Intégration de la gestion WBEM au sein du modèle OSI

Nizar Ben Youssef

UFR STMIA Département Informatique

Intégration de la gestion WBEM au sein du modèle OSI

MÉMOIRE

soutenu le 7 septembre 1998

pour l'obtention du

DEA de l'Université Henri Poincaré – Nancy I

(Spécialité Informatique)

par

Nizar Ben Youssef

Composition du jury

Jean-Paul Haton
Michaeël Rusinowitch
Didier Galmiche
Olivier Festor
Laurent Andrey
Hala SKAF

Laboratoire Lorrain de Recherche en Informatique et ses Applications

Mis en page avec la classe TheseCRIN.

Remerciements

De nombreuses personnes ont contribué au bon déroulement de ce stage, et m'ont aidé tout au long de ces trois derniers mois. Très touché par leurs encouragements, je tiens à les remercier tous.

Je tiens tout d'abord à remercier André Scaff, professeur à l'université Henri Poincarré de m'avoir acceuilli au sein de l'équipe RESEDAS du LORIA.

Je remercie Olivier Festor, Chargé de Recherche à l'Institut National de Recherche en Informatique et en Automatique, d'avoir dirigé mon étude.

Je remercie très sincérement les membres de l'équipe RESEDAS, qui ont rendu ce stage plus agréable en m'offrant leur amitié.

Table des matières

Table des figures

Chapitre 1

Introduction générale

1.1 L'intégration dans la gestion de réseaux

La croissance de l'utilisation de réseaux de plus en plus complexes entraîne de nouveaux besoins spécifiques envers leur administration. Ainsi, la gestion des réseaux et des services a été reconnue comme l'un des domaines vitaux des télécommunications et des réseaux d'entreprises.

Les ressources d'un réseau sont fréquemment issues de constructeurs différents avec des caractéristiques différentes, ce qui a conduit à l'apparition de plusieurs approches de gestion les plus connues étant la gestion OSI[1] pour les réseaux d'opérateurs et la gestion SNMP[2] pour les réseaux locaux.

Le plus grand souci des industriels du domaine (fournisseur de plates-formes, fournisseurs d'équipements réseaux,...) face à cette hétérogénéité tant au point de vue des composants des réseaux, que de leurs fonctionnalités de gestion, fut alors de développer des systèmes facilitant l'intégration de toutes ces approches au vu d'une gestion globale.

Dans ce domaine, un travail de recherche fondamentale et appliquée reste à développer. La recherche fondamentale devra aboutir à extraire des approches existantes, les concepts de base de la gestion de réseaux et en définir les règles de conception et de développement. La recherche appliquée devra pousuivre d'une part les travaux sur l'intégration des approches de gestion actuellement existantes et montrer la faisabilité de l'intégration.

1.2 Le cadre du travail

Les travaux entrepris pendant ce stage ont été réalisés dans le cadre d'une coopération entre BULL et l'équipe RESEDAS du LORIA: le projet ANTARES (Architectures et Nouvelles Technologies pour l'Administration de Réseaux et de Services).

RESEDAS développe des activités de recherche autour des réseaux, de leur gestion, des applications distriburées et services de télécommunication. Les efforts se portent principalement sur l'aspect ingénierie de ce domaine d'application. Cela comporte les différentes phases du développement d'un système de gestion incluant des approches de gestion multiples (SNMP, OSI, TL-1).

Les travaux récents du groupe portent notamment sur la validation des systèmes de gestion et principalement sur la validation et l'intégration en gestion. Dans ce cadre, le groupe a conçu

1. Open Systems Interconnection
2. Simple Network Management Protocol

et réalisé différents environnements logiciels facilitant le développement et la validation de tels systèmes.

WBEM pour Web-Based Enterprise Mangement, nouvelle initiative de gestion lancée en 1996, par un ensemble d'entreprises dont Microsoft Corp. Cisco Systems Inc., est rapidement passée dans le giron des standards incontournables en gestion de systèmes applicatifs.

L'objectif principal de ce stage est de proposer un modèle d'intégration de cette approche au sein du modèle OSI, c'est à dire de permettre l'inter-connexion des deux modèles de gestion tout en donnant à l'administrateur une vue homogène sur le nouveau système.

1.3 L'Organisation du rapport

Ce mémoire est composé de quatre chapitres correspondant chacun aux différentes phases du stage. Il est organisé de la manière suivante:

Le chapitre 2 donne un apperçu général sur la gestion des réseaux ainsi qu'une présentation des approches qui font l'objet de cette étude i.e. OSI et WBEM.

Le chapitre 3 comporte un parcours de l'état de l'art dans le domaine de l'intégration dans la gestion des réseaux. Il comporte alors une présentation des concepts généraux ainsi que des descriptions de certaines approches d'intégration.

Le chapitre 4 présente l'approche d'intégration que nous proposons et que nous avons implantée pendant ce stage.

Le chapitre 5 discute la validation du système de gestion WBEM par le langage SDL[3], et par la même occasion la validation de l'intégration que nous avons réalisée.

Le dernier chapitre rappelle brièvement les conclusions de l'étude et présente les différentes perspectives de ce travail.

3. Specifiaction and Description Langage

Chapitre 2

Présentation des approches de gestion OSI et WBEM

2.1 Introduction

La première étape de notre étude consistait à comparer les deux approches OSI et WBEM qui font l'objet de notre travail. Ainsi, nous présentons dans ce chapitre les concepts principaux de chacune d'entre elles tout en soulignant les points fondamentaux qui les diffèrencient.

Dans un premier temps nous donnons un apperçu général sur le domaine de la gestion des réseaux, puis nous présenterons les techniques spécifiques utilisées dans l'approche OSI, puis celles récemment proposés dans l'approche WBEM.

2.2 Concepts généraux de la gestion des réseaux

La gestion des réseaux regroupe toutes les activités nécessaires à la supervision et au contrôle des réseaux via notamment le suivi du bon fonctionnement de chacune des ressources logiques ou physiques sur les réseaux.

Pour satisfaire ce besoin, trois entités principales ont été définies par la communauté de la gestion des réseaux (figure 2.1) :

- La base de l'information de gestion (MIB[4]): dans un système de gestion, chacune des ressources du réseau est représentée par un (ou plusieurs) objet(s) géré(s). La base de gestion se définit comme un ensemble d'objets gérés ;
- L'agent: attaché à un ensemble de ressources, il se charge de collecter les informations utiles pour la gestion, afin de maintenir la base de gestion correspondant à ces ressources ;
- Le gestionaire: c'est l'application qui permet à un opérateur d'interroger ou de commander les ressources, par l'intermédiaire d'échanges d'informations de gestion (objets gérés) avec les différents agents.

Afin de décrire l'activité de gestion partagée par ces trois entités, la communauté a normalisé quatre modèles de conceptions, complémentaires, formant ainsi la base de toute approche de gestion.

- l'architecture fonctionnelle : elle précise les rôles pris par chaque entité de gestion ;

4. Management Information Base

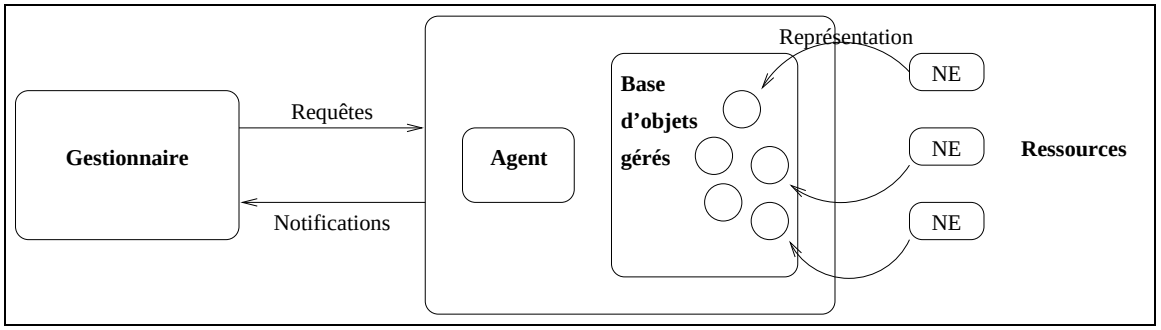

FIG. 2.1 – *Architecture générale d'un système de gestion de réseaux*

- le modèle fonctionnel : il définit un ensemble d'activités de base nécessaires à la gestion;
- le modèle de l'information : il spécifie formellement tous les objets gérés ;
- le modèle de communication : il présente l'ensemble des services et protocoles permettant l'échange de l'information de gestion.

Notre étude a porté essentiellement sur les modèles de l'information et de communication. Ainsi, la présentation des approches de gestion OSI et WBEM qui suit, ne portera que sur ces deux points.

2.3 Présentation de l'approche OSI

L'approche de gestion OSI a été retenue par l'ISO [5] pour subvenir, essentiellement, aux besoins de la gestion des réseaux de télécomunication.

Les composantes de l'approche OSI que nous présentons sommairement dans cette section sont:

- Le modèle de l'information: il est basé sur une approche orientée objet. Il est spécifié formellement par le langage GDMO [6] normalisé par le CCITT [9], ainsi que par un sous ensemble de types ASN-1 [7] [17].
- Le modèle de communication: les échanges d'information de gestion au sein du modèle OSI sont réalisés par des appels de services CMIS [8] [7].

2.3.1 Le langage GDMO

Les formulaires GDMO

Le langage GDMO offre neuf formulaires, correspondants aux élements du méta-modèle, pour spécifier les objets gérés d'une manière modulaire. Dans cette section nous ne détaillerons que deux de ces formulaires, les autres formulaires GDMO sont donnés en annexe A.

- Le formulaire de **classe d'objet géré** : il permet de spécifier une classe d'objet en définissant les modules qui les composent et en les liant aux super-classes dont elles héritent ;

5. International Standarization Organization
6. Guidelines for the Defintion of Managed Objects
7. Abstract Syntax Notation number One
8. Common Management Information Service

- Le formulaire de **module** : il est utilisé pour définir les modules en spécifiant pour chacun d'eux les attributs (et les opérations autorisées), actions, notifications et le comportement qui le composent ;
- Le formulaire d'**attribut** : il permet de définir des attributs types ainsi que les opérations qui lui peuvent lui être appliquées ;
- Le formulaire de **groupe d'attributs** : il permet de regrouper sous une étiquette un certain nombre d'attributs. Un groupe peut être par la suite référencé dans un module ;
- Le formulaire d'**action** : il est utilisé pour définir des actions autorisées sur des objets gérés ;
- Le formulaire de **notification** : il permet de spécifier des notifications pouvant être émises par des objets gérés. Ces notifications sont inclues dans des modules ;
- Le formulaire de **paramètre** : il permet de définir des paramètres d'attributs, d'actions et de notifications ;
- Le formulaire de **comportement** : il permet de définir et d'enregistrer sous une étiquette, la définition informelle d'un comportement ;
- Le formulaire de **corrélation de noms** : il est utilisé pour spécifier des liens de contenance entre les différents objets gérés au sein d'une base d'information de gestion. Ces liens sont à la base de la hiérarchie de cette dernière.

Le formulaire de classe d'objet géré

La classe d'objets spécifiée en GDMO est caractérisée par son nom, la liste des classes dont elle hérite, un ensemble de modules qui définissent l'extension des fonctionnalités et par un identificateur d'enregistrement. Le formulaire de classe d'objet géré est donné dans la figure 2.2.

```
<class-label> MANAGED OBJECT CLASS
    [ DERIVED FROM <class-label> [,<class-label>]*;
    ]
    [ CHARCTERIZED BY <package-label> [,<package-label>]*;
    ]
    [ CONDITIONNAL PACKAGES <package-label> [,<package-label>]*
                            PRESENT IF condition
                      [,<package-label> [,<package-label>]*
                            PRESENT IF condition]*;
    ]
 REGISTERED AS <object-identifier>;
```

FIG. 2.2 – *Formulaire de classe d'objet géré GDMO*

La clause DERIVED FROM précise la liste des super-classes dont on hérite des propriétés.

La clause CHARACTERIZED BY déclare le(s) modules(s) qui définissent les propriétés de la classe (attributs, actions et notifications).

La clause CONDITIONAL PACKAGES permet de référencer des modules optionnels: ils seront inclus à l'instanciation la classe si la condition de la clause PRESENT IF est vérifiée.

La clause REGISTERED AS permet de donner un identificateur à la nouvelle définition de classe: dans l'approche OSI, les formulaires sont identifiés sans aucune ambiguïté grâce à ces identificateurs regroupés dans un *arbre d'enregistrement.*

Architecture de la MIB OSI

La base de gestion au sein de l'OSI est organisée sous forme d'un arbre dont les nœuds et les feuilles représentent des instances de classes du modèle. L'arborescence est alors déterminée par une notion de contenance définie à l'aide du formulaire de corrélations de noms (Name-Binding: voir annexe A).

Une corrélation de noms entre deux classes spécifie que les instance de l'une peuvent "contenir" les instances de l'autre. Un attribut particulier de la classe *subordonnée* est spécifié par cette relation en tant qu'*attribut de nommage* afin de permettre l'identification des instances de la MIB. Un arbre dit *arbre de nommage* se construit avec ces relations de contenance, et l'identification d'une instance se fait par la suite des valeurs des attributs de nommage des instances rencontrées depuis la racine de l'arbre (figure 2.3).

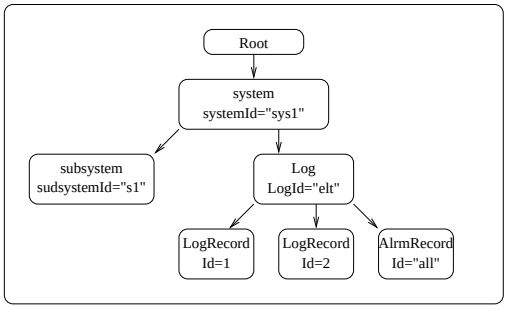

FIG. 2.3 – *Architecture d'une MIB OSI*

2.3.2 Les services CMIS

Les services CMIS, permettent de lire ou de modifier l'information de gestion contenue dans les MIB des agents OSI. A ces services, peuvent être associés des mécanismes supplémentaires dont le rôle est de faciliter l'accès à plusieurs objets gérés par un même appel. Dans cette section nous présentons brièvement chacun de ces services, ainsi que les mécanismes supplémentaires qui peuvent leurs être associés.

