Représentation des spécifications CIM dans un serveur d'annuaire LDAP

Olivier Festor, Nizar Ben Youssef

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

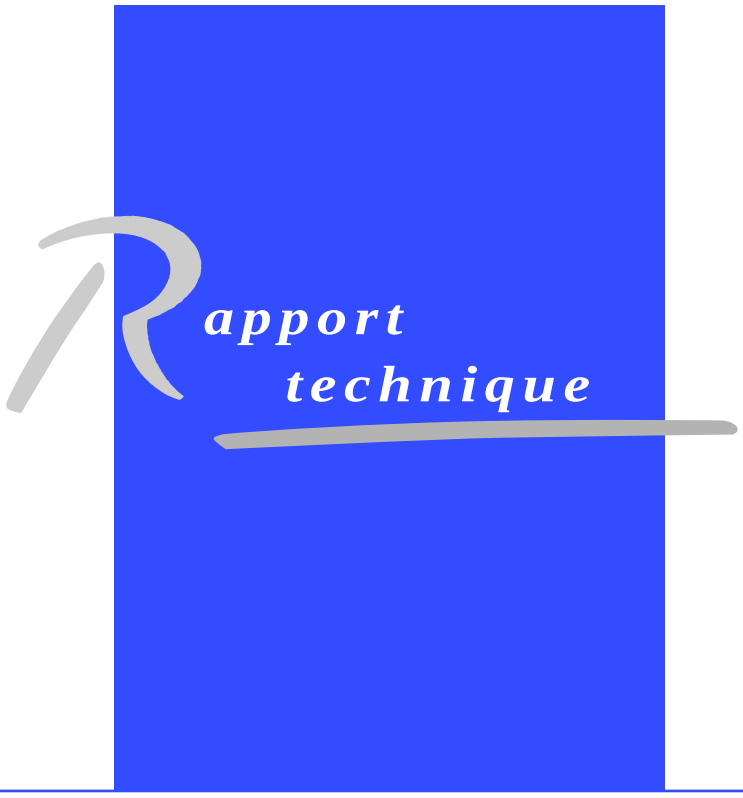
Représentation des spécifications CIM dans un serveur d'annuaire LDAP

Olivier Festor, Nizar Ben Youssef

N° 0250

24 juillet 2001

THÈME 1



ISSN 0249-0803

Représentation des spécifications CIM dans un serveur d'annuaire LDAP

Olivier Festor, Nizar Ben Youssef

Thème 1 — Réseaux et systèmes
Projet RÉSÈDAS

Rapport technique n° 0250 — 24 juillet 2001 — pages

Résumé : Ce rapport présente une étude sur le stockage des modèles de l'information CIM (Common Information Model) dans un serveur d'annuaire LDAP. Il propose un schéma permettant de représenter toute modélisation CIM dans un annuaire LDAP. Il décrit aussi l'implémentation qui a été réalisée et qui est disponible dans la distribution actuelle de MODERES.

Mots-clés : WBEM, CIM, Information de gestion, LDAP, Services d'annuaires

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

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Representation of CIM specifications whithin a LDAP directory

Abstract: This report presents our approach to store CIM (Comon Information Model) specifications within a LDAP directory. It proposes a schema for the representation of any CIM model within the directory. It also describes the implementation provided within the last distribution of the MODERES toolkit.

Key-words: WBEM, CIM, Management Information, LDAP, Directories

Table des matières

1 Introduction

Dans le monde de la gestion des réseaux, le partage des informations entre un gestionnaire et un agent est primordial. En effet, pour pouvoir effectuer ses opérations de gestion, le gestionnaire se doit de connaître les modèles des informations maintenus par ses agents, i.e. les structures modélisant les éléments gérés par l'agent. Traditionnellement, les modèles de l'information sont stockés localement par les applications dans une phase d'initialisation, limitant ainsi le domaine géré aux seuls éléments dont les modélisations ont été préalablement chargées, et ce malgré l'existance de normes offrant un service d'accès dynamique aux modèles de l'information.

Les services d'annuaires (*directory services*), longtemps utilisés de manière propriétaire pour le partage des données entre applications, offrent depuis l'avènement du protocole LDAP (Lightweight Directory Access Protocol) [Wahl 97b] et des serveurs LDAP sous-jacents, un moyen standardisé et facile à implémenter, pour le stockage et l'accès aux données entre différents types d'applications issues du monde Internet.

Partant de ces deux constats, nous avons mené une étude sur l'implémentation d'un service de partage et d'accès dynamique aux modèles de l'information par l'intermédiaire d'un service d'annuaire LDAP. Ainsi, nous proposons une solution permettant le stockage et l'accès aux modèles CIM (Common Information Model) [DMTF 99b], utilisés dans l'approche de gestion WBEM (Web-Based Enterprise Management) [Festor 00], s'appuyant sur les fonctionnalités offertes par un serveur d'annuaire LDAP. Dans ce rapport, nous présentons les résultats de ce travail effectué dans le cadre du projet `AntaresII` issu du `GIE DYADE`[1], coopération entre `Bull` et l'`INRIA`.

Ce rapport est organisé comme suit. Dans la section 2 nous donnons un rapide apperçu sur les services d'annuaires en général et sur les annuaires LDAP en particulier. La section 3 introduit le modèle que nous avons adopté pour la représentation de l'information CIM dans un serveur LDAP. Dans la section 4, nous détaillons chacun des éléments de cette représentation. La section 5 présente l'implémentation de notre solution. Enfin, dans la section 6 nous donnons les conclusions relatives à ce travail.

2 Présentation de LDAP

Apparus depuis une dizaine d'années, les services d'annuaires (*dircetory services*) normalisés X.500 [X.500 93], n'ont pas eu le succès escompté auprès des industriels de l'Internet, qui se sont souvent basés sur des solutions propriétaires pour leurs besoins de maintient de données. Avec l'avènement du protocole LDAP (Lightweight Directory Access Protocol) standardisé par l'IETF (RFC 1777 [Yeong 95]) et des serveurs sous-jacents, le monde de l'Internet ne cesse de voir de plus en plus d'applications s'appuyer sur ce service d'annuaire pour le stockage et l'accès aux données.

Dans cette section, nous donnons dans un premier temps un apperçu général sur les services d'annuaires. Nous en présentons les composantes principales ainsi que les fonctionnalités génériques. Ensuite nous décrivons de manière plus détaillée le protocole LDAP et les serveurs LDAP sous-jacents.

2.1 Le concept de directory

Les services d'annuaires sont le plus souvent décrits comme des bases de données spécialisées et optimisées pour des accès fréquents en lecture. Ils permettent ainsi de stocker les données statiques des applications, i.e. les données qui ne changent que rarement.

2.1.1 Les composants d'un service d'annuaire

Les services d'annuaires s'appuyent sur des interactions du type client/serveur. Trois entités participent alors à ces interactions tel que le montre la figure 1.

- l'annuaire : représente l'entité dans laquelle sont maintenues les données ;
- le serveur : il se charge de gérer les accès à la base d'annuaire et de répondre aux requêtes sur cette base ;
- le client : c'est l'application qui utilise le service d'annuaire pour le stockage de ses données. La communication entre un client et un serveur se fait exclusivement sous forme de requêtes et de réponses via une interface d'accès propre au service d'annuaire.

1. `http://www.dyade.fr`

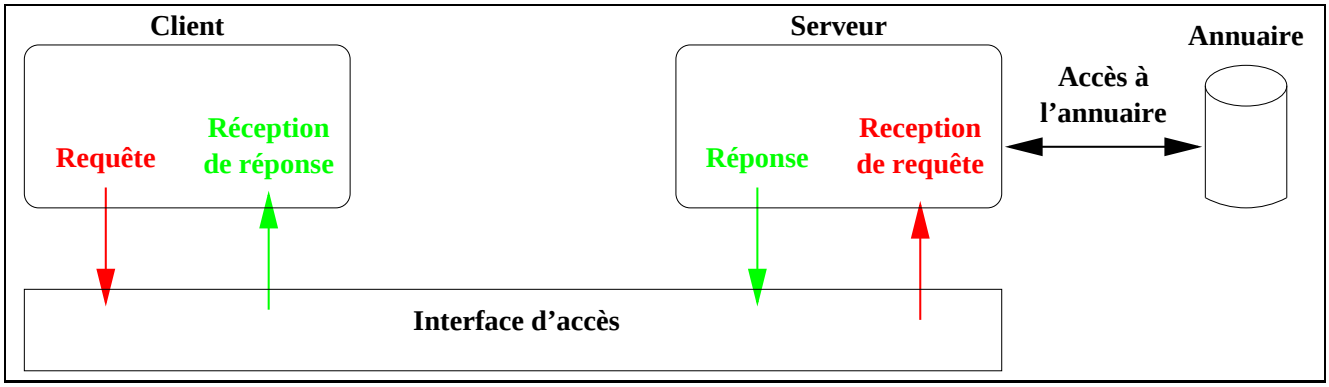


FIG. 1 – *Les intéractions client/serveur*

Les interactions client/serveur décrites ci-dessus nécessitent pour leur réalisation la définition des quatres modèles suivants appelés aussi composantes principales d'un service d'annuaire :

- le modèle de l'information : il définit la structure des données maintenues dans l'annuaire ;
- le modèle de nommage : il définit l'organisation des données dans l'annuaire (relations entre les objets), ainsi que le schéma de nommage des objets ;
- le modèle fonctionnel : il définit l'interface fonctionnelle d'un serveur, i.e. les méthodes et les opérations offertes par le serveur à ses clients ;
- le modèle de sécurité : il définit d'une part les droits d'accès aux objets (authentification et autorisations), ainsi que les besoins de confidentialité (intégrité et protection des données).

2.1.2 Le serveur d'annuaire

Le serveur d'annuaire joue un rôle central dans une implémentation d'un service d'annuaire. Ainsi nous présentons les fonctionnalités principales qu'il doit réaliser :

- le maintient de l'information dans la base de l'annuaire ;
- l'accès à l'information en offrant aux utilisateurs une interface et un protocole d'accès à la base d'annuaire ;
- la gestion de la distribution en offrant un mécanisme de ditribution de l'information sur plusieurs bases d'annuaires de serveurs distants ;
- la gestion de la sécurité et notamment les aspects d'authentification des utilisateurs et la confidentialité des échanges entre un serveur et un client.

