MASTER EN SCIENCES INFORMATIQUES

Etude de l'implémentation des standards ISO de performance au sein d'ISM/OpenMaster

Van Ryckeghem, Frédéric

Award date:
1996

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'informatique
Année académique 1995-1996

Etude de l'implémentation des standards ISO de performance au sein d'ISM/OpenMaster

Frédéric VAN RYCKEGHEM

Promoteur : Jean Ramaekers

Mémoire présenté en vue de
l'obtention du grade de Licencié
et Maître en Informatique

Rue Grandgagnage, 21, B-5000 NAMUR (BELGIQUE)

Résumé

Bon nombre d'entreprises deviennent de plus en plus dépendantes de leur système d'information. Elles en sont tellement dépendantes que leur fonctionnement peut être remis en question lorsque des applications ou des moyens de communication cruciaux se détériorent.

Ceci implique la présence de moyens de gestion efficaces au sein même de l'entreprise. Une majorité des systèmes informatiques sont construits autour d'une architecture distribuée, ce qui ne facilite pas la gestion. Nous parlerons dans un premier temps de ces deux aspects.

La gestion de tels systèmes comprend, entre autres, la gestion des performances qui occupe une place prépondérante dans le processus de gestion. Nous expliquerons cette gestion en détails et nous présenterons également un standard qui définit un modèle à suivre pour mesurer les performances d'un système.

Nous terminerons par la présentation d'un outil de gestion intégrée développé par BULL et à l'application de mesures de performances qu'il contient.

Mots-clés *: système distribué, gestion OSI, gestion des performances, métriques.*

Abstract

Businesses are becoming increasingly dependent on their networked information systems. So dependent, that their ability to function deteriorates rapidly when key communication and application services are interrupted

This implies the presence of management operations within the organization. A great part of computer systems are built around a distributed architecture and it doesn't simplify the management. At first, we will review these two aspects.

Management of such systems consist of, among others things, performance management which takes an important part in the management process. We will detail this kind of management and then we will present a standard which defines a model of systems performance measurement

The end of this work presents an integrated system management platform and his performance management application developed by BULL.

Keywords *:distributed systems, OSI management, performance management, metrics.*

Remerciements

Je tiens à remercier toutes les personnes qui, de près ou de loin, ont contribué à la rédaction de ce mémoire

Je remercie tout particulièrement Jean Ramaekers pour m'avoir permis, grâce aux contacts qu'il a établi, d'effectuer mon stage de maîtrise chez BULL situé aux Clayes-sous-Bois, ainsi que pour ses conseils utiles à la rédaction du mémoire.

Ensuite, Hugues Deghorain, qui m'a accueilli chez BULL et permis de garder de très bons souvenirs de ce stage. Toujours attentif et à l'écoute, il m'a aidé à progresser dans cet environnement qui pour moi appartenait à l'inconnu.

Ensuite, je remercie *également* toute l'équipe avec laquelle j'ai travaillé chez BULL, et plus particulièrement Messieurs Jean-Luc Tesseron, Alain Horte, Raphaël Doummar et Jean-Marie Pleven avec qui j'ai passé de très bons moments.

Je dois aussi ce mémoire aux conseils judicieux et à l'appui de Pascal Gossens et Pascal Libert.

Sans oublier mes parents pour leur soutien durant mes études et sans qui rien ne serait arrivé.

Table des matières

Chapitre 3 La Gestion des Performances 53

Chapitre 4 Les Objets Métriques 65

Chapitre 5 Présentation d'ISM/OpenMaster 77

Chapitre 6 Présentation d'ISM Performance 85

Table des illustrations

Table des tableaux

Introduction

L'information et la communication d'informations sont devenus deux termes incontournables en informatique. Transmission d'informations, stockage de l'information, partage d'informations, ... tout cela s'intègre dans ce que l'on appelle maintenant un système distribué.

Il est clair que nous sommes plongés dans une nouvelle ère, l'ère de l'information. Le phénomène Internet en est un exemple, tout un chacun peut accéder à des services ou à des informations présents à des milliers de kilomètres et cela de manière tout à fait transparente.

Mais même si tout cela paraît simple, il n'en est rien ! Des moyens de gestion considérables doivent être mis en œuvre pour assurer un fonctionnement continu de ce nouvel univers.

Ce mémoire est essentiellement divisé en deux grandes parties, une première partie théorique où nous présenterons un ensemble de concepts qui seront utiles à la compréhension de la seconde partie. Cette seconde partie contiendra une présentation de l'environnement dans lequel s'est effectué notre stage et des développements personnels qui ont été réalisés durant ce stage.

Aussi bien dans la première que dans la seconde partie, nous prendrons un concept en particulier et nous le détaillerons dans le chapitre suivant.

Le premier chapitre sera consacré à la présentation *des systèmes distribués*. Tout le monde en parle, mais savons-nous réellement de quoi il s'agit ? Pourquoi les préférer aux bonnes vieilles architectures centralisées ? Comment sont-ils organisés ? Autant de questions auxquelles nous nous efforcerons de répondre tout au long de ce chapitre.

La communication est un élément de base dans les systèmes distribués. L'ISO - International Standards Organization - propose un modèle de référence à partir duquel on peut bâtir une architecture de communication. Nous aborderons dès lors *la gestion OSI* - Open Systems Interconnection - normalisée par l'ISO. Nous mettrons principalement l'accent sur le cadre architectural et sur la gestion système de la gestion OSI

Le chapitre suivant, étudiera de manière plus détaillée et plus pratique un des aspects de la gestion OSI, *la gestion des performances*. Nous présenterons les principaux indicateurs utilisés dans cette gestion. Ensuite, nous détaillerons *les* trois architectures de gestion de performances qui sont actuellement utilisées dans un environnement distribué.

Pour la suite de ce mémoire, nous devrons prendre connaissance d'un standard, édicté par l'ISO, qui définit *les objets métriques* utilisés dans le cadre d'applications de gestion de performance. Le chapitre 4 commencera par la présentation des termes qui sont fréquemment utilisés dans le domaine de la gestion, et ensuite nous détaillerons le contenu du standard en prenant soin d'expliquer et d'insister sur les aspects les plus importants.

C'est avec le sixième chapitre que nous entamerons la seconde partie, où nous présenterons l'environnement dans lequel nous avons travaillé durant notre stage. Nous expliquerons les caractéristiques principales et le mode de fonctionnement de *la plate-forme ISM/OpenMaster*.

Le chapitre 6 décrira une des applications de cette plate-forme, il s'agit *d'ISM Performance*. Nous présenterons son fonctionnement ainsi que les modifications que nous y avons apporté durant pendant notre stage.

Le septième et dernier chapitre proposera une étude qui envisage de *faire respecter le standard* présenté au chapitre 4 par l'application ISM Performance. Pour ce faire, nous réaliserons une étude de l'existant et nous présenterons également les caractéristiques qui devraient être inclues dans *tout* progiciel de gestion de performances.

Pour terminer, la conclusion résumera les aspects importants de la gestion des performances et présentera l'avenir de cette activité.

Chapitre

1

Les Systèmes Distribués

« Decentralized Computing is sweeping business like a wave rolling onto a beach. Its advance is unstoppable ... »

JOHN DONOVAN

Les systèmes distribués représentent à l'heure actuelle un grand centre d'intérêt. Les utilisateurs travaillent actuellement dans des environnements hétérogènes, et composées d'un ensemble de réseaux (Internet est l'exemple le plus marquant à ce jour). C'est dans ce cadre que vont s'insérer les systèmes distribués.

C'est la raison pour laquelle nous avons décidé de les présenter dans ce mémoire. Nous allons dans un premier temps essayer de les définir et nous remarquerons que ce n'est pas chose aisée. Ensuite nous nous orienterons plutôt vers une énumération des caractéristiques de ces systèmes. Vu les coûts engendrés par le fonctionnement de ces systèmes et l'immense complexité qu'ils représentent[1], nous présenterons brièvement les moyens utilisés pour gérer de tels systèmes.
Nous clôturerons ce chapitre par une présentation de l'évolution de ces systèmes dans les années à venir.

Un essai de définition ...

Nous avons constaté que la plupart des auteurs d'ouvrages traitant des systèmes distribués ne les définissent pas facilement Nous allons présenter une définition d'un système distribué et nous montrerons en quoi cette définition est incomplète et n'englobent pas totalement le concept.

Tanenbaum et Van Renesse ont défini les systèmes distribués de la manière suivante :

« A **distributed** operating system is one that looks to its users like an ordinary centralized operating system, but runs on multiple, independent CPUs. The key concept here is **transparency**, in other words, the use of multiple processors should be invisible to the user. Another way of expressing the same idea is to say that the user views the system as a 'virtual uniprocessor', not as a collection of distinct machines. »[2]

Si nous prenons les termes de cette définition à la lettre, système multiprocesseur serait un système distribué, ce qui n'est pas le cas ! Tanenbaum et Van Renesse ont donc donné une condition nécessaire à l'obtention d'un système distribué mais cette condition ne paraît pas suffisante.

[1] Ne nous effrayons pas à la vue de ces caractéristiques plutôt pessimistes, l'objectif principal d'une architecture distribuée est de mettre à la disposition de ***chaque*** utilisateur les ressources (CPU, données, ...) dont il a besoin de façon à optimiser les performances.

[2] [Mullen,1989]

En effet, il manque des aspects de communication et de tolérance aux pannes qui sont essentiels dans les systèmes distribués.

Nous remarquons donc qu'il serait plus aisé de déterminer la nature d'un système distribué en énumérant une succession de *conditions nécessaires* à l'obtention d'un tel système. C'est ce que nous faisons dans la section suivante.

Ou plutôt une caractérisation ?

Nous pouvons donc remarquer que personne ne s'est encore mis d'accord sur une définition 'universelle' des systèmes distribués. Notre approche va consister en la définition de la centralisation/décentralisation et distribution, que nous allons ensuite affiner en donnant plusieurs caractéristiques que doivent présenter les systèmes distribués.

Un système est *centralisé* si toutes ses composantes sont réunies en un seul site, *décentralisé* lorsque les composantes se trouvent sur plusieurs sites avec un minimum de coordination. Un système est dit *distribué* si ses composantes sont des mécanismes autonomes qui coordonnent leur fonctionnement via un mécanisme de coordination global. Une représentation schématique d'un système distribué est illustré à la Figure 1-1. La communauté scientifique s'accorde sur le fait que les notions de *composantes autonomes* et de *coordination* font partie des caractéristiques essentielles des systèmes distribués.

Nous pourrions comparer un système distribué à un corps humain où les muscles effectuent des mouvements locaux qui sont coordonnés par les signaux provenant du cerveau

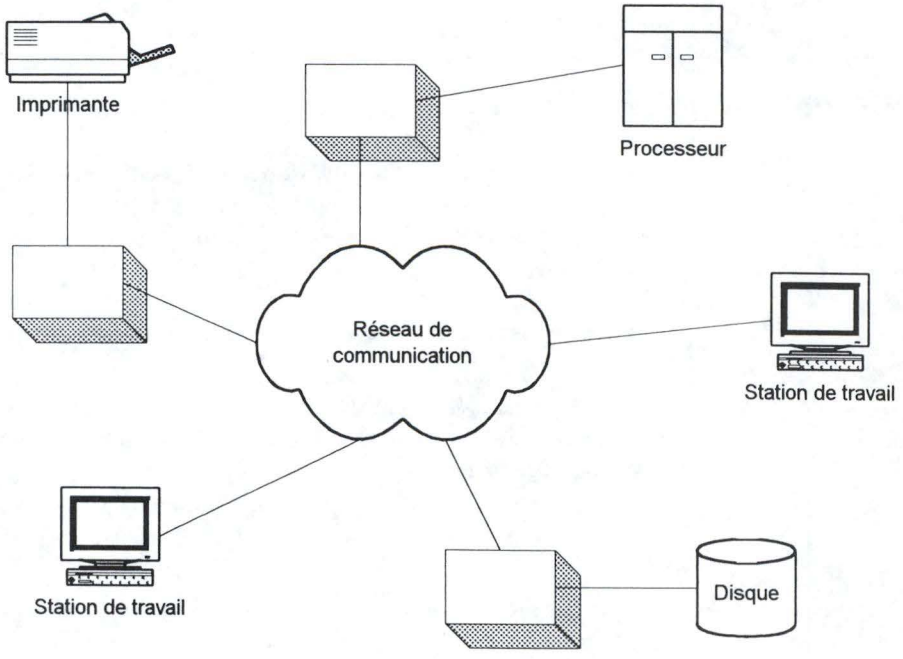


Figure 1-1 : Représentation d'un système distribué

Passons en revue les autres caractéristiques que doivent présenter les systèmes distribués :

- Partage des ressources
- Ouverture (openess)
- Aspects de concurrence
- Evolutif (scalability)
- Tolérance aux pannes

- Transparence

Les systèmes distribués doivent *partager leurs ressources*. Les ressources sont aussi bien matérielles (disques, imprimantes, ...) que logicielles (fichiers, fenêtres, ...). Un système distribué partage ses ressources matérielles afin de réduire les coûts engendrés et pour rendre le système plus confortable. Les ressources logicielles seront quant à elles partagées pour permettre un travail coopératif (le groupware), pour pouvoir réaliser des échanges d'informations par le biais de bases de données et enfin le partage des données va assurer cohérence (consistency) des données.

Les systèmes distribués doivent être *ouverts au monde extérieur*. L'ouverture englobe tout ce qui a trait à la manière d'étendre le système. Le système peut donc être ouvert ou fermé de par ses composantes logicielles ou matérielles. Pour permettre au système d'être ouvert nous nous assurerons de publier ses spécifications et de standardiser les interfaces. UNIX est un système d'exploitation que nous pouvons qualifier d'ouvert, pour s'en assurer, il suffit de constater le nombre d'entreprises qui l'utilise à ce jour.

Un système distribué présente des aspects de *concurrence*. Cette concurrence va permettre la multi-programmation[3] et la possibilité d'utiliser plusieurs processeurs pour réaliser une tâche.
Remarquons à cet effet que les exécutions parallèles sont choses courantes au sein des systèmes distribués. Citons à titre d'exemple :

- Plusieurs utilisateurs peuvent utiliser les mêmes ressources et les applications vont interagir entre elles
- Plusieurs serveurs répondront à une requête envoyée par un client.

Un système distribué est *évolutif*. C'est à dire qu'il doit être capable de gérer toute croissance. Les évolutions du système ne doivent pas engendrer des modifications logicielles. Si des modifications logicielles devaient avoir lieu, elles devraient avoir le moins d'influence possible sur l'ensemble du système distribué.

Un petit système distribué est composé de quelques ordinateurs et d'un serveur de fichiers présents sur un seul réseau.
Pour donner un exemple de gros système distribué, prenons l'exemple d'Internet.

La *tolérance aux pannes* est primordiale au sein d'un système distribué. Cette tolérance aux pannes est assurée par une redondance des composantes matérielles et par un système de récupération des données pour ce qui concerne les aspects logiciels.
Grâce la redondance des composantes matérielles nous avons l'assurance que si un composant tombe en panne, il sera possible de retrouver ailleurs sur le système la même composante. La récupération des données suit la même philosophie que pour les composantes matérielles.
Cette tolérance aux pannes rend le système distribué fiable dans pratiquement tous les cas de figures.

La *transparence* est obligatoire pour que nous puissions parler de système distribué. Cette caractéristique permet à l'utilisateur de voir le système comme formant un tout. Il a l'impression que toutes les ressources qu'il utilise sont contenues dans sa machine. A cet effet, le système distribué pourrait être vu comme ***une*** immense ***machine virtuelle***. Notons que la transparence est assurée à plusieurs niveaux :

- de l'accès aux données

---

[3] Des utilisateurs qui ne se trouvent pas forcément au même endroit pourront programmer différents modules d'une même application.

- de la localisation des machines
- de la concurrence
- de la duplication des données et des ressources matérielles
- des performances
- l'évolution du système

La gestion des systèmes distribués

La gestion d'un système distribué est l'activité qui permet de contrôler et de superviser le système afin qu'il remplisse correctement les tâches qui lui sont attribuées. Lorsque l'on parle de gestion des systèmes distribués, nous pouvons nous attarder sur deux problèmes. Nous pouvons nous poser la question de savoir comment gérer et organiser les technologies (citons par exemple les réseaux et les applications distribuées) qui vont former le système distribué et ensuite, nous pouvons nous demander comment utiliser ces nouvelles technologies afin d'améliorer les processus de gestion.

Nous pouvons appliquer aux systèmes distribués un processus de gestion classique composé des étapes suivantes :

- une phase de planification
- une phase d'organisation
- une phase de développement
- une phase de contrôle de ce qui a été réalisé

Notons également que ce processus de gestion doit s'intégrer dans le gestion générale de l'entreprise mais cela dépasse le cadre de ce mémoire. Il est important de veiller à ce que le gestionnaire du système distribué dispose de tous les éléments nécessaires à la réalisation de sa tâche et que le système distribué lui soit présenté de la manière la plus simple et la plus claire possible.

Nous nous limiterons à ce qui précède en ce qui concerne la gestion des systèmes distribués. Le lecteur intéressé pourra se référer à la littérature spécialisée[4] pour obtenir de plus amples informations et il pourra également lire ce qui est dit sur ***Les Aires Fonctionnelles de Gestion*** aux pages 26 et suivantes de ce mémoire.

L'évolution des systèmes distribués.

Nous allons parler ici de la situation actuelle des systèmes distribués et les attentes futures de ceux ci pour les dix années à venir.

Durant les années 1990, de grandes mutations se sont déjà produites au sein des systèmes distribués. L'engouement pour les télécommunications et les nombreuses recherches effectués dans le domaine des processeurs et des supports de stockage ont permis de mettre à la disposition de la plupart des personnes intéressées les moyens de se connecter à un système distribué.

Il suffit de constater les progrès réalisés en matière de client-serveur, en matière d'informatique mobile avec la prolifération de machine portables et des ordinateurs de bureau d'une puissance extraordinaire et à des prix qui le sont encore plus.
Mais ce que l'on retiendra le plus pour ces années 1990, c'est le phénomène client-serveur qui constitue la forme la plus populaire d'informatique distribuée.

[4] Voyez à ce propos [Sloman,1994]

Les recherches effectuées à l'heure actuelle tendent à créer des *'supercomputers'* qui seraient reliés par des réseaux pouvant atteindre des vitesses de plusieurs Gigabits par seconde. Les recherches effectuées en informatique neuronale sont également très intéressantes en matière de systèmes distribués. En effet, un cerveau humain est constitué de plus ou moins un milliard de neurones reliés par des nerfs (synapses). Si on assimile les neurones à un CPU (même si les neurones sont beaucoup plus lents que les CPU) et les synapses à un moyen de communication ... on arrive à la notion de réseaux neuronaux.

Imaginons dès lors la connexion d'ordinateurs ultra rapides par des réseaux ayant des vitesses incroyables ... et nous pouvons voir là les opportunités qui se présentent pour des nouvelles applications dans les domaines de l'intelligence artificielle, de la médecine, de la finance, de la gestion , etc.

Conclusion

L'informatique distribuée est un phénomène assez récent et les années 1990 l'ont vu s'épanouir de plus belle. Si nous y regardons de plus près, nous remarquerons qu'à terme pratiquement toute informatisation aura recours à cette distribution car elle présente bien plus d'avantages que d'inconvénients.

Il nous semble que le fait de ne pouvoir donner une définition bien précise d'un système distribué montre les possibilités qu'offre une telle architecture. Il suffit pour s'en convaincre d'énoncer ses points forts (ouverture, partage des ressources, transparence, évolution, tolérance aux pannes, etc.).

Les immenses progrès réalisés en matière informatique aidant et l'internationalisation des échanges font que la distribution a encore de beaux jours devant elle.

Chapitre

2

La Gestion OSI

« Everything should be as simple as possible, not simpler »

ALBERT EINSTEIN

L'objet de ce chapitre est d'expliciter et d'analyser la *gestion OSI (Open System Interconnexion)* . L'explication de la gestion OSI peut s'avérer être utile au lecteur soucieux d'acquérir les connaissances suffisantes pour la compréhension des chapitres qui suivront.

Nous présenterons tout d'abord le cadre architectural de la gestion OSI, en nous attardant sur les différents constituants de cette architecture. Ensuite, nous parlerons de l'aspect principal de la gestion OSI, la gestion-système en énumérant les différentes fonctionnalités.

Le cadre architectural pour la gestion OSI

Modèle de référence

Le quatrième additif au Modèle de Référence [ISO7498-4] détermine un modèle pour l'élaboration de normes de gestion OSI. Il spécifie les concepts architecturaux de base en structurant la gestion en fonctions. Il définit les aspects de la gestion des systèmes d'information qui entraînent des échanges protocolaires entre systèmes ouverts. Ces derniers peuvent être situés à trois niveaux différents du Modèle de Référence. Un protocole manipulant de l'information de gestion dans la couche Application est dénommé protocole de *gestion-système*. Dans les autres couches OSI, on distingue les protocoles de *gestion de couche (N)* des *protocoles de couche (N)*[5].