La liste des services CMIS

- Le service M-GET : permet de demander les valeurs des attributs d'une instance de la MIB de l'agent ;
- Le service M-CANCEL-GET : permet d'arrêter une requête M-GET si celle-ci a des réponses mutiples (le cas d'une requête M-GET avec un mécanisme de portée) ;
- Le service M-CREATE : permet de créer une nouvelle instance de classe dans la MIB de l'agent ;
- Le service M-DELETE : permet de supprimer une instance d'une classe dans MIB de l'agent ;

- Le service M-SET : permet de modifier les valeurs des attributs d'une instance d'une classe de la MIB ;
- Le service M-ACTION : permet d'exécuter une action sur une instance de la MIB de l'agent ;
- Le service M-EVENT-REPORT : permet de notifier un événement [9] au gestionnaire.

Les mécanismes supplémentaires

- Le mécanisme de portée : permet d'étendre la requête sur tout un sous-arbre de la MIB ;
- Le mécanisme filtrage : complémentaire au mécanisme de portée, il permet de restreindre la recherche dans un sous-arbre aux instances satisfaisant les conditions du filtre (assertions sur la présence ou les valeurs des attributs) ;
- Le mécanisme de synchronisation : permet de rajouter une condition supplémentaire sur l'exécution d'une opération de gestion multi-cibles. L'opération n'est effectuée que si toutes les instances sélectionnées (par un mécanisme de portée avec ou sans filtrage) l'autorisent.

2.4 Présentation de l'approche WBEM

L'approche naissante WBEM, soutenue par plusieurs industriels de l'informatique pour une gestion homogène de tout type de réseau, se définit aussi à partir de chacune des composantes principales de la gestion de réseaux.

Dans cette section, nous en présentons les modèles de l'information et de communication [30] [29] [32], et ce dans le but de les comparer à ce qui a été étudié pour l'approche OSI.

2.4.1 Le modèle CIM

Le modèle de l'information retenu dans l'approche de gestion WBEM est le modèle CIM [10] en cours de normalisation au DMTF [11][14].

Le modèle CIM repose sur une approche orientée objet avec héritage simple pour la spécification des ressources de gestion. L'élément principal du modèle CIM est le méta-modèle. Le méta-modèle définit l'ensemble des constructeurs de l'information de gestion. Ces constructeurs, formellement spécifiés à l'aide de la notation MOF [12] sont:

- le **schéma** qui regroupe un ensemble de spécifications de **classes**, d'**instances** et de **qualifiers** ;
- la **classe** d'objet: toute ressource est modélisée par une ou plusieurs classes d'objets gérés. Une classe est définie par des **attributs** (ou propriétés) et des **méthodes** qui représentent son interface ;
- l'**attribut** représente un état ou une propriété de l'objet dans lequel il est défini. L'attribut est défini par un type MOF ;
- la **méthode** définit des opérations que l'on peut invoquer sur une instance d'objet géré ;

9. exemple: disfonctionnement d'une ressource
10. Common Information Model
11. Desktop Management Task Force
12. Managed Object Format

- l'**association** est une classe particulière permettant de spécifier des dépendances entre classes. Une association comprend au moins deux attributs de type **référence** et peut comporter des méthodes et d'autres attributs ;
- la **référence** est un attribut particulier qui pointe vers un objet géré ;
- l'**instance** d'objet qui est une occurrence particulière d'une classe ;
- le **qualifieur** permet de compléter la définition d'un composant (classe, attribut, méthode, instance). Il permet notamment d'attacher des éléments de sémantique aux composants auxquels il est associé ;
- l'**indication** est une classe particulière dont les instances sont créées par des **déclancheurs** ;
- le **déclancheur**: opération invoquée lors d'un changement d'état dans une MIB.

La classe d'objet

La spécification d'une classe d'objets dans le formalisme MOF se fait de manière simple : la classe est définie par son nom, éventuellement la classe dont elle hérite les propriétés, et par un ensemble d'attributs et de méthodes (voir figure 2.4).

```
1  [ qualifiers] CLASS <class-label>
2        [ AS <alias-identifier>] [ : <super-class-label>]
3          {
4          [ property | method ]*
5          }
6  ;
7
8  property -> [ qualifiers]
9                  dataType <property-label> [ array] [default-value] ;
10 method -> dataType <method-label> ( parameter* ) ;
```

FIG. 2.4 – *Spécification d'une classe MOF*

La clause AS, permet de donner un nom d'alias à la classe ce qui pourra la référencer dans d'autres spécifications. La clause d'héritage indique le nom de la super-classe de laquelle la classe héritera des proriétés et des méthodes.

Le qualifieur

Le qualifieur, généralement utilisé pour rajouter une sémantique aux autres éléments, est défini par un nom, un type de données, une valeur par défaut, un champ d'application à savoir les composants d'une spécification donnée, et une validité vis-à-vis de la propagation[13]. Un certain nombre de qualifieurs de base[14] sont définis dans le modèle CIM. Les plus importants sont:

- `Association`: attaché à une classe, il sert à définir une association ;

13. Propagation au sens de l'héritage des propriétés.

14. Ils sont regroupés dans un **schéma** particulier appelé schéma Core. chaque entité WBEM doit inplanter ce shéma de spécifications de base.

- `Indication`: attaché à une classe, il sert à définir une indication ;
- `Reference`: attaché à un attribut, il sert à définir une référence vers un objet ;
- `Abstract(bool)`: associé à une classe, il permet de spécifier que celle-ci ne peut être instanciée ;
- `Key(bool)`: permet de préciser qu'un attribut d'une classe est utilisé comme clef pour le nommage des instances (voir *architecture de la MIB WBEM*) ;
- `Description(string)`: permet d'associer une description textuelle à un composant d'une spécification ;
- `Read(bool)`, `Write(bool)`: permettent de spécifier si un attribut ou une référence peuvent être lus, respectivement affectés.

L'architecture de la MIB WBEM

La MIB d'un agent WBEM est structurée par un concept d'espaces de nommage que nous pouvons comparer à l'arborescence des fichiers dans le système UNIX.

L'espace de nommage sert un regrouper un ensemble d'objets (classes et instances) sous une même étiquette. Un espace de nommage peut aussi contenir un autre espace de nommage et ainsi de suite. La création d'une instance d'une classe donnée ne se fait que dans l'espace de nommage contenant la définition de la classe. Un objet est alors identifié par son nom, ainsi que par le chemin[15] de l'espace de nommage dans lequel il est défini (figure 2.5).

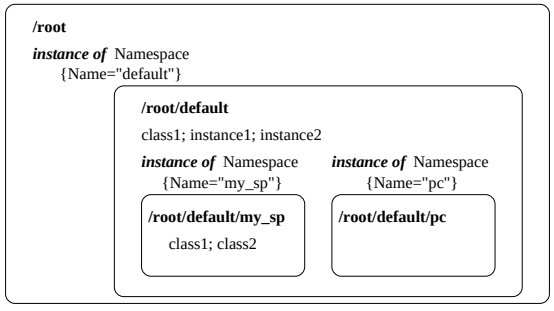

FIG. 2.5 – *Les espaces de nommage dans la MIB WBEM*

2.4.2 Les services HMMS

Les services offerts par la gestion WBEM sont les services HMMS[16] [32]. Les services HMMS permettent d'accéder à tous les éléments d'une MIB WBEM, et donc offrent la possibilité de manipuler aussi bien des définitions de classes que des instances de celles-ci.

- **Get Class:** permet de demander la définition d'une classe (liste des attributs, méthodes et qualifieurs) ;
- **Delete Class:** permet de supprimer une classe d'un espace de nommage ;

15. Chemin au sens du chemin dans les répertoires sous UNIX
16. Hyper-Media Management Services

- **Enumerate Classes:** permet de demander plusieurs définitions de classes suivant l'un des deux critères suivants: l'ensemble des sous-classes d'une classe donnée, ou l'ensemble des classes définies dans un espace de nommage donné ;
- **Put Class:** permet de rajouter ou de modifier une définition de classe. La distinction se fait à l'aide d'un champs particulier du service: `Modify/Create` ;
- **Get Instance:** permet de demander les valeurs des attributs d'une instance donnée ;
- **Delete Instance:** permet de supprimer une instance d'une classe ;
- **Enumerate Instances:** permet de demander toutes les instances d'une classe donnée ;
- **Put Instance:** permet aussi bien de créer une nouvelle instance, que de modifier les valeurs des attributs d'une instance déjà existante.
- **Execute Method:** permet d'invoquer une méthode sur une instance donnée ;
- **Cancel Enumeration:** permet d'annuler un service d'enumération (de classes ou d'instances) ;
- **Execute Query:** permet d'invoquer des requêtes sur la MIB de l'agent. Les langages de requêtes retenus pour le moment sont HMQL1[17] et HMQL2.
- **Délivrance d'indications:** permet à un producteur d'indications de les transmettre à un consommateur.

2.5 Conclusion

L'étude de chacune des approches, réalisée dans ce chapitre, nous permet de tirer un certain nombre de conclusions quand aux ressemblances et différences qui existent entre elles.

Du point de vue du modèle de l'information, les deux systèmes de gestion montrent une ressemblance de taille: elles sont toutes les deux basées sur des approches orientées objet et donc sont basées principalement sur les mêmes types de méta-constructeurs (la classe, l'attribut, la méthode,...)

Le modèle OSI paraît beacoup plus complet que le modèle WBEM. Pour combler ses lacunes, l'approche WBEM introduit la notion de qualifieur afin de rajouter une sémantique aux spécifications MOF.

Du point de vue du modèle de communication, l'approche WBEM, comparée à l'approche OSI, introduit beaucoup plus de services, et notamment les services d'accés aux définitions de classes. Mais les services CMIS s'avèrent plus complexes en supportant les mécanismes de portée, filtrage et synchronisation.

Dans les chapitres suivants, nous allons discuter de l'intégration de la gestion WBEM dans le cadre d'une gestion OSI, et nous allons nous appuyer sur ces remarques pour décrire notre modèle d'intégration.

17. Hyper-Media Query Langage 1

Chapitre 3

L'intégration dans la gestion de réseaux

3.1 Introduction

Comme dans toute autre application distribuée, l'interopérabilité a été l'une des préoccupations majeures des spécialistes de la gestion de réseaux. Ainsi, plusieurs études d'intégration ont été proposées afin d'assurer cette propriété entre les différents systèmes de gestion existants.

L'objet de ce chapitre est de présenter les différents concepts utilisés dans le cadre de ces approches d'intégration. Dans un premier temps, nous en présentons les concepts généraux, puis nous citerons quelques exemples d'intégration afin d'illustrer ces concepts. Un exemple parmis ceux-ci sera particulièrement développé: l'intégration OSI/CORBA[18] qui présente des caractéristiques intéressantes pour notre étude.

3.2 Les concepts généraux de l'intégration

L'objectif de l'intégration dans la gestion des réseaux est d'assurer l'interopérabilité entre deux entités implémentant des approches de gestion différentes. il est alors indispensable de réaliser cette interopérabilité, et donc une gestion hétérogène, au niveau de chacune des composantes principales définissant les approches de gestion.

3.2.1 Architecture fonctionnelle

Afin de permettre la communication entre un gestionnaire et un agent implémentant des approches de gestion différentes, quatre architectures de gestion hétérogène ont été proposées par la communauté de la gestion des réseaux [20]:

Un gestionnaire hétérogène

Dans cette approche, on implémente dans le gestionnaire chacun des deux modèles, et on offre aux applications un langage de gestion unifié NML (Network Management Language). Une couche intermédiaire (NMLS Network Management Language System) se chargera alors de traduire les requêtes NML selon la nature de l'agent cible vers l'un ou d'autre des modèles (figure 3.1).

18. Common Object Request Broker Architecture

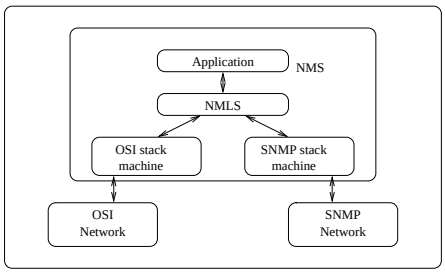

FIG. 3.1 – *Architecture d'un gestionnaire hétérogène*

Un agent hétérogène avec duplication de MIB

Cette fois, l'interopérabilité est assurée au niveau de l'agent qui implémente deux sous-agents ainsi que deux versions de la base d'information de gestion (MIB: Management Information Base) correspondants à chacun des modèles. Un gestionnaire pourra alors accéder à la MIB qui correspond au modèle qu'il implémente. L'intégration se fait dans ce cas entre les deux MIB (figure 3.2) à un niveau purement informationnel.

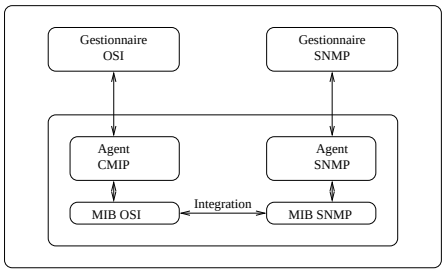

FIG. 3.2 – *Architecture d'un agent hétérogène avec duplication de MIB*

Un agent hétérogène avec une MIB unifiée

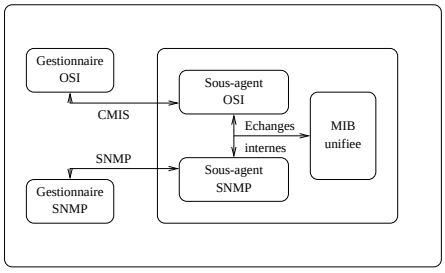

FIG. 3.3 – *Architecture d'un agent avec une MIB unifiée*

Analogue à l'approche précédante, cette technique propose un agent à double rôle (un sous-agent pour chacune des approches) mais avec une MIB unifiée. Chacun des sous-agents pourra consulter la MIB par des échanges internes à travers une interface générique (figure 3.3).