2.2 Les annuaires LDAP

Le protocole LDAP (*Lightweight Directory Access Protocol*) est apparu en 1993 pour pallier à la lourdeur du protocole père DAP (*Directory Access Protocol*) issu du monde OSI (norme X.500 [X.500 93]). Facile à implanter et à utiliser, il est devenu depuis, incontournable dans le monde Internet. Dans sa version actuelle, le protocole LDAPv3 est défini à l'aide de 6 RFC :

- RFC 2251 : *Lightweight Directory Access Protocol (v3)* [Wahl 97b] ;
- RFC 2252 : *Lightweith Directory Access Protocol (v3) : Attributes Syntax Definitions* [Wahl 97c] ;
- RFC 2253 : *Lightweigth Directory Access Protocol (v3) : UTF-8 String Representation of Distinguished Names* [Wahl 97d] ;
- RFC 2254 : *The String Representation of LDAP Search Filters* [Howes 97a] ;
- RFC 2255 : *The LDAP URL Format* [Howes 97b] ;
- RFC 2256 : *A Summary of the X.500(96) User Schema for Use With LDAPv3* [Wahl 97a] ;

Dans cette section, nous présentons chacune des composantes principales des annuaires LDAP. L'objet de cette présentation, est de familiariser le lecteur avec le protocole LDAP afin de faciliter la compréhension du travail présenté dans les sections suivantes. Pour plus de détail, nous renvoyons le lecteur aux références citées.

2.2.1 Le modèle de l'information

Les éléments principaux définissants le modèle de l'information, que nous appelerons par la suite éléments du méta-modèle, sont la classe d'objet et l'attribut.

L'**attribut** représente la construction élémentaire de l'annuaire. Il est défini par un nom et une syntaxe. Il permet de stocker dans la base d'annuaire une donnée élémentaire (par exemple le nom d'une personne ou son numéro de téléphone). Les types des attributs LDAP se limitent à un ensemble restreint de syntaxes dont nous donnons celles qui sont le plus fréquemment implémentés par les serveurs LDAP :

- `ces (Case Exact String)` : chaînes de caractères faisant la distinction entre majuscules et miniscules ;
- `cis (Case Ignore String)` : chaînes de caractères ne faisant pas de distinction entre majuscules et minuscules ;
- `tel (Telephone Number)` : représentation d'un numéro de téléphone ;
- `dn (Distinguished Name)` : représentation d'un identificateur d'objet ;
- `Generalized Time` : dates et heures ;
- `Postal Address` : adresses ;
- `bin (Binary)` : données binaires (une image au format `jpeg` par exemple).

Dans la figure 2.2.1 nous rappelons la structure donnée dans la norme [Wahl 97c] pour la définition d'un attribut LDAP.

```
1  ( <attribute-oid>
2      [ NAME <attribute-name> ]
3      [ DESC <attribute-description> ]
4      [ OBSOLETE ]
5      [ SUP <superiorattribute-oid> ]
6      [ EQUALITY <matchingrule-oid> ]
7      [ ORDERING <matchingrule-oid> ]
8      [ SUBSTR <matchingrule-oid> ]
9      [ SYNTAX <syntax-oid> [ { <length> } ] ]
10     [ SINGLE-VALUE ]
11     [ COLLECTIVE ]
12     [ NO-USER-MODIFICATION ]
13     [ USAGE <attribute-usage> ]
14 )
```

FIG. 2 – *La définition d'un attribut LDAP*

Les éléments principaux d'une définition d'attribut LDAP sont l'identificateur absolu (`attribute-oid`, ligne 1); le nom de l'attribut (`attribute-name`, ligne 2); le type de l'attribut donné soit par la clause `SYNTAX` (ligne 9) pour une déclaration explicite ou bien par la clause `SUP` (ligne 5) pour un héritage de type; et un ensemble de règles de comparaisons (*matching rules*) utilisées pour la recherche dans l'annuaire.

La **classe d'objet** permet de stocker dans l'annuaire des objets structurés en regroupant un ensemble d'attributs sous un nom commun. Tout objet stocké dans l'annuaire est appelé **entrée** de l'annuaire et représente une instanciation d'une classe d'objet. La classe d'objet permet de donner la composition d'une entrée en termes d'attributs obligatoires et d'attributs optionnels tel que le montre la figure 3.

La classe d'objet est caractérisée par un identificateur absolu `class-oid` (ligne 1). Elle est aussi très souvent appelée par un nom d'alias plus significatif `class-name` (ligne 2). La modélisation dans LDAP peut être hiérarchisée via un mécanisme d'héritage d'attributs spécifié à l'aide de la clause `SUP` (ligne 5). Les attributs obligatoires de la classe sont spécifiés par la clause `MUST` (ligne 7). Les attributs optionnels sont spécifiés par la clause `MAY` (ligne 8). L'héritage conserve cette propriété des attributs, i.e. les attributs obligatoires dans une super-classe sont aussi obligatoires pour la sous-classe, ceux optionnels dans une super-classe le restent aussi pour la sous-classe.

```
( <class-oid>
    [ NAME <class-name> ]
    [ DESC <class-description> ]
    [ OBSOLETE ]
    [ SUP <superiorclass-oid> [, <superiorclass-oid>]* ]
    [ ABSTRACT | STRUCTURAL | AUXILIARY ]
    [ MUST <attribute-oid> [, <attribute-oid>]* ]
    [ MAY <attribute-oid> [, <attribute-oid>]* ]
)
```

FIG. 3 – *La définition d'une classe d'objet*

Lors de la conception, la définition d'un modèle de l'information, consiste à spécifier les classes et les attributs qui représenterons les données à stocker. Dans le jargon LDAP, un modèle d'information, i.e. l'ensemble des classes et attributs du modèle est appelé *schéma*.

2.2.2 Le modèle de nommage

La base d'annuaire LDAP est structurée de manière arborescente, à l'aide d'un modèle de contenance entre objets. Chaque objet est stocké dans le contexte d'un autre objet quelle qu'en soit la nature, c'est à dire qu'il est stocké dans l'arbre de nommage sous n'importe quel noeud représentant tout autre type d'objet sans aucune contrainte sur la contenance. La définition d'un modèle de nommage comprend alors un choix sur le modèle de contenances entre objets.

Chaque objet est identifié de manière unique par un nom absolu DN (*Distinguished Name*) défini par une liste de couples attribut-valeur représentant le chemin vers l'objet depuis la racine de l'arbre de nommage. Chaque couple attribut-valeur forme un nom relatif RDN (*Relative Distinguished Name*), et correspond à un noeud dans l'arbre de nommage (voir figure 4). La seconde composante du modèle de nommage consiste alors à identifier les attributs permettant d'identifier les objets dans l'arbre de nommage.

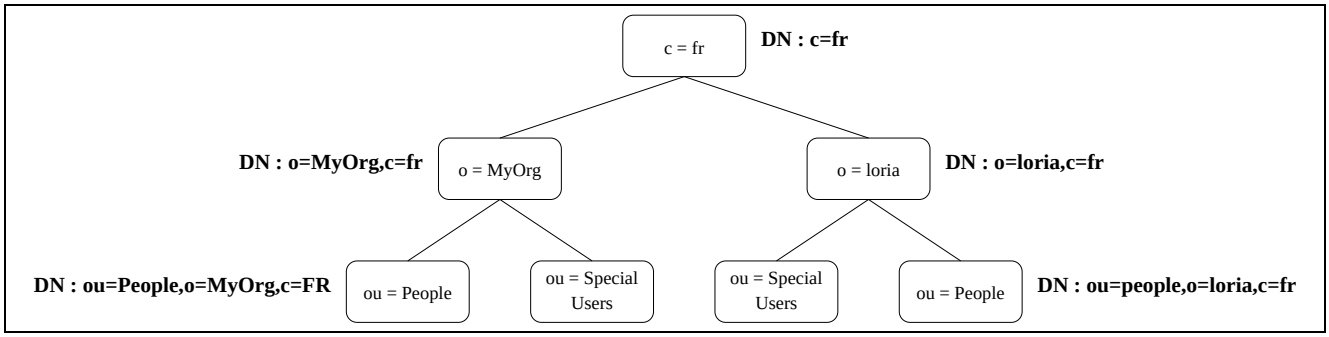


FIG. 4 – *L'arbre de nommage dans un annuaire LDAP*

2.2.3 Le modèle fonctionnel

Le protocole LDAP permet à un client d'accéder à l'information maintenue par un serveur LDAP.

Contrairement à son prédécesseur DAP (Directory Access Protocol) utilisé par les serveurs d'annuaires X.500, LDAP ne définit qu'un nombre restreint d'opérations classées dans trois catégories :

- **la recherche dans l'annuaire** : elle est définie par une unique opération générique **search** permettant de fouiller dans l'arbre de nommage de l'annuaire et de récupérer les données qui y sont enregistrées. A cet effet l'opération définit des mécanismes de portée et de filtrage permettant d'affiner les recherches ;

- **la mise à jour des données** : les opérations de mise à jour au nombre de quatre (`add, delete, modify et modifyDN`) peremettent aux utilisateurs la modification, l'ajoût et la suppression des données (les attributs) stockées dans la base d'annuaire ainsi que la modification des noms des objets (DN) ;
- **l'authentification** : les opérations d'authentifications au nombre de trois (`Bind, Unbind et Abandon`) permettent aux utilisateurs d'établir des sessions de travail avec un serveur LDAP.

En plus de cet ensemble d'opérations, le modèle fonctionnel définit un mécanisme de références pour supporter la distribution dans la base d'annuaire. Les références dans LDAP (*referrals*) sont des entrées particulières dans l'arbre permettant de référencer un arbre (ou sous-arbre) de nommage sur une base d'annuaire distante.

2.2.4 Le modèle de sécurité

Comme tout protocole d'échange de données sur un réseau, LDAP définit des mécanismes permettant de sécuriser ses échanges. Dans sa version 3, le protocole LDAP définit des mécanismes pour supporter l'authentification, l'intégrité et la confidentialité. Le protocole définit alors trois types de sessions correspondants à trois modes d'échange de données :

- Aucune authentification : c'est le mode le plus simple qui ne supporte aucun des mécanismes de sécurité ;
- L'authentification de base : dans ce mode, le client s'identifie auprès du serveur en lui envoyant son identificateur (DN) et son mot de passe. Ces informations passent en clair sur le réseau.
- Le mode SASL (*Simple Authentication and Security Layer*) : rajouté dans la version 3 de LDAP, ce système permet d'établir des sessions LDAP sécurisées. SASL [Myers 97] permet au client et au serveur de négocier l'utilisation d'un mécanisme de sécurisation des échanges. Le mécanisme le plus souvent utilisé pour LDAP est SSL (*Secure Socket Layer*)[2].