Pour ces trois types de gestion, illustrés par la Figure 2-1, la normalisation porte sur les services et sur les protocoles utilisés pour transférer les informations de gestion, sur la syntaxe abstraite, et sur la sémantique manipulée par les protocoles de gestion.

La Gestion - Système

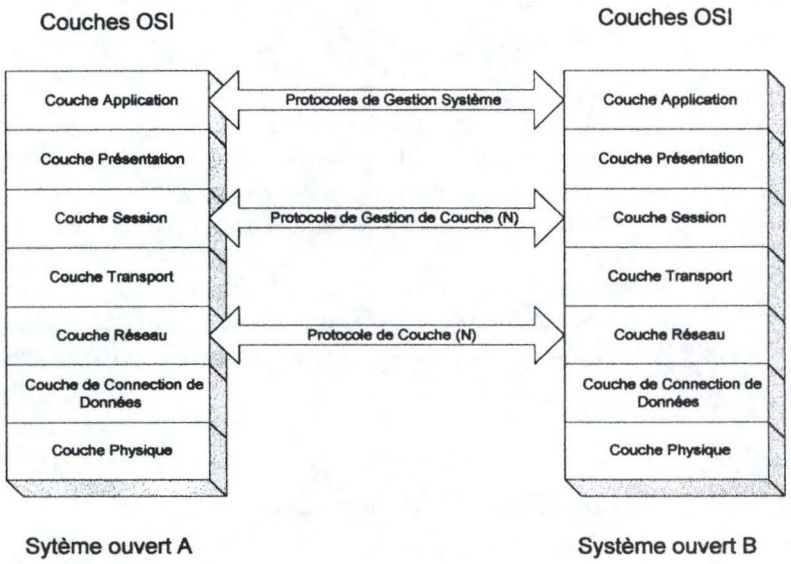
La gestion-système permet un contrôle global des ressources utilisées dans le réseau par un système particulier.

Elle fournit des mécanismes pour la surveillance, le contrôle et la coordination des objets de gestion par l'utilisation de protocoles de gestion-système de la couche Application. Elle manipule de l'information relative à l'ensemble des sept couches d'un système ouvert. C'est la manière privilégiée de réaliser des échanges de données de gestion entre entités homologues de gestion distantes (de la couche 7) dénommées SMAE (**S**ystem **M**anagement **A**pplication **E**ntity) ou entités d'application de gestion-système.

[5] N détermine une couche OSI de niveau 1 à 6.

Les protocoles au niveau application sont basés sur des services fiables et en mode connecté. C'est d'ailleurs la raison pour laquelle on surnomme de moyen d'échanger de l'information de gestion la *'route royale'*. La décision d'utiliser les protocoles de la couche Application est basée sur l'hypothèse qui dit que les informations de gestion doivent être échangées comme tout autre type d'information. Cette optique implique que la gestion devrait être considérée comme une application au sommet du réseau.

Figure 2-1 Types de protocoles véhiculant les informations de gestion



La Gestion de Couche (N)

Bien que la gestion-système soit considérée comme la méthode préférée pour échanger des informations de gestion, ce n'est pas la seule méthode. Un protocole de gestion de couche (N) manipule l'information de gestion relative aux ressources de la couche (N). La gestion de couche (N) supervise le fonctionnement d'un niveau particulier de l'architecture OSI, au moyen des communications entre entités de gestion de couche(N) homologues.

Elle fournit des mécanismes de surveillance, de contrôle et de coordination des ressources de gestion, liés aux activités de communication dans la couche (N) (par exemple, la communication de paramètres et d'informations décrivant des anomalies). Le contrôle des échanges de données sur plusieurs connexions ou instances de communication est possible. Selon le principe de répartition de tâches, chaque couche possède un rôle déterminé et rend un service différent de celui des autres. Les services de gestion de couche (N) ne doivent donc pas être redondants avec d'autres fonctions OSI. Ainsi, chaque protocole de gestion de couche (N) est indépendant des autres.

Un distinction importante entre la gestion-système et la gestion de couche (N) est que la gestion-système utilise le service de présentation pour l'échange d'information, alors que la gestion de couche (N) utilise les services de la couche (N-1)

Remarque : De manière générale, on peut dire que la gestion de couche (N) ne devrait être utilisée que si les protocoles de gestion-système sont indisponibles ou inappropriés à des besoins spéciaux.

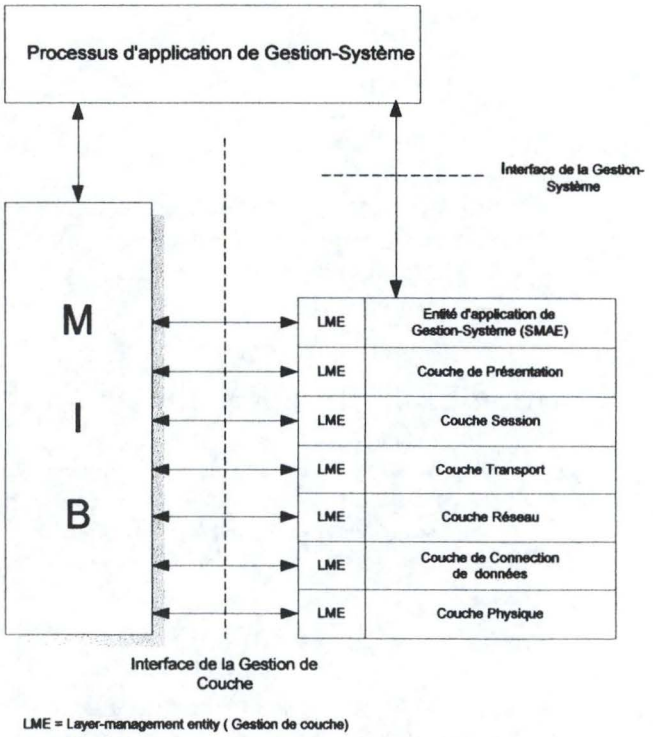
Exemple : La gestion de couche est souvent utilisée pour l'échange d'informations de routage. Dans de nombreux cas les informations de routage doivent être diffusées dans tout un domaine de routage. Vu que la couche présentation ne contient pas de service permettant une diffusion des messages (Broadcast), il sera moins performant d'utiliser la gestion-système et donc on se basera alors sur des protocoles de gestion de couche.

Les gestionnaires de couche sont responsables du maintien de la concordance entre les informations contenues dans la base d'informations de gestion (MIB, elle détaillée dans la section suivante) et les objets gérés, c'est ce qu'indique la Figure 2-2. Ce qui signifie qu'en cas de problème avec les gestionnaires de couche, il se peut que l'information contenue dans la MIB ne reflète pas exactement la réalité.

L'Opération de Couche

Le dernier cas d'échange d'informations administratives, véhiculées par un protocole OSI de niveau (N), est défini par des opérations sur des ressources de la couche (N), pour *une instance* de communication *unique*[6]. Il n'y a pas de spécification de protocole spécifique de gestion. Le pouvoir d'action de ces opérations est alors plus limité dans le temps que celui de la gestion de couche (N).

Figure 2-2 : Modèle architectural de la gestion OSI



La Base d'Informations de Gestion

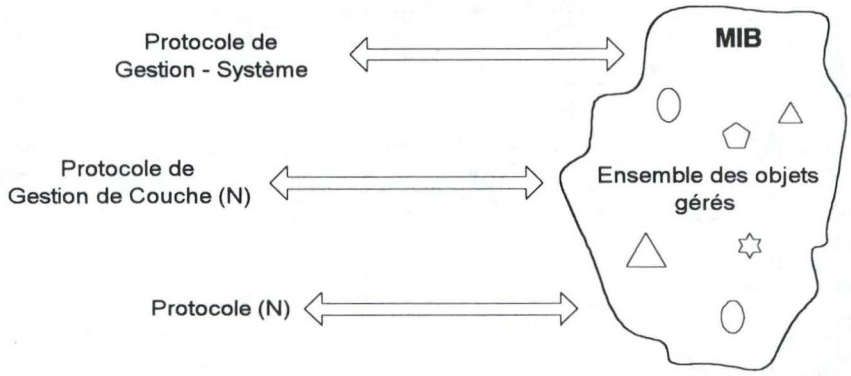
La base d'informations de gestion (ou MIB pour **M**anagement **I**nformation **B**ase) représente, dans un système ouvert, l'ensemble conceptuel des informations de gestion.

[6] Une seule instance de communication est une connexion unique dans le cas d'un service en mode connecté et c'est une seule demande-réponse dans le cas d'un service en mode non-connecté.

La MIB est considérée comme un ensemble d'objets gérés accessibles par trois types de gestion (cfr. Modèle de référence, page 23). Lorsque l'on parle des objets de gestion, seuls les objets de gestion relatifs à l'environnement OSI entrent dans la cadre de la normalisation qui ne s'applique qu'a leur structure logique. De même, la normalisation ne spécifie pas les formes physiques $^{et}/_{ou}$ logiques de stockage de l'information dans la base, ni même l'état des données sauvegardées (état brut ou après traitement). Seule la *syntaxe abstraite* des données de la MIB importent lors des échanges protocolaires.

Il n'y a donc pas de contrainte de normalisation sur la structure interne ou sur l'organisation locale, ni d'impératif de mise en œuvre. La Figure 2-3 met en évidence les types d'accès possibles à la base d'informations.

Figure 2-3 : Accès par des systèmes ouverts distants à la MIB par trois types de protocoles.



Ainsi les processus de gestion reçoivent et contrôlent les informations de gestion provenant de personnes ou de logiciels agissant comme agents administratifs du système local et des systèmes distants, via les entités d'application de gestion-système, les entités de gestion de couche (N) et les entités (N).

Les Aires Fonctionnelles de Gestion

D'un point de vue fonctionnel, les trois niveaux de gestion cités ci-dessus peuvent être structurés en cinq aires fonctionnelles (ou SMFA pour **S**pecific **M**anagement **F**unctional **A**rea) : la gestion des anomalies, la gestion des informations comptables, la gestion de la configuration, la gestion des performances et la gestion de la sécurité.

La Gestion des Anomalies

Afin de garder un réseau complexe en état de fonctionnement correct, on doit veiller à ce que tout le système fonctionne ainsi que chaque composant pris individuellement. Lorsqu'une anomalie est détectée, il est primordial d'y répondre aussi vite que possible :

- En déterminant la localisation exacte de l'anomalie
- En isolant le reste du réseau de l'anomalie afin qu'il puisse encore fonctionner
- En reconfigurant le réseau en vue de minimiser l'impact du (ou des) composant(s) manquant(s) sur le fonctionnement du réseau.
- En réparant ou en remplaçant les composants défectueux pour que le réseau retrouve son état initial.

La gestion des anomalies[7] (ou Fault Management) permet au gestionnaire de réseau de détecter les problèmes de communications sur le réseau et dans l'environnement OSI. Ces moyens permettent de détecter, d'isoler et de corriger les opérations anormales au sein de n'importe quel composant du réseau ou des couches OSI. La gestion des anomalies comprend les fonctions suivantes :

1. Détecter et rapporter la présence d'anomalies. Ces procédures vont permettre au système géré d'avertir son administrateur (manager) de la présence d'anomalies par le mécanisme de notification (*l'event-reporting*),en utilisant un protocole standard.

2. Inscrire dans un fichier de log les événements reçus. Ce fichier de log peut être consulté ou être l'objet d'un traitement.

3. Programmer et exécuter des tests de diagnostic, tracer les anomalies et lancer la correction de ces anomalies. Ces procédures peuvent être lancées comme résultat de l'analyse du fichier de log.

La Gestion d'Informations Comptables

La gestion d'informations comptables (ou Accounting Management) est l'ensemble des fonctions permettant d'établir les factures d'utilisation des ressources de l'OSIE (**OSI E**nvironment) et d'identifier les coûts liés à l'utilisation des ressources.

Ces ressources peuvent être, le fournisseur du service de réseau qui est responsable du transfert des données de l'utilisateur ou encore les applications du réseau.

La gestion d'informations comptables comprend les fonctions suivantes :

1. Informer les utilisateurs sur les coûts encourus ou sur les ressources utilisées par le mécanisme de rapport d'événement (*event reporting*)

2. Permettre de fixer des limites comptables et des prévisions de tarif, associées à l'utilisation de ces ressources

3. Permettre un regroupement des coûts lorsque plusieurs ressources sont utilisées pour mener à bien une communication

La Gestion de la Configuration

La gestion de la configuration (Configuration Management), encore appelée gestion des noms [Stallings,1993], regroupe l'ensemble des moyens qui permettent au gestionnaire de réseau d'exercer un contrôle sur la configuration des composantes du réseau ou sur les entités des couches OSI.

La gestion de la configuration est considérée comme étant ***le*** processus ***central*** de la gestion de réseau. En effet, toutes les autres aires fonctionnelles nécessitent les données fournies par la gestion de la configuration.

Un des aspects importants de la gestion de la configuration est l'assignation de noms, c'est la raison pour laquelle on utilise parfois le terme de gestion des noms. La configura-

---

[7] Les concept d'anomalie et d'erreur sont différents. Une *anomalie* est une situation anormale nécessitant un mécanisme de gestion pour être réparée. Une anomalie est généralement constatée lors d'un dysfonctionnement ou lorsqu'il y a trop d'erreurs. L'*erreur* résulte de problèmes survenus durant une exécution, un transfert ...

tion peut être modifiée afin de diminuer la congestion, d'isoler des anomalies ou tout simplement pour satisfaire les nouveaux besoins des utilisateurs. La gestion de la configuration comprend les fonctions suivantes :

1. Rassembler et mettre à la disposition des autres aires fonctionnelles les données relatives à l'état courant des ressources. Les changements effectués sont communiqués aux moyens de gestion par l'intermédiaire de protocoles standards.

2. Assigner ou modifier les paramètres relatifs aux composantes du réseau ou logiciels des couches OSI.

3. Changer la configuration.

4. Associer un nom à un ensemble d'objets.

La Gestion des Performances

La gestion des performances (Performance Management) est l'ensemble des moyens permettant au gestionnaire du réseau de surveiller, d'évaluer les performances du système et des entités des différentes couches.

La gestion des performances sera l'objet du chapitre 3, nous renvoyons donc le lecteur à ce chapitre pour obtenir plus d'informations. La gestion des performances comprend les fonctions suivantes :

1. La collection et répartition des données concernant le niveau de performance courant des ressources du réseau.

2. Tenir à jour et examiner les journaux chronologiques de l'état du système pour des besoins tels que la planification ou l'analyse.

3. Modification des modes de fonctionnement du système pour mener des activités de gestion de performance.

4. Définition de la performance du système dans des conditions naturelles et artificielles

La Gestion de la Sécurité

La gestion de la sécurité (ou Security Management) est l'ensemble des moyens permettant au gestionnaire de réseau de gérer les services autorisant l'accès aux ressources du réseau. La gestion de la sécurité comprend les services suivants :

1. Des moyens d'autorisation

2. Le contrôle d'accès

3. La gestion du chiffrement et des clés

4. L'authentification

5. Le compte rendu d'événements relatifs à la sécurité

6. Diffusion des informations relatives à la sécurité

La Gestion - Système

Comme nous l'avons signalé auparavant, la gestion-système est l'un des 'éléments clé' de la gestion OSI, c'est la raison pour laquelle nous allons l'examiner plus en détail dans les pages qui suivent.

La description de la gestion-système fera la distinction entre les quatre aspects suivants :

- informationnels
- organisationnels
- fonctionnels
- de communication

Les sections suivantes seront consacrées à chacun de ces aspects. Le point de départ de discussion sera le standard [ISO10040] mais nous ne nous limiterons pas à ce dernier car nous ferons également référence aux standards qui en sont dérivés et ces références seront également expliquées.

Aperçu Général et Architecture

La gestion-système est considérée comme étant une application. Puisque l'environnement à gérer est distribué, les composants individuels des activités de gestion-système sont eux aussi répartis. Les processus de gestion-système exécutent donc leurs activités de façon répartie et coopérante.

On distingue deux types de processus de gestion : les gestionnaires et les agents. Les *gestionnaires* ont la responsabilité d'une ou de plusieurs activités de gestion. Ils effectuent des opérations et reçoivent des notifications (résultant éventuellement d'opérations). Les *processus agents* exécutent des opérations de gestion sur les objets gérés qui sont situés dans la MIB. Ils peuvent émettre des notifications d'événements ou faire suivre des notifications d'événement émises par les objets gérés.

Ces deux entités sont couramment appelées MIS-Users dans le cadre de la gestion OSI. Toute interaction a lieu entre deux MIS-Users, l'un prenant le rôle de gestionnaire et l'autre prenant le rôle d'agent. Ces interactions sont illustrées par la Figure 2-4.

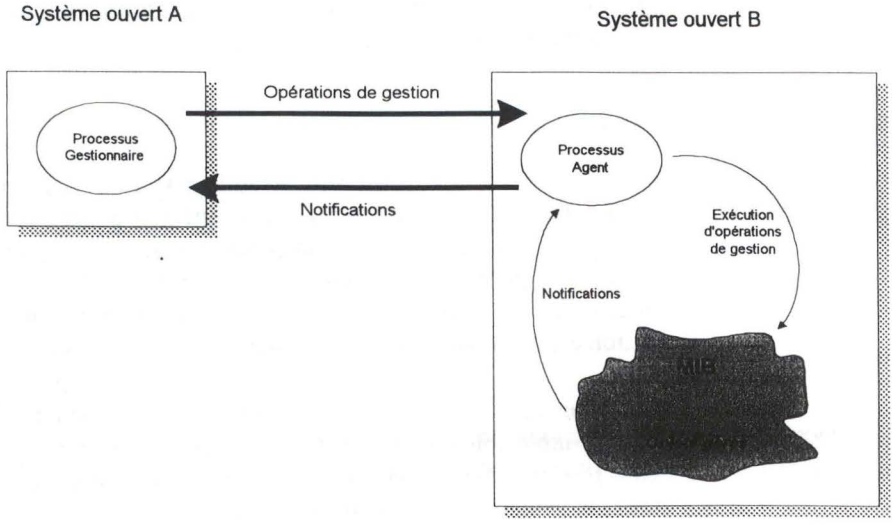
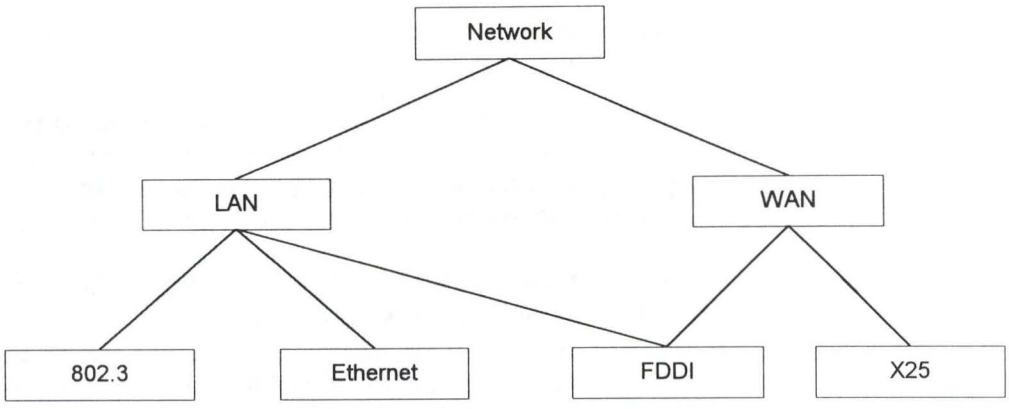


Figure 2-4 : Processus gestionnaire et processus agent

ses différentes d'objets gérés peuvent également partager certaines propriétés. On parlera d'héritage simple ou multiple si une classe, dite *sous-classe*, hérite des propriétés d'une ou plusieurs autres classes dénommées alors *super classes*. Lorsqu'une sous classe hérite de toutes les propriétés d'une super classe, l'héritage sera qualifié de *strict*. La notion d'héritage permet en fait de spécialiser et de raffiner une classe en définissant une sous classe (notion de spécialisation).

Les liens de parenté des classes peuvent se modéliser par une structure arborescente dite *arbre d'héritage*. Le sommet de l'arbre d'héritage qui est la classe objet dont toutes les autres héritent directement ou indirectement, est appelée « TOP ». La Figure 2-6 illustre une partie de cet arbre.