Un agent d'adaptation ou passerelle

Cette technique consiste à intercaler entre le gestionnaire et l'agent une troisième entité qui se chargera d'effectuer les traductions nécessaires pour les faire interopérer. Cette architecture est de loin la plus utilisée car elle est la plus modulaire et donc moins coûteuse (figure 3.4).

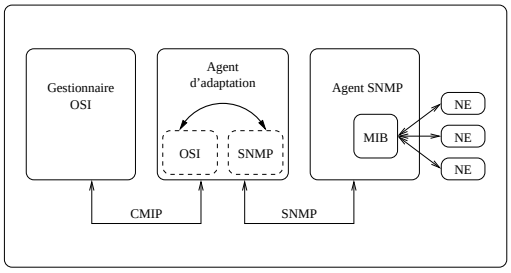

FIG. 3.4 – *Architecture de gestion avec un agent d'adaptation*

Avantages et inconvénients de ces architectures

L'architecture la plus intéresante dans le domaine de l'intégration des approches de gestion est celle de l'agent d'adaptation. En effet, elle permet l'interopérabilité entre des agents et des gestionnaires différents sans avoir à les modifier.

Les trois premières architectures que l'on peut appeler techniques de "sous-agent"[19] présentent un inconvénient majeur: la nécessité d'implémenter deux approches dans chaque agent, ce qui revient à augmenter la taille et la complexité du système de gestion. De plus, elles nécessitent une révision des implémentations à chaque tentative d'intégration d'une nouvelle approche.

3.2.2 L'intégration du modèle de l'information

C'est l'étape qui est à la fois la plus importante mais aussi la plus délicate dans l'intégration des approches de gestion. Son objectif est de permettre aux différents systèmes de gestion d'utiliser un "vocabulaire" commun pour la gestion des ressources: *le modèle de l'information.* L'étude des concepts généraux de l'intégration du modèle de l'information que nous présentons dans cette section est tirée des travaux d'A.Rivière et M.Sibilla [24].

Caractéristiques du modèle de l'information

Le modèle de l'information se définit comme l'ensemble des règles de construction de la base de gestion (MIB) à partir des caractéristiques utiles - pour la gestion en question - de chacune des ressources physiques ou logiques. A partir de cette définition, on distingue dans le modèle de l'information 3 niveaux conceptuels :

- La structure de l'information de gestion (SMI: Structure of Management Information) : c'est la spécification des concepts de gestion et des concepts opérationnels qui seront à la base des descriptions des ressources à gérer ;
- La spécification de la MIB (MIM : Management Information Model) : description de chacune des ressources à gérer conformément aux recommandations de la SMI ;

19. bien que la première propose un sous-gestionnaire.

- L'implémentation de la MIB (MIB : Management Information Base) ou encore une instance d'une MIB.

Caractéristiques des approches d'intégration des modèles

L'intégration des modèles de l'information consiste à déterminer un mécanisme de traduction des objets d'un modèle *source* vers des objets d'un modèle *destination*. Pour qualifier les approches d'intégration existantes, trois caractéristiques principales ont été identifiées:

Philosophie: Elle caractérise le niveau conceptuel (SMI, MIM, MIB) à partir duquel la traduction est effectuée. On distingue 3 philosophies d'intégration:

- Technique: la traduction de technique consiste à décrire le méta-modèle de l'approche source avec des composants du méta-modèle de l'approche destination. Ainsi, le méta-modèle de l'approche destination devient un " méta-méta-modèle" de l'approche source ;
- Recast: cette philosophie agit aussi au niveau des méta-modèles mais cette fois on cherche à déterminer des correspondances entre les méta-constructeurs. Ainsi, tous les concepts du modèle source seront conservés. Cette philosophie est la plus utilisée dans le domaine de l'intégration (cf. propositions du IIMC [16] et du JIDM [19]) ;
- Domaine : Cette technique diffère des deux précédantes dans la mesure où elle n'opère plus sur les méta-modèles mais sur les modèles mêmes. On cherche alors à déterminer des équivalences entre les éléments de la MIB de chacune des approches. Cette technique reste limitée car elle ne peut être appliquée qu'entre des modèles proches dans leur conception. De plus, elle ne permet pas de développer des traductions automatiques.

Principe: Pour effectuer une intégration entre les modèles de l'information, on a besoin de définir toutes les règles de traduction entre les éléments de chacune des approches. Ces règles peuvent être complètement génériques, ou alors spécifiques à certains cas de figures. On distingue alors deux principes de traduction:

- Traduction directe : on parle alors de projection simple des éléments du modèle source vers des éléments du modèle destination. Ce principe permet de réaliser des traductions complètement automatisées, mais son utilisation reste limitée aux modèles de conceptions semblables.
- Traduction abstraite : plus appropriée quand les modèles sont de natures très différentes. Ce principe repose sur la modification du modèle source (par ajoût ou suppréssion d'éléments) afin de fournir un modèle cohérent avec les concepts du modèle destination.

Dynamique: Dans l'intégration des modèles de l'information on peut identifier une recherche d'interopérabilité dans deux contextes différents: un contexte purement informationnel et un contexte d'appels de services. Ceci permet de distinguer deux étapes dans l'intégration:

- Traduction statique : elle s'opère sur un niveau purement informationnel (SMI et MIM);
- Traduction dynamique : elle intègre les notions dynamiques d'une MIB. Typiquement elle comprend les traductions des nommages des instances qui n'apparaîssent pas au niveau des définitions SMI et MIM.

3.2.3 Intégration dans les modèles de communication

Cette intégration concerne la traduction des services et protocoles de communication : une requête émise par un gestionnaire implémentant une approche donnée doit être traduite vers son équivalent dans l'approche qu'implémente l'agent. Dans ce paragraphe, nous ne nous intéresserons qu'aux approches d'intégration introduisant un agent d'adaptation - les plus utilisées - et qui nécessitent ce type de traduction.

Stratégie du transfert des requêtes

En introduisant un agent d'adaptation intercalé entre le gestionnaire et l'agent, on affecte directement le couplage entre ces deux derniers. Ainsi, deux types de passerelles ont été identifiées [1]:

- *stateless gateway* (figure 3.5): elle garde un couplage fort entre le gestionnaire et l'agent en transmettant immédiatement toute requête de l'un vers l'autre. Ainsi elle ne joue qu'un simple rôle d'intermédiaire de traduction;

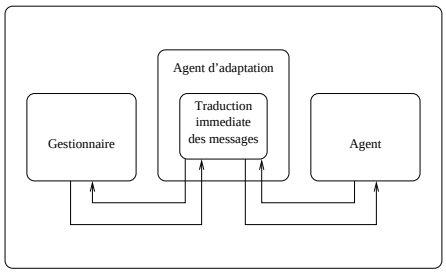

FIG. 3.5 – *Stateless gateway*

- *statefull gateway* (figure 3.6): de part son principe, elle découple les deux entités de gestion. En effet, cet agent maintient une MIB image de celle de l'agent, et répond aux requêtes du gestionnaire par simple consultation de cette MIB. Il implémente alors un mécanisme de mise à jour de cette MIB (ainsi que celle de l'agent) par un dialogue généralement périodique avec l'agent.

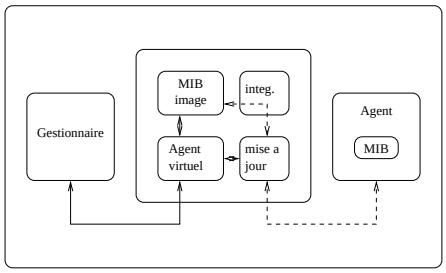

FIG. 3.6 – *Statefull gateway*

Chacun de ces agents d'adaptation a ses avantages et ses inconvénients: l'agent *stateless* permet de garder une consistance entre le gestionnaire et l'agent mais peut provoquer une augmentation du trafic de gestion; en contre partie, l'agent *statefull* permet d'optimiser le trafic de gestion mais peut introduire des incohérences temporaires entre l'information sur l'état des ressources vue par le gestionnaire et l'état réel de ces ressources.

Ainsi, un agent *statefull* sera plus adapté dans le cas d'une information statique[20], alors qu'un agent *stateless* sera plus adapté dans le cas d'une information plus dynamique. Le meilleur compromis serait alors de définir un agent d'adaptation qui regroupe les avantages de chacun des modèles, en identifiant la nature (statique ou dynamique) de l'information qu'il manipule[21].

D'autres aspects de cette discussion sont donnés dans le cadre spécifique de l'intégration OSI/SNMP (voir §3.3).

Intégration des services

Bien qu'intuitivement plus simple à réaliser que l'intégration des modèles de l'information, dans la mesure où l'on retrouve dans toutes les approches les mêmes philosophies de services (lire ou écrire dans une MIB, notifier un événement,...), un certain nombre de problèmes reste à résoudre dans le cadre de l'intégration des services pour une approche avec un agent d'adaptation. Le point le plus sensible à ce sujet est la complexité des mécanismes complémentaires que certaines approches de gestion rattachent aux services primitifs. Ainsi un appel d'un service complexe dans une approche peut se traduire par plusieurs appels de services simples dans la deuxième.

Pour cette raison, nous n'avons pas relevé de schéma générique pour l'intégration des services : les travaux se font souvent au cas par cas suivant les mécanismes proposés par les différentes approches. A titre d'exemple, le service GET avec un mécanisme de portée de l'approche OSI a souvent été traduit par une suite d'appels GET de l'approche SNMP dans les intégrations OSI/SNMP.

3.3 Approches d'intégration existantes

Dans cette section nous présentons sommairement certaines des approches d'intégration afin de mieux illustrer l'aplication des concepts que nous avons rappelés dans la section précédante. Ceci nous permettra aussi de dégager certaines des techniques qui ont déjà fait leurs preuves dans le domaine de l'intégration.

L'agent PIMA : Le Protocol Independant Management Agent (PIMA) proposé par l'équipe Network Management Systems d'IBM [21], réalise l'intégration SNMP/OSI par l'intermédiaire d'un agent double avec une vision unifiée d'une MIB (voir figure 3.3).

L'interopérabilité entre les deux modèles est réalisée à travers la MIB générique composée d'objets non typés, et d'une interface générique pour les appels de services.

L'agent PIMA permet d'intégrer les deux modèles de gestion SNMP et OSI de manière simple à travers un mécanisme de tables de pointeurs sur les objets de la MIB unifiée, sans s'occuper des problèmes de l'intégration dans le modèle de communication. Il présente cependant un inconvénient que nous rappelons: il nécessite l'implémentation de chacun des deux modèles SNMP et OSI dans chaque agent du réseau.

20. Ne change pas fréquemment tel qu'une adresse IP d'une station
21. Encore faut-il pouvoir faire cette distinction!

Le projet ICM : Le projet ICM (Integrated Communication Management) [22] propose une gestion des agents SNMP à partir d'une plate-forme OSI par l'intermédiaire d'un agent de Q-Adaptation (voir figure 3.4).

L'intégration du modèle de l'information est basée sur les travaux du Network Management Forum ISO/CCITT et du ISO Internet Management Coexistance (IIMC), et plus particulièrement sur les algorithmes donnés dans IIMCIMIBTRANS (Translation of Internet MIBs) [16]. Elle a été réalisée de manière statique basée sur une philosophie de *recast*: les algorithmes de traduction définissent des correspondances entre les éléments des méta-modèles des deux gestions OSI et SNMP. De cette intégration de l'information nous retiendrons les 3 directives suivantes :

- définition des procédures de pré-traductions: elles ont pour objectifs la créations des éléments de base pour la construction toutes les arborescences définies au sein du modèle OSI[22], ainsi que la création des modules ASN-1 nécessaires pour les références de types ;
- définition des procédures de traduction proprements dites, c'est à dire les règles de construction des formulaires GDMO à partir des spécifications d'objets SNMP :
- définition des procédures de post-traductions: elles concernent notamment la vérification manuelle des clauses BEHAVIOUR (utilisées ici pour inclure tout ce qui ne peut pas être directement traduit) et la définition d'un arbre de contenance (ce qui revient à spécifier des corrélations de noms).

Parallèlement à l'intégration du modèle de l'information, l'intégration des services est basée sur les travaux de l'IIMC: IIMCPROXY (ISO/CCITT to Internet Management Proxy) [15]. Pour cette intégration L'ICM a opté pour un agent *statefull* jugé plus avantageux que l'agent *stateless* proposé dans IIMCPROXY, et ce relativement à chacun des aspects suivants:

- traduction des notifications: l'agent SNMP n'utilise les traps[23] que pour des événements "exceptionnels", alors que le modèle OSI utilise les notifications plus fréquemment afin de limiter les scrutations de ses agents. Un agent *stateless* ne permet pas d'intégrer cet aspect fonctionnel, alors qu'un agent *statefull* pourra s'aider de l'état des variables dans la MIB image de celle de l'agent pour générer les notifications requises par l'OSI;
- traduction des mécanismes de portée et de filtrage (voir 2.3.2): le protocole SNMP n'offre pas ces mécanismes. Ainsi sur une requête CMIS de ce type émise par le gestionnaire OSI, l'agent *stateless* est contraint d'émettre vers l'agent SNMP autant de requêtes que d'objets ciblés, alors que l'agent *statefull* pourrait optimiser ce trafic en répondant tout de suite à la requête CMIS par consultation de sa MIB interne.