En plus de ces mécanismes, certaines implémentations de serveur LDAP supportent un mécanisme d'autorisation sur l'accès aux données (droits d'accès à un sous-arbre de la base d'annuaire), même si ce dernier ne fait pas partie de la dernière version du protocole.

3 Le modèle de la représentation de CIM

Dans cette section, nous discutons du choix du modèle de représentation des spécifications CIM dans une base d'annuaire LDAP. Dans un premier temps nous présentons les différentes stratégies possibles. Ensuite nous motivons notre choix sur le modèle que nous avons implanté. La dernière partie de cette section introduit les modèles de l'information et de nommage conformes à la stratégie choisie.

3.1 La repésentation de l'information

La définition d'un modèle de l'information pour la base d'annuaire d'un serveur LDAP constitue une tâche fondamentale dans le processus de conception des applications LDAP. Concrètement, cette activité consiste à définir le schéma LDAP qui sera utilisé par l'application, c'est à dire spécifier les classes et les attributs définissant les entrées de l'annuaire que l'application aura à manipuler.

Ainsi, pour notre application d'enregistrement de spécifications CIM dans un serveur LDAP, nous avons défini un schéma de représentation des objets CIM par des entrées de l'annuaire LDAP. Pour cette représentation, nous avons identifié deux stratégies possibles :

- la représentation de méta-modèle qui consiste à définir un schéma de représentation du méta-modèle CIM ;
- la représentation de modèle qui consiste à définir un schéma de représentation pour chaque modèle d'information CIM.

3.1.1 La représentation de modèle

La représentation de modèle consiste à générer un schéma LDAP pour chaque modèle CIM, c'est à dire représenter chaque élément d'un modèle CIM (classes et associations) par un ensemble de classes et d'attributs d'un schéma LDAP. Les entrées du serveur LDAP représenterons alors les instances des classes CIM traduites dans le schéma.

2. SSL peut être remplacé dans le futur par son successeur TLS (*Transport Layer Security*)

Pour automatiser une telle stratégie, l'algorithme consiste à définir des corrspondances entre les éléments du méta-modèle CIM (classes, propriétés, méthodes ...) et les éléments du méta-modèle LDAP (classes, attributs ...).

Pour illustrer l'utilisation de cette technique, nous proposons par exemple les algorithmes de correspondances suivants :

- à toute classe CIM on fait correspondre une classe LDAP ;
- à toute propriété CIM on fait correspondre un attribut LDAP ;

Ceci permet de générer automatiquement un ensemble de classes LDAP, i.e. un schéma, correspondant à un modèle CIM donné.

Cette stratégie a été "manuellement" utilisée dans le cadre de l'approche DEN [Judd 98] : certaines classes CIM du schéma de base (*Core Schema* [DMTF 98a]) ont été représentées par des classes LDAP [Moore 00], afin d'utiliser les fonctionnalités d'un serveur LDAP pour des besoins de gestion de services.

3.1.2 La représentation de méta-modèle

La représentation de méta-modèle consiste à décrire le méta-modèle de CIM par un schéma LDAP, c'est à dire décrire chaque élément du méta-modèle CIM (classe, propriété, méthode, ...) par un ou plusieurs éléments du méta-modèle LDAP. Les entrées du serveur LDAP représenterons alors les définitions de classes, de propriétes, etc ....

L'algorithme de réalisation de cette stratégie consiste à définir un schéma LDAP décrivant le méta-modèle de CIM. Ce schéma sera par la suite utilisé pour donner un "description LDAP" de toute modélisation CIM.

Une stratégie analogue a été utilisée par le DMTF dans le cadre de la représentation des spécifications CIM en XML [DMTF 98b] [DMTF 99d] [DMTF 99a] : un document DTD (*Document Type Definition*) a été défini pour permettre l'encapsulation de toute modélisation CIM dans un document XML.

3.1.3 Discussion

Chacune des deux stratégies que nous venons de présenter a ses avantages et ses inconvénients. Nous avons opté pour une représentation de méta-modèle pour les raisons suivantes :

- elle réduit la complexité au niveau des serveurs LDAP :

  avec une représentation de méta-modèle, un schéma unique, de petite taille, permet de charger les spécification CIM dans les serveurs LDAP ;

  avec une représentation de modèle, plusieurs schémas (un pour chaque modèle CIM), de tailles assez conséquentes (directement proportionnelles aux tailles des modèles CIM) , doivent être chargés dans les serveurs LDAP ;
- elle facilite la gestion des serveurs LDAP :

  avec une représentation de méta-modèle, un changement au niveau des modèles CIM se traduit par un changement aux niveau des entrées de l'annuaire. La mise à jour peut exploiter les fonctionnalités des serveurs LDAP (distribution et mise à jour) ;

  avec une représentation de modèle, un changement des modèles CIM se traduit par un changement des schémas LDAP, tâche relevant souvent de la configuration des serveurs (c'est le cas pour le serveur LDAP de Netscape par exemple), ce qui est nettement plus fastidieux qu'une simple mise à jour de la base de données d'un serveur ;
- elle peut permettre d'implanter des fonctionnalités de manipulation de schéma dans un CIMOM :

  la représentation de modèle, essentiellement orientée représentation d'instances d'objets, nous mène tout naturellement à la question : peut-on implanter un CIMOM à l'aide d'un serveur LDAP, ou encore peut-on remplacer un CIMOM par un serveur LDAP ? nous n'en sommes pas convaincus pour une raison essentielle inhérente à la philosophie même de LDAP : le protocole ainsi que le serveur sont concus et optimisés pour traiter de l'information du type statique, ce qui n'est pas le cas d'un grand nombre d'instances d'objets gérés ;

  la représentation de méta-modèle est orientée représentation des classes du modèle, ce qui peut profiter à un CIMOM [3], qui en reportant cette tâche de maintient du modèle à un serveur LDAP, n'aura plus qu'à se consacrer (i.e. être optimisé) au traitement des instances d'objets gérés.

3. Et c'est, rappelons-le, dans cette perspective que nous avons entrepris ce travail!

d'un autre côté, la représentation de méta-modèle peut, si le besoin s'en fait sentir, être facilement étendue à la représentation des instances (tout en restant moins performante que la stratégie de modèle) ;

3.2 L'architecture du serveur LDAP

Conformément à notre choix de stratégie de représentation des modélisations CIM, nous avons été en mesure de définir deux des composantes du service d'annuaire LDAP que nous proposons : le modèle de l'information et le modèle de nommage. Ces deux modèles, décrits brièvement dans cette section, sont détaillés dans la section 4.

3.2.1 Les classes d'objets

Notre schéma de représentation définit pour chaque élément du méta-modèle CIM une classe d'objet de l'annuaire LDAP. Ces classes sont détaillées dans la section 4.2.

- **cim-namespace** est utilisée pour la représentation d'un espace de nommage ;
- **cim-class** est utilisée pour la représentation d'une définition de classe CIM (classe d'objet géré ou association) ;
- **cim-property** est utilisée pour la représentation d'une définition de propriété de classe ;
- **cim-method** est utilisée pour la représentation d'une définition de méthode de classe ;
- **cim-parameter** est utilisée pour la représentation d'une définition de paramètre d'une méthode ;
- **cim-association** est utilisée pour la représentation d'une définition d'association ;
- **cim-reference** est utilisée pour la représentation d'une définition de référence d'une classe d'association ;
- **cim-qualifier** est utilisée pour la représentation d'une instance de qualifieur ;
- **cim-qualifierdefinition** est utilisée pour la représentation d'une définition de qualifieur ;
- **cim-instance** est utilisée pour la représentation d'une définition d'instance.

Les classes définies dans le schéma ainsi que les relations qui existent entre elles sont représentées en UML dans la figure 5.

3.2.2 L'arbre de nommage

Le modèle de nommage que nous avons choisi défini un arbre de nommage avec trois niveaux conceptuels :

- un niveau de représentation d'un espace de nommage qui peut contenir dans son sous-arbre tout type d'élément de représentation y compris la représentation d'un autre espace de nommage (espace de nommage fils) ;
- un niveau de représentation de tout élément de modélisation CIM exepté le qualifieur : la classe, l'association, la propriété, la méthode, le paramètre, la référence et l'instance. Chaque élément de ce niveau contient la (les) représentation(s) de(s) différents composants de la définition CIM (exemple : une propriété est un composant d'une classe CIM) ;
- un niveau de représentation des qualifieurs formant les feuilles de l'arbre de nommage ;

A ces trois niveaux conceptuels correspond un arbre de nommage avec cinq niveaux hiérarchiques possibles décrits dans la figure 6.

1. le premier niveau correspond à la représentation d'un espace de nommage : toute définition CIM appartiendra au sous arbre d'un "nœud espace de nommage" ;
2. le second niveau contient les représentations des classes, associations et instances CIM ainsi que celles des définitions des qualifieurs ;
3. les propriétés, méthodes et références sont représentées dans l'arbre de nommage sous les noeuds correspondants aux classes et associations qui les définissent dans les modélisations;
4. le dernier type de noeuds possibles correspond aux représentations des paramètres des méthodes ;
5. les instances de qualifieurs sont représentées par des feuilles dans l'arbre de nommage. Elles sont rattachées au noeud correspondant à la définition CIM qu'elles qualifient.