Figure 2-6 : Un arbre d'héritage



Les opérations

Les opérations sont appliquées aux attributs des objets gérés ou alors à l'objet géré dans son entièreté. Une opération effectuée sur un objet géré ne peut réussir que si le système administrateur a les droits d'accès nécessaires pour réaliser l'opération et si les contraintes de consistance ne sont pas violées.

Le comportement

Le comportement est caractérisé par certaines caractéristiques comportementales, y compris la manière de réagir lorsque les opérations sont effectuées. La modification du comportement peut être due à des stimuli aussi bien internes qu'externes.

Pour ce qui concerne les *stimuli internes*, il s'agit d'événements internes à l'objet géré et à la ressource qui y est associée, comme par exemple des horloges.

Passons maintenant aux *stimuli externes* qui consistent en des opérations de gestion-système transmises sous la forme de *messages*[8].

Toutes les instances d'une classe d'objets gérés ont la même définition. Le comportement défini :

- La sémantique des attributs, des opérations et des notifications
- La réponse aux opérations de gestion effectuées sur l'objet géré
- Les circonstances dans lesquelles des notifications doivent être émises
- Les dépendances entre les valeurs de certains attributs

---

[8] Nous verrons par la suite qu'il s'agit de messages utilisant le protocole CMIP (cfr. Protocole commun d'information de gestion CMIP à la page 41).

- Les implications des relations entre les objets gérés participants

Les notifications

Les objets gérés doivent émettre des notifications lorsque certains événements se manifestent.

Les notifications peuvent être transmises à l'extérieur via un protocole ou alors elles peuvent être rapportées dans un journal. Les systèmes gestionnaires peuvent demander que tout ou partie des notifications leurs soient envoyées par l'objet géré. Les notifications ainsi envoyées sont contenues dans un rapport d'événement (event report).

Les paquets conditionnels

Un paquet conditionnel est un paquet formé d'attributs, de notifications, d'opérations et de comportements optionnels qui sont ou bien tous présents ou bien tous absents dans un objet géré.

Notion d'allomorphisme

L'allomorphisme, tel que défini dans la gestion OSI, est un cas particulier du concept de polymorphisme utilisé dans les approches orientées objets.

L'allomorphisme renvoie à la possibilité pour une instance d'une sous-classe (appelée sous-classe allomorphique) d'émuler ou de ressembler au comportement de sa super-classe (appelée super-classe allomorphique).

On utilise ce concept notamment lors de l'évolution des MIB, en effet, un nouvel objet peut être défini pour émuler le comportement d'un ancien.

Principes de Contenance et de Dénomination

Nous avons déjà vu précédemment un moyen de représenter les classes et leur position les unes par rapport aux autres grâce à l'arbre d'héritage (cfr. Figure 2-6). Mais cet arbre d'héritage n'était simplement qu'un moyen de représenter l'ensemble des objets sans écrire trop de texte et il ne reflétait pas la structure réelle de la MIB. Cette structure sera définie en utilisant la notion de *contenance*.

La structure de contenance

La contenance permet à un objet de *contenir* un ou plusieurs autres objets. La contenance est représentée par l'inclusion de la référence de l'objet subordonné dans l'objet supérieur (l'objet contenant).

Un objet subordonné ne peut être contenu que par un seul supérieur, obligeant ainsi la MIB à avoir une structure d'arbre. Par contre, un objet contenant peut être lui-même contenu dans un autre objet, permettant ainsi la construction d'un arbre de profondeur arbitraire.

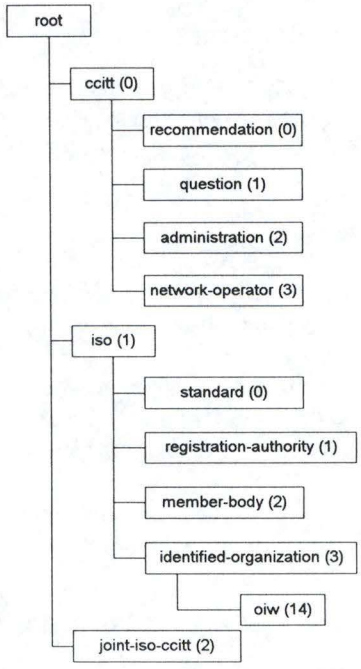
La dénomination

Comme nous venons de le voir, il y a une différence entre la hiérarchie d'héritage et de contenance, l'une représentant les relations entre les classes d'objets et l'autre représentant les relations entre les instances des objets au sein de la MIB. Cette distinction se fait également dans la manière de désigner les objets.

Considérons tout d'abord les classes d'objets. Chaque classe d'objet présente dans l'arbre d'enregistrement (Registered Tree) est identifiée par un identifiant unique.

Chaque identifiant est une succession d'entiers obtenue en parcourant l'arbre de enregistrement[9] (cfr. Figure 2-7).

Figure 2-7 : Arbre d'enregistrement.



Par exemple, les objets enregistrés sous BULL auront un identifiant d'objet commençant par 1 3 6 1 4 1 107.

Le schéma de désignation (naming)pour les instances d'objets est différent, il se base sur les relations de contenance.
Tout d'abord, chaque classe d'objet comprend un attribut qui est utilisé pour désigner les instances de cet objet. Ensuite nous allons introduire deux nouvelles notions, celle de *relative distinguished name* et celle de *distinguished name.*

Le *relative distinguished name (RDN)* d'une instance d'objet correspond à une assertion sur la valeur de l'attribut de désignation, utilisée pour désigner un objet dans l'arbre de contenance. Ce RDN possède une valeur non ambiguë dans le domaine de dénomination du supérieur auquel il est rattaché.

Le *distinguished name (DN)* correspond à la concaténation de tous les RDN depuis la racine. C'est ce qui donne le nom global de l'objet géré.

---

[9] C'est le même mécanisme utilisé en SNMP (Simple Management Protocol) excepté qu'en SNMP, il n'y a pas de différence entre les classes d'objets et les instances d'objet.

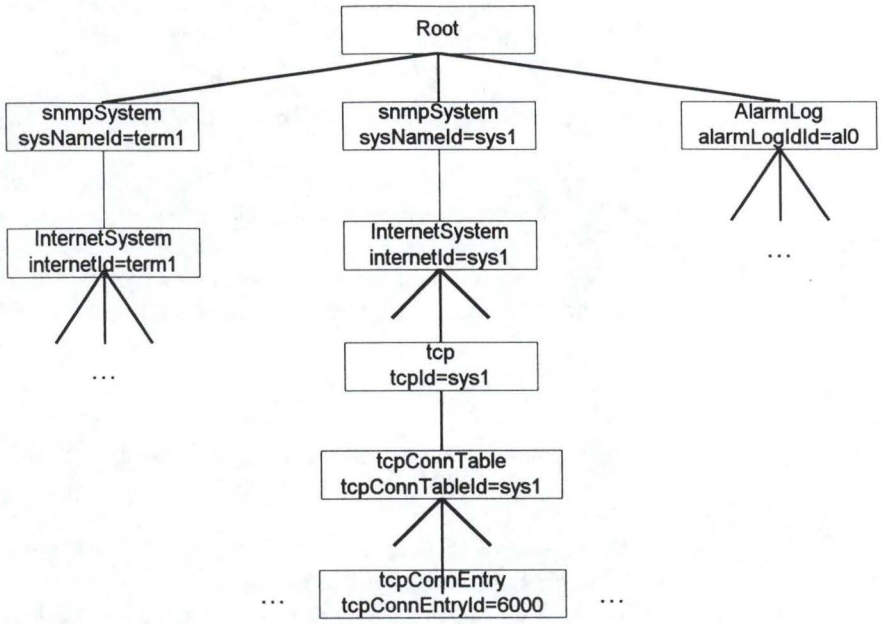


Figure 2-8 : Un arbre de contenance avec ses RDN

La Figure 2-8 nous montre un exemple d'arbre de contenance. Pour chaque instance d'objet, le nom de sa classe d'objet et son RDN sont montrés.

Relative Distinguished Name	*Distinguished Name*
snmpId=term1	snmpId=term1
InternetId=sys1	snmpId = sys1,InternetId=sys1
tcpConnEntryId=6000	snmpId = sys1,InternetId=sys1,tcpId=sys1, tcpConnTableId=sys1, tcpConnEntryId=6000

Il est important de noter que les noms des instances des objets gérés est créé lorsque l'instance est créée donc qu'il n'est pas nécessaire d'enregistrer ou de rendre ces noms publiques.

Les trois 'arbres' de la gestion OSI.

Vu l'importance des concepts précédents dans la gestion OSI, il est bon de les synthétiser en quelques lignes :

Il y a donc trois types *différents* et *indépendants* d'arbres utilisés dans la gestion OSI :

1. ***L'arbre d'enregistrement :*** c'est une sorte d'index qui donne une vue unique des identifiants d'objets. L'identifiant d'objet est en fait le cheminement de la racine jusqu'à la localisation de l'objet dans l'arbre.

2. ***L'arbre d'héritage :*** Cet arbre montre comment la définition des classes d'objet peut être dérivée d'autres classes d'objets en utilisant des principes orientés objet. Une réutilisation des classes d'objets gérés avec un raffinement de leurs définitions.

3. ***L'arbre de contenance :***Cet arbre représente la structure de la MIB. Il montre les objets gérés contenus par un agent et la 'hiérarchie' de ces objets. Cet arbre n'est

pas seulement utilisé pour définir la MIB mais également pour référencer sans équivoque possible les instances d'objets gérés.

Les opérations de gestion-système

Les opérations sont également spécifiées. Elles sont envoyées par l'intermédiaire de messages et elles sont effectuées sur des objets.

Nous pouvons distinguer deux types d'opérations, celles qui sont appliquées sur les attributs de l'objet et celles qui sont appliquées sur l'entièreté de l'objet.

Les opérations *orientées attribut*

Les opérations suivantes peuvent être appliquées à un ou plusieurs attributs d'un objet :

- Prendre la valeur d'un attribut
- Remplacer la valeur d'un attribut par sa valeur par défaut
- Assigner une valeur à un attribut
- Ajouter un 'membre' à un attribut ensemble (set-value attribute)
- Retirer un 'membre' à un attribut ensemble

Les opérations *orientées objet*

Les opérations suivantes peuvent être effectuées sur un objet :

- La création
- L'effacement
- Une action sur l'objet

La sémantique de ces opérations fait partie de la définition de la classe d'objet et les effets sur les objets corrélés doivent également être spécifiés.

Les aspects organisationnels

La gestion OSI est organisée d'une manière centralisée. Un processus gestionnaire peut contrôler plusieurs agents. Le processus gestionnaire effectue des opérations sur ses agents et les agents se chargent de renvoyer les notifications vers leur processus gestionnaire, c'est le concept *Gestionnaire-Agent (Manager-Agent)*.

Nous avions vu que l'environnement de gestion OSI pouvait être partitionné en un certain nombre de domaines de gestion (Management Domains). Ce partitionnement se base sur des besoins fonctionnels (par exemple, la sécurité, la comptabilité et la gestion des anomalies), mais aussi sur d'autres besoins (par exemple, géographiques et technologiques).

Les aspects fonctionnels

La rédaction des brouillons (drafts) du cadre architectural a été directement suivie par la rédaction de protocoles standards pour chacune des cinq aires fonctionnelles.

Peu de temps après, ils ont remarqué que la plupart des protocoles relatifs aux aires fonctionnelles utilisaient des ensembles similaires de fonctions élémentaires. C'est pourquoi, ils ont décidé d'arrêter la spécification des protocoles des aires fonctionnelles et qu'ils se sont concentrés sur la définition de fonctions de gestion élémentaires, que nous

décrirons dans la suite de ce chapitre dans la partie réservée aux *fonctions de gestion-système.*.

Les aspects liés aux communications

OSI a défini le '**C**ommon **M**anagement **I**nformation Service **E**lement' (**CMISE**) comme étant le service de prédilection pour l'échange des informations de gestion.

Le rôle de CMISE est limité au transfert d'informations de gestion, le contrôle réel du système est laissé aux MIS-users se trouvant au-dessus de CMISE.

CMISE est spécifié en deux étapes :

A. *L'interface avec l'utilisateur*[10], spécifiant les services offerts. C'est le Common Management Information Service (CMIS) défini dans [ISO9595].

B. *Le protocole*, spécifiant le format des Protocol Data Unit (PDU) et les procédures associées. C'est le Common Management Information Protocol (CMIP) défini dans [ISO9596-1].

La Figure 2-9 nous montre les contextes d'utilisation de CMIS et CMIP. Comme nous le verrons au point suivant, CMIS offre sept services pour effectuer les opérations de gestion, sous forme de primitives de services.

Pour décrire ces deux modules, CMIS et CMIP, nous nous inspirerons de [ISO9595] et de [ISO9596].

Elément de service commun d'information de gestion CMISE

CMIS [ISO9595]définit les services offerts pour la gestion OSI. Ces services sont accessibles par des processus de gestion qui veulent communiquer à distance.

Les services CMIS sont spécifiés en termes de primitives qui peuvent être vues comme des 'appels de procédure'. Ces services sont de deux types. Soit ils sont confirmés auquel cas il doit y avoir une sorte d'accusé de réception avec une indication pour voir si l'opération a réussi ou non. Les services non confirmés quant à eux ne demandent pas d'accusé de réception.

[10] Il ne s'agit pas d'une interface utilisateur au sens de l'interface Homme-Machine.

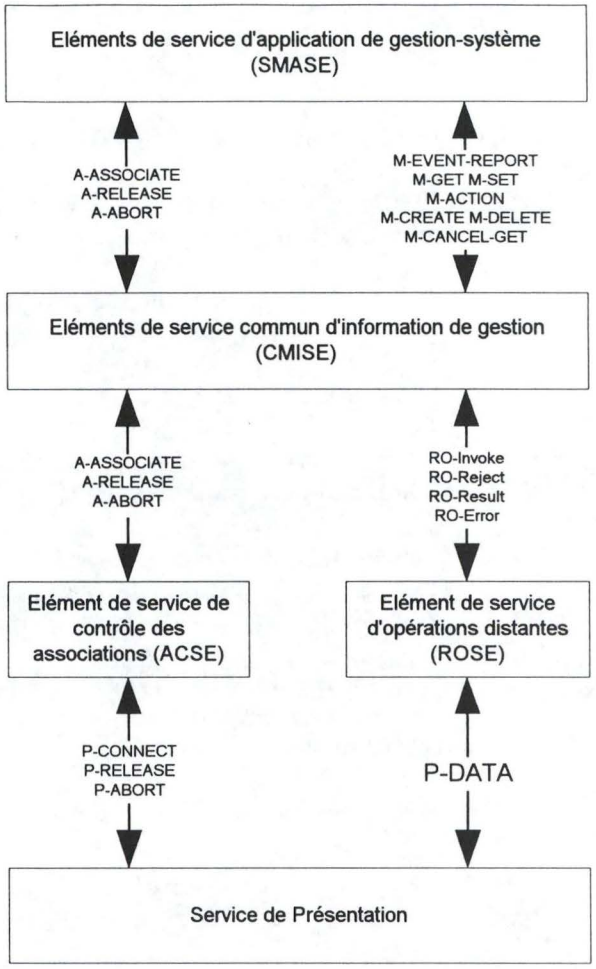


Figure 2-9 : Eléments de service offerts et utilisés par CMISE

Les services CMIS peuvent être classés dans trois catégories :

1. **Les services d'association** : *(Association Service)*
   Les utilisateurs CMIS doivent établir une association d'application afin de communiquer. Ils se basent sur les éléments de service de contrôle d'association pour contrôler les associations d'application (ACSE pour Application Control and Service Element).

2. **Les services des gestion des notifications** *(Management-notification Service)*
   Ce service est utilisé pour véhiculer des informations qui sont des notifications. La définition de la notification ainsi que le comportement des entités communicantes dépend de la spécification de l'objet géré qui a émis la notification. Ils sont présentés dans le Tableau 1.

3. **Les services de gestion des opérations** : *(Management-operation Service)*
   Ils sont au nombre de six et ont pour fonction de véhiculer les informations de gestion applicables aux opérations de gestion. Ces services sont présentés dans le Tableau 1.

Tableau 1 : Les services CMIS

Les services de gestion des notifications		
Service	***Type***	***Définition***
M-EVENT-REPORT	Confirmé/non confirmé	Fait part d'un événement survenu pour un objet géré *Exemple :* Utilisé pour signaler qu'une des liaisons est coupée.
Les services de gestion des opérations		
Service	***Type***	***Définition***
M-GET	Confirmé	Demande l'extraction d'une information de gestion *Exemple :* on veut prendre l'adresse réseau d'un agent.
M-SET	Confirmé/non confirmé	Demande la modification d'une information de gestion *Exemple :* Utilisé pour changer l'adresse réseau de l'agent
M-ACTION	Confirmé/non confirmé	Demande l'exécution d'une action particulière. *Exemple :* utilisé par le gestionnaire pour réinitialiser un autre réseau
M-CREATE	Confirmé	Sert à créer une nouvelle instance d'un objet *Exemple :* Utilisé pour ajouter une entrée à une table de routage
M-DELETE	Confirmé	Sert à supprimer une instance d'un objet. *Exemple :* Utilisé pour retirer une entrée dans une table de routage
M-CANCEL-GET	Confirmé	Sert à annuler un M-GET *Exemple :* Utilisé par exemple lorsqu'un M-GET renvoie trop d'informations.

Les opérations représentées sur un fond grisé correspondent à des opérations 'orientées objet'.

CMIS offre également un mécanisme de *réponse multiple* suite à une demande unique et un mécanisme permettant qu'une opération soit effectuée sur plus d'un objet à la fois en les ayant sélectionnés auparavant (voit infra).

La sélection des objets gérés.

CMIS dispose d'outils puissants pour sélectionner un ou plusieurs objets sujets à des opérations de gestion. Comme nous l'avons déjà vu, les noms des objets gérés sont organisés de manière hiérarchique dans un arbre d'information de gestion (MIT pour Management Information Tree) et la manipulation d'un objet passe donc par sa sélection, sa localisation et son accès dans le MIT. Ce sont ces mécanismes que nous allons présenter.

Ces outils sont référencés en tant que paramètre(s) au sein des primitives de service M-GET, M-SET, M-ACTION, M-DELETE. Trois concepts sont présents ; la portée ou la profondeur (scope), le filtrage (filtering) et la synchronisation (synchronization).

La portée ou profondeur de sélection (Scoping)

Le « scoping »permet de localiser le ou (les) niveau(x) hiérarchique(s) de l'arbre d'information de gestion où se trouve(nt) l'(les) objet(s) sujet(s) à manipulation. Le « scoping » est défini avec une référence à une instance d'objet spécifique, appelée *objet géré de base (base managed object)*. Cet objet de base est le point de départ à partir duquel le filtre sera appliqué.
On peut donc avoir quatre types de spécification pour le « scoping » :

1. L'objet de base seul
2. Le $n^{ième}$ niveau en dessous de l'objet de base, sachant que l'objet de base représente le niveau 0
3. L'objet de base et tous ses descendants jusqu'au $n^{ième}$ niveau y compris
4. L'objet de base et tous ses descendants

La Figure 2-10 montre des exemples de sélection d'objet par réduction de la profondeur de sélection.

Voici des exemples de sélection :

Base = B, niveau =(Base) → Espace = {B}
Base = A, niveau =(1^{er} niveau) → Espace = {B,C,D}
Base = C, niveau =(Base + 1^{er} niveau) → Espace = {C,H,I,J}
Base = D, niveau =(Base + fils) → Espace = {D,K,L,M,T,U,V,W}

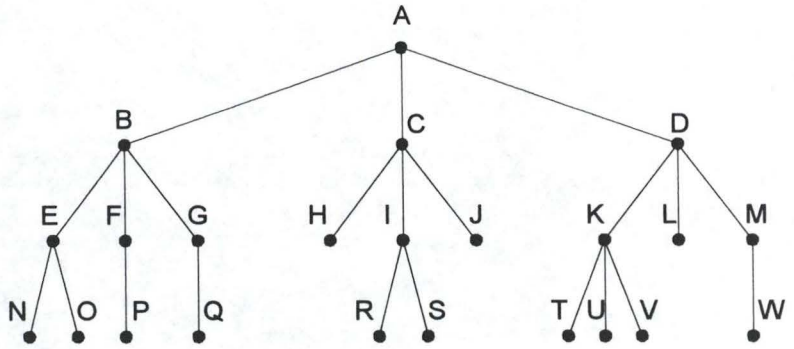
Le filtrage

Le processus de filtrage autorise le test de la présence ou de l'absence de certaines valeurs d'attributs d'un objet géré préalablement « scopé » afin de pouvoir le sélectionner. Le filtre n'est rien d'autre qu'un ensemble d'assertions reliées par des opérateurs logiques.