L'intégration SNMP/OSI/CORBA : Cette approche d'intégration a été proposée par le JIDM (Join Inter-Domain Management) [19] dans le cadre de la mise en place d'un agent d'adaptation permettant d'inter-connecter trois réseaux avec des approches de gestion différentes: CORBA IDL, OSI et SNMP. De ces travaux, nous avons retenus les algorithmes d'intégration du modèle de l'information de CORBA IDL vers le celui de la gestion OSI, le

22. arbre d'enregistrement, arbre d'héritage et arbre de nommage
23. notification envoyée par l'agent pour informer le gestionnaire de l'occurence d'un évenement particulier

modèle CORBA IDL étant assez semblable avec celui de la gestion WBEM. Ces algorithmes peuvent se résumer par les points suivants :

- Spécification d'un fichier de base (`JIDMbasedocument`) contenant les défintions nécessaires pour la construction des formulaires GDMO (racine de l'héritage, racine de l'enregistrement,...) ;
- Chaque interface IDL est traduite par une définition de classe d'objet géré ;
- Chaque attribut de l'interface IDL sera traduit par un attribut GDMO ;
- Chaque opération IDL est traduite par une action GDMO ;
- L'arbre de contenance est défini de manière simple : toutes les classes sont appelées par une unique classe (`CorbaSystem`) à l'aide d'un attribut générique `corbaName`.
- Des tables de traductions sont données par le JIDM afin de traduire les types de données IDL vers des types ASN-1 (types simples et structures).

3.4 Conclusions

Dans ce chapitre nous avons présenté certaines techniques et directives qui ont déjà été utilisées pour d'autres approches d'intégration. Nous avons tenu compte de ces directives pour concevoir notre intégration OSI/WBEM. Nous résumons ici les directives les plus importanes qui nous ont servi de point départ.

Du point de vu des concepts de l'intégration, nous avons d'ores et déjà retenu deux modèles présentant plusieurs avantages :

- une architecture d'intégration avec un agent d'adaptation entre un gestionnaire OSI et un agent WBEM ;
- une philosophie de recast pour la traduction des modèles de l'information.

Du point de vu des approches étudiées, nous avons également retenu certaines techniques, issues notemment des approches d'intégration du JIDM et du IIMC, pour l'intégration dans le modéle OSI :

- génération d'un fichier de "base" contenant toutes les définitions de base pour la création des différentes hièrarchies du modèle OSI ;
- l'approche de gestion CORBA présente plusieurs similitudes avec l'approche WBEM, de ce fait nous retiendrons des travaux du JIDM la plupart des algorithmes présentés ;
- pour les mêmes raisons que celles présentées dans les travaux de l'ICM, nous retiendrons l'idée d'un agent d'adaptation *statefull* qui permet de satisfaire la majorité des mécanismes supplémentaires des services CMIS.

Chapitre 4

Intégration OSI/WBEM

4.1 Introduction

Dans les sections précédantes nous avons identifié les problèmes qui doivent être résolus afin de réaliser l'intégration de la gestion WBEM dans le cadre d'une gestion OSI. Aussi, grâce à l'étude des approches d'intégration réalisées dans la communauté de gestion, nous avons été en mesure de dégager certaines techniques que nous avons utilisées dans le cadre de notre application. Ainsi, nous avons conçu des algorithmes de traduction que nous avons implémentés pour réaliser l'intégration WBEM/OSI.

L'objet de ce chapitre est la présentation de ces algorithmes. Nous l'avons structuré comme suit: dans un premier temps nous rappelons ces algorithmes dans leurs grands axes; puis nous nous intéresserons aux traductions des spécifications MOF vers des spécifications GDMO; ensuite nous parcourerons les traductions des types MOF vers des types ASN-1; et finalement nous présenterons les traductions des services de gestion.

4.2 Intégration du modèle de l'information

4.2.1 Présentation générale

Dans la section de présentation des concepts généraux (cf §3.2.2), nous avons noté que l'intégration des modèles de l'information peut être réalisée par l'une des trois philosophies suivantes: *technique, recast* ou *domaine*. Dans notre approche d'intégration du modèle CIM vers le modèle OSI, nous avons opté pour une philosophie de *recast* plus avantageuse pour une traduction complétement automatisée.

En fonction de ce choix, nous avons développé des algorithmes de traduction qui définissent des correspondances entre les constructeurs du modèle[24] CIM spécifiés en MOF vers ceux du modèle OSI spécifiés en GDMO/ASN-1.

Dans leurs grandes lignes, les directives de traduction que nous présentons dans cette section sont :

- construction de la structure de base pour la reconstitution de l'ensembles des hiérarchies OSI ;
- définition des correspondances entre les types MOF et les types ASN-1 utilisés par le langage GDMO ;

24. composants du méta-modèle

– définition des correspondances entre les constructeurs des spécifications MOF et les formulaires GDMO

Les spécifications de base

Les spécifications GDMO étant organisées d'une manière hiérarchique (héritage entre classes, arbre d'enregistrement, arbre de nommage), nous avons spécifié un fichier GDMO contenant toutes les définitions de base nécessaires pour créer de telles structures dans le modèle de l'information de gestion issu des traductions. Dans ce fichier nous avons défini les objets suivants:

– la classe MOF_MOC : racine de l'arbre d'héritage des classes. Elle ne contient aucun attribut spécifique ;

– la classe CIM_agent : elle donnera la racine de l'arbre de nommage. Les instances de cette classe représenterons un agent WBEM (adresse IP, état).

– la classe MOF_MetaMOC : c'est la classe qui servira pour contenir des définitions de classes MOF. Les instances de cette classe seront alors utilisées afin d'accéder à une définition de classe comme le propose l'approche WBEM. La spécification de cette classe est donnée en annexe B ;

– la classe MOF_NameSpace : classe qui servira pour représenter les espaces de nommage du modèle CIM. La classe possède un unique attribut qui représente le nom de l'espace de nommage ;

– la définitions d'une corrélation de noms entre les classe MOF_MetaMOC, CIM_agent et MOF_NameSpace. Ces corrélations (schématisées dans la figure 4.1) servent à construire l'architecture de la MIB de l'agent d'adaptation.

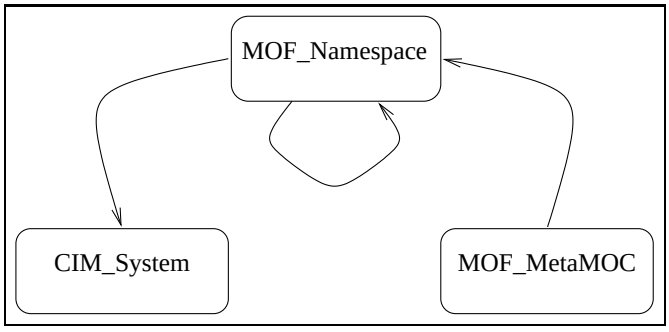

FIG. 4.1 – *Corrélations génériques du module de base*

Enregistrement des formulaires GDMO

Le modèle CIM ne supporte pas la notion d'enregistrement des objets comme dans le modèle OSI. Nous devons alors construire un arbre d'enregistrement pour identifier de manière unique la liste des spécifications GDMO issues des traductions afin de pouvoir les manipuler dans le contexte OSI. La génération de l'arbre se fait de la manière suivante (voir figure 4.2):

– la racine de l'arbre est définie dans le module de base;

- à chaque schéma correspond une entrée dans l'arbre;
- à chaque type de formulaire GDMO (classe, attribut, action,...) correspond une entrée dans l'arbre sous le brin du schéma dans lequel il est déclaré;
- à chaque objet est attribué un identificateur qui correspond à son ordre d'apparition dans les spécifications d'un même schéma.

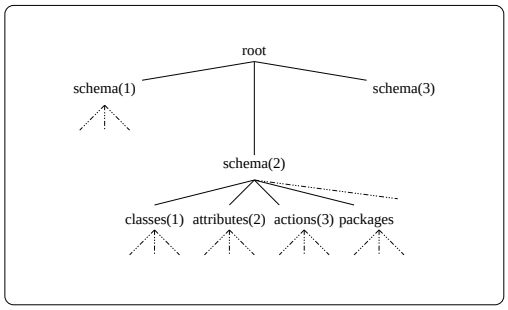

FIG. 4.2 – *Architecture de l'arbre d'enregistrement*

4.2.2 Traductions MOF/GDMO

Traduction d'une classe

Elément de base du modèle de l'information de l'approche WBEM, la classe d'objet est utilisée pour contenir toute l'information correspondante à une ressource donnée.

Les traductions que nous proposons se font alors par analyse des spécifications contenues dans la classe pour construire les différents éléments du modèle OSI. La classe MOF est traduite par une classe GDMO en deux temps:

- génération d'un formulaire GDMO de classe d'objet géré (MANAGED OBJECT CLASS) pour la déclaration de la classe et son enregistrement (figure 4.3);
- génération d'un formulaire de module (PACKAGE) déclaré en tant de module obligatoire de la classe et qui contiendra les spécifications des propriétés [25] de la classe (figure 4.4);

Un formulaire de comportement (BEHAVIOUR) peut être associé à la classe si dans les spécifications MOF, des qualifiers sont rattachés à la déclaration de celle-ci (typiquement le qualifier Description sera inclus dans ce formulaire).

Par défaut, chaque classe hérite de la classe MOF_MOC, sauf si un lien d'héritage est précisé dans la spécification MOF. Dans ce cas, le formulaire issu de la traduction de la super-classe sera l'unique déclaration de la clause DERIVED FROM.

De plus, à chaque nouvelle déclaration de classe, nous instancions la classe MOF_MetaMOC dans la MIB interne de l'agent d'adaptation (voir 4.4.1).

25. Attributs, actions, et notifications

```
MOF_<schema-name>_<class-label>Class MANAGED OBJECT CLASS
    DERIVED FROM MOF_MOC | MOF_<schemaName>-super-class-label;
    CHARACTERIZED BY MOF_<schema-name>_<class-label>Package;
 [ BEHAVIOUR MOF_<schema-name>_<class-label>BehaviourPackage;]
REGISTERED AS { <schema-name> classes xx };
```

FIG. 4.3 – *La spécification de classe*

```
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 { <schema-name> packages xx } ;
```

FIG. 4.4 – *Le package de classe*

Traduction d'une propriété

La propriété dans les spécifications MOF est caractérisée par son nom, son type, sa valeur par défaut, et par la liste des qualifieurs qui lui peuvent lui être attachés. Ainsi, pour la traduction d'une propriété MOF en un attribut GDMO nous opérons en deux étapes:

- définition de l'attribut GDMO: elle se fait à partir des caractéristiques de base de la propriété MOF (nom, type) dans un formulaire d'attribut GDMO (figure 4.5);

- nous complétons le formulaire d'attribut, ainsi que le formulaire de module correspondant à la classe déclarant cet attribut, par analyse de la liste des qualifieurs associés à la propriété MOF: certains de ces qualifieurs seront traduits par des éléments du modèle GDMO tel que propriété d'attribut, et les autres seront rajoutés à la définition de l'attribut sous forme d'un formulaire de comportement.

```
MOF_<schema-name>_<class-label>_<property-label>Attribute ATTRIBUTE
    WITH ATTRIBUTE SYNTAX reference-type;
 [ BEHAVIOUR
     MOF_<schema-name>_<class-label>_<property-label>AttributeBehaviour;]
REGISTERED AS { <schema-name> attributes yy };
```

FIG. 4.5 – *La spécification d'attribut*

Le type de l'attribut est donné par la clause WITH ATTRIBUTE SYNTAX, qui fait référence à un type ASN-1 défini par la traduction des types MOF en ASN-1.

Traduction d'une méthode

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. Le composant du modèle GDMO que nous associons à cette définition est l'action. Elle est rajoutée aux spécifications GDMO à deux niveaux:

- déclaration de l'action dans le formulaire de module associé à la classe ;
- création d'un formulaire d'action qui servira à donner les paramètres d'invocation et de retour de la méthode (figure 4.6).

```
MOF_<method_label>Action ACTION
    MODE CONFIRMED
    WITH INFORMATION SYNTAX reference-type;
    WITH REPLY SYNTAX reference-type;
REGISTERED AS { <schema-name> actions zz } ;
```

FIG. 4.6 – *La spécification d'une action*

Toutes les actions que nous déclarons dans nos traductions sont définies dans le mode comfirmé.

Les paramètres d'appels sont donnés par la clause WITH INFORMATION SYNTAX. Si la méthode MOF spécifie plusieurs paramètres d'appels, le type donné dans cette clause sera défini par la SEQUENCE de ces paramètres.

Le paramètre de retour de la méthode est donné par la clause WITH REPLY SYNTAX.

Traduction des qualifieurs

Certains de qualifieurs du modèle CIM jouent un rôle très important dans les spécifications MOF en leurs attribuant une sémantique particulière. Ainsi, nous avons essayé de tenir compte de ces qualifieurs en tant qu'éléments de l'information directement exploitables (par opposition aux descriptions en langage naturel des formulaires de comportement). A cet effet, nous avons défini des algorithmes de traduction pour certains des qualifieurs du schéma Core:

- le qualifieur `read`: attaché à une propriété, il précise le mode d'accès lecture seule. Nous l'avons traduit en tant que propriété "GET" attachée à un attribut ;
- le qualifieur `write`: analogue au qualifier read, nous l'avons traduit par une propriété "REPLACE" attachée à l'attribut. Par défaut, ces qualifieurs ont la valeur "true"; la propriété attachée par défaut à l'attribut GDMO sera donc GET-REPLACE ;
- le qualifieur `description`: il est rajouté à la spécification de l'objet GDMO auquel il est attaché comme une description de comportement ;
- le qualifieur `key`: voir §`traduction du nommage des instances` ;
- le qualifieur `abstract`: attaché à une classe, il précise qu'elle ne peut pas être instanciée. En présence de ce qualifieur, nous ne générerons pas de corrélation de nom de cette classe avec la classe MOF_Namespace, ainsi, elle ne pourra pas être instanciée.

Traduction du nommage d'instances

Le modèle CIM permet le nommage d'instances à partir de plusieurs attributs clefs (attributs spécifiés avec le qualifieur `Key`): une instance d'une classe sera caractérisée par l'espace de nommage dans lequel elle est créé ainsi que par la (les) valeur(s) de son (ses) attribut(s) clef(s).