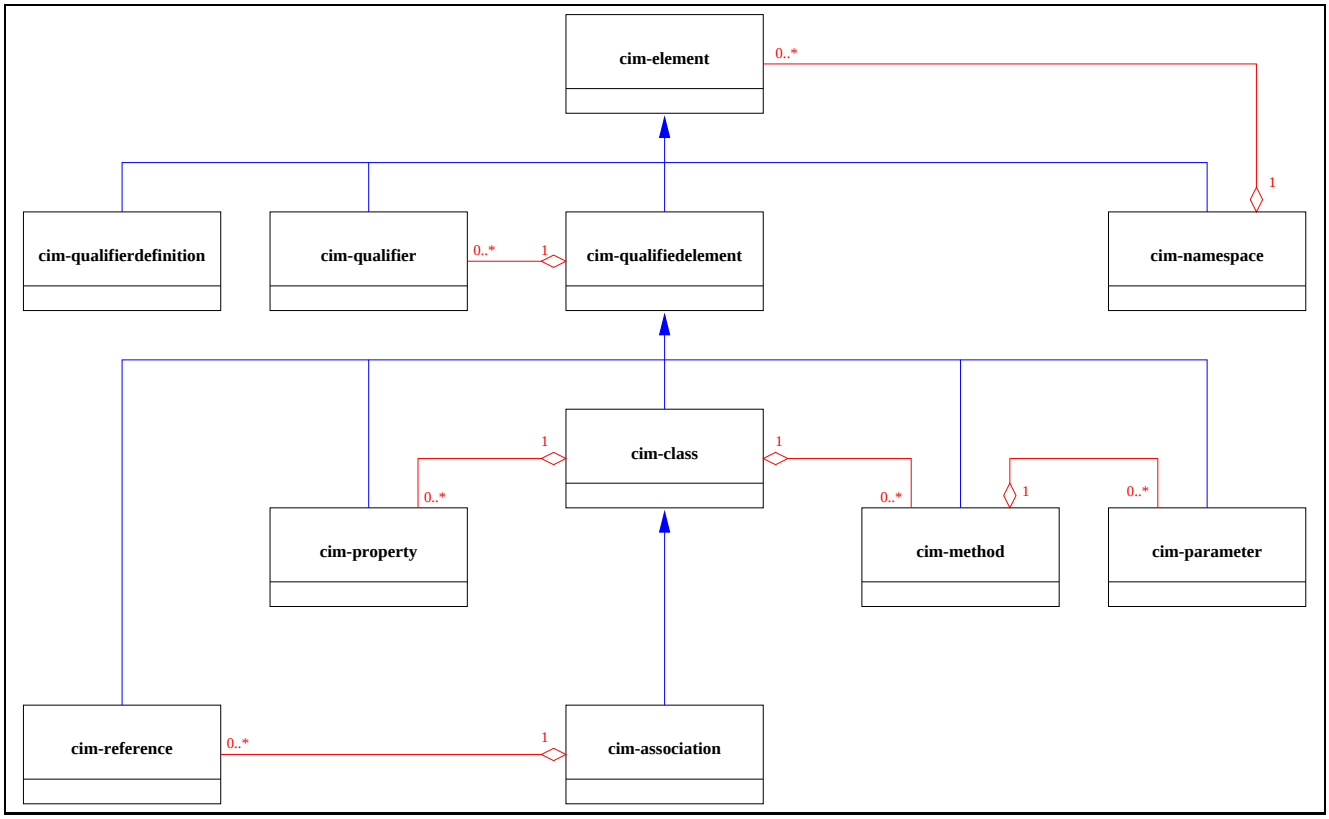


FIG. 5 – *Les classes du schéma LDAP*

4 Le schéma LDAP pour la représentation de CIM

Dans cette section, nous présentons le schéma LDAP que nous avons utilisé pour la représentation de l'information CIM. Ce schéma définit la représentation de chacun des éléments du méta-modèle CIM, permettant ainsi de transcrire une spécification CIM en un ensemble d'entrées dans un annuaire LDAP.

Dans la suite de cette section nous donnons dans un premier temps la liste des attributs utilisés dans le cadre de cette représentation LDAP de CIM. Ensuite nous donnons la liste des classes d'objets définissant les entrées de l'annuaire, ainsi que quelques exemples d'entrées écrits dans le format LDIF (*LDAP Data Interchange Format*) utilisé pour le chargement d'un grand nombre de données dans la base d'annuaire.

4.1 Les définitions d'attributs

Les spécifications des attributs données dans cette section sont basées sur la notion de sous-typage d'attribut. Une telle syntaxe définie par spécialisation est autorisée dans la version LDAPv3, sans être obligatoire dans une implémentation de serveur LDAP. Ainsi, nous signalons que cette notion de sous-typage, gardée dans le présent rapport pour des raisons de simplification, n'est absolument pas fondamentale dans nos définitions et peut être facilement remplacée par une syntaxe explicite des types dans les définitions des attributs.

4.1.1 L'attribut cim-name

L'attribut **cim-name** est défini comme une spécialisation de l'attribut générique **name** du shéma standard [Wahl 97a], et hérite sa syntaxe de chaîne de caractères **Directory String** (ligne 4). La mention **SINGLE-VALUED** (ligne 3) précise que l'attribut est de type simple et ne pourra comporter qu'une unique valeur[4]. La figure 7 donne la définition de l'attribut **cim-name** ainsi que celle de l'attribut **name**.

4. Les attributs LDAP sont par défaut multi-valués.

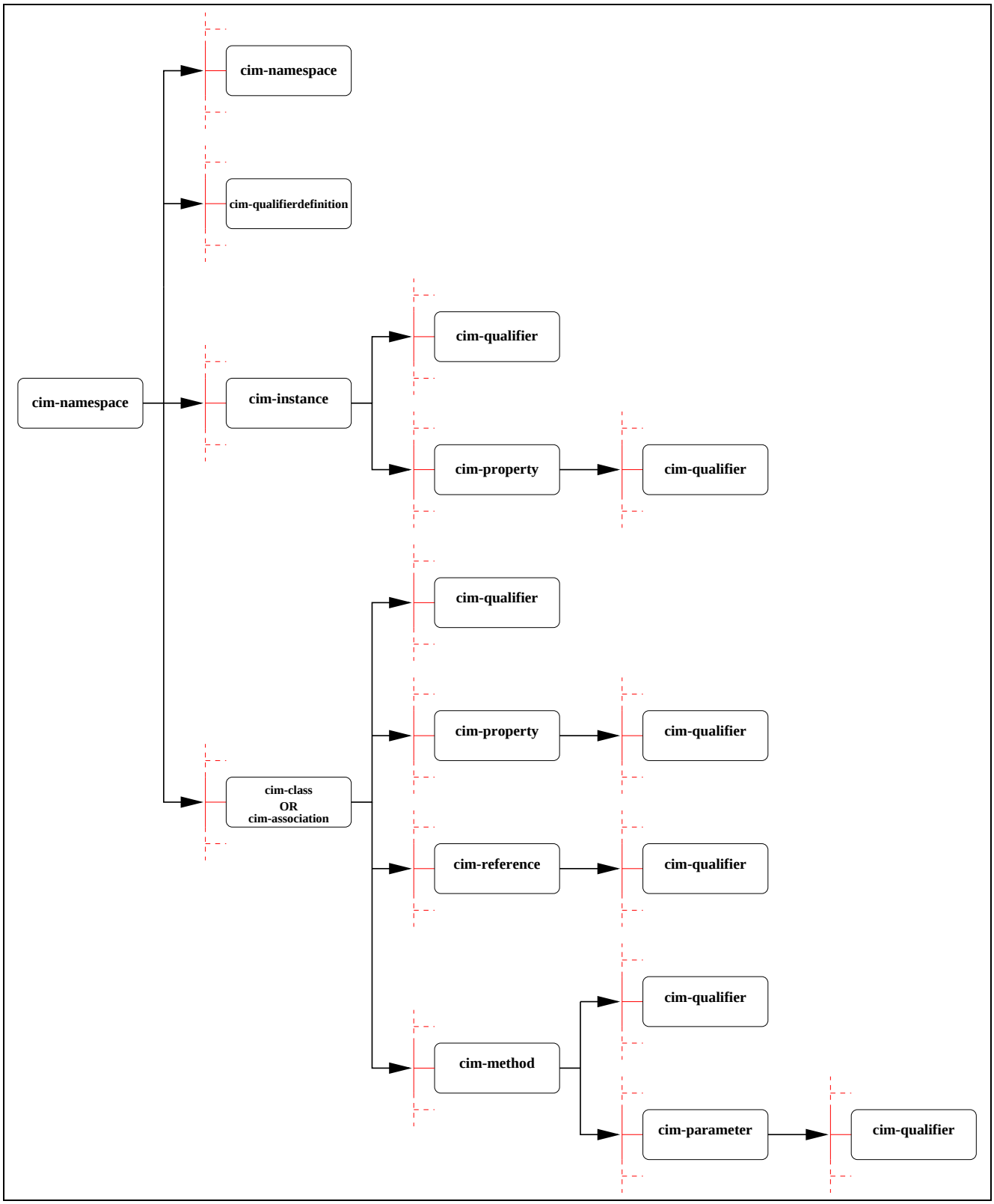


FIG. 6 – *L'arbre de nommage des objets CIM dans un serveur LDAP*

L'attribut name

```
( 2.5.4.41  NAME  ' name '
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 {32768}
)
```

L'attribut cim-name

```
( <cim-name-oid>  NAME  ' cim-name '
    SUP name
    SINGLE-VALUE
)
```

FIG. 7 – *L'attribut de nommage* **cim-name**

Cet attribut est utilisé comme attribut de nommage d'une représentation d'un espace de nommage CIM (`namespace`). Il est également utilisé comme racine de tous les attributs de nommage des objets définis dans le schéma LDAP de la repésentation des modèles CIM.

4.1.2 L'attribut cim-classname

L'attribut **cim-classname** est une spécialisation de de l'attribut **cim-name** et permet de donner le nom d'une classe CIM. Dans une représentation de classe, il est utilisé comme attribut de nommage.

```
( <cim-classname-oid>  NAME  'cim-classname '
    SUP cim-name
)
```

4.1.3 L'attribut cim-superclassname

L'attribut **cim-superclassname** est une spécialisation de l'attribut **cim-name** et permet de donner le nom de la super-classe dans la définition d'une classe CIM. Il est défini de la même manière que l'attribut **cim-classname**.

4.1.4 L'attribut cim-propertyname

L'attribut **cim-propertyname** est une spécialisation de l'attribut **cim-name** et permet de donner le nom d'une propriété CIM. Dans une représentation de propriété, il est utilisé comme attribut de nommage. Il est défini de la même manière que l'attribut **cim-classname**.

4.1.5 L'attribut cim-methodname

L'attribut **cim-methodname** est une spécialisation de l'attribut **cim-name** et permet de donner le nom d'une méthode CIM. Dans une représentation de méthode, il est utilisé comme attribut de nommage. Il est défini de la même manière que l'attribut **cim-classname**.