La synchronisation

Si plusieurs objets ont été sélectionnés, l'utilisateur des services CMISE, doit pouvoir spécifier la façon dont les opérations doivent être synchronisées.

Figure 2-10 : sélection d'objet



Deux types de synchronisation existent. La synchronisation dite *atomique* qui assure que l'exécution des opérations n'est appliquée sur l'ensemble des objets que si toutes les opérations peuvent être exécutées. Par opposition, la synchronisation dite « *au mieux* » (best effort) permet d'exécuter au moins les opérations réalisables même si toutes ne le sont pas.

Protocole commun d'information de gestion CMIP

Les utilisateurs de CMISE doivent être capables d'établir des associations afin d'effectuer ces opérations de gestion. Ces associations sont offertes par les éléments de service de contrôle d'association (ACSE pour Association Control Service Element) et par CMISE. Pour les services d'opération de gestion, CMISE a recours aux PDU d'échange de CMIP. CMIP se fiant aux services des éléments de service d'opérations distantes (ROSE pour Remote-Operation-Service Element). ACSE et ROSE se fient au service de présentation.

Sur la Figure 2-9 on voit apparaître les trois services suivants : A-ASSOCIATE pour établir une association entre des utilisateurs homologues de services CMIS, A-RELEASE pour libérer une association et finalement A-ABORT pour couper brutalement une association.

CMIP va utiliser les services de ROSE pour véhiculer les *Application Protocol Data Unit (APDU)* des ACSE. Les transferts s'effectueront à l'aide de la primitive RO-INVOKE. Et on peut considérer que les trois autres primitives RO-RESULT, RO-ERROR et RO-REJECT-U sont des confirmations de la primitive RO-INVOKE.

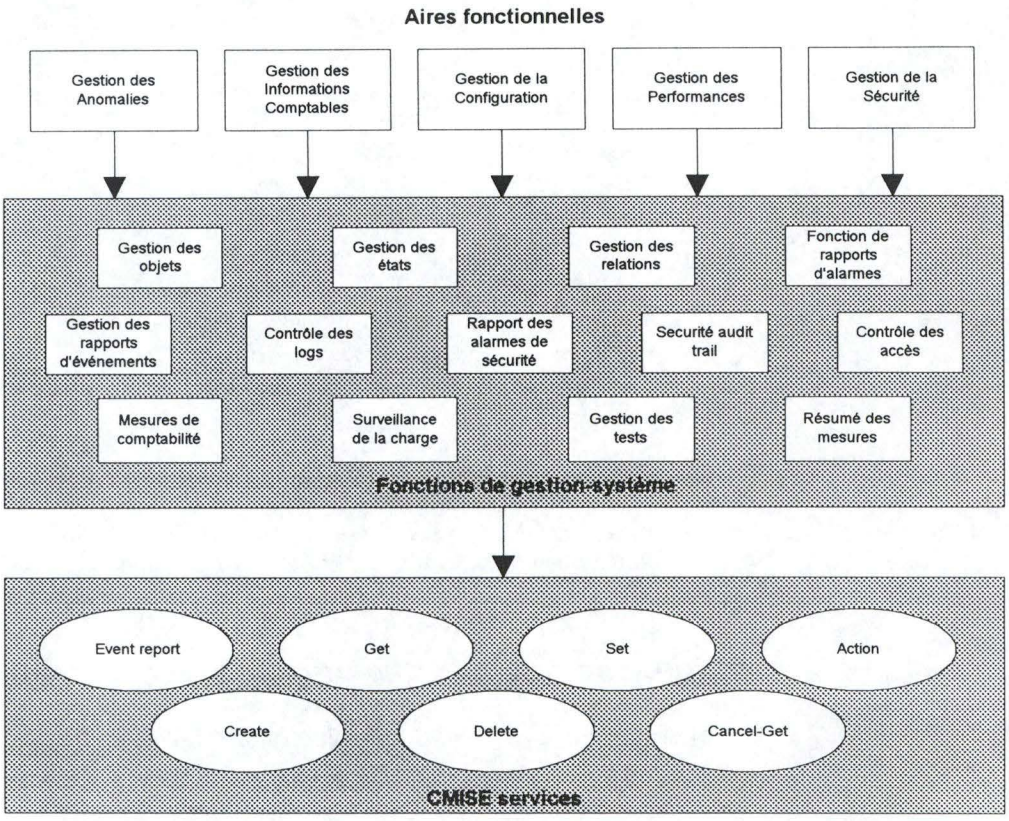
Les fonctions de gestion-système

Les aires fonctionnelles décrites précédemment dans le cadre architectural de la gestion OSI décrivaient de grands domaines de responsabilité en matière de gestion de réseau.

Chacune de ces aires nécessite l'utilisation de fonctions spécifiques. Ces fonctions ont été appelées *Fonctions de gestion-système (SMF pour System Management Function)*. L'ensemble des standards de l'ISO commençant par 10164 spécifient chacun à leur tour une fonctionnalité [ISO10164-1]...[ISO10164-14].

Il se peut que certaines SMF supportent les besoins de plusieurs aires fonctionnelles, prenons par exemple la fonctionnalité de rapport d'événement[11]. Chaque fonctionnalité de gestion-système peut utiliser les services d'autres fonctions de gestion-système. Nous allons présenter brièvement chacune de ces quatorze fonctions qui ont été spécifiées par l'OSI.

Figure 2-11 : Aperçu de la gestion-système



Fonction de gestion d'objet

La norme [ISO10164-1] définit et spécifie des outils de manipulation et de notification pour les objets gérés.

La fonction de gestion d'objets permet de créer ou de supprimer des objets, de lire ou de modifier des valeurs d'attributs ainsi que de notifier la création et la suppression d'objets, la modification d'un nom et des valeurs d'attributs des objets.

Les objets peuvent être créés et effacés et la valeur de leurs attributs peut être changée de trois manières différentes :

- A travers le processus de configuration dans l'environnement local, cela sort du contexte de la standardisation
- Grâce aux opérations de couche (N) et aux entités de gestion de couche (N)
- Avec des fonctions de gestion d'objet comme faisant partie des services de gestion OSI.

---

[11] D'autres exemples sont présentés dans l'annexe 1.

Le service de gestion des objets utilise directement les primitives de CMISE[12]. C'est à dire que les paramètres sont associés directement aux paramètres des primitives CMISE. La correspondance de ces services de gestion est illustrée dans le Tableau 2.

Chaque fonction spécifique de gestion (états, relations, journal, etc.) traite d'événements qui lui sont propres. Chaque type d'événement est transmis à l'aide de la primitive M-EVENT-REPORT et du protocole associé. Cette primitive possède un paramètre qui permet de spécifier le type d'événement et pour chaque événement spécifique ce dernier est détaillé dans le paramètre « information d'événement » de la primitive M-EVENT-REPORT.

Tableau 2 : Correspondance des services de gestion

Services de gestion	*Service transparent*
CREATE	PT-CREATE
DELETE	PT-DELETE
ACTION	PT-ACTION
REPLACE	PT-SET
ADD	PT-SET
REMOVE	PT-SET
REPLACE-WITH-DEFAULT	PT-SET
GET	PT-GET
NOTIFICATION	PT-EVENT-REPORT

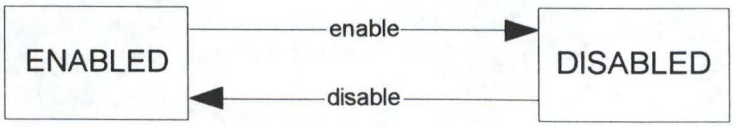
Fonction de gestion des états

La norme [ISO10164-2] définit et spécifie des outils de manipulation et de notification pour les objets gérés.

Trois attributs de base ont été définis pour caractériser l'état d'un objet et donc de la ressource associée. Ces attributs sont regroupés dans un groupe d'attributs d'état (State attribute group). Les états possibles étant les suivants : l'état opérationnel, l'état administratif et l'état d'utilisation.

L'état opérationnel d'un objet indique la possibilité de pouvoir physiquement utiliser ou non la ressource représentée par l'objet. Cet état comporte un attribut d'état qui peut prendre deux valeurs, soit disponible (enabled), soit indisponible (disabled). Cet attribut ne peut pas être modifié par des opérations de gestion.(cfr. Figure 2-12)

Figure 2-12 : Diagramme d'état opérationnel



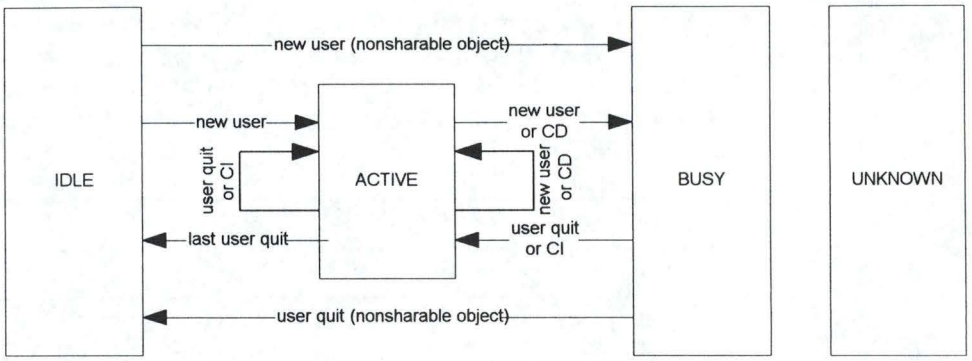
L'état d'utilisation définit la capacité présente de la ressource pour son utilisation. Cet attribut peut prendre quatre valeurs différentes : occupé (busy), actif (active), inactif (idle) ou inconnu (unknown).

---

[12] Les services CMISE utilisés par les fonctions de gestion-système sont présentés à l'annexe 1.

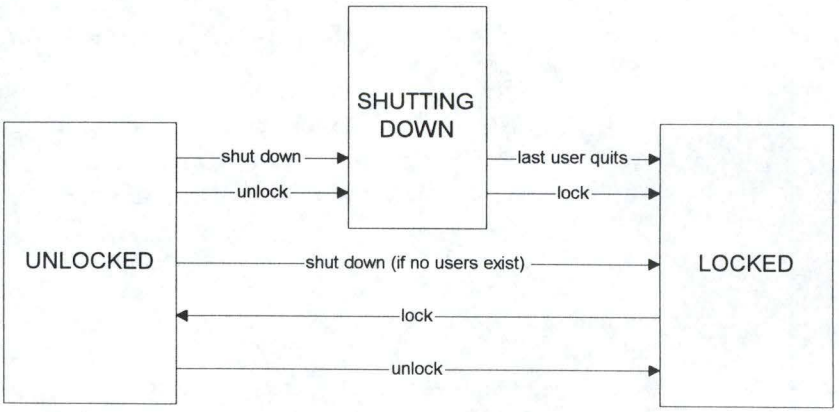
L'état inactif signifie que la ressource n'est pas utilisée. L'état actif signifie que la ressource est utilisée mais qu'elle dispose encore d'une capacité suffisante pour satisfaire d'autres utilisateurs. L'état occupé quant à lui signifie que la ressource est occupée pour l'instant et qu'elle ne dispose plus d'une capacité suffisante pour satisfaire d'autres utilisateurs.(cfr. Figure 2-13)

Figure 2-13 : Diagramme d'état d'utilisation



L'état administratif détermine la possibilité d'utiliser la ressource de façon logique. L'attribut d'état administratif peut prendre trois valeurs : verrouillé (locked), déverrouillé (unlocked) et « en cours de verrouillage » (shutting-down).
Cet état est modifié par des opérations de gestion. Quand une ressource est verrouillée, cette ressource ne peut administrativement pas exécuter des opérations pour les utilisateurs. L'état « en cours de verrouillage n'autorise plus l'entrée de nouveaux utilisateurs de la ressource. (cfr. Figure 2-14)

Figure 2-14 : Diagramme d 'état administratif



Cette norme définit les paramètres qui doivent figurer dans le service M-EVENT-REPORT. La norme définit trois paramètres : Event type, Event information et Event reply.

Le paramètre *event type* aura pour valeur 'State change'. Cet événement sera utilisé pour notifier le fait que la valeur d'un attribut d'état a changé.

L'*event information* est composé de plusieurs paramètres.

- Le *Source indicator* donne l'origine de l'opération qui a donné lieu à cette notification.
- *L'attribute identifier list* donne la liste des attributs qui ont changé de valeur
- Le paramètre *State Change definition* contient à son tour plusieurs autres paramètres. Il y a l'identifiant,l'ancienne et la nouvelle valeur de l'attribut.

La norme ne spécifie rien pour ce qui est du contenu du paramètre *event reply.*

Le lecteur intéressé trouvera à l'annexe 1 un exemple de digramme où nous pouvons retrouver les différents états combinés.

Fonction de gestion des relations

La norme [ISO10164-3] définit et identifie les types de relations qui existent entre les objets gérés.

Des services sont définis pour établir, examiner et surveiller les relations entre les objets gérés. Nous définirons la *relation* par un ensemble de règles qui décrivent comment le fonctionnement d'un objet peut influencer le fonctionnement d'un autre objet. Les relations entre les objets gérés s'expriment par des attributs d'objets. Gérer les relations entre les différents objets revient à gérer les attributs des objets ainsi que leur valeur.

Le type de relation définit la nature de la relation entre deux ou plusieurs objets. Voici les trois différentes catégories de relation définies par la gestion OSI :

- La relation *réciproque :* c'est un lien représenté par une valeur d'attribut dans chacun des objets gérés qui identifie l'autre objet géré en relation.
- La relation *unidirectionnelle :* elle est exprimée par une valeur d'attribut d'un seul membre de la relation.
- La relation de *contenance :* Cette relation n'est pas définie dans [ISO10164-3] mais bien dans la norme [ISO10165-1], en même temps que la définition des objets. C'est une relation que l'on pourrait caractériser de structurelle car elle représente le fait que l'existence d'un objet 'contenu' dépend de l'existence de l'objet 'contenant'.

Fonction de rapports d'alarme

La norme [ISO10164-4] modélise les rapports d'alarmes. Cette norme spécifie les notifications émises ainsi que la sémantique et les paramètres des différentes opérations.

Cette fonction permet de notifier qu'une alarme a été détectée dans l'un des objets gérés. Cette fonction est bien évidemment essentielle dans un environnement ouvert, où tout dysfonctionnement doit être constaté à temps et localisé avec précision.

Tableau 3 : Types de causes d'alarmes probables

Types d'alarmes	*Définition*	*Exemple*
Communication	Associée à la détection, par un objet, d'une erreur lors de la communication.	Perte de signal Erreur locale Erreur d'établissement d'appel
Qualité de service	Utilisée pour relater la présence d'une dégradation de la qualité de service ou à la présence d'anomalies.	Temps de réponse trop long Taux de retransmission trop important
Traitement logiciel	Utilisée pour notifier la présence d'une erreur lors de l'exécution d'un traitement.	Erreur logicielle Place mémoire insuffisante
Equipement	Utilisée pour rapporter un problème dans l'équipement	Alimentation Interface Problème au processeur
Environnement	Utilisée pour rapporter un problème dans l'environnement[13]	Détection de fumée Température, ventilation Feu

La norme cite également les informations principales devant se trouver dans les alarmes mais les gestionnaires de réseaux sont encouragés à y ajouter des informations pertinentes afin de pouvoir réagir rapidement à ces alarmes. L'ISO a identifié cinq catégories d'alarmes qui sont détaillées dans le Tableau 3.

Fonction de gestion des rapports d'événements

Les rapports d'événements sont générés lorsqu'un événement est survenu sur un objet (modification d'un objet, violation de seuil, etc.).

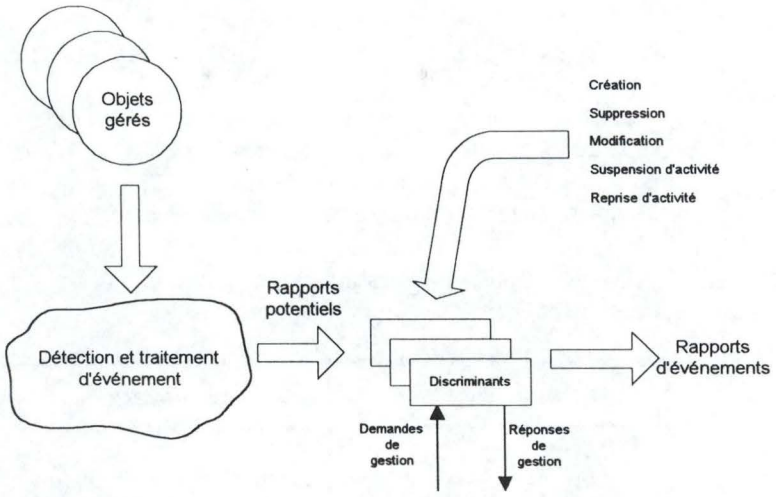
La fonction de gestion rapport d'événement, qui est spécifiée dans [ISO10164-5], doit permettre aux systèmes de gestion de choisir parmi les rapports d'événements ceux qui devront être envoyés et à quels destinataires. Ainsi, le service permet de spécifier les destinataires (identification des systèmes de gestion) et des mécanismes de contrôle d'envoi de rapports d'événements (par exemple, en suspendant ou en reprenant leurs activités d'envoi). Ceci offre la possibilité pour un système de gestion distant de modifier les conditions d'envoi des rapports d'événements. Pour cela, on a défini un objet de gestion de support appelé *le discriminant.*

Comme nous le montre la Figure 2-15, un discriminant peut agir comme un filtre en sélectionnant l'information pertinente qui doit être envoyée au système gestionnaire, pour la prise en compte de notifications particulières relatives au comportement d'un objet ou d'un ensemble d'objets gérés. La fonction de gestion de rapports d'événement consiste en fait en une gestion des discriminants et de leurs attributs spécifiques.

---

[13] On entend par environnement, tout ce qui est relatif à l'enceinte ou se trouve l'équipement.

Figure 2-15 : Fonction de rapport d'événement.



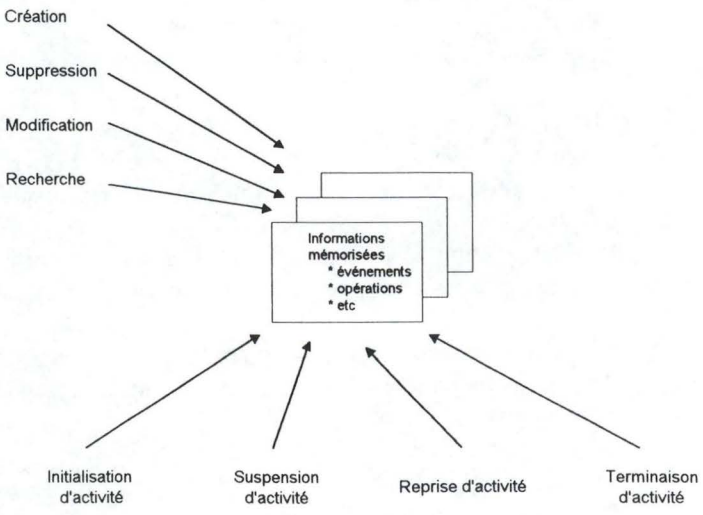
Fonction de contrôle du journal

La norme [ISO10164-6] spécifie un modèle pour créer les journaux. Un filtre peut être établi pour sélectionner les événements devant apparaître dans le journal.

Les ressources réelles de mémorisation sont modélisées pour la gestion OSI par la classe d'objets de nom « log » (journal). Ce journal est lui-même constitué d'enregistrements modélisés par une classe d'objets « log-record » . La définition d'un service flexible de contrôle de journalisation permet la sélection d'enregistrements qui doivent être mémorisés par un système de gestion dans un journal particulier.

Les services de la fonction de rapport de journal permettent de créer un journal, de le supprimer et d'en modifier des attributs. Plus précisément, la gestion de ces attributs autorisent l'initialisation, la suspension, la reprise et la terminaison d'activité de journalisation, ainsi que la recherche et la suppression d'enregistrements. Ces services sont résumés par la Figure 2-16 .

Figure 2-16 : Fonction de rapport de journalisation



Fonction de rapports d'alarmes de sécurité

La norme [ISO10164-7] modélise la fonction de rapport d'alarmes de sécurité.

Elle spécifie la sémantique et les paramètres des différentes notifications d'alarmes de sécurité. Les alarmes de sécurité ne sont rien d'autre que des notifications particulières. La fonction de rapport d'alarme de sécurité offre les mêmes services que ceux offerts dans la fonction de rapport d'alarmes mais ils sont relatifs aux paramètres de sécurité.

Dans le Tableau 4 nous présentons les différents types d'alarmes de sécurité, en les définissant et en donnant leurs principales causes.

Tableau 4 : Types et causes d'alarmes de sécurité.