Dans le modèle OSI, l'identification d'une instance de classe se fait par la donnée de chemin dans l'arbre de nommage ainsi que par la valeur que prend un unique attribut appelé attribut de nommage.

Pour reproduire la stratégie de nommage du modèle CIM dans le contexte OSI, nous devons résoudre les deux aspects du problème:

- la correspondance entre les espace de nommage et l'arbre de nommage;
- la correspondance entre une identification multi-attributs et un unique attribut de nommage.

Nous avons résolu le premier point d'une manière statique en définissant pour chaque classe instanciable[26] une corrélation de nom avec la classe MOF_Namespace (cette dernière étant la classe appelante).

Pour résoudre le second point, nous rajoutons un nouvel attribut à la classe: gdmoNaming. Il sera spécifié comme attribut de nommage dans la corrélation de noms avec MOF_Namespace. Sa syntaxe sera une séquence de tous les attributs de nommage définis dans la classe MOF.

4.3 Traduction des types MOF vers des types ASN-1

L'objectif de cette traduction est de permettre de faire des correspondances entre les spécifications des types du langage MOF vers les types ASN-1 qui seront utilisés dans les formulaires GDMO. Ainsi, nous avons distingué deux étapes dans cette intégration: la traduction des types de base, et la traductions des constructeurs de structures.

4.3.1 Traduction des types de base

Voir tableau 4.1.

Type MOF	*Type ASN.1*	Commentaire
(u/s)int**xx**	**INTEGER**	(u/s) = **u**nsigned/**s**igned. xx = 16, 32 ou 64. On peut sous-classer le type INTEGER ASN.1 afin de retrouver exactement les types MOF correspondants.
real**xx**	Real	real
boolean	BOOLEAN	boolean
char16	GraphicString	16 bits character UCS-2 coding.
string	GraphicString	UCS-2 string.
datetime	GeneralizedTime	date: année, mois, jour, heure avec une précision d'une fraction de seconde.

TAB. 4.1 – *Correspondance des types*

26. L'attribut `abstract` spécifie les classes non instanciables.

4.3.2 Traduction des structures

Le seul type structuré dans le langage MOF est le vecteur d'objets d'un même type simple. Nous avons donc traduit le vecteur MOF par le constructeur SEQUENCE OF. Si la taille du vecteur MOF est donnée, elle le sera pour la séquence ASN-1 (voir exemple du tableau 4.2).

Type MOF	Type ASN-1
uint64 un_vecteur[2]	un_vecteur SEQUENCE SIZE(2) OF INTEGER

TAB. 4.2 – *Exemple de traduction d'un vecteur MOF*

4.4 Intégration des services

4.4.1 Algorithme général des traductions des services

Dans notre agent d'adaptation, les traductions des services s'opèrent à l'aide de deux stratégies différentes suivant la nature de l'objet ciblé par la requête du gestionnaire OSI:

- une stratégie *statefull* pour des demandes de définitions de classes c'est à dire ciblant des instances de la classe MOF_MetaMOC[27] : l'agent d'adaptation maintient une MIB contenant les instances de MOF_MetaMOC, et la réponse à une requête du gestionnaire est donnée directement par consultation de cette MIB. Une mise à jour est faite séparemment pour garder une consistance entre cette MIB et celle de l'agent WBEM.

- une stratégie *stateless* pour des requêtes ciblant les autres instances d'objets gérés : la requête est traduite et transférée immédiatement vers l'agent, et la réponse de celui-ci est renvoyée au gestionnaire.

Mais d'une manière générale (en tenant compte de la procédure de mise à jour), le schéma de la traduction d'une requête du gestionnaire correspond à l'algorithme suivant (figure 4.7):

- réception d'une requête gestionnaire par l'intermédiaire de l'interface CMISoverJava[28] ;

- analyse de la requête, et détermination de la stratégie ;

- émission d'une requête équivalente vers l'agent WBEM ;

- réception de la réponse de l'agent par l'intermédiaire de l'interface Java du protocole HMMP[29] ;

- mise à jour de la MIB de l'agent d'adaptation selon la réponse de l'agent ;

- traduction et transfert de la réponse vers le gestionnaire OSI.

27. La classe MOF_MetaMOC définie dans les traductions du modèle de l'information permet les accés aux définitions de classes.

28. Implémentation des services CMIS en Java, développée au sein de l'équipe dans le cadre du projet ANTARES.

29. Hyper-Media Management Protocol: Le protocole applicatif qui a été retenu au depart dans l'initiative WBEM.

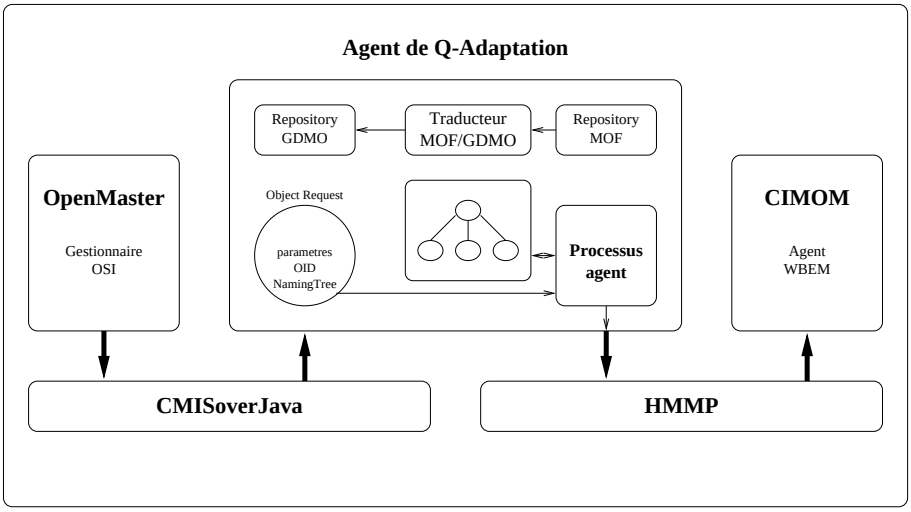

FIG. 4.7 – *Structure de l'agent d'adaptation*

Dans la suite de cette section nous présentons les algorithmes retenus pour traduire chacun des services CMIS ainsi que les mécanismes supplémentaires de portée et de filtrage. Seul le mécanisme de synchronisation n'est pas traduit car les instances d'une classes GDMO issus des traductions du modèle de l'information ont toutes les mêmes propriétés, ce qui n'est pas forcément le cas dans une spécification GDMO d'une classe contenant des modules conditionnels [30].

4.4.2 Le service M-GET

Le service M-GET simple

Le service M-GET dans sa forme simple permet de demander les caractéristiques d'un objet donné de la MIB. ainsi nous l'avons traduit de deux manières différentes suivant la nature de l'objet ciblé:

- demande d'une définition de classe : nous utilisons le service *Get Class* ;
- demande d'une instance de classe : nous utilisons le service *Get Instance*.

Le service M-GET avec portée et filtrage

Le service M-GET avec portée permet de demander un ensemble d'objets d'un même sous arbre de la MIB. Les services HMMS nous offrent alors des services semblables qui permettent de demander tous les objets appartenant à un même espace de nommage. Nous utiliserons ces services car les corrélations de noms que nous avons définies pour la traduction du modèle de l'information permettent de construire une arborescence avec des noeuds représentants les espaces de nommage. Nous avons donc implémenté la traduction suivante:

- déterminer à partir de la MIB interne de l'agent d'adaptation les classes appartenantes à l'espace de nommage désigné par le gestionnaire ;

30. La notion de définition conditionnelle n'existe pas dans le modèle CIM.

- lancer un appel *Enumerate Instances* pour chaque classe retenue par la recherche précédante ;

- si un filtre est précisé par la requête, il sera simplement inclu dans les paramètres de la requête *Enumerate Instances* qui supporte ce mécanisme ;

- la mise à jour peut se faire par un appel *Enumerate Classes* sur l'espace de nommage désigné.

4.4.3 Le service M-CREATE

Le service M-CREATE simple, permet au gestionnaire de créer un nouvel objet dans la MIB de l'agent. Dans notre approche d'intégration, ce service peut être traduit de deux manières différentes suivant la nature de l'objet ciblé.

- une définition de classe: l'objet ciblé appartient à la classe MOF_MetaMOC. Dans ce cas c'est une demande de création de classe qui est émise par l'appel du service *Put Class* avec le champ `Create` ;

- une instance d'une classe: l'objet ciblé appartient à tout autre classe que MOF_MetaMOC. Dans ce cas nous faisons appel au service *Put Instance* avec le champ `Create`.

4.4.4 Le service M-DELETE

Le service M-DELETE simple

Le service M-DELETE simple permet de détruire un objet de la MIB. Il est alors traduit de deux manières différentes suivant la nature de l'objet ciblé:

- destruction d'une définition de classe: il est traduit par la destruction dans la MIB de l'agent d'adaptation de l'instance de MOF_MetaMOC correspondante à la classe ciblée et par l'appel du service *Delete Class* pendant la mise à jour ;

- destruction d'une instance de classe: il est traduit par un appel du service *Delete Instance* directement envoyé vers l'agent.

Le service M-DELETE avec portée

Le mécanisme de portée permet de cibler tout un ensemble d'objets de la MIB OSI dans un même sous arbre de cette MIB. Les services HMMS n'offrent pas cette possibilité de destruction de groupes d'objets. Mais comme dans notre approche d'intégration du modèle de l'information, la MIB de l'agent d'adaptation est basée sur des nœuds formés par des instances d'objets représentant les espaces de nommage, le service M-DELETE avec portée sera traduit par une opération *Delete Instance* de l'espace de nommage racine du sous-arbre ciblé.

Remarque: dans la version actuelle de l'approche WBEM, la destruction d'un espace de nommage conduit à la destruction de tous les objets contenus dans cet espace.

Le service M-DELETE avec portée et filtrage

Le service M-DELETE avec portée et filtrage se traduit de deux manières différentes selon la nature des objets cibles:

- destruction de définitions de classes : il se fait de manière interne sur les instances de MOF_MetaMOC qui sont sélectionnés par la requête, puis des appels successif du service *Delete Class* ;
- destruction d'instances : la destruction d'instances avec filtrage nécessite une connaissance de chacune de ces instance, et donc nous passons par une opération M-GET filtrée, et puis des appels successifs du service *Delete Instance* pour chacune des instance sélectionnées.

4.4.5 Le service M-ACTION

Le service M-ACTION simple

Le service M-ACTION simple permet de lancer une action sur un objet. Tel que nous l'avons défini lors des traductions du modèle de l'information, l'action GDMO correspond à une méthode MOF. Il est tout naturel de traduire ce service par un appel du service *Execute Method.*

Le service M-ACTION avec portée

Le service M-ACTION avec portée permet de lancer l'action sur toutes les instances d'une classe donnée appartenantes à un sous arbre donné. Les services HMM ne proposent pas ce mécanisme. Nous l'avons alors traduit avec la même philosophie que pour le service M-SET avec portée: nous récupèrons toutes les instances satisfaisant à l'appel, ensuite nous lancons autant d'appels *Execute Method* qu'il y a d'objets issus de ce travail préliminaire.

Le service M-ACTION avec portée et filtrage

Il se traduit de manière analogue au service M-SET avec portée et filtrage: récupération des instances satisfaisant au filtre puis appels du service *Execute Method* sur chacune de ces instances.

4.5 Conclusion

Dans ce chapitre, nous avons présenté notre approche d'intégration de la gestion WBEM au sein d'une gestion OSI. Dans un premier temps nous avons réalisé l'intégration du modèle de l'information c'est à dire les traductions MOF vers GDMO/ASN-1. Nous avons ensuite présenté les algorithmes de traduction des services CMIS vers des services HMMS.

Avec ces deux applications, nous sommes en mesure d'implanter un agent d'adaptation capable de maintenir une MIB et de réaliser le transfert des services. Mais avant ceci, nous essayerons de valider le modèle de l'intégration ce qui fait l'objet du chapitre suivant.

Chapitre 5

Validation de l'intégration

Dans le chapitre précédant nous avons présenté notre approche d'intégration de l'approche naissante WBEM au sein de la gestion OSI plus ancienne et qui a déjà fait ses preuves. Ce travail étant effectué, les questions que nous nous sommes posées tout naturellement à ce stade de l'étude sont: valider? quoi? et pourquoi?

Nous avons essayé de répondre à ces questions en nous appuyant sur un langage formel très utilisé dans le domaine pour la validation des protocoles de communications et des réseaux de télécommunications : SDL'92 normalisé par le CCITT en 1993 [4].

Dans de la gestion de réseaux, SDL'92 a été souvent utilisé pour modéliser formellement les entités de gestion (gestionnaire, agent,...) afin d'en étudier le comportement. Disposant déjà de la modélisation en SDL'92 d'un agent OSI[31], nous avons définit une modélisation en SDL'92 d'un agent WBEM afin d'étudier le comportement du système { gestionnaire OSI , agent d'adaptation , agent WBEM }.

Dans ce chapitre nous présentons cette modélisation. Dans un premier temps nous rappelons brièvement les concepts généraux du langage SDL'92, puis nous citons quelques travaux de validation dans la gestion des réseaux avec SDL'92, ensuite nous présenterons nos algorithmes de modélisation de la gestion WBEM avec SDL'92, enfin nous discuterons l'intérêt d'une telle modélisation.

5.1 Présentation de SDL'92

SDL'92 est un langage de spécification formelle, utilisé notamment dans le domaine des systèmes distribués, et basé sur les automates étendus communicants. L'entité de base du langage est le processus: c'est un automate à états finis qui communique avec les autres processus ou le monde extérieur à l'aide de canaux ou acheminements de signaux.