4.1.6 L'attribut cim-referencename

L'attribut **cim-referencename** est une spécialisation de l'attribut **cim-name** et permet de donner le nom d'une référence CIM. Dans une représentation de référence, il est utilisé comme attribut de nommage. Il est défini de la même manière que l'attribut **cim-classname**.

4.1.7 L'attribut cim-referencedclassname

L'attribut **cim-referencedclassname** est une spécialisation de l'attribut **cim-name** et permet de donner le nom de la classe référencée dans la définition d'une référence d'une association CIM. Il est défini de la même manière que l'attribut **cim-classname**.

4.1.8 L'attribut cim-parametername

L'attribut **cim-parametername** est une spécialisation de l'attribut **cim-name** et permet de donner le nom d'une paramètre CIM. Dans une représentation de paramètre, il est utilisé comme attribut de nommage. Il est défini de la même manière que l'attribut **cim-classname**.

4.1.9 L'attribut cim-type

L'attribut **cim-type** est utilisé pour représenter la définition d'un type de données CIM. Sa syntaxe de définition est une chaîne de caractères simple. Les valeurs possibles que cet attribut peut prendre étant l'une de celles données dans la spécification des types CIM (`uint8`, `sint8`, `uint16`, `sint16`, `uint32`, `sint32`, `uint64`, `sint64`, `real32`, `real64`, `boolean`, `string`, `char16`, `datetime`).

```
( <cim-type-oid> NAME 'cim-type'
    EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 {16}
    SINGLE-VALUE
)
```

4.1.10 L'attribut cim-value

L'attribut **cim-value** permet de représenter une valeur associée à une donnée CIM (valeur par défaut de propriété ou de qualifieur, ou valeur d'initialisation d'une propriété ou qualifieur, ou valeur d'initialisation d'une référence vers un objet). Sa syntaxe est une chaîne de caractères multi-valuée.

```
( <cim-value-oid> NAME 'cim-value'
    EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
)
```

4.1.11 L'attribut cim-array

L'attribut **cim-array** permet de représenter la taille d'un vecteur de données CIM. Sa syntaxe est une chaîne de caractères simple.

```
( <cim-array-oid> NAME 'cim-array'
    EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
    SINGLE-VALUE
)
```

4.1.12 L'attribut cim-scope

L'attribut **cim-scope** permet de représenter la portée d'un qualifieur dans une définition de qualifieur. Sa syntaxe est une chaîne de caratères multi-valuée. Les valeur possibles sont celles données dans la spécification CIM pour la portée d'un qualifieur (`CLASS`, `ASSOCIATION`, `INDICATION`, `PROPERTY`, `METHOD`, `REFERENCE`, `PARAMETER`, `ANY`).

```
( <cim-scope-oid> NAME 'cim-scope'
    EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 {16}
)
```

4.1.13 L'attribut cim-flavor

L'attribut **cim-flavor** permet de représenter les règles de propagation d'un qualifieur. Sa syntaxe est une chaîne de caractères multi-valuée. Les valeurs possibles sont celles données dans la spécification CIM pour les règles de propagation d'un qualifieur (`EnableOverride`, `DisableOverride`, `ToSubclass`, `Restricted`, `Translatable`).

```
( <cim-flavor-oid> NAME 'cim-flavor '
    EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 {16}
)
```

4.1.14 L'attribut cim-qualifierslist

L'attribut **cim-qualifierslist** permet de donner la liste des noms de qualifieurs associés à une définition CIM. Sa syntaxe est une chaîne de caractères multi-valuée.

```
( <cim-qualifierslist-oid> NAME 'cim-qualifierslist '
    SUP name
)
```

4.1.15 L'attribut cim-propertieslist

L'attribut **cim-propertieslist** permet de donner la liste des noms de propriétés d'une classe CIM. Sa syntaxe est une chaîne de caractères multi-valuée.

```
( <cim-propertieslist-oid> NAME 'cim-propertieslist '
    SUP name
)
```

4.1.16 L'attribut cim-methodslist

L'attribut **cim-methodslist** permet de donner la liste des noms de méthodes d'une classe CIM. Sa syntaxe est une chaîne de caractères multi-valuée.

```
( <cim-methodslist-oid> NAME 'cim-methodslist '
    SUP name
)
```

4.1.17 L'attribut cim-parameterslist

L'attribut **cim-parameterslist** permet de donner la liste des noms de paramètres d'une méthode de class CIM. Sa syntaxe est une chaîne de caractères multi-valuée.

```
( <cim-parameterslist-oid> NAME 'cim-parameterslist '
    SUP name
)
```

4.1.18 L'attribut cim-referenceslist

L'attribut **cim-referenceslist** permet de donner la liste des noms de références d'une association CIM. Sa syntaxe est une chaîne de caractères multi-valuée.

```
( <cim-referenceslist-oid> NAME 'cim-referenceslist '
    SUP name
)
```

4.2 Les définition des classes

De même que pour les attributs, les définitions des classes d'objets s'appuyent sur un mécanisme d'héritage autorisé dans le protocole LDAPv3 mais pas toujours implémenté. Ce mécanisme d'héritage a été gardé dans le présent rapport, mais peut facilement être éliminé à l'implémentation sans changer la nature de la représentation LDAP de CIM. Ceci est obtenu en reportant, pour chaque classe d'objet du schéma LDAP proposé, tous les attributs appartenant à ses super-classes.

4.2.1 La classe racine cim-element

Elle est directement issue de la classe **Named Element** racine du méta-modèle CIM, et ne définit aucun attribut spécifique. La spécification de la classe **cim-element** est donnée dans la figure 8.

```
1 ( <cim-element-oid>  NAME  'cim-element '
2     SUP top
3     ABSTRACT
4 )
```

FIG. 8 – *Spécification de la classe* **cim-element**

Comme toute classe LDAP, **cim-element** hérite de la classe générique **top** (ligne 2) définie dans le schéma standard de LDAPv3 [Wahl 97a].

La classe est abstraite (ligne 3), i.e. aucune entrée de ce type ne sera enregistrée dans le directory. L'intérêt de cette définition est de donner une racine pour les définitions de classes LDAP utilisées dans notre schéma de représentation de CIM.

4.2.2 La classe cim-namespace

Elle spécialise la classe **cim-element** (ligne 2) et permet de représenter un espace de nommage CIM dans la base d'information du serveur LDAP. Elle ne définit qu'un attribut obligatoire **cim-name** (ligne 4) permettant de donner le nom relatif de l'espace de nommage. La définition de la classe **cim-namespace** ainsi qu'un exemple d'une entrée correspondant à un espace de nommage dans le format LDIF sont donnés dans la figure 4.2.2.

La spécification de la classe cim-namespace

```
1 ( <cim-namespace-oid>  NAME  ' cim-namepsace '
2     SUP cim-element
3     STRUCTURAL
4     MUST <cim-name-oid>
5 )
```

L'entrée LDAP correspondant à l'espace de nommage /root/cimv2

```
1 dn: cim-name=cimv2,cim-name=root,o=loria.fr
2 objectclass: top
3 objectclass: cim-element
4 objectclass: cim-namespace
5 cim-name cimv2
```

FIG. 9 – *La classe* **cim-namespace**

Dans l'exemple, l'entrée correspondant à l'espace de nommage `/root/cimv2` initialise l'attribut `cim-name` avec la valeur `cimv2` qui est le nom relatif par rapport à l'espace de nommage père `/root`.

4.2.3 La classe cim-qualifiedelement

Définie comme classe abstraite, elle spécialise la classe **cim-element** pour donner une racine de la représentation de tout élément CIM auquel peut être associé une liste de qualifieurs, i.e. la classe, la propriété, la méthode, le paramètre, la référence et l'instance. La spécification de la classe **cim-qualifiedelement** est donnée dans la figure 10.

```
1 ( <cim-qualifiedelement-oid>  NAME  'cim-qualifiedelement '
2     SUP <cim-element-oid>
3     ABSTRACT
4     MAY <cimqualifierslist-oid>
5 )
```

FIG. 10 – *Specification de la classe* **cim-qualifiedelement**

La classe **cim-qualifiedelement** hérite de **cim-element** (ligne 2) et ne définit qu'un unique attribut supplémentaire **cim-qualifierslist**. Elle est abstraite (ligne 3), et ne peut pas être instanciée.

4.2.4 Représentation d'une définition de classe

La définition d'une classe CIM, est représentée par une entrée dans l'annuaire du type **cim-class**. La spécification de cette classe d'objet est donnée dans la figure 11.

```
1 ( <cim-class-oid>  NAME  'cim-class '
2     SUP cim-qualifiedelement
3     STRUCTURAL
4     MUST cim-classname
5     MAY   ( cim-superclassname  $
6             cim-propertieslist  $
7             cim-methodslist   )
8 )
```

FIG. 11 – *La classe* **cim-class**

La classe **cim-class** hérite les attributs de la classe **cim-qualifiedelement** (ligne 2).

Elle définit un attribut obligatoire **cim-classname** (ligne 4) permattant de donner le nom de la classe CIM ; et trois attributs optionnels **cim-superclassname** (ligne 5), **cim-propertieslist** (ligne 6) et **cim-methodslist** permettant de donner le nom de super-classe ainsi que les listes des propriétés et des méthodes de la classe CIM.

Dans la figure 12 nous donnons un exemple d'entrée LDAP du type **cim-class**, correspondant à une définition de classe CIM (la classe CIM donnée ici est issue du modèle de base *Core Schema* du DMTF [DMTF 98a]).

4.2.5 Représentation d'une définition de propriété

La représentation d'une définition de propriété d'une classe CIM ainsi que la représentation d'une initialisation de propriété dans une instance sont données par des entrées du directory du type **cim-property**. La spécification de cette classe est donnée dans la figure 13.

La classe **cim-property** hérite les attributs de la classe **cim-qualifiedelement** (ligne 2).

Elle définit en plus deux attributs obligatoires **cim-propertyname** (ligne 3) et **cim-type** permettant de donner le nom et le type de la propriété CIM; et deux attributs optionnels **cim-value** (ligne 5) et **cim-array** (ligne 6) permettant de donner la valeur de la propriété (par défaut pour une declaration, et effective pour une instanciation) et la taille du vecteur pour les propriétés de ce type.

Dans la figure 14 nous donnons un exemple d'entrée LDAP du type **cim-property**, correspondant à une définition de propriété CIM (la définition CIM correspondante est donnée dans la figure 12).