Types de violation	*Signification*	*Cause*
d'intégrité	Indique qu'il y a eu une interruption potentielle du flot informationnel,tel que l'information a été illégalement modifiée, insérée ou effacée	Information dédoublée Information manquante Information hors de séquence Modification d'information détectée
opérationnelle	Indique que le service requis n'a pu être réalisé suite à une indisponibilité, un dysfonctionnement ou à une invocation incorrecte de service.	Refus de service Erreur de procédure
physique	Indique qu'une manipulation non autorisée de la ressource physique a été détectée	Ecoute sur la ligne Détection d'intrusion
de service de sécurité	Indique la présence d'un problème de sécurité.	Problème d'authentification Accès non autorisé
des plages horaires	Indique qu'un événement est survenu en dehors de sa plage horaire « normale »	Information retardataire Activité en dehors des heures « légales ».

Fonction d'audit-trail

La norme [ISO10164-8] spécifie les types de rapports d'événements qui devraient figurer dans un journal lorsque celui-ci est destiné à l'évaluation de la sécurité d'un système ouvert ou à l'évaluation des performances des mécanismes de sécurité sous-jacents.

La fonction de vérification des traces peut être utilisée pour déceler les attaques à la sécurité du système, lorsque celles-ci n'ont pu être détectées lors de leur survenance.

Cette fonction pourrait être considérée comme étant une extension de la fonction de contrôle des journaux. Les types d'événements qui pourraient faire l'objet de cette vérification seraient :

- Les connexions
- Les déconnexions
- L'utilisation des mécanismes de sécurité
- Les opérations de gestion
- L'utilisation de la comptabilité

Fonction de contrôle d'accès

La norme [ISO10164-9] spécifie un modèle de contrôle d'accès aux informations et aux opérations de gestion.

Elle spécifie les objets gérés ainsi que les attributs qui permettent d'autoriser ou de refuser l'accès selon une politique de contrôle d'accès représentée par les informations de gestions relatives au contrôle d'accès (access-control-management information).

Différents niveaux d'accès sont possibles. Certains ont des droits en écriture, en lecture sur tout ou partie des attributs alors que d'autres n'ont aucun droit sur ces attributs. Le contrôle d'accès se chargera également de s'assurer que des notifications ne seront pas envoyées à des réceptionnaires qui n'y sont pas autorisés et il s'assurera également que les opérations seront effectuées par les utilisateurs de droit.

Fonction de mesures de comptabilité

La norme [ISO10164-10] spécifie un modèle pour comptabiliser l'utilisation des ressources du système et des mécanismes destinés à fixer des limites comptables.

La norme définit des métriques comptables (account meters) et des journaux, elle spécifie également des services pour extraire, rapporter et enregistrer les informations relatives à l'utilisation des ressources . Elle déterminera également quelles données sont à extraire et sous quelles conditions elles doivent apparaître dans le rapport.

Le métrique comptable représente l'abstraction de la fonction de gestion des informations comptables. Deux aspects sont à considérer dans cette fonction :

- Le contrôle et le rapport d'informations associées à l'utilisation d'une ressource
- Les informations spécifiques enregistrées

Fonction de surveillance de la charge

La norme [ISO10164-11] définit un modèle pour surveiller les attributs des objets gérés.

Elle définit des objets qui peuvent rapporter des événements basés sur des valeurs de compteurs, de jauges reflétant les performances du système.

Nous n'irons pas plus loin dans la description de la fonction de surveillance de la charge à ce niveau car cette fonction sera l'objet principal du chapitre 4 destiné à l'étude des objets métriques.

Fonction de gestion des tests

La norme [ISO10164-12] définit un modèle fonctionnel de processus de gestion autorisant la réalisation de tests de diagnostique et de confiance sur des systèmes ouverts distants.

Cette norme définit une classe d'objets gérés qui est utilisée pour contrôler les tests qui peuvent être réalisés de manière synchrone ou asynchrone avec des résultats qui sont rapportés par la suite.

Un test sera dit *synchrone* lorsque le résultat du test est retourné en réponse à l'opération d'initiation du test. Un test sera dit asynchrone lorsque la réponse sera retournée par des opérations additionnelles de gestion ou par des notifications issues des objets testés.

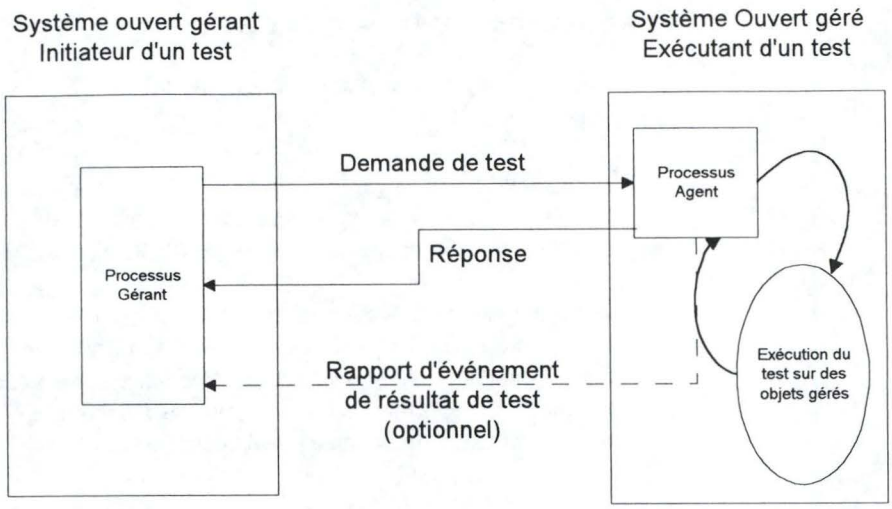


Figure 2-17 : Modèle fonctionnel d'une application de gestion de test

Un objet de gestion sur lequel portent des opérations de test est dénommé MOT pour *Managed Object under Test*. Un test est modélisé par un objet, encore appelé TO pour *Test Object*. Une fonction de gestion de test revient à mettre à la disposition des utilisateurs de ce service, des facilités de manipulation de ces objets et des attributs qui représentent l'ensemble des opérations de test et leurs résultats appliqués sur une ressource.

Fonction de résumé des mesures

La norme [ISO10164-13] définit un modèle et les classes d'objets utilisées pour résumer et pour appliquer des analyses statistiques sur les informations gérées.

Les valeurs des attributs de la fonction de résumé sont des instances d'objets spécifiques au cours du temps (time average) et un *ensemble* d'instances d'objet à un moment déterminé (ensemble average). Les services doivent spécifier les objets gérés devant être incorporés dans les rapports, la programmation des observations sur ces objets et ces attributs ainsi que la programmation de la synthèse des observations.

La fonction résumé va extraire les informations des objets gérés et ensuite les mettre dans un objet « résumé ». L'objet « résumé » spécifie un algorithme de « synthèse » qui permettra de réaliser un résumé à partir de valeurs observées.

Toutes les fonctions de résumé sont basées sur le concept de « scanner ». Le « scanner » est un processus d'échantillonnage des valeurs d'attributs observés à certains moments dans le temps. Trois types de « scanners » sont définis.

- *Scanner homogène :* Il sonde un ensemble d'attributs au sein d'instances d'objets choisies afin de faire des statistiques.
- *Scanner hétérogène :* Il rassemble des statistiques à partir d'ensemble d'attributs qui ont été collectés sur un ensemble d'objets gérés.
- *Scanner hétérogène (avec tampon) :* C'est le même que le scanner hétérogène excepté qu'il permet de mettre une certaine partie des résultats dans un tampon et ainsi d'autoriser la possibilité la présence de résultats de plusieurs périodes.

Conclusion

Ce que nous venons d'exposer à travers ce chapitre ne représente qu'une infime partie de la gestion OSI. Pour s'en convaincre, il suffit de constater la multitude de standards qui émanent de cette organisation.

La quantité impressionnante de standards est justifiée par le fait que l'OSI veut offrir tous les outils utilisés dans la gestion. Ceci implique que la gestion OSI s'avère être ***la solution*** dans le cadre d'une gestion intégrée.

L'ennui réside dans le fait que face à CMIP, qui privilégie l'exhaustivité à la facilité de mise en œuvre, d'autres organismes élaborent des protocoles qui sont plus simples et accessibles à tout un chacun.

Nous constatons que les implémentations basées sur des standards de l'ISO sont pratiquement inexistantes aux Etats-Unis et quelques unes voient le jour en Europe.

Reste donc à espérer que CMIP parviendra à pénétrer le marché et peut-être que les administrateurs de systèmes regretteront alors de ne pas être passés plus rapidement à une gestion s'appuyant sur les normes édictées par l'ISO ...

Chapitre

3

La Gestion des Performances

Aujourd'hui, les besoins des utilisateurs en terme de gestion de performances sont énormes. En effet, face à la diversité des environnements systèmes sur un même centre informatique, la gestion de performances est devenue plus complexe. Le fait de configurer 'correctement' des systèmes UNIX, par exemple, nécessite une multitude d'analyses des centres vitaux du système pour aboutir à une configuration optimale. De plus, il est nécessaire de procéder à une analyse de la charge du système dans le temps, de façon à déterminer un réglage (tuning) adéquat. C'est dans ce contexte que nous présentons la gestion des performances.

Dans un premier temps nous présentons les problèmes auxquels est confrontée la gestion des performances. Ensuite nous présenterons quelques indicateurs les plus fréquemment utilisés dans ce domaine. Nous passerons enfin à l'exposé de trois architectures distribuées de gestion de performance car nous verrons que les outils disponibles pour une gestion des performances dans un environnement de type 'mainframe' ne sont plus valables pour les systèmes distribués.

Qu'est ce que la gestion des performances ?

Lorsque nous parcourons la littérature informatique relative à la gestion des performances, nous constatons que les auteurs ne donnent que très rarement une définition explicite. Bien souvent les auteurs présentent une panoplie d'indicateurs et se limitent à cet aspect. Prenons néanmoins la définition que donne [Terplan,1993] :

« **Performance management** is a set of activities required to continuously evaluate the principal performance indicators of network operation, to verify how service levels are maintained, to identify actual and potential bottlenecks, and to establish and report on trends for management decision making and planning. Maintaining the performance database, networking models, and preparing automation procedures for fault management are also supported. »

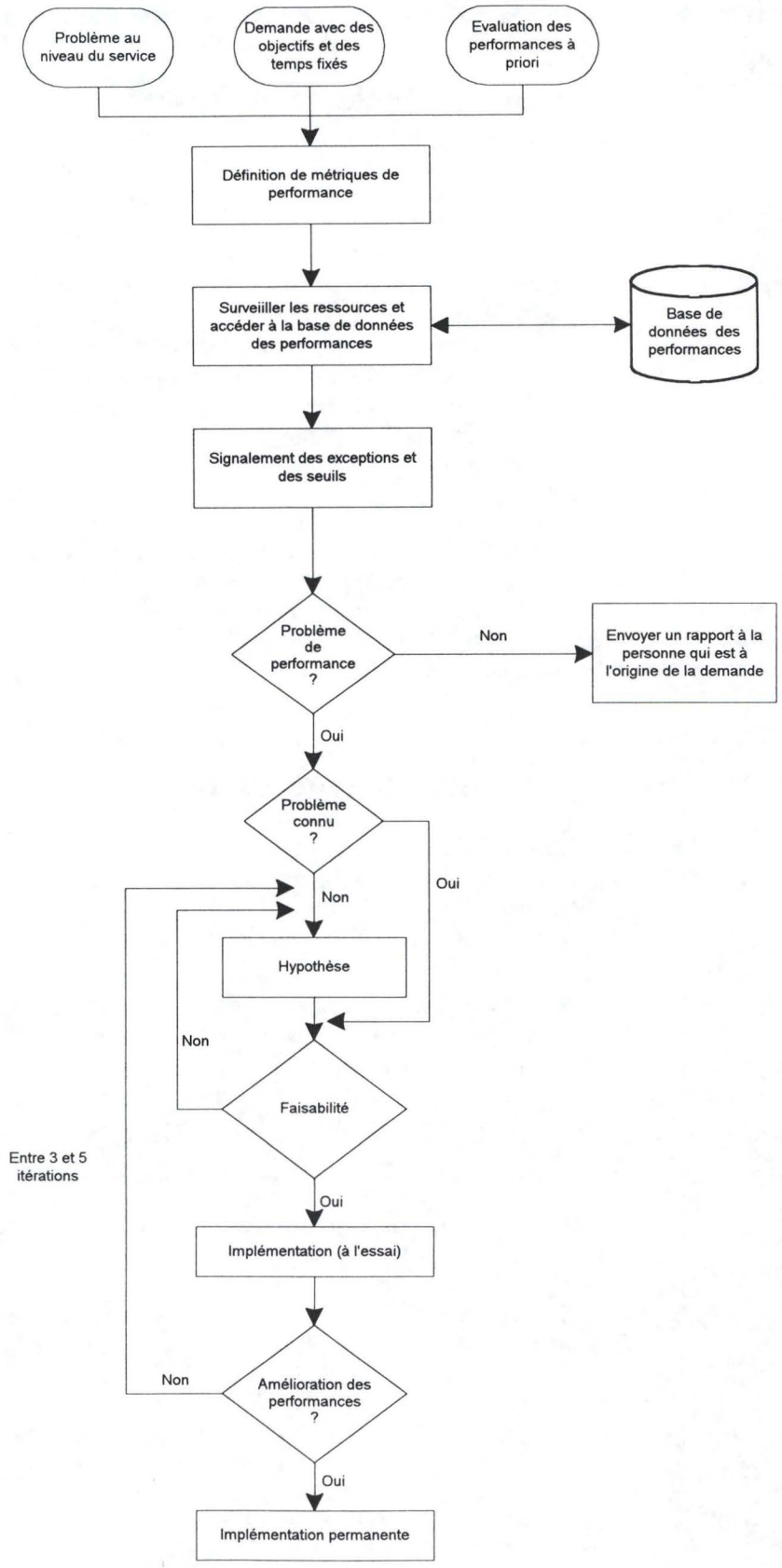
Cette définition nous paraît à la fois aisée à comprendre et elle identifie correctement tous les cas auxquels la gestion des performances doit prendre en compte.

La gestion des performances n'est pas simplement une activité qui requiert un relevé de mesures au travers des ressources du système, elle constitue à elle seule tout un processus comme illustré par la Figure 3-1 .

Une demande d'analyse et d'amélioration des performances à lieu lorsque le niveau de service rendu à l'utilisateur n'est plus satisfaisant ou lorsque des études de performances sont demandées ou enfin lorsque évaluations de performances à priori sont initiées. Le processus débute par la définition de métriques.[14]

[14] Nous avons traduit le terme *'metric'* en anglais par *'métrique'*. Les métriques sont des mesures faites sur un système en activité et qui sont utilisées par les gestionnaires de performances.

Figure 3-1 : Processus de gestion de performances



Ce processus continue par l'interrogation des ressources du système ou par l'interrogation de la base de données relative aux performances. Cette base de données est considérée comme essentielle en ce qui concerne la gestion des seuils et les rapports de

performances. On vérifie si il s'agit d'un problème déjà rencontré auparavant, si il s'agit d'un problème nouveau, on émettra de hypothèses pour le solutionner et on vérifiera que ces hypothèses soient réalisables. On fera un implémentation provisoire des modifications à prendre en considération et si une amélioration des performances est constatée, on fera une implémentation définitive.

La gestion de performances n'est pas une activité simple et nous pouvons relever quelques problèmes auxquels elle est confrontée, c'est l'objectif du paragraphe suivant.

Les problèmes généraux de la gestion des performances

Nous l'avons déjà signalé dans le premier chapitre de ce mémoire, l'objectif d'une architecture distribuée est de mettre à la disposition de chaque utilisateur les ressources (CPU, données, imprimantes, ...) dont il a besoin de façon à optimiser les performances.

Des problèmes de performances sont aggravés par le manque de rapidité des communication dans un système distribué. Il faut en effet, envoyer un message à travers le réseau et en attendre une réponse, ce délai étant sensiblement augmenté par la gestion du protocole de communication. Nous nous trouvons alors devant un paradoxe : le meilleur moyen d'améliorer les performances d'un système est d'exécuter des événements en parallèle sur différents processeurs, mais ceci entraîne l'envoi de nombreux messages ralentissant ainsi la rapidité du système. Mais d'un autre côté, empiler toutes les applications au sein d'une architecture centralisée entraînera également des problèmes de performances.

Nul ne doutera de l'importance d'obtenir des données précises lorsqu'on effectue les mesures de performances. La plupart des outils, en place actuellement, collectent des informations relatives à une seule plate-forme alors que les systèmes distribués vont engendrer le besoin de collecter des données synchrones (sur les applications et sur les réseaux) à partir de plusieurs plates-formes.

Tout système dispose d'un certain nombre d'indicateurs de performances. Mais si on considère tous les indicateurs ils sont très nombreux et les utilisateurs peuvent alors être confrontés aux problèmes suivants :

- Il y trop d'indicateurs
- La signification de tous les indicateurs n'est pas claire pour tout le monde
- Certains indicateurs ne sont pas standardisés
- La plupart des indicateurs ne peuvent pas être comparés entre-eux
- Bien souvent les indicateurs mesurés avec précision mais mal interprétés

Un dernier problème, qui est néanmoins de taille, est de présenter de manière intelligible les données récoltées à l'utilisateurs. Si une application de gestion de performances présente les données brutes, les unes à la suite des autres, le gestionnaire des performances ne saura pas discerner les éventuels problèmes.

C'est pour toutes ces raisons qu'il est nécessaire d'élaborer une architecture de mesure de performances qui prendra en considération toutes les complexités. Nous présenterons trois architectures dans les pages qui suivent, mais avant toute chose passons en revue quelques indicateurs de performances.

Définition d'indicateurs de performances

Lorsque l'on présente les indicateurs de performances, nous avons souvent tendance à n'exposer que des indicateurs relatifs à l'administration du réseau. A juste titre ! En effet,

comme nous le verrons par la suite la plupart des architectures existantes sont 'orientées réseau', nous verrons que c'est le cas de SNMP et de CMIP. Mais on s'aperçoit que les paramètres à analyser sont à la fois du domaine de l'administration des réseaux et du domaine de l'administration des systèmes.

Nous allons exposer les différents indicateurs courants[15]. Nous pouvons distinguer deux classes d'indicateurs : les indicateurs orientés vers le niveau de service offert aux utilisateurs et les indicateurs orientés vers l'efficacité du système.

Les indicateurs orientés vers le niveau de service offert

Un des rôles de l'administrateur est d'assurer un *service global aux utilisateurs*. La fourniture de ce service est liée à un suivi des tâches et à une analyse pointue au niveau répartition CPU, mémoire et ressources I/O, consommation CPU et équilibrage des charges entre les systèmes. Nous présentons les indicateurs suivants : la disponibilité (availability), le temps de réponse et l'exactitude (accuracy).

La disponibilité

La disponibilité caractérise le fonctionnement du système comme un tout, c'est-à-dire qu'elle indique le pourcentage de temps pendant lequel l'utilisateur peut accéder à une composante. Les composantes spécifiques qui fonctionnent ou qui ne fonctionnent pas ne nous intéressent pas, c'est la disponibilité d'une ressource *en général* qui importe.

La disponibilité dépend bien entendu de la fiabilité des composantes, cette fiabilité peut être très élevée (par exemple 99,999%) mais elle ne sera jamais maximale. Afin d'assurer une plus grande disponibilité, nous pouvons augmenter le nombre de composantes identiques, assurant ainsi leur redondance[16] en cas de dysfonctionnement tout en essayant de ne pas en arriver à des coûts excessifs.

La disponibilité peut être calculée au moyen de la formule suivante :

$$AV(availability) = \frac{MTBF}{MTBF + MTTD + MTTR + MTOR}$$

où

MTBF : Mean Time Between Failure
MTTD : Mean Time To Diagnosis
MTTR : Mean Time To Repair
MTOR : Mean Time Of Repair

Nous remarquons que pour améliorer la disponibilité, il faut essayer de garder des valeurs assez faibles pour MTTD, MTTR et MTOR en comparaison avec MTBF.