Données: Pour décrire les données, SDL'92 est fondé sur le langage ACT One. Ce dernier est un langage de description des données basé sur les types abstraits. Dans sa nouvelle version, le langage autorise aussi des descriptions de données par les types ASN-1.

Hiérarchie: un système est décrit comme étant un ensemble de blocs. Un bloc peut contenir un ensemble de machines à états finis. Ces machines peuvent communiquer entre elles ou avec l'environnement du système. Le comportement de la machine à états finis est représenté par un processus. Un processus possède un ensemble d'états, dont un état particulier dit

31. Grâce à l'outil MODERES développé au sein de l'équipe RESEDAS

de départ, et un ensemble de transitions. Un processus passe d'un état à un autre par l'exécution d'une transition déclanchée par la réception d'un stimulus.

Communication: La communication peut être faite de deux manières différentes: asynchrone par l'intermédiaire de signaux envoyés à travers des canaux de communication; ou synchrone par des appels de procédures distantes sur d'autres processus.

Identificateur de processus: Un PID[32] exprime les informations de nommage relatives au processus. La valeur de PID identifie une instance de processus. Toutes les valeurs de PID sont uniques. Toute instance de processus possède quatre valeurs de PID pré-définies: **self** est l'identificateur de l'instance, **sender** est l'instance à partir de laquelle le processus a recu le signal le plus récent, **parent** est l'identificateur de l'instance de processus créateur et **offspring** est l'instance de processus le plus récemment créé.

Processus:
- Initialisation: l'interprétation d'une instance de processus débute par l'affectation de valeurs des paramètres du processus, la création ou éventuellement l'initialisation des variables du processus, et l'interprétation de l'état initial **start**.
- Comportement: Le comportement d'un processus est spécifié par une machine à états finis. Le passage d'un état à un autre se fait par la réception d'un stimulus et par l'exécution d'une transition. Une transition peut contenir une instruction d'affectation, un appel à une procédure et/ou une création d'une instance de processus. Une transition termine par un passage vers un autre état ou par une instruction de terminaison.
- Terminaison: L'interprétation d'une instance de processus termine lorsque le **stop** est traîté. A ce moment là, l'instance cesse d'exister.

Procédure: Une procédure est une partie paramétrée d'un processus. Une procédure possède le même corps qu'un processus. Lorsqu'une procédure est appelée, l'interprétation débute par l'état intial **start**. Le contrôle retourne au processus appelant ou à la procédure appelante lorsque **return** est exécutée.

Service: Un service est une machine à états finis qui sert à définir une partie du comportement d'un processus.

5.2 Le langage SDL'92 dans la gestion de réseaux

L'utilisation du langage SDL'92 pour spécifier formellement le comportement des objets gérés a été proposé à plusieurs reprises. Nous nous sommes intéressés à deux de ces propositions:

- La traduction des spécifications GDMO vers un système SDL'92 réalisée au sein de l'équipe RESEDAS et implantée depuis dans l'outil MODERES [27], [26].
- La traduction des spécifications CORBA IDL vers un système SDL'92 réalisée par la société Telelogic [3].

Dans cette section, nous présenterons brièvement chacune de ces deux approches afin d'en dégager les différents concepts et méthodes, que nous appliquerons dans notre approche de traduction des spécifications MOF vers un système SDL'92.

32. Process IDentity

5.2.1 GDMO vers SDL'92

L'objet de cette étude est de modéliser un agent de gestion OSI par le langage SDL'92 afin d'en tester le comportement. L'approche présente alors trois phases:

1. Traduction: l'objet de cette phase est de décrire le modèle de la MIB de l'agent OSI (objets gérés) sous forme d'un système SDL'92. Elle fournit des squelettes SDL'92 à partir des spécifications GDMO et GRM[33].

2. Spécification des éléments génériques de l'agent: cette phase comprend la modélisation des services CMIS ainsi que celle de l'arbre de nommage.

3. Simulation: la simulation se fait en chargeant les spécifications SDL'92 de chacune de ces deux phases dans un simulateur SDL'92 (ObjectGeode), et puis en fournissant un scénario de tests permettant de vérifier les propriétés du système.

Notre intérêt s'est particulièrement porté sur les deux premières phases. De cette étude nous avons noté les points suivants:

- Traduction d'un objet géré par un processus SDL'92 : les attributs de la classe sont déclarés en tant que variables internes, les actions en tant que procédures distantes, et les notifications en tant que procédures internes. De plus des procédures supplémentaires sont rajoutées afin de donner une interface d'accés aux attributs.

- Traduction des services CMIS : ils sont représentés par des services SDL'92 dans un processus simulant l'agent OSI.

- Modélisation de l'arbre de nommage : il est représenté par un type abstrait sur lequel sont définies les opérations d'ajoût et de suppression des processus représentants les objets gérés de la MIB. Nous noterons alors que l'identification des objets est faite par les PID des processus et non pas par les attributs de nommage des spécifications GDMO.

5.2.2 CORBA vers SDL'92

La proposition de Telelogic que nous avons étudiée, présente une utilisation combinée de CORBA et du langage SDL'92. Deux approches sont alors introduites : l'utilisation de SDL pour implémenter et tester des descriptions CORBA IDL, et l'utilisation de CORBA comme support d'exécution des processus SDL. Nous nous sommes intéressé à la première approche, c'est à dire à la validation d'un agent CORBA par SDL.

Nous avons retrouvé dans cette approche les mêmes concepts que la modélisation SDL'92 des spécifications GDMO :

- La méthode générale consiste à produire, dans un premier temps des squelettes SDL'92 à partir des spécifications CORBA IDL. Ces squelettes sont complétés ensuite par le comportement ;

- Les algorithmes de traduction proposés définissent des correspondances entre une interface IDL - assimilée à une classe GDMO dans les traductions IDL/GDMO du JIDM (cf. 3.3) - et un processus SDL ; les attributs et opérations IDL sont traduits respectivement par des variables internes et des procédures SDL.

33. General Relationship Model

5.3 Le traducteur MOF/SDL'92

5.3.1 Présentation générale

Le traducteur que nous avons réalisé est conçu de la même manière que le traducteur MOF/GDMO, c'est à dire que nous avons défini des correspondances entre les éléments du méta-modèle CIM et les constructeurs du langage SDL'92. Ceci nous permet de construire un squelette SDL'92 à partir d'un fichier de spécifications MOF. Ce squelette nous servira par la suite à créer un agent SDL'92 avec un comportement équivalent à l'agent WBEM implémentant ces mêmes spécifications MOF (figure 5.1).

Pour modéliser cet agent nous avons opéré en deux temps: traduction des objets gérés (i.e. le modèle de l'information), et traduction des services HMMS qui nous permettrons d'accéder aux objets SDL'92 issus des traductions.

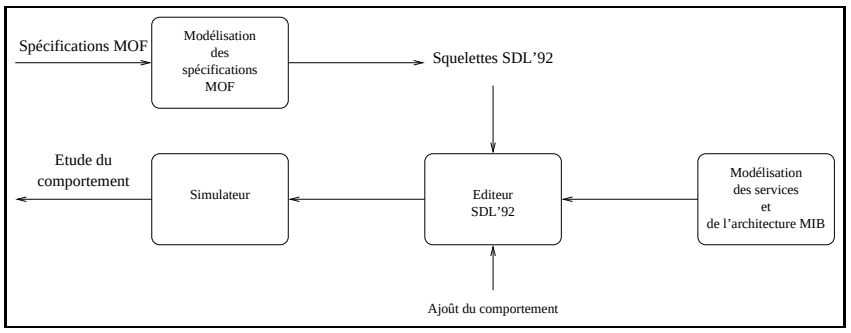

FIG. 5.1 – *Modélisation d'un agent WBEM par un système SDL'92*

5.3.2 Traduction de l'information

Modélisation d'une classe d'objet géré

A une classe d'objet géré MOF, nous faisons correspondre un processus SDL'92. Le processus est alors déclaré au sein d'un paquetage. Le processus héritera des propriétés du processus issu de la traduction de la super-classe MOF si elle existe. Les attributs (*property*) et les méthodes déclarés au sein de cette classe, seront alors déclarés au sein du processus en tant que paramètres, variables ou procédures internes au processus (figure 5.2).

Modélisation d'un attribut

Nous avons traduit un attribut déclaré dans une classe en deux temps (figure 5.2):

- par une variable interne du processus issu de cette classe. Le type de l'attribut sera alors issu de la correspondance entre les types MOF et les types SDL'92.
- par une procédure interne d'accès à l'attribut : cette procédure sera définie à partir de l'analyse des qualifieurs `read` et `write` éventuellement attachés à cet attribut. Si aucun de ces qualifieurs n'est présent avec la valeur `false`, nous déclarons une procédure pour l'accès en écriture (`<attribute-label>_WRITE`), et une deuxième pour l'accès en lecture (`<attribute-label>_READ`)[34].

34. Accés autorisé par défaut dans les spécifications MOF.

```
USE Package_Types;
PACKAGE package_<schema-name>_<class-label>
    PROCESS TYPE <schema-name>_<class-label>
      [ INHERITS <super-class-label>]
        DCL
            [<type-reference> <attribute-label>;]*
      [ EXPORTED PROCEDURE <attribute-label>_ACCESS
        ENDPROCEDURE; ]*
      [ EXPORTED PROCEDURE <method-label>
        ENDPROCEDURE; ]*
      [ /* Qualifiers descriptions */]
    ENDPROCESS TYPE <schema-name>_<class-label>
 ENDPACKAGE package_<schema-name>_<class-label>
```

FIG. 5.2 – *Traduction d'une classe MOF par un processus SDL'92*

Modélisation d'une méthode

La méthode MOF est modélisée par une procédure interne au processus issu de la traduction de la classe déclarant cette méthode. Les paramètres de la procédure seront les paramètres de la méthode MOF. Le type de retour sera déclaré en tant que paramètre de retour avec l'identificateur `<method-label>_return`.

Modélisation d'une association

L'association MOF est représentée dans notre application par un canal de communication entre les processus issus des classes liées par l'association.

Modélisation d'un qualifieur

Dans notre approche, seuls les qualifieurs `read` et `write` sont traduits par des éléments d'information directement exploitables par la machine (voir modélisation d'un attribut).

Les autres qualifieurs MOF, sont présents dans les spécifications SDL'92 sous forme de commentaires.

Nous noterons à cet effet que même le qualifieur `Key` indispensable dans le formalisme MOF pour le nommage des instances est passé en commentaire : en effet, dans l'agent SDL'92, nous nous appuyons sur les PID pour l'identification des instances de processus. Ainsi, lors de la création d'une nouvelle instance d'objet (nouvelle instance de processus), le "gestionnaire" reçoit en réponse le PID du nouveau processus, qu'il utilisera ultérieurement dans ses requêtes.

5.3.3 Traduction des services

Afin de compléter la modélisation d'un agent WBEM par un agent SDL'92, nous devons implanter la traduction les services HMMS par des services SDL'92. Ainsi, chaque appel de service HMMS sera représenté par une activation du service SDL'92 correspondant, et qui se chargera d'appeler le (les) pocessus cible(s) pour simuler la requête du gestionnaire.

5.4 Les apports de la modélisation

Cette modélisation par le langage SDL'92 d'un fichier de spécifications MOF présente un intérêt immédiat pour la validation formelle de ces spécifications. Elle permettera en effet, de simuler le comportement de l'agent WBEM décrit par ces spécifications. Ainsi, nous pourrons prouver des propriétés sur cet agent, et tester la cohérence des spécifications MOF.

D'autre part, nous proposons d'utiliser cette modélisation dans une autre approche de validation. Pour cela, elle sera combinée avec l'application de modélisation des spécifications GDMO de l'outil MODERES ainsi qu'avec le traducteur MOF/GDMO qui a fait l'objet de notre stage.

L'objectif de cette approche est d'étudier le comportement du système de gestion incluant le gestionnaire OSI, l'agent d'adaptation issu de notre approche d'intégration et l'agent WBEM. La méthode que nous proposons pour cette appliation est décrite par la figure 5.3).

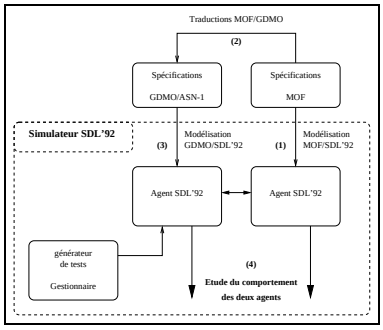

FIG. 5.3 – *Méthode de validation des traductions MOF/GDMO*

1. Construction d'un agent SDL'92 à partir des spécifications MOF d'un agent WBEM ;
2. Génération des formulaires GDMO correspondants à ces mêmes spécifications MOF à l'aide du traducteur MOF/GDMO ;
3. Construction d'un agent SDL'92 à partir de ces spécifications GDMO à l'aide de l'outil MODERES ;
4. Etude du comportement de chacun de ces deux agents SDL'92 à l'aide d'un scénario de tests simulant des requêtes gestionnaires.

5.5 Conclusion

Dans ce chapitre nous avons proposé une modélisation des spécifications MOF par un système SDL'92. Cette modélisation à un intérêt double: valider les agents WBEM décrits par ces spécifications, et valider l'intégration WBEM/OSI que nous avons proposée dans le chapitre précédant.

En effet, nous pouvons maintenant charger le système SDL'92 issu de cette modélisation dans le simulateur ObjectGeode, ceci nous permettra de vérifier les propriétés soit de l'agent WBEM soit de l'agent d'adaptation. Mais pour cela nous devons compléter l'implantation des processus simulant l'agent puis définir des scénario de tests adéquats.

Chapitre 6

Conclusions et perspectives

Dans le cadre des activités de gestion de réseaux, plusieurs approches sont actuellement utilisées suivant les caractéristiques des composants des réseaux. Cette diversité dans les réseaux et dans les approches proposées pour leur administration pose un problème de taille: l'interopérabilité dans le but d'une supervision homogène. De nombreux efforts d'intégration ont été accomplis, mais l'apparition continue de nouvelles approches de gestion pousse les chercheurs du domaine à poursuivre ces travaux.