La spécification de la classe CIM_ManagedSystemElement

```
   [ Abstract, Description("CIM_ManagedSystemElement is the base class...")]
class CIM_ManagedSystemElement {
      [ MaxLen(256), Description("The Name property defines the label...")]
  string Name;
      [ Description("The description property provides a textuel description...")]
  string Description;
}
```

La définition de l'entrée LDAP correspondante

dn: cim-classname=CIM_ManagedSystemElement,cim-name=cimv2,cim-name=root,o=loria.fr
objectclass: top
objectclass: cim-element
objectclass: cim-qualifiedelement
objectclass: cim-class
cim-classname: CIM_ManagedSystemElement
cim-qualifierslist: Abstract
cim-qualifierslist: Description
cim-propertieslist: Description
cim-propertieslist: Name

FIG. 12 – *Un exemple d'entrée du type* **cim-class**

```
( <cim-property-oid>  NAME  'cim-property '
    SUP cim-qualifiedelement
    STRUCTURAL
    MUST  ( cim-propertyname  $
            cim-type  )
    MAY   ( cim-value  $
            cim-array  $  )
)
```

FIG. 13 – *La class* **cim-property**

dn: cim-propertyname=Name,cim-classname=CIM_ManagedSystemElement,
cim-name=cimv2,cim-name=root,o=loria.fr
objectclass: top
objectclass: cim-element
objectclass: cim-qualifiedelement
objectclass: cim-property
cim-propertyname: Name
cim-type: string
cim-qualifierslist: MaxLen
cim-qualifierslist: Description

FIG. 14 – *Un exemple d'entrée du type* **cim-property**

4.2.6 Représentation d'une définition de méthode

La définition d'une méthode est représentée par une *entrée* du directory du type **cim-method**. La spécification de cette classe est donnée dans la figure 15.

```
1 ( <cim-method-oid>  NAME  'cim-method '
2     SUP cim-qualifiedelement
3     STRUCTURAL
4     MUST  ( cim-methodname  $
5               cim-type  )
6     MAY   ( cim-array  $
7               cim-parameterslist  )
8 )
```

FIG. 15 – *La class* **cim-method**

La classe **cim-method** hérite les attributs de la classe **cim-qualifiedelement** (ligne 2).

Elle définit en plus deux attributs obligatoires **cim-methodname** (ligne 3) et **cim-type** (ligne 4) permettant de donner le nom de la méthode CIM ainsi que son type de retour; et 2 attributs optionnels **cim-array** (ligne 5) et **cim-parameterslist** (ligne 6) permettant de représenter les méthodes avec une donnée de retour de type vecteur ainsi que de représenter la liste des paramètres des méthodes.

Dans la figure 16 nous donnons un exemple d'entrée LDAP du type **cim-method**, correspondant à une définition de méthode CIM (la définition CIM correspondante est issue du schéma de base CIM).

La spécification de la méthode CIM

```
   [ Description("SetPowerState defines the power state for a LogicalDevice...")]
uint32 SetPowerState( [IN] uint16 PowerState, [IN] datetime Time );
```

La définition de l'entrée LDAP correspondante

dn: cim-methodname=SetPowerState,cim-classname=CIM_LogicalDevice,
cim-name=cimv2,cim-name=root,o=loria.fr
objectclass: top
objectclass: cim-element
objectclass: cim-qualifiedelement
objectclass: cim-method
cim-methodname: SetPowerState
cim-type: uint32
cim-qualifierslist: Description
cim-parameterslist: PowerState
cim-parameterslist: Time

FIG. 16 – *Un exemple d'entrée du type* **cim-method**

4.2.7 Représentation d'une définition de paramètre

La définition d'un paramètre est représentée par une entrée du directory du type **cim-parameter**. La spécification de cette classe est donnée dans la figure 17.

La classe **cim-parameter** hérite les attributs de la classe **cim-qualifiedelement** (ligne 2).

Elle définit en plus un attribut obligatoire **cim-parametername** (ligne 4) permettant de donner le nom du paramètre CIM; et deux attributs optionnels **cim-type** (ligne 5) permettant de donner le type du paramètre pour les paramètres de type simple et **cim-referencedclassname** (ligne 6) permettant de donner le nom de la classe référencée pour les paramètres de type référence vers un objet.

Dans la figure 18 nous donnons un exemple d'entrée LDAP du type **cim-parameter**, correspondant à une définition de parameter CIM (la définition CIM correspondante est donnée dans la figure 16).

```
1 ( <cim-parameter-oid>  NAME  'cim-parameter '
2     SUP cim-qualifiedelement
3     STRUCTURAL
4     MUST cim-parametername
5     MAY  ( cim-type  $
6              cim-referencedclassname  )
7 )
```

FIG. 17 – *La classe* **cim-parameter**

```
dn: cim-parametername=PowerState,cim-methodname=SetPowerState,
          cim-classname=CIM_LogicalDevice,cim-name=cimv2,cim-name=root,o=loria.fr
objectclass: top
objectclass: cim-element
objectclass: cim-qualifiedelement
objectclass: cim-parameter
cim-parametername: PowerState
cim-type: uint16
cim-qualifierslist: IN
```

FIG. 18 – *Un exemple d'entrée du type* **cim-parameter**

4.2.8 Représentation d'une définition d'association

La définition d'une association CIM, est représentée par une entrée du directory du type **cim-association**. La spécification de cette classe d'objet est donnée dans la figure 19.

```
1 ( <cim-association-oid>  NAME  'cim-association '
2     SUP cim-class
3     STRUCTURAL
4     MAY cim-referenceslist
5 )
```

FIG. 19 – *La classe* **cim-association**

La classe **cim-association** hérite les attributs de la classe **cim-class** (ligne 2).

Elle définit en plus un attribut optionnel **cim-referenceslist** permettant de donner la liste des références de l'association CIM.

Dans la figure 20 nous donnons un exemple d'entrée LDAP du type **cim-association**, correspondant à une définition de classe CIM (la définition CIM donnée ici est issue du modèle de base *Core Schema*).

4.2.9 Représentation d'une définition de référence

La définition d'une référence est représentée par une entrée du directory du type **cim-reference**. La spécification de cette classe est donnée dans la figure 21.

La classe **cim-reference** hérite les attributs de la classe **cim-qualifiedelement** (ligne 2).

Elle définit en plus deux attributs obligatoires **cim-referencename** (ligne 3) et **cim-referencedclassname** (ligne 4) permettant de donner le nom de la référence ainsi que le nom de la classe des objets référencés.

Dans la figure 22 nous donnons un exemple d'entrée LDAP du type **cim-reference**, correspondant à une définition de référence CIM (la définition CIM donnée ici est donnée dans la figure 20).

La spécification de la classe CIM_Dependency

```
   [ Association, Abstract, Description("CIM_Dependency is ageneric association...")]
class CIM_Dependency {
      [ Description("Antecedent represents the independent object...")]
  CIM_ManagedSystemElement REF Antecedent;
      [ Description("Dependent represents the object dependent...")]
  CIM_ManagedSystemElement REF Dependent;
}
```

La définition de l'entrée LDAP correspondante

dn: cim-classname=CIM_Dependency,cim-name=cimv2,cim-name=root,o=loria.fr
objectclass: top
objectclass: cim-element
objectclass: cim-qualifiedelement
objectclass: cim-class
objectclass: cim-association
cim-namespace: /root/default
cim-classname: CIM_Dependency
cim-qualifierslist: Association
cim-qualifierslist: Abstract
cim-qualifierslist: Description
cim-referenceslist: Antecedent
cim-referenceslist: Dependent

FIG. 20 – *Un exemple d'entrée du type* **cim-association**

```
1 ( <cim-reference-oid>  NAME  'cim-reference '
2     SUP cim-qualifiedelement
3     STRUCTURAL
4     MUST  ( cim-referencename  $
5               cim-referencedclassname  )
6 )
```

FIG. 21 – *La classe* **cim-reference**

dn: cim-referencename=Antecedent,cim-classname=CIM_Dependency,
cim-name=cimv2,cim-name=root,o=loria.fr
objectclass: top
objectclass: cim-element
objectclass: cim-qualifiedelement
objectclass: cim-reference
cim-referencedclassname: CIM_ManagedSystemElement
cim-referencename: Antecedent
cim-qualifierslist: Description

FIG. 22 – *Un exemple d'entrée du type* **cim-reference**

4.2.10 Représentation d'une définition de qualifieur

La définition d'un qualifieur est représentée par une entrée du directory du type **cim-qualifierdefinition**. La spécifictaion de cette classe est donnée dans la figure 23.

```
1  ( <cim-qualifierdefinition-oid>  NAME  'cim-qualifierdefinition '
2      SUP cim-element
3      STRUCTURAL
4      MUST  ( cim-qualifiername  $
5                cim-type  $
6                cim-scope  )
7      MAY   ( cim-array  $
8                cim-value  $
9                cim-flavor  )
10 )
```

FIG. 23 – *La classe* **cim-qualifierdefinition**

La classe est directement issue de la classe abstraite **cim-element**. Elle définit trois attributs obligatoires **cim-qualifiername** (ligne 3), **cim-type** (ligne 4) et **cim-scope** (ligne 5) permettant de donner le nom, le type et le champs d'application du qualifieur (*scope*); et trois attributs optionnels **cim-array** (ligne 6), **cim-value** (ligne 7) et **cim-flavor** (ligne 8) permettant de donner la taille d'un qualifieur de type vecteur, une valeur par défaut et les règles d'héritage du qualifieur (*flavor*).

Dans la figure 24 nous donnons un exemple d'entrée LDAP du type **cim-qualifierdefinition**, correspondant à une définition de qualifieur CIM (la définition CIM donnée ici est schéma de base CIM).

La spécification d'une définition de qualifieur CIM

```
Qualifier : Association : boolean = false, Scope(class, association),
                          Flavor(DisableOverride);
```

La définition de l'entrée LDAP correspondante

dn: `cim-qualifiername=Association,cim-name=cimv2,cim-name=root,o=loria.fr`
objectclass: `top`
objectclass: `cim-element`
objectclass: `cim-qualifierdefinition`
cim-qualifiername: `Association`
cim-type: `boolean`
cim-scope: `CLASS`
cim-scope: `ASSOCIATION`
cim-value: `false`
cim-flavor: `DisableOverride`

FIG. 24 – *Un exemple d'entrée du type* **cim-qualifierdefinition**

4.2.11 Représentation d'une instance de qualifieur

Une instance de qualifieur (associé à un élément CIM) est représentée par une entrée du directory du type **cim-qualifier**. La spécification de cette classe est donnée dans la figure 25.