---

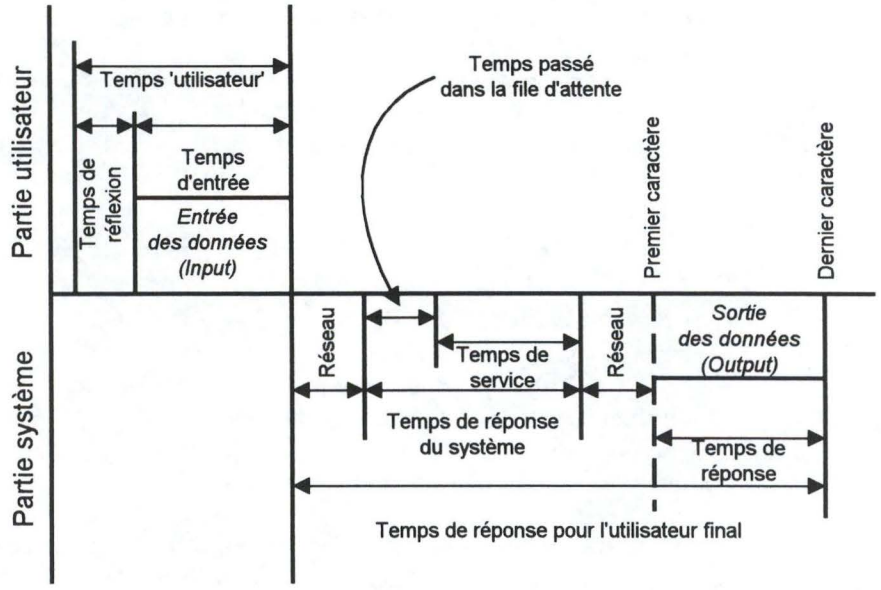
[15] Nous ne présenterons pas tous les indicateurs car ils sont bien trop nombreux. Certaines applications de gestion de performances ont jusqu'à 150 indicateurs ...

[16] A ce propos, notons qu'il est préférable d'utiliser des configurations parallèles lors de la redondance des composantes, de cette manière si un composant est mis en échec, on peut passer à un autre sans problème, ce qui n'est pas le cas d'une configuration séquentielle.

Le temps de réponse

Le temps de réponse est un des éléments essentiels du point de vue de l'utilisateur final. Nous définissons le temps de réponse comme étant l'intervalle de temps écoulé entre l'initiation d'une requête par l'utilisateur et la réception du dernier caractère de la réponse sur l'écran. Ce temps de réponse est illustré à la Figure 3-2

Figure 3-2 : Schématisation du temps de réponse



L'exactitude

Nous entendons par exactitude la livraison de réponses précises et sans erreur à l'utilisateur. La livraison d'informations peut être influencée par :

- La réception de caractères erronés (CH_E)
- Des caractères transmis mais non-reçus (CH_U)
- Des caractères reçus qui n'avaient pas été envoyés (CH_N)
- Des caractères envoyés et reçus en plusieurs exemplaires (CH_D)

Le taux d'erreur résiduel peut servir de mesure pour l'exactitude et il se calcule de la manière suivante :

$$ER(tauxd'erreur) = \frac{CH_E + CH_U + CH_N + CH_D}{CH_T}$$

où CH_T représente le nombre total de caractères transmis

Les indicateurs orientés vers l'efficacité du système

Les indicateurs orientés vers l'efficacité du système représentent les intérêts de l'organisation, en offrant des services à l'ensemble de la communauté des utilisateurs. Les indicateurs couramment présentés sont le *throughput*, que nous traduisons par débit et *l'utilisation* des ressources.

Le débit

Le débit ou throughput représente simplement une mesure globale de la capacité du serveur. Nous entendons par capacité *la limite théorique supérieure* de la ressource dans les *circonstances idéales.*

Nous donnerons par exemple des informations sur les processeurs en terme de MIPS (million d'instructions par seconde) ou encore la capacité maximale d'une ligne en kilobits par seconde.
Le débit est une mesure orientée 'application', en voici des exemples :

- Le nombre de transactions A, B ou C pendant une certaine période
- Le nombre de sessions client pour une application pendant un intervalle de temps défini
- Le nombre de jobs exécutés pendant une certaine période.

Le taux d'utilisation

Le taux d'utilisation représente une mesure dynamique des ressources du système. Le taux d'utilisation se base sur des mesures pratiques et il donne le pourcentage utilisé de la capacité théorique de la composante. C'est un indicateur intéressant afin de prévoir les goulots d'étranglement et les zones de congestion.

C'est une mesure qu'il faut surveiller de très près car comme nous le savons, le temps de réponse d'une ressource croît exponentiellement avec le taux d'utilisation d'une ressource.

Trois architectures utilisées pour des mesures de performances

Comme nous l'avons déjà dit auparavant, la gestion des performances est un processus complexe. C'est la raison pour laquelle, il s'est avéré nécessaire d'élaborer une architecture (distribuée) complète de mesures de performances afin de ne plus se retrouver avec de multiples applications propriétaires qui ne se basaient sur aucun standard.
Nous allons présenter trois architectures : SNMP ; CMIP et l'Universal Measurement Architecture(UMA[17]). Deux de ces architectures, SNMP et CMIP, sont connues par la plupart des informaticiens mais UMA est une architecture récente réalisée par le Performance Measurement Working Group (PMWG) qui travaille avec l'aide du Computer Measurement Group (CMG).

SNMP et CMIP sont des protocoles principalement destinés à une gestion du réseau, avec néanmoins une architecture solide. D'autre part, UMA, propose une architecture destinée à l'administration des systèmes.
Mais ces standards évoluent de telle manière que les frontières conceptuelles, qui étaient présentes à l'origine, ont tendance à devenir de plus en plus confuses. Ce rapprochement des standards n'est pas sans risques car des termes identiques sont parfois utilisés par des standards différents et représentent des notions différentes[18].

---

[17] A l'origine, UMA signifiait *Unix Measurement Architecture* mais par la suite on a préféré remplacer le terme *Unix* par *Universal*

[18] Citons par exemple la notion de *throughput* qui peut représenter *des paquets par seconde* ou *des transactions par seconde* selon que l'on parle de performance d'un réseau ou de performance d'un serveur de bases de données.

Le standard 'de fait' SNMP, développé par des groupes membres de l'Internet engineering task force (IETF) est bien présent en Amérique du Nord, tandis que CMIP influence plus les européens. UMA quant à elle est trop récente pour dire où elle à trouvé ses marques.

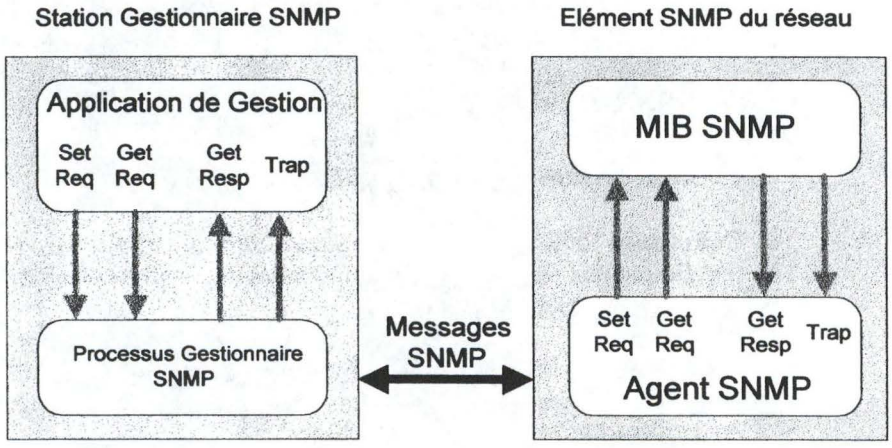
SNMP, CMIP et UMA représentent en fait des pièces différentes du « puzzle de la gestion distribuée ». Nous verrons donc dans la suite ce chapitre les similarités et les différences entre ces différentes architectures. Etant donné que l'objectif de ce mémoire n'est pas de présenter les différents protocoles (SNMP, UMA, ...) nous ne ferons qu'aborder les éléments les plus importants concernant ces deux architectures. Nous conseillons au lecteur désirant en savoir plus de consulter [Stallings,1993], [Xopen,1995-1], [Xopen,1995-2], [Xopen,1995-3], [Xopen,1995-4].

L'architecture SNMP

Présentation de l'architecture

L'architecture SNMP utilise le modèle manager/agent, comme illustré à la Figure 3-3, avec une relation maître-esclave. Les agents résident sur les noeuds du réseau et ils doivent être interrogés par des noeuds gestionnaires. Il y a néanmoins une exception à ceci, il s'agit du message TRAP, qui peut être envoyé de manière autonome par un agent lorsque la survenance de certains événements se produit.

Figure 3-3 : modèle architectural SNMP



Le mécanisme de transport sous-jacent à SNMP est UDP/IP[19] car SNMP utilise un protocole de communication non connecté. Il y a plusieurs raisons pour lesquelles on utilise un mode non connecté. Ce mode permet de garder la simplicité recherchée par SNMP en termes de types de messages qui sont générés.

L'interface via laquelle les agents rassemblent leurs données s'appelle la MIB. Il s'agit d'une forme très élémentaire de base de données. Les données sont regroupées sous forme d'objets[20].

---

[19] UDP signifie User Datagram Protocol, c'est un protocole qui travaille en mode non connecté.

[20] Il est important de ne pas confondre les objets de la MIB SNMP, qui sont simplement des variables ASN.1, avec les objets de la MIB vus lors de la présentation de la gestion OSI.

Une des extensions importante de la MIB SNMP concernant la gestion distribuée des performances est la Remote Network Monitoring MIB (RMON MIB). Un RMON peut examiner les paquets sur les sous-réseaux et il peut répertorier immédiatement tous les éléments du réseau. Le RMON ne perturbe en rien la charge du réseau car il ne fait qu'inspecter les entêtes des paquets évite ainsi de charger le réseau avec des requêtes en tout genres.

Les limitations

Le schéma de contrôle Maître/Esclave centralisé signifie que les agents doivent être interrogés pour obtenir des données. C'est-à-dire que chaque message contenant des données doit être précédé d'un message contenant une requête.

De manière générale, les messages et les datagrammes doivent être courts en raison des limitations inhérentes à la construction des datagrammes UDP.
Nous comprendrons bien vite que si le nombre de noeuds augmente dans le réseau, le protocole d'interrogation va charger considérablement le réseau.

Il n'y a pas de possibilité de faire communiquer les agents entre eux . Notons aussi que les noeuds ne garderont pas des données 'historiques' les concernant. Il n'y a qu'un simple tampon circulaire de 255 enregistrements, qui est prévu dans le RMON, pour conserver les données. Le fait de ne pouvoir garder des historiques pose beaucoup de problèmes pour comparer de nouvelles interrogations avec les anciennes ...

Une autre limitation est que les TRAPs SNMP sont envoyées sans accusé de réception.

L'architecture CMIP

Présentation de l'architecture

Nous nous limiterons à faire quelques rappels pour cette partie car le détail de l'architecture et des fonctionnalités offertes a été décrit au chapitre 1 et les outils de mesures seront décrits au chapitre 4.

Le standard ISO CMIP s'étend bien au-delà des standards SNMP. L'ISO utilise des classes d'objets explicites avec une structure d'héritage et de contenance. La MIB de l'OSI est basée sur des concepts orientés-objets.

La gestion des performances constitue l'une des aires fonctionnelles de la gestion OSI (cfr. **La Gestion des Performances**, page 28) et elle utilise les fonctions de gestion système suivantes :

- La fonction de gestion des objets (cfr. page 42)
- La fonction de gestion des rapports d'événements (cfr. page 46)
- La fonction de surveillance de la charge (cfr. page 49)
- La fonction de gestion des tests (cfr. page 49)
- La fonction de gestion des résumés (cfr. page 50)

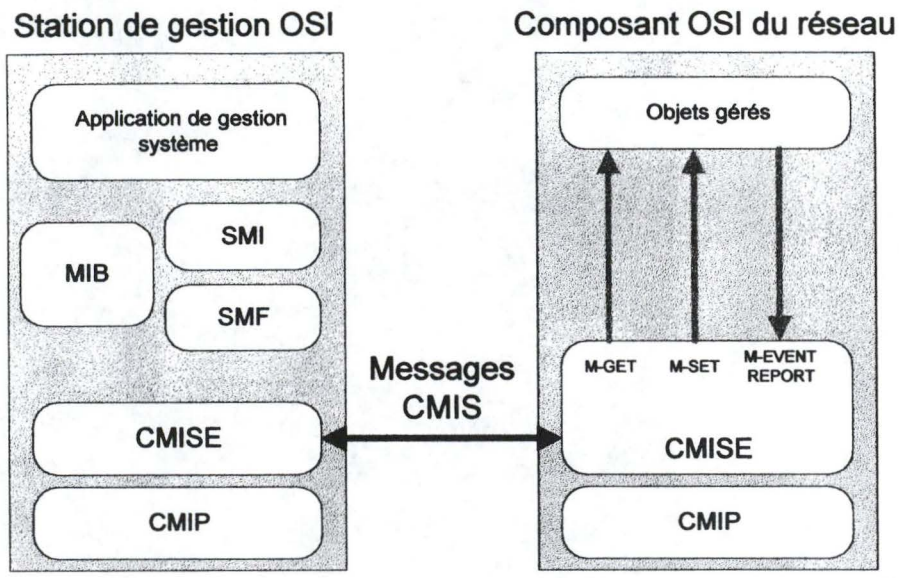


Figure 3-4 : Le modèle architectural CMIP

Les limitations

Autant l'architecture OSI est impressionnante de par ses fonctionnalités, autant elle est peu présente sur le marché ! Cela peut sembler bizarre mais l'ISO a la mauvaise habitude de travailler sur ses standards pendant de longues années et en comité restreint, alors qu'au contraire SNMP lance plus vite ses propositions de standardisation sur le marché.

Notons que cette limitation est purement commerciale et non pas technique. BULL figure parmi une des seules sociétés à utiliser CMIP et à importer des données SNMP via des intégrateurs d'agents (cfr. **Un modèle de gestion unique** à la page 79).

L'architecture UMA

Présentation de l'architecture

Tout comme le standard SNMP, l'architecture UMA utilise également un modèle architectural Manager/Agent (illustré à la Figure 3-5) à la différence que l'UMA n'utilise pas un protocole maître/esclave. Les agents UMA peuvent recevoir des messages de contrôle mais ils sont autonomes en ce qui concerne la livraison des données.

La transmission des données relatives aux performances est asynchrone et on peut véhiculer de grandes quantités d'informations. Pour ce faire, UMA utilise le protocole TCP/IP (Transfer Control Protocol/Internet Protocol). L'architecture UMA permet également aux agents de communiquer entre eux au niveau du Data Service Level (DSL).

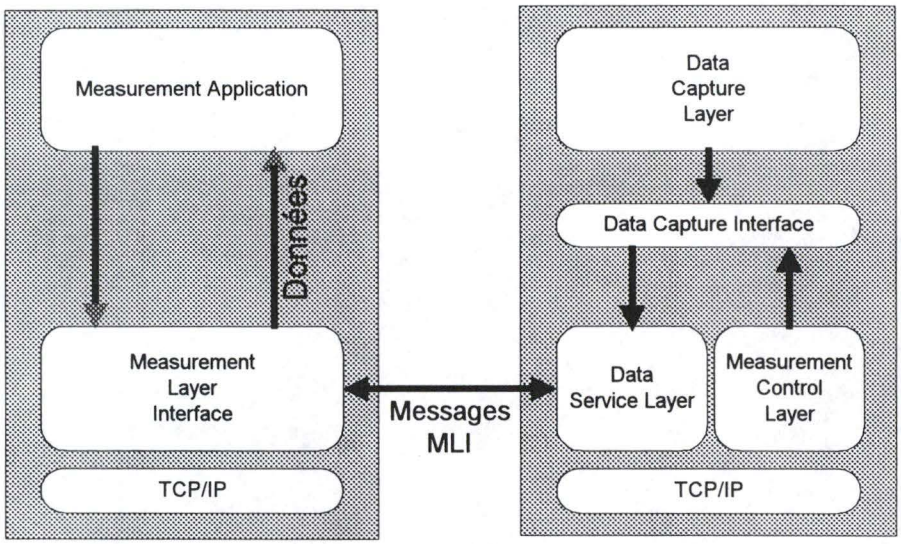


Figure 3-5 : Le modèle architectural UMA

Contrairement à SNMP, les données représentées dans cette architecture sont plus complexes car elles ont une structure de classe hiérarchique mais par contre elles sont plus simples que dans l'architecture de l'OSI car il n'y a pas d'héritage ni de contenance.

Le modèle architectural de l'UMA définit quatre couches et deux interfaces. Nous allons détailler les différentes couches et interfaces suivantes : Data capture Layer, Data Capture Interface, Measurement Control Layer, Data Service Layer, Measurement Layer Interface et enfin la Measurement Application Layer.

La **Data Capture Layer (DCL)** est responsable de la collecte des données brutes. Son architecture couplée avec la *Data Capture Interface(DCI)* permet de collecter des données à partir de plusieurs sources au dessus de la DCI et ceci afin d'améliorer la synchronisation entre la collecte des données.

La **Data Capture Interface (DCI)** est l'interface entre la *Measurement Control Layer* et la DCL. Elle permet d'étendre dynamiquement la collecte de données vers de nouveaux fournisseurs, comme par exemple les bases de données, sans affecter les programmes existants.

La **Measurement Control Layer (MCL)**. Cette couche va synchroniser et planifier la collecte de données au travers de la DCI.

La **Data Services Layer (DSL).** Cette couche accepte les requêtes de mesures provenant de programmes de mesures de performances (MAP pour Measurement Application Program) et elle va assembler les messages de données et les renvoyer aux applications concernées.

La **Measurement Layer Interface (MLI)** est l'interface entre la *Measurement Application Layer* et la couche DSL. Elle est le lieu de toutes les interactions entre un MAP et UMA.
Elle permet également une communication transparente à travers tout le réseau ? De cette manière, une MAP tournant sur un système peut requérir et examiner de données provenant d'un autre système. Associée avec la DSL, elle permettra d'avoir une infrastructure pour la distribution de données à travers un grand nombre de sites et de plates-formes hétérogènes.

La **Measurement Application Layer (MAL).** Cette couche est constituée par plusieurs MAP. Nous retrouverons à ce niveau l'ensemble des applications requises par le responsable de la gestion des performances.

Les limitations

Le standard UMA n'est pas prévu pour couvrir tous les aspects de la gestion système, comme les backups et le trouble ticketing mais il est utilisé pour s'occuper de la gestion des performances qui est un sous-ensemble de la gestion-système.

Conclusion

Nous avons pu constater tout au long ce chapitre que malgré l'intérêt grandissant pour tous les domaines se rapportant à la gestion des performances, cette activité n'est pas encore maîtrisée aussi bien par les concepteurs d'applications que par les gestionnaires.

Nous avons passés en revue les indicateurs classiques en montrant chaque fois leur importance dans le processus de gestion de performance.

Vu l'importance que l'on accorde à l'heure actuelle aux systèmes distribuées et au phénomène Client-Serveur, force est de constater que les outils de gestion de performances présents sur les mainframes ne sont plus valables dans ces nouveaux contextes.
Nous avons donc présenté trois architectures de mesures de performances dans le cadre de systèmes distribués : SNMP, CMIP et UMA.

Nous en retiendrons ceci : SNMP est actuellement très répandu et est apprécié pour sa simplicité. CMIP est présent dans très peu de sociétés et vu que l'engouement pour ce protocole, pourtant complet, n'est est pas des plus importants. Et UMA est le petit nouveau de la bande et s'oriente plutôt vers une administration du système que vers l'administration du réseau comme le font SNMP et CMIP.

A terme, il serait possible d'envisager un rapprochement entre SNMP et UMA, ce qui permettrait de fusionner les avantages présents dans les deux architectures et nous obtiendrions ainsi une architecture qui contiendrait tous les outils nécessaires à l'administration du système et du réseau.

Chapitre

4

Les Objets Métriques

Tout au long de ce chapitre, nous essayerons de présenter de la manière la plus claire possible le standard sur les objets métriques [ISO10164-11]. Il sera en effet essentiel de bien comprendre cette partie car elle sera le fondement du chapitre 7 consacré à l'étude de l'implémentation de cette norme dans une application de gestion de performances.

Mais avant de détailler les objets métriques nous pensons qu'il serait préférable pour le lecteur de présenter brièvement les informations de gestion définies dans [ISO10165-2]. En effet, les termes définis dans ce standard se retrouveront dans [ISO10164-11] et il nous a semblé important de bien comprendre chacun des termes.

Nous terminerons par une conclusion sur le standard concernant les objets métriques ou nous essayerons de pointer les éléments principaux de ce standard.

Définition des informations de gestion

Nous allons présenter les différentes classes d'objets, les attributs et les notifications qui sont le plus couramment utilisées lors du développement des MIS. Le but de cette partie n'est pas de présenter en détails les différents termes mais tout simplement de se mettre d'accord sur la définition à donner à ceux-ci.
Des exemples de ces différents attributs pourront être examinés à l'annexe 2.

Les attributs génériques

Vu l'utilisation intensive des opérations de dénombrement dans quelque contexte que ce soit, l'ISO/CCITT a défini un ensemble d'attributs génériques qui peuvent être utilisés ultérieurement pour redéfinir des attributs spécifiques concernant les opérations de dénombrement.