L'étude que nous avons effectuée lors de ce stage ce situe dans ce contexte: l'intégration de la nouvelle approche WBEM apparue en 1996, au sein de la gestion OSI.

Dans un premier temps nous avons étudié les caractéristiques de chacune des approches afin d'en dégager les différences principales. Puis nous avons étudié les propositions d'intégration dans le but d'identifier les concepts et les techniques confirmés dans le domaine.

Ces deux études préliminaires, nous ont permis alors de concevoir un modèle pour l'intégration WBEM/OSI. Dans notre proposition nous avons opté pour une approche basée sur un agent d'adaptation intercalé entre un gestionnaire OSI et un agent WBEM. Nous avons ensuite défini les composants d'un tel système. En premier lieu, Nous avons implanté un traducteur des modèles de l'information, afin de donner au gestionnaire une vision OSI sur les éléments de la base de gestion de l'agent WBEM. Puis nous avons établi des algorithmes de transfert de l'information de gestion, en termes de services, entre les différentes entités du nouveau système.

Suite à ce travail de conception, nous nous sommes intéressés aux possibilités de validation de notre modèle. Nous avons alors étudié certaines approches de validation par le langage formel SDL'92. De cette étude nous avons été en mesure de construire un système SDL'92 modélisant un agent WBEM. Ce système nous permettra dans un premier temps d'étudier le comportement d'un agent WBEM seul, puis de le comparer avec la modélisation SDL'92 (implantée dans l'outil MODERES développé au sein de l'équipe RESEDAS) de l'agent OSI issu de notre approche d'intégration.

Dans un avenir proche, nous allons compléter l'implantation de l'agent d'adaption par un traducteur inverse, et ce afin de pouvoir administrer les deux types de réseaux OSI et WBEM par n'importe quel gestionnaire supportant l'une ou l'autre des approches.

A long terme, nous envisageons de formaliser et valider l'intégration globale, c'est à dire valider le nouveau système de gestion composé des différents éléments OSI et WBEM.

Bibliographie

[1] S. Abeck, A. Clemm, and U. Hollberg. Simply Open Network Management ; An Approach for the Integration of SNMP into OSI Management Concepts. *Integrated Network Mangement*, 1993.

[2] A.Clemm. SNMP and TL-1: Simply integrating management of legacy systems? *Integrated Network Mangement*, 1997.

[3] M. Bjorkander. Using sdl to develop corba object implementations. *Telelogic*, 1997. Technical Paper.

[4] CCITT-Z-100-92. Specification and description langage (sdl) 1992, 1992.

[5] CCITT.X.700. Technologies de l'information - interconnexion de systèmes ouverts: Cadre de gestion pour l'interconnexion de systèmes ouverts pour les applications du ccitt, 1992.

[6] CCITT.X.701. Technologies de l'information - interconnexion de systèmes ouverts: Aperçu général de la gestion de systèmes, 1992.

[7] CCITT.X.710. Data communication networks - open systems interconnexion (osi) management - common management information service definition for ccitt applications, 1991.

[8] CCITT.X.720. Technologies de l'information - interconnexion de systèmes ouverts - structure des informations de gestion: Modèle de l'information de gestion, 1992.

[9] CCITT.X.722. Technologies de l'information - interconnexion de systèmes ouverts - structure des informations de gestion: Directives pour la définition des objets gérés, 1992.

[10] O. Festor. The gdmo and grm modules semantic checker of the moderes java toolkit. *INRIA Lorraine*, July 1997. Technical Report no. 0205.

[11] O. Festor. Moderes java: Architecture and core packages. *INRIA Lorraine*, May 1997. Technical Report no. 0205.

[12] O. Festor. The managed object format specification parser of the moderes java toolkit. *INRIA Lorraine*, Mars 1998. Technical Report no. 0218.

[13] Olivier Festor. *Formalisation du comportement des objets gérés dans le cadre du modèle OSI.* Université Henri Poincaré - Nancy I, Octobre 1994. Mémoire de thèse.

[14] Desktop Management Task Force. *Common Information Model (CIM) Specification - Version 1.1.* 1997.

[15] IIMC. *ISO/CCITT to Internet Management Proxy.* February 1994.

[16] IIMC. *Translation of Internet MIBs to ISO/CITT GDMO MIBs* . February 1994.

[17] ITU-T.X.680. Data networks and open system communication - osi networking and system aspects - abstract syntax notation one, January 1998.

[18] M. Bolsinger J. Horowitz. *Accessing TMN Through Web-Based Entreprise Mangement.* VERTEL, February 1997. White paper.

[19] JIDM. *CMIP/SNMP TO OMG IDL TRANSLATION* . November 1996.

[20] P. Kalyanasundarm and A. S.Sethi. An Application Gateway Design for OSI-Internet Management. *Integrated Network Mangement*, 1993.

[21] S. Mazumdar, S. Brady, and D. W.Levine. Design of Protocol Independant Management Agent to Support SNMP and CMIP Queries. *Integrated Network Mangement*, 1993.

[22] Kevin McCarthy, George Pavlou, and Saleem Bhatti. Exploiting the power of OSI Management for the control of SNMP-capable ressources using generic application level gateways. *Integrated Network Management*, 1992.

[23] David T. Perkins. *User's guide for SMICng, The SNMP MIB Information Compiler.* http://smurfland.cit.buffalo.edu/ftp/pub/utils/SNMP/smicng/docs/, November 1994.

[24] Anne-Isabelle Rivière and Michelle Sibilla. Management Information Models Integration: From Existing Approaches to new Unifuing Guidelines.

[25] N. Soukouti and U. Hollberg. Join Inter Domain Management: CORBA,CMIP and SNMP. *Integrated Network Mangement*, 1997.

[26] S. Tata. *Modélisation d'agents OSI en SDL'92.* Université H.Poincaré - Nancy I, Juillet 1997. Mémoire de DEA.

[27] S. Tata, L. Andrey, and O. Festor. A practical experience on validating gdmo-based information models with sdl'88 and '92. *8th SDL Forum, Evry France*, September 1997.

[28] S. Todd. *Draft HMMP Events.* Microsoft Corporation, May 1997. Not yet assigned.

[29] S. Todd. *Draft HMMP Mandatory Schema.* Microsoft Corporation, May 1997. Not yet assigned.

[30] S. Todd. *Draft HMMP Overview.* Microsoft Corporation, May 1997. Not yet assigned.

[31] S. Todd. *Draft HMMP Protocol Encoding.* Microsoft Corporation, May 1997. Not yet assigned.

[32] S. Todd. *Draft HMMP Protocol Operations.* Microsoft Corporation, May 1997. Not yet assigned.

[33] S. Todd. *Draft HMMP Query Definitions.* Microsoft Corporation, May 1997. Not yet assigned.

[34] S. Todd. *Draft HMMP Security and Administration.* Microsoft Corporation, May 1997. Not yet assigned.

[35] Marshall T.Rose. *The Simple Book: An Introduction to Management of TCP/IP-based Internets.* Pentice Hall, 1991.

Glossaire

ASN-1 : Abstract Syntax Notation One

CIM : Common Information Model
CMIP : Common Mnagement Information Protocol
CMIS : Common Management Information Service
CORBA : Common Object Request Broker Architecture

DMTF : Desktop Management Task Force

GDMO : Guidelines for the Definition of Managed Objects
GRM : General Relationship Model

HMMP : Hyper-Media Management Protocol
HMMS : Hyper-Media Management Services
HMQL : Hyper-Media Query Langage

ISO : International Standardization Organization

MIB : Management Information Base
MOF : Managed Object Format

OSI : Open Systems Interconnection

SDL : Specification and Description Langage
SNMP : Simple Network Management Protocol

WBEM : Web-Based Enterprise Management

Annexe A

Le langage GDMO

Le but de cette annexe est de donner un aperçu général de certains des formulaires du langage GDMO. Une description plus détaillée de la totalité des formulaires pourra être trouvée dans [9].

A.1 Le formulaire de module

Le formulaire de module permet de regrouper un ensemble de spécifications d'attributs, actions et notifications sous une même étiquette. Ainsi, la déclaration d'un module dans une classe permet de spécifier les propriétés de celle-ci. La spécification du formulaire de module est donnée dans la figure A.1.

```
<package-label> PACKAGE
    [ BEHAVIOUR <behaviour-label> [,<behaviour-label>]*;
    ]
    [ ATTRIBUTES <attribute-label>  property-list [<parameter-label>]*
                  [,<attribute-label>  property-list [<parameter-label>]*]*;
    ]
    [ ATTRIBUTE GROUPS <group-label> [<attribute-label>]*
                        [,<group-label> [<attribute-label>]*]*;
    ]
    [ ACTIONS <action-label> [<parameter-label>]*
              [,<action-label> [<parameter-label>]*]*;
    ]
    [ NOTIFICATIONS <label-notification> [<parameter-label>]*
                    [,<label-notification> [<parameter-label>]*]*;
    ]
[ REGISTERED AS <object-identifier>];
```

FIG. A.1 – *Formulaire de module GDMO*

La clause **ATTRIBUTES** permet de référencer les attributs du module ainsi que les propriétés que peut avoir chacun de ces attributs. Les propriétés définissent les opérations autorisées sur ces attributs (exemples: GET, REPLACE, ADD, REMOVE) ou encore les opérations d'initialisation (exemples: REPLACE-WITH-DEFAULT, INITIAL VALUE, DEFAULT VALUE).

La clause **ATTRIBUTE GROUPS** permet de regrouper sous une même étiquette plusieurs attributs.

La clause **ACTIONS** permet de référencer une liste de définitions d'actions. De même la clause **NOTIFICATIONS** permet de référencer une ou plusieurs notifications d'événements.

A chacun des éléments (attributs, actions ou notifications) peut être associé un ou plusieurs paramètres. Ils servent à définir généralement le traitement d'erreur lors d'une défaillance d'une opération sur l'élément.

A.2 Le formulaire d'attribut

```
<attribute-label> ATTRIBUTE
      DERIVED FROM <attribute-label> | WITH ATTRIBUTE SYN-
TAX type-reference~;
    [ MATCHES FOR qualifier [, qualifier]*;
    ]
    [ BEHAVIOUR <behaviour-label> [,<behaviour-label>]*;
    ]
    [ PARAMETERS <parameter-label> [,<parameter-label>]*;
    ]
[ REGISTERED AS <object-identifier>];
```

FIG. A.2 – *Formulaire d'attribut GDMO*

La définition d'un attribut peut être effectuée de deux manières: une extension d'un attribut déjà défini (**DERIVED FROM**), ou par un type ASN-1 (**WITH ATTRIBUTE SYNTAX**).

La clause **MATCHES FOR** permet de spécifier les opérations de comparaison autorisées sur l'attribut (EQUALITY, ORDERING, SUBSTRINGS, SET-COMPARAISON, SET-INTERSECTION).

Le formulaire d'action

```
<action-label> ACTION
    [ BEHAVIOUR <behaviour-label> [,<behaviour-label>]*;
    ]
    [ MODE CONFIRMED;
    ]
    [ PARAMETERS <parameter-label> [,<parameter-label>]*;
    ]
    [ WITH INFORMATION SYNTAX type-referenced;
    ]
    [ WITH REPLY SYNTAX type-referenced;
    ]
 REGISTERED AS <object-identifier>;
```

FIG. A.3 – *Formulaire d'action GDMO*

La clause **WITH ATTRIBUTE SYNTAX** permet de spécifier le paramètre d'appel de l'action. Elle fait référence au type ASN-1 de ce paramètre. Si l'action nécessite plusieurs paramètres d'appel, le type référencé sera la SEQUENCE des types de ces paramètres.

La clause **WITH REPLY SYNTAX** permet de spécifier les paramètres de retour de l'action.

A.3 Le formulaire de corrélation de nom

```
<name-binding-label> NAME BINDING
    SUBORDINATE OBJECT CLASS <class-label> [ AND SUBCLASSES];
    NAMED BY SUPERIOR OBJECT CLASS <class-label> [ AND SUBCLASSES];
    WITH ATTRIBUTE <attribute-label>;
   [  BEHAVIOUR <behaviour-label> [,<behaviour-label>]*;]
   [  CREATE [ create-modifier [, create-modifier]] [<parameter-label>]*;]
   [  DELETE [ delete-modifier] [<parameter-label>]*;]
 REGISTERED AS <object-identifier>
```

FIG. A.4 – *Formulaire de corrélation de nom GDMO*

La corrélation de noms définit un lien d'inclusion entre deux classes du modèle ce qui permet de construire une hièrarchie dans la MIB OSI: dans l'arbre représentant la MIB, les instances de la classe subordonnée formeront le sous-arbre des instances de la classe supérieure.

Ce lien est défini par la classe appelante **NAMED BY SUPERIOR OBJECT CLASS**, la classe subordonnée **SUBORDINATE OBJECT CLASS**, et par un attribut de nommage **WITH ATTRIBUTE**. Cet attribut sert à distinguer les instances d'un même niveau hièrarchique dans un sous-arbre de la MIB, et donc à nommer toute instance de la MIB.

L'option **AND SUBCLASSES** permet d'étendre la corrélation de nom aux sous-classes des classes *superior* et *subordinate*.