La classe hérite des attributs de la classe **cim-element**. Elle définit en plus deux attributs obligatoires **cim-qualifiername** (ligne 3), **cim-value** (ligne 4) et **cim-type** permettant de donner le nom, la valeur de l'instance de qualifieur ainsi que le type du qualifieur; et d'un attribut optionnel **cim-flavor** (ligne 5) permettant de donner les règles d'héritage du qualifieur.

```
1 ( <cim-qualifier-oid>  NAME  'cim-qualifier '
2     SUP cim-element
3     STRUCTURAL
4     MUST  ( cim-qualifiername  $
5               cim-value  $
6               cim-type  )
7     MAY cim-flavor
8 )
```

FIG. 25 – *La classe* **cim-qualifier**

Dans la figure 26 nous donnons un exemple d'entrée LDAP du type **cim-qualifier**, correspondant à une instance de qualifieur CIM.

```
dn: cim-qualifiername=Association,cim-class=CIM_LogicalDevice,
          cim-name=cimv2,cim-name=root,o=loria.fr
objectclass: top
objectclass: cim-element
objectclass: cim-qualifier
cim-qualifiername: Association
cim-value: true
cim-type: boolean
```

FIG. 26 – *Un exemple d'entrée du type* **cim-qualifier**

4.2.12 Représentation d'une instance

La définition d'une instance CIM est représentée par une entrée dans l'annuaire du type **cim-instance**. La spécification de cette classe d'objet est donnée dans la figure 4.2.12.

```
1 ( <cim-instance-oid>  NAME  ' cim-instance '
2     SUP cim-qualifiedelement
3     STRUCTURAL
4     MUST  ( cim-classname  $
5               cim-instancename  )
6     MAY cim-propertieslist
7 )
```

FIG. 27 – *La classe* **cim-instance**

La classe **cim-instance** hérite des attributs de la classe **cim-qualifiedelement** (ligne 2).
Elle définit en plus deux attributs obligatoires et un attribut optionnel :

- **cim-classname** (ligne 4) permettant de donner le nom de la classe dont l'instance est issue ;
- **cim-instancename** (ligne 5) permettant de donner le nom complet de l'instance tel qu'il a été défini par le DMTF, i.e. la concaténation du nom de la classe avec la liste des attributs clef et de leurs valeurs ;
- **cim-propertieslist** (ligne 6) donnant la liste des propriétés qui ont été initialisées par l'instance.

Dans la figure 4.2.12 nous donnons un exemple de l'entrée LDAP du type **cim-instance** correspondant à une déclaration d'instance CIM.

La spécification d'une instance CIM

```
instance of CIM_ComputerSystem {
    Name = "dolcourt.loria.fr"
    NameFormat = "IP"
    CreationClassName = "CIM_ComputerSystem"
}
```

La définition de l'entrée LDAP correspondante

dn: cim-instancename=CIM_ComputerSystem.Name="dolcourt.loria.fr",
cim-name=cimv2,cim-name=root,o=loria.fr
objectclass: top
objectclass: cim-element
objectclass: cim-qualifiedelement
objectclass: cim-instance
cim-classname: CIM_ComputerSystem
cim-instancename: CIM_ComputerSystem.Name="dolcourt.loria.fr"
cim-propertieslist: Name
cim-propertieslist: NameFormat

FIG. 28 – *Un exemple d'entrée de type* **cim-instance**

5 Implémentation

MODERES est l'environnement développé au sein du projet RESEDAS pour la manipulation des modèles de l'information de gestion. Il comprend un ensemble d'analyseurs syntaxiques et sémantiques, ainsi qu'un ensemble d'outils d'intégration et de visualisation.

MODERES permet de charger et d'étudier des modélisations CIM. Il est muni d'un analyseur syntaxique [Festor 98] et d'un analyseur sémantique [Festor 99a] du langage MOF (Managed Object Format) utilisé pour les modélisations CIM ainsi que d'un éditeur graphique CIM [Festor 99b].

Dans le cadre de notre étude, nous avons enrichi l'outil par une passerelle CIM/LDAP s'appuyant sur les modèles que nous avons spécifié dans les sections précédentes. Cette passerelle offre les fonctionnalités suivantes :

- l'enregistrement des modélisations CIM chargées par l'outil MODERES dans un annuaire LDAP ;
- la récupération dans l'outil MODERES des spécifications CIM depuis un serveur LDAP.

La figure 29 donne l'architecture de la passerelle CIM/LDAP. Dans cette architecture nous distinguons 4 composants :

- le provider de service LDAP développé par Sun permet de générer et de récupérer les unités de données LDAP ;
- l'interface JNDI [Sun Microsystems 99] (*Java Naming and Directory Interface*) développée par Sun offre un cadre unificateur pour le développement d'applications Java basées sur les services de nommage et d'annuaires ;
- le client LDAP de MODERES développé dans le cadre de cette étude, s'appuye sur l'interface JNDI pour offrir ses deux principales fonctionnalités. Il est implémenté par deux classes Java `MODERES_LDAPtoMOF` et `MODERES_MOFtoLDAP` du package **FR.loria.resedas.moderes.ldapmappings** inclu dans la distribution actuelle de MODERES ;
- l'interface CIM/LDAP est une généralisation du client LDAP de MODERES. Elle s'appuye sur les méthodes de traductions CIM/LDAP pour implanter l'interface de communication WBEM proposée par le DMTF [DMTF 99c] au dessus du protocole LDAP.

Dans la suite de cette section nous donnons quelques exemples d'utilisation de la passerelle CIM/LDAP ainsi qu'un apperçu rapide de l'interface WBEM.

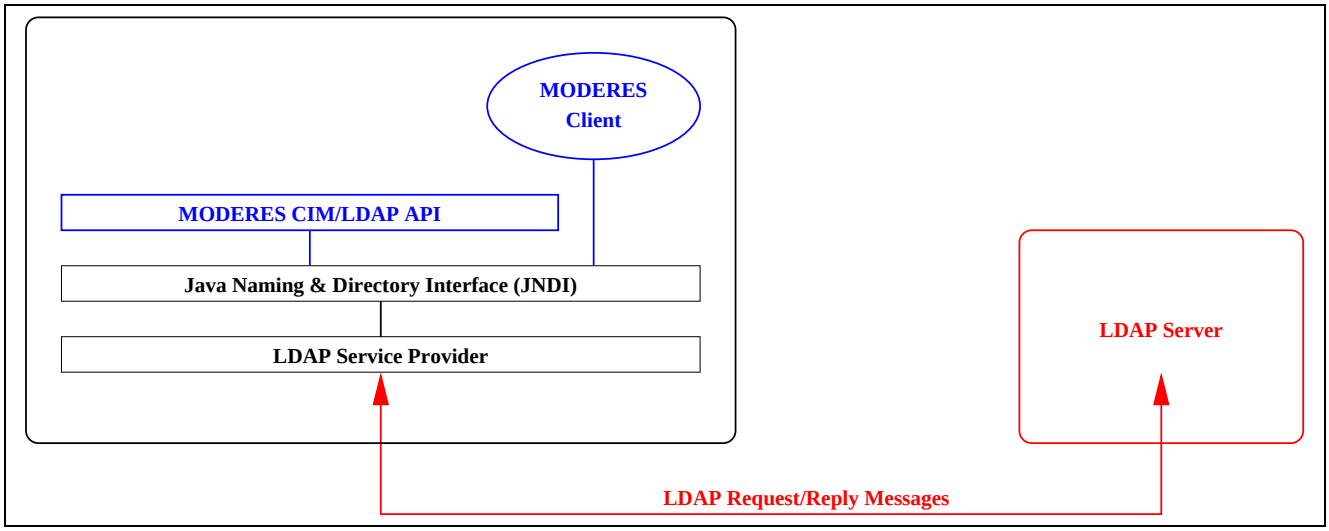


FIG. 29 – *L'implémentation de la passerelle CIM/LDAP*

5.1 Enregistrement de modélisations CIM dans un serveur LDAP

Le programme Java donné dans cette section permet de charger un ensemble de modélisations CIM depuis un fichier MOF, pour ensuite les enregistrer dans un serveur d'annuaire LDAP. Les lignes en gras sont celles qui permettent la sauvegarde des spécifications dans la base d'annuaire.

```
// JAVA related packages
import java.io.*;
import com.objectspace.jgl.*;

// JNDI related packages
import javax.naming.*;
import javax.naming.directory.*;

// MODERES related packages
import FR.loria.resedas.cimom.*;
import FR.loria.resedas.moderes.moffrontend.*;
import FR.loria.resedas.moderes.mofrepository.*;

public class SaveExemple {

  public static void main(String[] args) {

    System.out.println("Loading MOF specifications from a file");
    MODERESCoreMOFParser parser = null;
    try {
      parser = new MODERESCoreMOFParser(new File("example.mof"));
    }
    catch ( FileNotFoundException f )  {
      System.out.println("File not found!");
      System.exit(0);
    }

    // The mofRepository object will contain the MOF specifications
    // parsed from the file
    MOF_Repository mofRepository = new MOF_Repository();

    // parse the file
    try {
```

```
      parser.ParseMOFFile(mofRepository);
    }
    catch ( ParseException e ) {
      System.out.println("Parse Exception");
      System.exit(0);
    }

    System.out.println("Saving specifications in the LDAP directory ...");

    // LDAP URL specifying the directory server and the base DN which
    // will contain the CIM subtree
    String  url = "ldap://dolcourt.loria.fr:389/
                    ou=Management Information,o=loria.fr";

    // The user DN
    String  userId = "uid=moderes,ou=People,o=loria.fr";

    // The user password
    String  userPasswd = "moderes";

    // The base namespace used for the connection to the server
    String  namespace = "/root/example";

     try {
       // Instanciates a new client, and connect to the directory
       // at the specified namespace entry
       CIM_Client client =
         new CIM_Client(url,userId,userPasswd,namespace);

      System.out.println("Connected to the server !");

       // Saving the specifications in the directory
       MODERES_MOFtoLDAP.translateAll(mofRepository,client.getContext());

       client.close();

     }
     catch (CIM_Exception e) {
       System.out.println("CIM Exception : "+e.getMessage());
     }

    }

  }

}
```