Les attributs génériques peuvent être considérés comme des types d'attributs ou des macros qui sont redéfinis pour construire des attributs spécifiques. La définition des attributs spécifiques peut étendre la définition des attributs génériques pour d'une part, relier le comportement des attributs aux opérations des ressources représentées par l'objet géré, d'autre part pour associer des paramètres supplémentaires à l'attribut et enfin pour relier l'attribut à d'autres attributs.

Nous pouvons distinguer deux catégories d'attributs génériques. Une catégorie relative aux ***compteurs*** et l'autre aux ***jauges***.

Les compteurs

De manière générale, le compteur est une abstraction d'un processus de dénombrement sous-jacent. Nous pouvons isoler les caractéristiques principales du compteur :

- La valeur d'un compteur est toujours entière et non négative
- Le compteur ne peut être qu'incrémenté et jamais décrémenté
- Une valeur maximale doit être spécifiée

- Lorsque la valeur maximale est atteinte, le compteur est réinitialisé à 0.

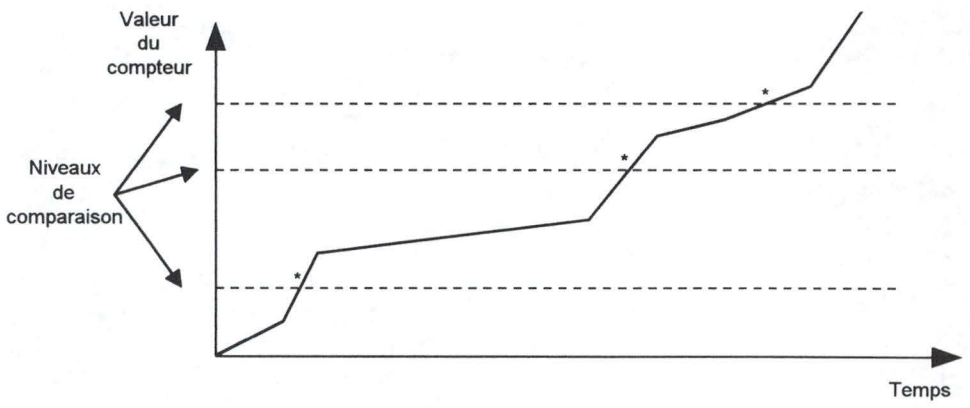
Nous devons discerner deux types de compteurs pour les besoins de gestion. D'une part les compteurs non modifiables et d'autre part les compteurs modifiables.
Les *compteurs non modifiables* ont la propriété de ne pouvoir être modifiés par des opérations de gestion mais uniquement suite à l'occurrence d'événements relatifs à l'objet auxquels ils se rapportent.
Les *compteurs modifiables* quant à eux peuvent être modifiés suite à des opérations de gestion.

Il est possible de générer une ou plusieurs notifications sur base des valeurs des compteurs en utilisant un attribut *counter-threshold* (seuil du compteur). La propriété du seuil est qu'une fois que la valeur du compteur atteint la valeur faisant office de seuil, une notification est émise. Nous pouvons également assimiler ce seuil à un niveau de comparaison.

Un *counter-threshold* peut avoir un ou plusieurs niveaux de comparaison. Dans le cas où un ensemble de valeurs forment le seuil, chaque valeur de l'ensemble déclenchera l'émission d'une notification.

Par exemple, les différents niveaux de comparaison représenteront différents degrés de sévérité d'une faute et des notifications seront émises lorsque la valeur du compteur atteindra chacun des niveaux de comparaison. (voir Figure 4-1)

Figure 4-1 : Ensemble de niveaux de comparaison

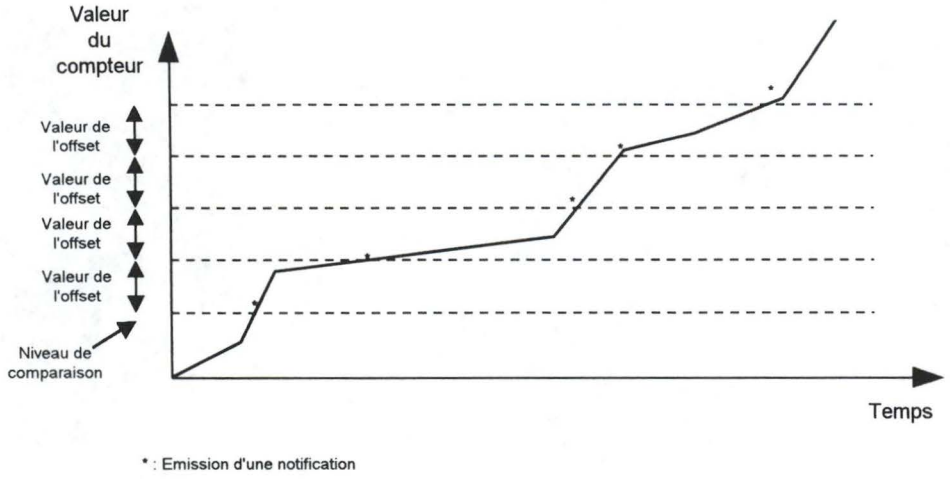


* : Emission d'une notification

En plus des seuils, nous pouvons inclure également des *offsets* (ou intervalles). Le fonctionnement de ces offsets est le suivant : lorsque le compteur atteint une des valeurs du *counter-threshold*, le niveau de comparaison est incrémenté de la valeur de l'offset, ce qui définit un nouveau niveau de comparaison. Une fois ce nouveau niveau de comparaison atteint, on ajoute la valeur de l'offset au niveau de comparaison. Ce mécanisme se répète indéfiniment (voir Figure 4-2).

Par exemple, un objet géré peut être configuré pour générer une notification chaque fois que 100 paquets sont reçus par un noeud.

Figure 4-2 : Niveau de comparaison avec offset



Si nous définissons des niveaux de comparaison multiples, alors chacun de ces niveaux de comparaison sera incrémenté de la valeur de l'offset lorsque sa valeur est atteinte par le compteur.

Nous remarquerons qu'un interrupteur relatif à l'émission des notifications est associé à chaque seuil. Le *counter-threshold* est activé et peut envoyer des notifications lorsque l'interrupteur est en position ***on***. Aucune notification ne sera émise dans le cas contraire.

Les jauges

La *jauge* est une abstraction d'une variable dynamique sous-jacente, comme par exemple le nombre de connexions qui sont ouvertes à un moment déterminé.

La valeur de la jauge est toujours un nombre entier non négatif qui peut augmenter ou diminuer tant qu'il reste entre les bornes[21] spécifiées. Des événements ayant pour but d'augmenter la jauge alors qu'elle avait atteint son maximum n'auront pas d'effet. C'est-à-dire qu'elle restera à sa valeur maximale. Un comportement similaire aura lieu pour valeur minimale de la jauge.

Une jauge est non modifiable, c'est-à-dire que sa valeur ne peut être modifiée par des opérations de gestion mais seulement par des changements de la variable sous-jacente.

Il est également possible de générer des notifications sur base des valeurs de la jauge en utilisant un *gauge-threshold* (un seuil de jauge). Deux types de notifications sont associés au gauge-threshold. Lorsque la jauge croise le seuil dans le sens positif, une notification signalant cet événement est émise. Lorsque la jauge croise le seuil dans le sens négatif, une notification signalant cet autre événement est émise (voir Figure 4-3).

Afin d'éviter des notifications répétées lorsque la valeur de la jauge oscille autour de la valeur du seuil, le seuil est défini par paire. Nous entendons par là que la valeur la plus élevée su seuil est utilisée pour déclencher un événement de sens positif et la valeur la plus basse du seuil déclenche un événement de sens négatif.

Le *gauge-threshold* se compose d'un ensemble de valeurs et chacune des valeurs de cet ensemble est à son tour composée d'un sous ensemble de deux valeurs : la première indiquant le NotifierBas et la seconde indiquant le NotifierHaut. A chacune de ces deux valeurs est associé également un interrupteur ON/OFF pour les notifications.

---

[21] Nous devons donc spécifier une valeur minimale ainsi qu'une valeur maximale.

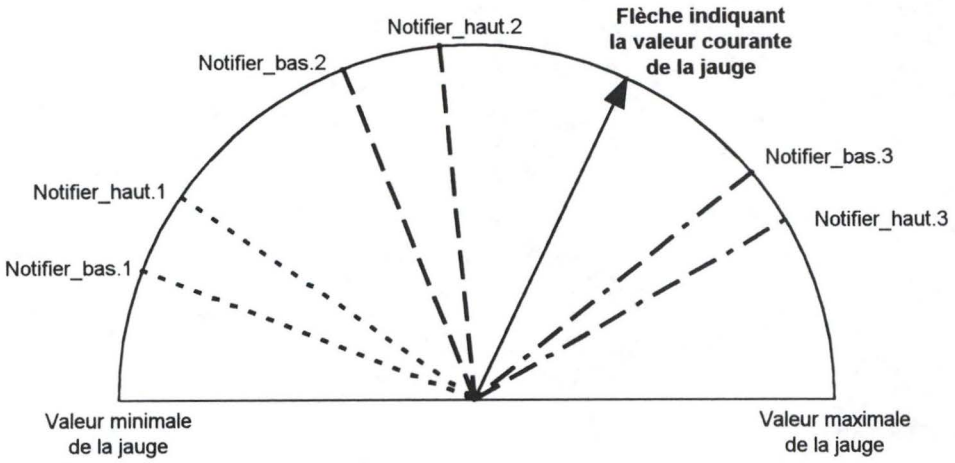


Figure 4-3 : Jauge et seuils de jauge. La jauge ci-contre possède trois seuils.

Nous allons maintenant expliquer le fonctionnement d'une jauge. Dès que la valeur de la jauge contient une valeur supérieure ou égale à la valeur du seuil NotifierHaut, la notification correspondante est générée. Lorsqu'un événement NotifierHaut est généré, une nouvelle notification de ce type ne pourra être générée avant que la valeur de la jauge ne devienne inférieure ou égale à la valeur du seuil NotifierBas[22].

Nous pouvons également associer un autre attribut aux jauges, il s'agit d'un attribut *tidemark*. Le *tidemark* est un mécanisme qui enregistre la valeur maximale et minimale atteinte par la jauge durant une période de mesure. Le *tidemark* contient trois valeurs :

- Sa valeur actuelle
- Sa valeur juste avant la dernière réinitialisation[23]
- L'heure de la dernière réinitialisation

Les attributs spécifiques

La norme [ISO10165-2] définit un certain nombre d'attributs spécifiques qui sont utilisés par les fonctions de gestion définies dans [ISO10164-X].

Les attributs spécifiques ont la caractéristique d'être complètement spécifiés et peuvent donc être utilisé directement dans les définitions d'objets gérés sans amendement supplémentaire.

Le lecteur intéressé pourra consulter l'annexe 1 afin de trouver des exemples concrets de ces attributs.

Les notifications et classes d'objets gérés

La norme [ISO10165-2] définit quelques types de notifications qui pourront être appliquées à une grande variété de classes d'objets.

---

[22] Le même principe est appliqué à la génération d'une notification suite à l'apparition d'un événement NotifierBas.

[23] La réinitialisation a pour fonction de remettre le *tidemark* a la valeur courante de la jauge et d'effacer les valeurs maximales et minimales mémorisées précédemment.

Cette norme définit également les classes d'objet qui sont référencées par les fonctions de gestion système définies par les normes [ISO10164-X]

Nous renverrons le lecteur intéressé par des exemples concrets à l'annexe 2 ou une liste non-exhaustive d'exemple est donnée.

Présentation du standard sur les objets métriques

Le standard [ISO10164-11] définit un modèle de surveillance des attributs des objets gérés. Il définit les objets gérés qui peuvent provoquer des événements sur base de valeurs contenues dans les compteurs et des jauges qui indiquent les performances du système.

Nous allons essayer de présenter ce standard de la manière la plus claire possible. Le lecteur qui souhaitera obtenir de plus amples informations[24] sur des points qui ne seraient pas assez détaillés se référera à la norme en question[25].

Généralités

La norme propose un certain nombre de services à l'utilisateur pour débuter, terminer, suspendre ou reprendre la surveillance de la charge (Workload Monitoring). Un des aspects essentiels de la surveillance est de déterminer les situations potentielles de surcharge d'une ressource, ce qui est un élément déterminant pour la prévention des anomalies.

Un concept clé qui sera récurrent tout au long de cet exposé est celui de la capacité.

Nous entendons par **capacité** la quantité de ressources qui peut être allouée à l'utilisateur. Cette quantité comprend les ressources déjà allouées ainsi que les ressources encore disponibles.

Le gestionnaire du système doit pouvoir examiner les ressources de son système. La fonction de surveillance (Workload Monitoring Function) lui permettra de vérifier à tout moment l'état de demande des ressources qui sont sous surveillance. Cette fonction peut être découpée en trois dimensions :

- *Un modèle d'utilisation des ressources* : utilisation des ressources, taux de refus, taux de demande
- *Un attribut d'objet géré* : Compteur modifiable et non modifiable, jauge
- *Un objet métrique* : un *gauge-monitor metric object* et un *mean-monitor metric object*

Un **objet métrique** est un objet géré qui contient au minimum un attribut dont la valeur est calculée à partir de valeurs observées dans les objets gérés.

Nous allons maintenant détailler ces trois dimensions.

Il y a tout d'abord le modèle d'utilisation des ressources (Ressource-Usage Model). L'architecture de ce modèle est subdivisée en trois sous-modèles.
L'utilisation des ressources (Resource Utilization) permet de surveiller la quantité utilisée de la ressource observée.
Le taux de refus mesure le nombre de demandes qui n'ont pas été satisfaites suite à une

[24] Entre autres sur la description en syntaxe ASN.1. des différentes classes d'objets.

[25] Auparavant le standard portait le titre suivant : *"Workload Monitoring Function"*.

surcharge de la ressource par exemple.
Le taux de demande représente la mesure du nombre de demandes qui ont été satisfaites pour la ressource observée.

Le gestionnaire peut utiliser un seul de ces trois modèles comme il peut en utiliser plus d'un. Si il en utilise plus d'un, les valeurs des attributs seront corrélées. En effet, il s'agit de mesures effectuées sur une même ressource et donc les variations d'utilisation se feront ressentir dans tous les modèles utilisés. Lorsque les trois modèles sont utilisés communément, nous pouvons obtenir une estimation de la charge de la ressource.

Une autre dimension de la fonction de surveillance est relative à la nature de l'attribut qui est utilisé pour refléter les performances du système. On choisira selon les besoins un compteur ou une jauge. Ces termes ont été expliqués dans la section concernant la définition des informations de gestion aux pages 65 et suivantes.

La dernière dimension est le type d'objet métrique utilisé. Les deux types seront expliqués par la suite.

Le processus de surveillance des objets métriques

Ce processus de surveillance est couramment divisé en quatre phases : la capture des données, la conversion des données, l'amélioration des données et enfin l'analyse des données.

1. La *capture des données.* Au cours de cette phase les données sont extraites des objets observés. Les données sont extraites à des intervalles réguliers. Ces intervalles sont conservés dans la *granularity period.*
2. La *conversion des données.* Les données peuvent être converties en une forme qui est adéquate à leur utilisation. Les objets métriques peuvent disposer d'algorithmes pour effectuer ces conversions. Si ces algorithmes n'existent pas, aucune conversion n'est réalisée[26].
3. *L'amélioration des données (Data enhancement).* Il est parfois utile d'observer des tendances dans les valeurs observées. A cette fin, il est possible de faire appel à un algorithme pour lisser les valeurs observées. On effacera ainsi les variations aléatoires des données. Le résultat de ce lissage est stocké dans une jauge.
4. *L'analyse des données.* Dans le cadre de la surveillance des objets métriques, cette analyse des données ne consiste qu'en la comparaison des valeurs des jauges avec les seuils afin de déclencher des alarmes.

La **granularity period** est le temps entre le lancement de deux interrogations successives.

Un **métrique** est une valeur qui est calculée à partir de valeurs observées

Un **attribut métrique** est l'attribut d'un objet métrique dont la valeur est ou bien utilisée comme paramètre d'un ou plusieurs algorithmes métriques ou bien représente le résultat d'un tel algorithme.

Un **algorithme métrique** représente le comportement d'un objet métrique et il modélise un processus de calcul pour obtenir des résultats bien spécifiés.

---

[26] Nous pouvons par exemple préférer une jauge à un compteur... Une conversion devra être réalisée. Cette conversion consistera à faire la différence entre les valeurs observées dans le compteur à des intervalles déterminés.

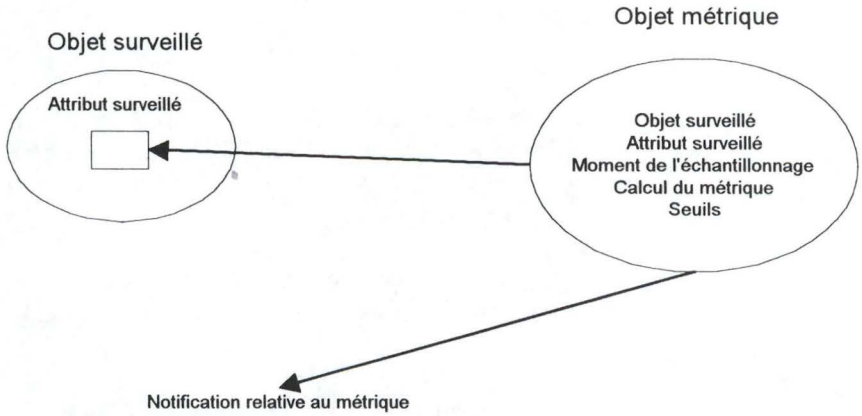
Les objets métriques doivent nécessairement avoir des caractéristiques propres, voici quelques-unes de leurs caractéristiques :

- l'identification de l'objet métrique
- l'identification des objets gérés observés et de un ou plusieurs de leurs attributs
- l'identification de l'algorithme métrique utilisé dans les observations
- le nombre d'échantillons, la fréquence des observations et le moment de la dernière observation
- le planning des observations
- l'indication des résultats des algorithmes métriques
- les valeurs de seuils
- la gestion des états d'administration

Le système de gestion peut demander la création d'un objet métrique. Au moment de sa création, l'objet métrique envoie lui-même une notification afin d'indiquer au système de gestion que l'objet métrique a bel et bien été crée. Un comportement similaire est effectué pour la suppression de l'objet métrique.

La relation entre les objets métriques et les autres objets gérés est également très importante. Les objets métriques disposent d'attributs relationnels qui ne peuvent être accédés qu'en lecture et qui sont établis lors de la création de ces objets. La relation de l'objet métrique avec un objet géré est une relation unidirectionnelle asymétrique[27]. La Figure 4-4 nous montre la relation qu'il existe entre l'objet surveillé et l'objet métrique.

Figure 4-4 : Relation entre l'objet surveillé et l'objet métrique



Afin d'avoir une idée du comportement futur de certains objets observés, nous pouvons avoir recours à l'utilisation d'algorithmes de "lissage". Les annexes B et C de [ISO10164-11] décrivent des algorithmes. Nous verrons rapidement en quoi consistent ces algorithmes dans la suite de ce chapitre.

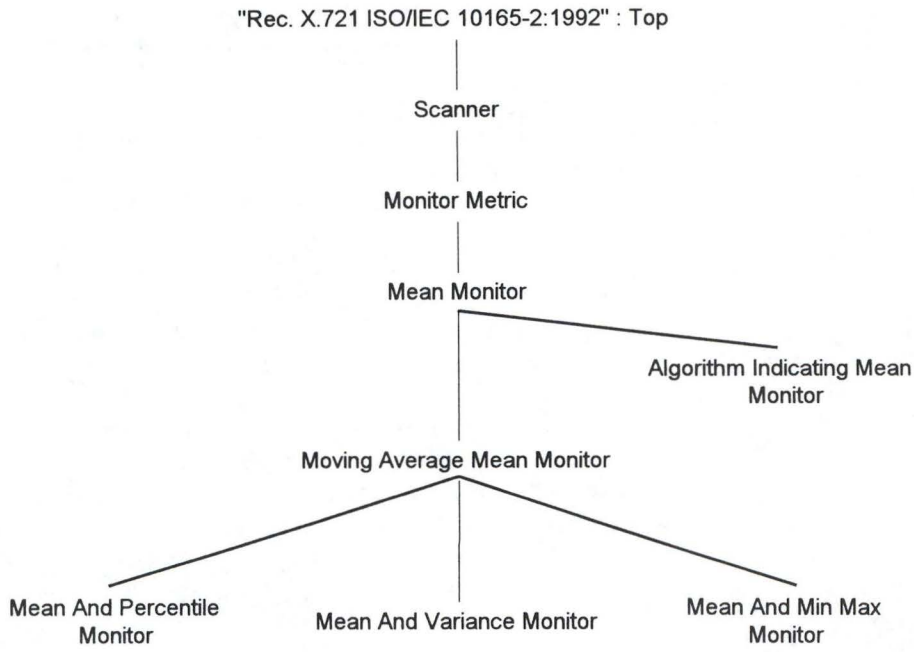
Les objets métriques supportés

Nous présenterons ici les objets métriques qui sont définis par le standard [ISO10164-11]. Les objets métriques ont pour particularité de pouvoir surveiller chacun des trois types d'attributs de surveillance de performance (compteur non modifiable, compteur modifiable et jauge). Ces objets peuvent également supporter chacun des trois sous-modèles (taux d'utilisation, taux de refus, taux de demande) vus un peu plus haut.
Les objets métriques se partagent des attributs et on peut donc les organiser au sein d'une

[27] L'explication de ce terme est données dans la présentation de la **Fonction de gestion des relations** à la page 45

structure d'héritage comme indiqué à la Figure 4-5. Partons du haut de l'arbre expliquons chacune de ses composantes (Scanner, monitor metric, mean monitor, algorithm indicating mean monitor, moving average mean monitor, mean and variance monitor, mean and percentile monitor et enfin mean and max monitor).

