralizedTime

Reference ::= ObjectInstance

--
-- Types used for the GDMO basic definitions
--

-- Defintions for MOF_MetaMOC attributes

PropertyBufferList ::= SEQUENCE OF PropertyBuffer
PropertyBuffer ::= SEQUENCE {
    name GraphicString,
    dataType GraphicString,
    defaultValue ANY,
    qualifierList QualifierList
}

MethodBufferList ::= SEQUENCE OF MethodBuffer
MethodBuffer ::= SEQUENCE {
    name GraphicString,
    returnType GraphicString,
    parameterList ParameterList
}
```

```
ParameterList ::= SEQUENCE OF Parameter
Parameter ::= SEQUENCE {
    Name GraphicString,
    DataType GraphicString
}

QualifierList ::= SEQUENCE OF Qualifier
Qualifier ::= SEQUENCE {
    Name GraphicString,
    Value SEQUENCE OF QualifierValue,
    Flavor SEQUENCE OF GraphicString
}

QualifierValue ::= SEQUENCE {
    TypeId INTEGER,
    Value ANY DEFINED BY TypeId
}

-- Defintions for MOF_QualifierDef attributes

ScopeList ::= SEQUENCE OF GraphicString

FlavorList ::= SEQUENCE OF GraphicString

END
```

Unit´e de recherche INRIA Lorraine, Technopˆole de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhˆone-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399