5.2 Chargement de modélisations CIM depuis un serveur LDAP

Le programme Java donné dans cette section permet de charger un ensemble de modélisations CIM depuis un serveur d'annuaire LDAP. Les lignes en gras sont celles qui permettent le chargement des spécifications depuis la base d'annuaire.

```
// JAVA related packages
import com.objectspace.jgl.*;

// JNDI related packages
import javax.naming.*;
import javax.naming.directory.*;
```

```
// MODERES related packages
import FR.loria.resedas.cimom.*;
import FR.loria.resedas.moderes.mofrepository.*;

public class LoadExemple {

  public static void main(String[] args) {

    System.out.println("Loading specifications from LDAP directory ...");

    String  url = "ldap://dolcourt.loria.fr:389/
                  ou=Management Information,o=loria.fr";
    String  userId = "uid=moderes,ou=People,o=loria.fr";
    String  userPasswd = "moderes";

    String  namespace = "/root/default";

     try {

       CIM_Client client =
        new CIM_Client(url,userId,userPasswd,namespace);

      System.out.println("Connected to the server !");

       // Loading all definitions from the directory
       MOF_Specification mofSpecification =
        MODERES_LDAPtoMOF.translateAll(client.getContext());

      if ( mofSpecification != null )
        System.out.println("MOF specifications loaded");

       client.close();

    }
    catch ( CIM_Exception e) {
      System.out.println("CIM Exception : "+e.getMessage());
    }

  }
}
```

5.3 L'interface de programmation CIM/LDAP de MODERES

Developpée dans la cadre de cette étude, cette interface permet d'implanter les opérations CIM définies par le DMTF au dessus de l'interface JNDI. Cette approche nous a permis d'implanter facilement un *provider* de schéma dans un gestionnaire CIM (*CIMOM*) en utilisant la base d'annuaire LDAP comme base d'objets gérés CIM (*CIM Object Repository*) tel que nous le montre la figure 30.

L'interface de programmation CIM/LDAP de MODERES est disponible actuellement dans le package Java **FR.loria.resedas.cimom**. Elle est composée :

- d'un ensemble d'interfaces Java correspondantes aux profils fonctionnels d'un serveur CIMOM définis par le DMTF ;
- d'une classe de base **CIM_Client** implantant les opérations de connection et de déconnexion du serveur d'annuaire ;
- d'une classe **CIM_SchemaProvider** implémentant l'interface **CIM_SchemaManipulationInterface** comprenant les opérations d'accès aux définitions des classes et des instances ;

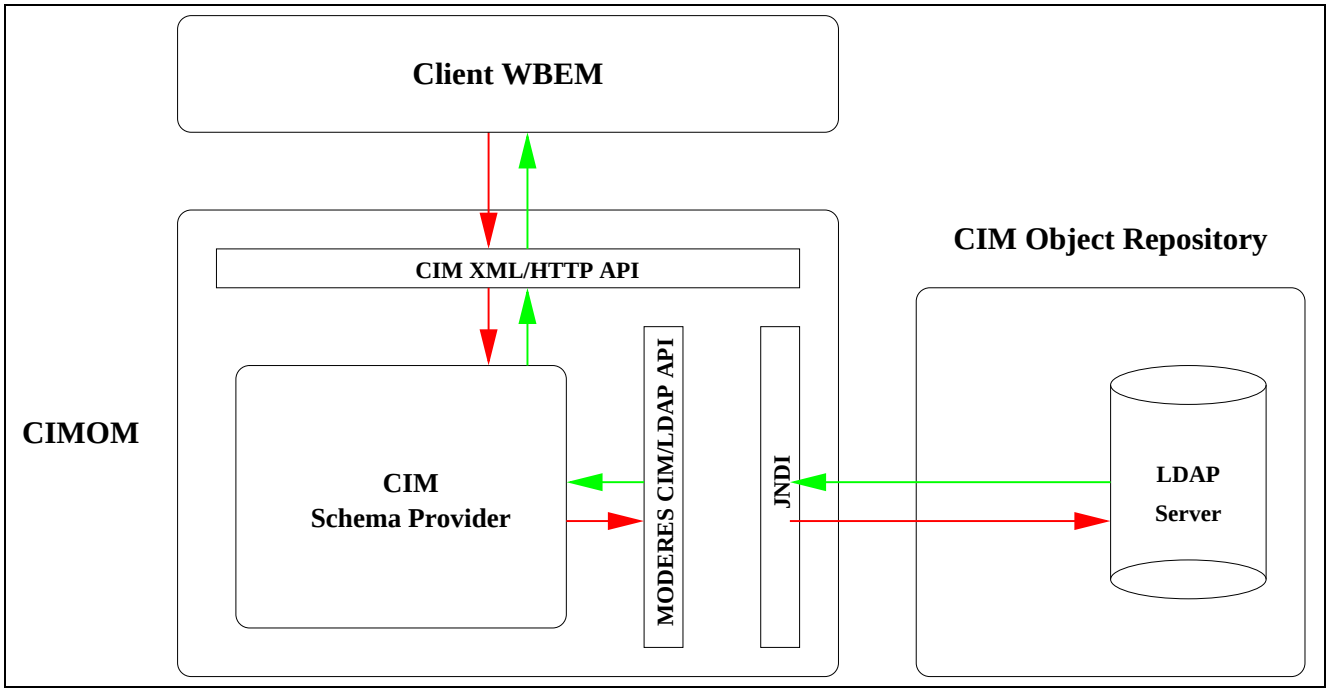


FIG. 30 – *L'implémentation d'un CIMOM*

– d'un ensemble d'exceptions correspondantes aux erreurs CIM définies par le DMTF.

Pour plus de détails sur cette interface nous renvoyons le lecteur à la documentation Java générée (`javadoc`) disponible sur le site de MODERES[5].

6 Conclusions

Les services d'annuaires en général, et les serveurs LDAP en particulier, offrent un moyen simple et efficace pour le stockage et le partage des données statiques des applications distribuées. Dans le monde de la gestion de réseau, cette fonctionnalité est très intéressante pour plusieurs types d'applications. Dans l'approche DEN, poussée par plusieurs indutriels et notamment Cisco, les serveurs LDAP sont utilisés pour le stockage des politiques de gestion issues de certains modèles CIM du DMTF (le modèle de base et le modèle de politiques). Dans le cadre de ce travail, nous proposons une généralisation de cette approche, en utilisant les serveurs LDAP pour le stockage de tout modèle d'information CIM.

Cette étude s'appuye sur la définition d'un schéma LDAP, directement issu du méta-modèle de CIM, permettant de stocker dans une base d'annuaire LDAP des descriptions des modèles CIM. Cette stratégie nous a permis dans un premier temps d'implanter une passerelle CIM/LDAP entre le toolkit MODERES (développé au sein du projet RESEDAS) et un serveur LDAP; et dans un second temps d'implanter la base d'information CIM (*CIM Object Repository*) d'un gestionnaire CIM (*CIMOM*) à l'aide du serveur LDAP.

Références

[DMTF 98a] DMTF. Common Information Model (CIM), Core Model. Rapport, Distributed Management Task Force, August 1998.

[DMTF 98b] DMTF. XML as a Representation for Management Information, a White Paper. Rapport, Desktop Management Task Force, September 1998.

[DMTF 99a] DMTF. CIM XML DTD, version 2.0. Rapport, Distributed Management Task Force, July 1999.

[DMTF 99b] DMTF. Common Information Model (CIM) version 2.2. Rapport, Distributed Management Task Force, June 1999.

5. `http://www.loria.fr/ festor/MODE.html`

[DMTF 99c] DMTF. Specification for CIM Operations over HTTP, version 1.0. Rapport, Distributed Management Task Force, August 1999.

[DMTF 99d] DMTF. Specification for the Representation of CIM in XML, version 2.0. Rapport, Distributed Management Task Force, July 1999.

[Festor 98] O. Festor. The Managed Object Format specification parser of the MODERES Java Toolkit. Rapport technique no. RT-0218, INRIA, Février 1998.

[Festor 99a] O. Festor et N. Ben Youssef. Un analyseur sémantique pour MOF. Rapport Technique no. RT-0233, INRIA, Juillet 1999.

[Festor 99b] O. Festor et N. Ben Youssef. Un Editeur de Spécifications MOF. Rapport Technique no. RT-0234, INRIA, Septembre 1999.

[Festor 00] O. Festor et N. Ben Youssef. WBEM. Rapport Technique no. RR-3927, INRIA, Avril 2000.

[Howes 97a] T. Howes, *The String Representation of LDAP Search Filters*, RFC2254, Décembre 1997.

[Howes 97b] T. Howes et M. Smith, *The LDAP URL Format*, RFC2255, Décembre 1997.

[Judd 98] S. Judd et J. Strassner. Directory-Enabled Networks. Rapport, Fevrier 1998. Draft.

[Moore 00] B. Moore, E. Ellesson et J. Strassner. Policy Core Information Model. Rapport, March 2000. draft-ietf-policy-core-info-model-05.txt.

[Myers 97] J. Myers, *Simple Authentication and Security Layer (SASL)*, RFC2222, Octobre 1997.

[Sun Microsystems 99] Inc. Sun Microsystems. Java Naming and Directory Interface : Application Programming Interface (JNDI API). Rapport, Sun Microsystems, Inc., Juillet 1999.

[Wahl 97a] M. Wahl, *A Summery of the X.500(96) User Schema for use with LDAPv3*, RFC2256, Décembre 1997.

[Wahl 97b] M. Wahl, T. Howes et S. Kille, *Lightweight Directory Access Protocol (v3)*, RFC2251, Décembre 1997.

[Wahl 97c] M. Wahl, T. Howes et S. Kille, *Lightweight Directory Access Protocol (v3) : Attribute Syntax Definition*, RFC2252, Décembre 1997.

[Wahl 97d] M. Wahl, T. Howes et S. Kille, *Lightweight Directory Access Protocol (v3) : UTF-8 String Representation of Distinguished Names*, RFC2253, Décembre 1997.

[X.500 93] ITU-T, *The Directory : Overview of Concepts, Models and Services*, ITU-T. X.500, 1993.

[Yeong 95] W. Yeong, T. Howes et S. Kille, *Lightweight Directory Access Protocol*, RFC2251, Mars 1995.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803