Annexe B

Les modules de base pour les traductions MOF/GDMO

B.1 Les formulaires GDMO

```
"BasicModule":MOF_MOC MANAGED OBJECT CLASS
 REGISTERED AS { BasicModule classes 1 }

"BasicModule":MOF_MetaMOC MANAGED OBJECT CLASS
     DERIVED FROM "BasicModule":MOF_MOC ;
     CHARACTERIZED BY "BasicModule":MOF_MetaMOCPackage ;
 REGISTERED AS { BasicModule classes 2 };

"BasicModule":CIM_Agent MANAGED OBJECT CLASS
     DERIVED FROM "BasicModule":MOF_MOC ;
     CHARACTERIZED BY "BasicModule":CIM_AgentPackage;
 REGISTERED AS { BasicModule classes 3 }

"BasicModule":MOF_MetaMOCPackage PACKAGE
     ATTRIBUTES
      "BasicModule":ClassName GET_REPLACE,
      "BasicModule":SuperClassName GET-REPLACE,
      "BasicModule":PropertyBufferList GET-REPLACE,
      "BasicModule":MethodBufferList GET-REPLACE,
      "BasicModule":QualifierList GET-REPLACE;
 REGISTERED AS { BasicModule packages 2 };

"BasicModule":CIM_AgentPackage PACKAGE
     ATTRIBUTES
      "BasicModule":IPAdress GET-REPLACE,
      "BasicModule":State GET-REPLACE;
 REGISTERED AS { BasicModule packages 3 };

"BasicModule":ClassName ATTRIBUTE
     WITH ATTRIBUTE SYNTAX GRAPHIC STRING;
 REGISTERED AS { BasicModule attributes 1 };

"BasicModule":SuperClassName ATTRIBUTE
```

```
    WITH ATTRIBUTE SYNTAX GRAPHIC STRING;
 REGISTERED AS { BasicModule attributes 1 };

"BasicModule":PropertyBufferList ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SEQUENCE OF BasicTypes.PropertyBuffer;
 REGISTERED AS { BasicModule attributes 2 };

"BasicModule":MethodBufferList ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SEQUENCE OF BasicTypes.MethodBuffer;
 REGISTERED AS { BasicModule attributes 3 };

"BasicModule":QualfierList ATTRIBUTE
    WITH ATTRIBUTE SYNTAX SEQUENCE OF BasicTypes.Qualifier;
 REGISTERED AS { BasicModule attributes 4 };

"BasicModule":IPAdress ATTRIBUTE
    WITH ATTRIBUTE SYNTAX GRAPHIC STRING;
 REGISTERED AS { BasicModule attributes 5 }

"BasicModule":State ATTRIBUTE
 WITH ATTRIBUTE SYNTAX BOOLEAN;
 REGISTERED AS { BasicModule attributes 6 }
```

B.2 Les types ASN-1

```
PropertyBuffer ::= SEQUENCE {
                     Name GRAPHIC STRING,
                     DataType GRAPHIC STRING,
                     DefaultValue ANY,
                     QualifierList SEQUENCE OF Qualifier
                     }

MethodBuffer ::= SEQUENCE {
                   Name GRAPHIC STRING,
                   ReturnType GRAPHIC STRING,
                   ParameterList SEQUENCE OF Parameter
                   }

Parameter ::= SEQUENCE {
                DataType GRAPHIC STRING
                }

Qualifier ::= SEQUENCE {
                Name GRAPHIC STRING,
                DataType GRAPHIC STRING,
                Value ANY
                }
```

Annexe C

Un exemple de traduction MOF/GDMO

C.1 Les spécifications MOF

```
[Abstract(true),
Desciption(“The ManagedSystemElement is...”),
Schema(“Core”)]
 class ManagedSystemElement
{
[Description(“The Name property...”),
Write(false)]
    string Name;
[Description(“A boolean value...”)]
    bool Installed;
};

[Description(“Any component of a system...”),
Schema(“Core”)]
 class PhysicalElement:ManagedSystemElement
{
[Description(“A manufacturer-allocated number...”),
Key(true)]
    string SerialNumber;
};
```

C.2 Les spécifications GDMO

```
MODULE “Core”;

“Core”:MOF_Core_ManagedSystemElementClass MANAGED OBJECT CLASS
 DERIVED FROM “BasicModule”:MOF_MOC;
 CHARACTERIZED BY “Core”:MOF_Core_ManagedSystemElementPackage;
 BEHAVIOUR “Core”:MOF_Core_ManagedSystemElementBehaviourPackage;
```

```
 REGISTERED AS { schemas(2) classes 1 };

"Core":MOF_Core_PhysicalElementClass MANAGED OBJECT CLASS
 DERIVED FROM "Core":MOF_Core_ManagedSystemElementClass;
 CHARACTERIZED BY "Core":MOF_Core_PhysicalElementPackage;
 BEHAVIOUR "Core":MOF_Core_PhysicalElementBehaviourPackage;
 REGISTERED AS { schemas(2) classes 2 };

"Core":MOF_Core_ManagedSystemElementPackage PACKAGE
 ATTRIBUTES
    "Core":MOF_Core_ManagedSystemElement_NameAttribute GET,
    "Core":MOF_Core_ManagedSystemElement_InstalledAttribute GET-REPLACE;
 REGISTERED AS { schemas(2) packages 1 };

"Core":MOF_Core_PhysicalElementPackage PACKAGE
 ATTRIBUTES
    "Core":MOF_Core_PhysicalElement_SerialNumberAttribute GET-REPLACE;
 REGISTERED AS { schemas(2) packages 2 };

"Core":MOF_Core_ManagedSystemElement_NameAttribute ATTRIBUTE
 WITH ATTRIBUTE SYNTAX GraphicString;
 BEHAVIOUR "Core":MOF_Core_ManagedSystemElement_NameAttributeBehaviourPAckage;
 REGISTERED AS { schemas(2) attributes 1 };

"Core":MOF_Core_ManagedSystemElement_InstalledAttribute ATTRIBUTE
 WITH ATTRIBUTE SYNTAX BOOLEAN;
 BEHAVIOUR "Core":MOF_Core_ManagedSystemElement_InstalledAttributeBehaviourPAckage;
 REGISTERED AS { schemas(2) attributes 2 };

"Core":MOF_Core_PhysicalElement_SerialNumberAttribute ATTRIBUTE
 WITH ATTRIBUTE SYNTAX GraphicString;
 BEHAVIOUR "Core":MOF_Core_PhysicalElement_SerialNumberAttributeBehaviourPAckage;
 REGISTERED AS { schemas(2) attributes 3 };

"Core":MOF_Core_PhysicalElementNameBinding Name Binding
 SUBORDINATE OBJECT CLASS "Core"_MOF_Core_PhysicalElementClass;
 NAMED BY SUPERIOR OBJECT CLASS "BasicModule":MOF_Namespace;
 WITH ATTRIBUTE "Core":MOF_Core_ManagedSystemElement_SerialNumberAttribute;
 REGISTERED AS { schemas(2) name-binding 2 }

"Core":MOF_Core_ManagedSystemElementBehaviourPackage BEHAVIOUR
 DEFINED AS "Description: The ManagedSystemElment is ...";
```

Annexe D

Algorithmes de traduction des services CMIS/HMMS

D.1 Traduction du service M-GET

D.1.1 Spécification du service

Nom du paramètre	Req/Ind	Rsp/Cnf
Invoke identifier	M	M
Linked identifier	-	C
Base object class	M	-
Base object instance	M	-
Scope	U	-
Filter	U	-
AccessControl	U	-
Synchronization	U	-
Attribute Identifier List	U	-
Managed object class	-	C
Managed object instance	-	C
Current time	-	U
Attribute list	-	C
Errors	-	C

M : obligatoire
U : optionnel (au choix de l'utilisateur)
C : conditionnel

TAB. D.1 – *Les paramètres du service M-GET*

Tel que nous l'avons défini précédemment, le service M-GET simple peut se traduire par un appel *Put Instance*. Dans cette section nous donnons un exemple de cette traduction avec la construction de tous les paramètres de l'appel *Put Instance* qui sera envoyée vers l'agent, ainsi que les paramètre de la réponse qui sera envoyé au gestionnaire.

Nom du paramètre	Req/Ind	Rsp/Cnf
Request identifier	M	M(=)
Security level	-	C
Access token	M	-
MD5 signature	M	-
Instance path	U	-
Preferred locale	U	-
Properties	U	-
Qualifiers	U	-
Error info	U	-
Instance	-	C
Error status	-	C
Error object	-	U

M : obligatoire
U : optionnel (au choix de l'utilisateur)
C : conditionnel

TAB. D.2 – *Les paramètres du service Get Instance*

D.1.2 Requête envoyée vers l'agent WBEM

Les paramètres de cette requête *Put Instance* sont déduites essentiellement des paramètres de la requête M-GET recue par l'agent d'adaptation :

- **Request identifier :** c'est l'identificateur de requête. Une tâche annexe de l'application se charge de le déterminer et faire le lien avec les identificateurs de requêtes du gestionnaire ;
- **Security level :** c'est l'indicateur du niveau de sécurité demandé pour la transaction. Il est déterminé par le gestionnaire à l'initialisation de l'agent d'adaptation. Il en est de même pour les champs **Access token** et **MD5 signature** qui servent pour les transactions d'un niveau de sécurité élevé ;
- **Instance path :** c'est l'identificateur de l'instance objet de la requête. Il est composé par le chemin de l'espace de nommage, suivi du nom de la classe et des valeurs des attributs clefs de l'instance. Le nom de la classe est tiré du champs **Base object class** de la requête M-GET alors que le chemin de l'espace de nommage et la valeurs des attributs clefs sont tirés du champs **Base object instance** suivant les règles que nous avons défini pour les traductions de l'organisation de la MIB ;
- **Preferred locale :** il permet de demander un format particulier pour les réponses. Il est fixé en phase d'initialisation de l'agent ;
- **Properties :** Il permet de désigné les attributs dont le gestionnaire cherche à connaitre les valeurs. Il est directement déduit du paramètre **Attribute identifier list** de la requête M-GET ;
- **Qualifiers :** nous avons traduit les qualifieurs par d'autres formes d'informations (attributs, propriétés d'attributs, comportement,...), ce champs ne sera dons pas rempli pour l'appel *Put Instance* ;

– **Error info :** Il permet de demander au serveur de lui fournir en cas d'erreur, des informations supplémentaires sur la cause de l'erreur. Ce champs est fixé par le gestionnaire en phase d'initialisation.

D.2 Les autres services CMIS

D.2.1 Le service M-CANCEL-GET

Le service M-CANCEL-GET permet d'interrompre une requête multi-réponses. Ainsi, elle ne sera lancée que dans le cas d'un précédant appel de service *Enumerate Instances*[35] .

La requête sera traduite par un appel *Cancel Enumeration* avec en paramètre l'identificateur de l'appel *Enumerate Instances* lancé par l'agent d'adaptation. Pour faciliter une telle opération, nous gardons en mémoire l'historique des correspondances entre les services lancés par l'agent d'adaptation et les services CMIS lancés par le gestionnaire OSI.

D.2.2 Le service M-SET

Le service M-SET simple

Le service M-SET permet de modifier des attributs d'objets déjà existant dans la MIB. Dans le protocole HMMS, la modification d'objets est considérée comme un cas particulier du service *Put Instance*: seul un paramètre dans l'appel du service permet de faire la différence.

Ainsi le service M-SET dans sa forme simple sera traduit d'une manière similaire au service M-CREATE avec le paramètre `Modify` ainsi que les nouvelles valeurs des attributs de l'objet ciblé.

Le service M-SET avec portée

Le service M-SET avec portée permet de modifier tout un ensemble d'objets de la même classe appartenant à un sous-arbre particulier de la MIB. Les seuls services permettant de modifier des objets offerts par le protocole HMMS sont les services *Put Instance* et *Put Class* avec le paramètre *Modify*.

Ainsi pour traduire un appel du service M-SET avec portée nous devons récupérer tous les objets de la même classe dans le sous arbre (c'est à dire tous les objets contenus dans l'espace de nommage), ce qui revient à effectuer un appel *Enumerate Instances* pour les instances de la classe désignée, puis nous lancons la demande de modification sur chacun des objets retenus.

Le service M-SET avec portée et filtrage

La traduction se fait de manière analogue que celle du service avec un simple portée: seules certaines instances satisfaisant aux critères du filtre feront l'objet de la modification par l'appel *Put Instance* avec le champs `Modify`.

35. Le service *Enumerate Classes* se fait pendant la phase de mise à jour et donc reste invisible pour le gestionnaire

Résumé

Dans le cadre de la gestion de réseaux hétérogènes, il est indispensable d'assurer l'interopérabilité entre les différentes approches de gestion implantées dans chaque sous-réseau. Plutôt que de développer un modèle unique de gestion, les spécialistes se sont alors penchés sur l'intégration des approches existantes.

Dans ce mémoire, nous proposons l'intégration de la gestion WBEM au sein du modèle OSI, par l'intermédiaire d'un agent d'adaptation intercalé entre un gestionnaire OSI et un agent WBEM.

Dans un premier temps, nous avons définit des algorithmes de traduction des modèles de l'information. L'objectif de ce travail est de donner au gestionnaire une vision OSI sur les ressources gérées par l'agent WBEM. Dans un deuxième temps , nous avons développé des algorithmes pour transfert des requêtes du gestionnaire OSI vers l'agent WBEM, à travers l'agent d'adaptation.

La dernière partie de ce rapport traîte de la validation d'un tel système de gestion incluant l'agent d'adaptation. A cet effet, nous proposons une modélisation par le langage SDL'92 d'un agent WBEM, ce qui permettra d'étudier le comportement de ce dernier.

Mots-clé : Gestion de réseaux, Agent d'adaptation, WBEM, MOF, OSI, GDMO, SDL

Abstract

Within the framework of the management of heterogeneous network, it is essentiel to ensure interoperability between the various approaches of management implemented in each subnetwork. Rather than to develop a single model of management, managers are interested in integrating the existing approaches.

In this report, we propose the integration of WBEM management within the OSI model, using a gateway between an OSI manager and a WBEM agent.

Initially, we have defined a mapping algorithms for the information models. The aim of this work is to give to the manager an OSI vision on the network elements managed by the WBEM agent. Then, we heve developed algorithms for transferring requests through the gateway, from the OSI manager towards yhe WBEM agent.

The last part of this report deals with the validation of such a management system including the gateway. To this end, we propose the use of SDL'92 langage for modelling the WBEM agent behaviour.

Keywords: Network management, Gateway, WBEM, MOF, OSI, GDMO, SDL