Figure 4-5 : Structure d'héritage des objets métriques



Tous les objets métriques présentés ci-dessous héritent de la classe ***scanner***. Cette classe définit les moyens de sonder périodiquement les valeurs d'un ensemble d'attributs présents dans un objet géré. Les intervalles entre les différents moments de sondage sont déterminés par un planning.[28]

Notons que tous les objets de surveillance de la charge descendent de l'objet scanner. Le *scanner* consiste donc en un objet de base rapatriant les données relatives aux objets gérés.

Le premier objet métrique défini par le standard est l'objet géré ***monitor metric***. Ce métrique est utilisé pour observer attribut de type compteur ou jauge et pour mettre à jour les attributs de jauges dérivés après chaque observation.
Les attributs obligatoires sont donc l'identifiant de l'objet géré à surveiller ainsi que les identifiants des attributs qu'il faut observer.
L'attribut derivedGauge (jauge dérivée) contient l'observation la plus récente et il est dérivé des attributs des objets gérés suivant les règles suivantes :

- Si l'attribut est une jauge, la valeur de la jauge dérivée est tout simplement la valeur de la dernière jauge observée.
- Si l'attribut est un compteur non modifiable, alors la valeur de la jauge dérivée est la différence entre deux observations successives du compteur.
- Si l'attribut est un compteur modifiable, la valeur de la jauge dérivée est la valeur du compteur juste avant sa dernière réinitialisation.

---

[28] Nous avons déjà parlé des scanners au chapitre 2 sur la gestion OSI dans la section relative aux **Fonction de résumé des mesures** à la page 50.

En ce qui concerne le calcul de la jauge dérivée dans le cas d'un compteur simple, on doit tenir compte de son éventuel retour à zéro (lorsque le compteur est arrivé à son maximum, il est remis à zéro).

La formule pour calculer la jauge dérivée est alors la suivante :

$$V(t) = [compteur(t) - compteur(t - GP) + CWV] \text{ modulo } CWV$$

avec

V(t) :	*La valeur de la jauge dérivée*
compteur(t) :	*La valeur du compteur au temps t*
compteur(t - GP):	*La valeur du compteur au temps (*t-GP)
GP :	*L'intervalle d'échantillonnage*
CWV :	*Maximum du compteur*

La classe d'objets gérés ***mean monitor*** constitue le deuxième objet métrique. Il est dérivé de la classe d'objet *monitor metric managed object.* Il vient ajouter aux attributs de cette classe, un attribut contenant l'estimation de la moyenne de la valeur de l'attribut de jauge dérivée. La différence avec la classe précédente est que la valeur dérivée servira de donnée au calcul de l'estimation de la valeur moyenne.

On peut distinguer différents types d'objets métriques *mean monitor* , ils sont situés en sous de la classe Mean Monitor dans la hiérarchie.

La classe d'objets gérés ***algorithm indicating mean monitor*** est dérivée de la *mean monitor managed object class.* Cette classe fournit en plus l'identifiant d'un algorithme calculant l'estimation de la moyenne.

La classe d'objets gérés ***moving average mean monitor*** est dérivée de la classe d'objets gérés *mean monitor.* Cette classe est la plus simple, elle ne sert qu'a donner une estimation de la moyenne. Elle a pour unique fonction de spécifier un des deux algorithmes suivants pour calculer cette estimation :

1. l'algorithme **U**niformly **W**eighted **M**oving **A**verage (UWMA)

2. l'algorithme **E**xponentially **W**eighted **M**oving **A**verage (EWMA)

Le premier algorithme donne une importance égale à toutes les observations. La formule de calcul de la moyenne devient donc :

$$\sim V(t) = \frac{\sum_{i=0}^{N-1} V(t - (i * GP))}{N}$$

où		
	V(t)	*est la valeur de la jauge dérivée au temps t.*
	*V(t - (i * GP))*	*est la valeur de la jauge dérivée au moment* t - *(i * DT)*
	~V(t)	*Estimation de la moyenne de V(t) au moment t*
	GP	*Temps entre deux observations successives de V(t)*

Remarquons que pour effectuer ce calcul, il est nécessaire de sauvegarder les N dernières valeurs observées.

Généralement on essaiera de donner plus d'importance aux dernières valeurs observées car elles reflètent mieux le comportement actuel et futur de la ressource observée. Nous

utiliserons dans ce cas la seconde technique dite EWMA. Pour réaliser ce calcul, nous utiliserons la formule suivante :

$$\sim V(t+GP) = \alpha * V(t+GP) + (1-\alpha) * (\sim V(t))$$

En utilisant une valeur constante et indépendante des observations passées pour α, nous considérons les valeurs antérieures mais nous attribuons moins d'importance aux moins récentes.

Dans la pratique, nous remarquerons que l'algorithme EWMA estime de manière plus efficace le comportement de la ressource par rapport à une estimation effectuée avec l'algorithme UWMA.

La classe d'objets gérés ***mean and variance*** fournit une estimation de la moyenne et de la variance d'un attribut observé. Cette classe est dérivée de la classe *moving average mean monitor.* En plus des attributs de la classe *moving average mean monitor,* la classe *mean and variance* fournit un attribut contenant l'identifiant d'un algorithme utilisé pour calculer la variance.

La formule pour calculer la variance utilise également un algorithme exponentiel :

$$\sim S(t) = \frac{\sum_{i=0}^{N-1} [V(t) - \sim V(t)]^2}{N-1}$$

où ~S(t) représente l'estimation de la variance au temps t

La classe d'objets gérés ***mean and percentile monitor***. Cette classe d'objet donne une estimation de la moyenne, du médian, du $n^{ième}$ percentile, du $100^{ième}$ percentile, de la plus petite et de la plus grande valeur de la jauge dérivée. Le *n* dans $n^{ième}$ représente un entier entre 1 et 49.

La classe d'objets gérés ***mean and min max monitor*** donne les valeurs de la moyenne, et des valeurs minimales et maximales de l'attribut observé. Elle est dérivée de la classe *mean monitor.*

Le modèle de la gauge-threshold avec indication de la sévérité

La norme [ISO10164-11] définit un attribut de type *severity indicating gauge-threshold* qui est utilisé pour le déclenchement de notifications pour les ressources surveillées. Cet attribut a un comportement similaire à celui du gauge-threshold (voir Les jauges, page 67).

Nous ajoutons aux caractéristiques du gauge-threshold un attribut optionnel qui contient une indication de la sévérité pour chacun des NotifierHaut et NotifierBas des seuils définis.

Si un paramètre d'indication de sévérité est présent, sa valeur est utilisée lors de toutes les alarmes déclenchées pour le seuil en question. Typiquement, on aura les sévérités suivantes :

- EWCT = Early-warning-clear threshold
- EWT = Earlry-warning threshold

- SCT = Severe-clear threshold
- ST = Severe threshold

Le fonctionnement de la jauge est identique à celui expliqué dans la section consacrée aux jauges à la page 67.

Conclusion

Nous retiendrons de cette norme [ISO10164-11] qu'elle regroupe un grand nombre de concepts intéressants. En effet, une application qui utiliserait une MIB telle que celle définie dans cette norme serait à la fois complète et efficace pour gérer les performances d'un système.

Mais nous savons à quel point les normes édictées par l'ISO sont complexes et par le fait même souvent très difficiles à implémenter dans la réalité.

Cette complexité est en partie due à l'énorme interaction qu'il existe entre les différentes fonctions de gestion système définies par l'ISO. Dans le cas de la norme précédemment expliquée, elle fait appel aux alarmes, à la définition d'autres objets gérés, aux relations entre objets gérés ... tous ces éléments sont à prendre en compte si on veut respecter le standard tel qu'il est présenté par l'ISO.

Néanmoins, différents degrés d'implémentation de ce standard peuvent exister dans une MIB. Plus la fonction de surveillance de la charge devra être fine, plus la mise en œuvre du standard dans la MIB devra être importante.

Nous pensons donc que cette norme a beaucoup d'avenir devant elle et il ne reste donc plus qu'à lui trouver un environnement adéquat afin de la développer.

Chapitre

5

Présentation d'ISM/OpenMaster

« A framework must be judged by its ability to share information among its applications ; object-oriented software may be the key. »

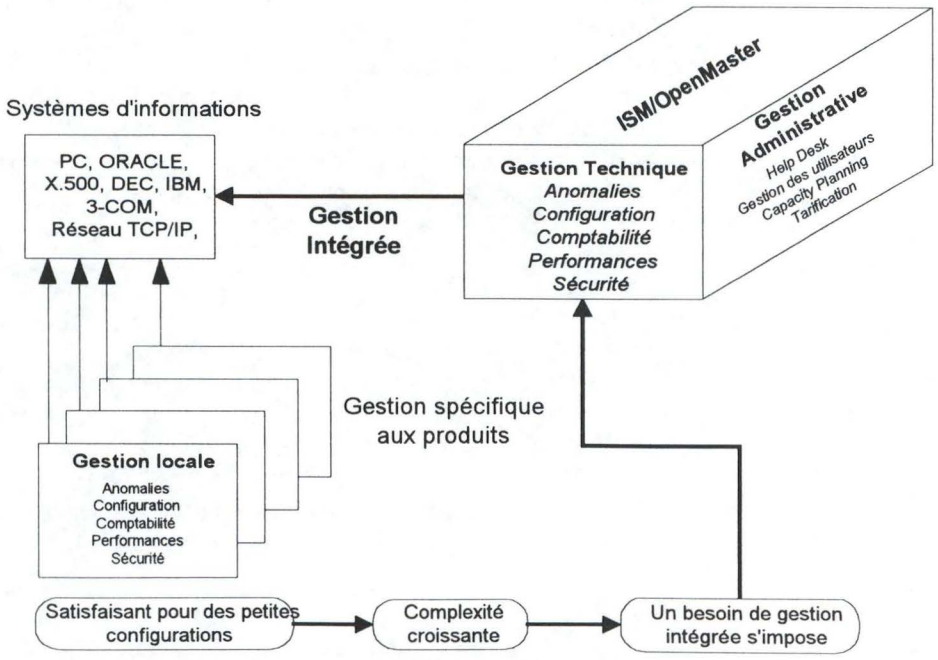
Mary Jander

Dans ce chapitre[29], nous présenterons la plate-forme sur laquelle nous avons travaillé durant notre stage chez Bull. Cette plate-forme, appelée ISM/OpenMaster, comporte plusieurs applications, que nous ne détaillerons pas ici.
Seule l'application, appelée Performance Monitoring Service (ou encore ISM Performance), avec laquelle nous avons travaillé sera présentée de manière plus complète dans le chapitre 6 car elle fera l'objet d'une étude plus approfondie dans la suite de ce mémoire.

Présentation d'ISM/OpenMaster

ISM/OpenMaster, plate-forme de gestion intégrée est apparu suite à la complexification et à la diversité des applications de gestion. Contrairement à une multitude d'outils isolés, ISM/OpenMaster permet aux administrateurs d'avoir une vue homogène d'un système distribué. La Figure 5-1 illustre le contexte dans lequel évolue ISM/OpenMaster.

Figure 5-1 : Contexte de développement d'ISM/OpenMaster



[29] La rédaction de ce chapitre a été réalisée avec la collaboration de Pascal Libert, nous retrouverons donc le même chapitre dans [Libert, 1996].

ISM/OpenMaster est composé d'un ensemble d'outils de gestion de systèmes et de réseaux opérant dans un environnement multi-protocolaire et multi-domaine.

C'est-à-dire une plate-forme de gestion intégrée capable de supporter des protocoles de gestion tels que :

- Des protocoles propriétaires — AEP[30] (Bull DSA Administrative Exchange Protocol)
- Les standards de l'ISO — CMIP
- Les standards de fait — SNMP
- Des protocoles spécifiques — SMT pour les Fiber Distributed Data Interface
  SNA pour IBM
  GMP[31], ...

L'environnement multi-domaines permet de gérer les différents domaines suivants :

- ISM SQL Master pour les bases de données
- ISM TMN Master pour les télécommunications
- ISM Operation Matser pour les systèmes
- ISM TransMaster pour les réseaux
- ISM Access Master pour la sécurité
- ISM PC Operation Master pour les groupes de travail

Architecture d'ISM/OpenMaster.

L'architecture d'ISM/OpenMaster est basée sur des standards qui ont été établis par l'ISO, le NM Forum (OMNIPoint), POSIX/IEEE, X/Open, l'Internet Engineering Task Force (IETF) et ITU-T. L'un des concepts clé dans ISM/OpenMaster est celui de la MIB. Nous détaillerons ce concept à la section suivante.

Dans un souci de compréhension, nous avons choisi d'illustrer nos propos au moyen de nombreux schémas explicatifs. Ces schémas, issus de la formation que nous avons reçue durant notre stage chez Bull, offrent une vision évolutive de l'architecture d'ISM/OpenMaster.

L'architecture globale d'ISM/OpenMaster repose en majeure partie sur le principe *Manager/Agent*. Il s'agit donc, comme nous l'avons déjà vu[32], d'une architecture à deux niveaux, d'une part l'agent et d'autre part de manager.
Le mécanisme *Manager/Agent* est basé sur un modèle orienté objet. Les objets réels d'un système distribué sont représentés par des objets abstraits formant eux-même la MIB. Les caractéristiques des objets peuvent être retrouvées grâce aux attributs des objets.

L'ISM/Manager perçoit le système distribué au travers des objets de la MIB et contrôle ce système via la manipulation des objets et de leurs attributs.

---

[30] Le protocole AEP (Administrative Exchange Protocol) est le protocole véhiculaire de base pour la communication des informations de gestion d'un réseau DSA. On distingue trois principaux types d'informations de gestion. Il s'agit des commandes, des réponses à ces commandes, et des messages non sollicités signalant des événements ou des erreurs se produisant sur un site.

[31] GMP GCOS Management Protocol.

[32] Voyez ce qui en est dit dans l'**Aperçu Général et Architecture** à la page 29.

Les ISM/Agents vont permettre à l'ISM/Manager de visualiser les objets du monde réel en représentant l'état d'un objet réel par ses attributs dans la MIB. Comme illustré à l'annexe 3 - **Gérer le monde réel**, l'ISM/Manager envoie des commandes vers l'ISM/Agent, qui à son tour renverra des réponses et des notifications en utilisant des protocoles de gestion (SNMP,CMIP, DSAC/AEP, ...).

Notons qu'il est également possible de distribuer les composantes d'ISM/Manager sur différents systèmes afin de satisfaire différents besoins :

- ***Performance*** : pour accroître la quantité de ressources disponibles au niveau central
- ***Résilience*** : Afin d'éviter les désagréments de la panne du composant central qui pourrait avoir comme conséquence le dysfonctionnement du système.
- ***Exécution Distribuée*** : pour avoir plusieurs centres de gestion reliés entre eux.

ISM/OpenMaster Manager

Les applications de gestion

Les applications sont les composantes d'ISM/OpenMaster qui ont une interface homme-machine. Plusieurs copies d'une même application peuvent être actives. Pour chaque fenêtre ouverte par un utilisateur, il existe une copie active de l'application correspondante.

Dans la plupart des cas, les applications sont développées dans le langage SML (System Management Language), un langage de développement qui permet d'ajouter de nouvelles fonctionnalités à ISM/OpenMaster.

Ces applications de gestion interagissent avec les ressources gérées par les ISM/Agents au moyen d'une interface de gestion. Cette interface de gestion représente le cœur d'ISM/OpenMaster.
Les différents concepts que nous venons d'exposer sont illustrés à l'annexe 3- **Les applications de gestion**.

Il nous semble inutile dans le cadre de ce mémoire d'énumérer toutes les applications existantes. Nous nous contenterons de présenter, à titre d'exemple, ISM/Monitor.

L'utilisation d'ISM/Monitor permet à l'administrateur de visualiser en temps-réel le statut de toute composante existant dans le système. ISM/Monitor affiche un système au moyen d'une carte dans laquelle les différents objets gérés par l'agent peuvent être représentés et animés graphiquement.

Un modèle de gestion unique

Ce modèle de gestion contient les deux mécanismes principaux de l'ISM/Manager :

- le *CMIS Dispatcher*
- les *Objects Managers* qui se décomposent en services d'ISM/OpenMaster , en Intégrateurs d'agents (AI) et en Supra Management Interface (SMI)

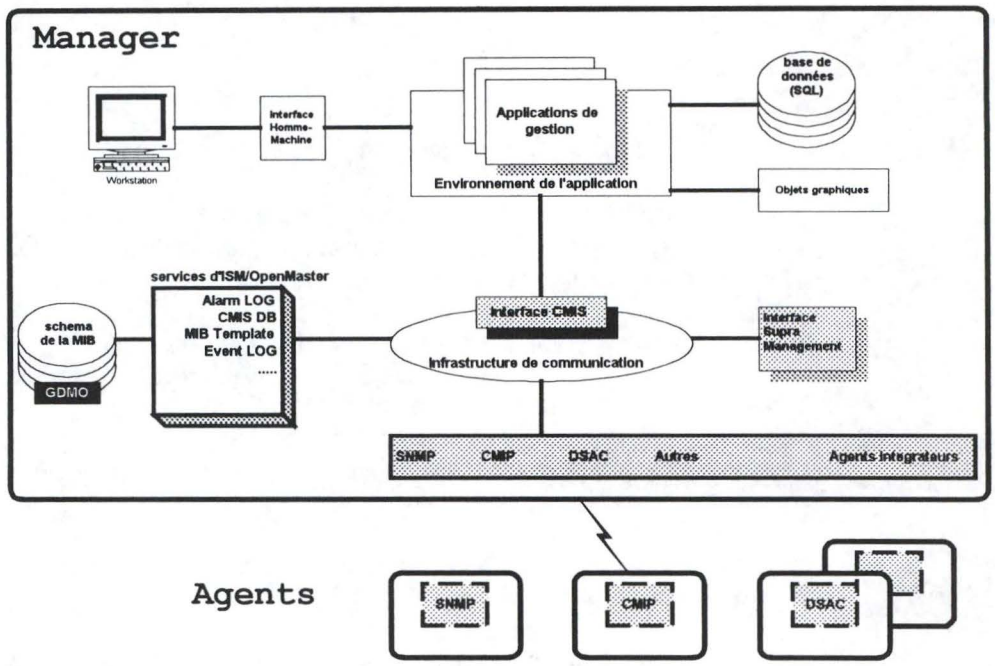
Nous allons détailler chacun des mécanismes présents à la Figure 5-2, sans être exhaustif en ce qui concerne les services

Le *CMIS Dispatcher* permet la communication entre les applications ISM/OpenMaster et tous les Objects Managers. Tous les composantes de l'ISM/Manager vont communiquer entre elles grâce au CMIS Dispatcher.

Le CMIS Dispatcher, illustré par l'infrastructure de communication, comprend les fonctions suivantes :

- le **CMIS Application Programmatic Interfaces**, ces API fournissent une interface-CMIS donnant accès au *Internal Messaging Mechanism*
- l'**Internal Messaging Mechanism**, ce mécanisme est utilisé pour permettre la communication des applications, des services et des AI au sein de l'ISM/Manager.
- le **Command Router** qui oriente les requêtes et les réponses entre les applications et les Objects Managers.
- l'**Event Router** qui oriente les notifications entre des Objects Managers vers les applications
- Le **Root Object Manager (ROM)** qui est gestionnaire d'objet pour l'objet ROOT.

Figure 5-2 : Une vue unique du modèle de gestion d'ISM/OpenMaster



Les *services d'ISM/OpenMaster* sont les composantes d'ISM/Manager qui n'ont pas d'interface homme-machine et qui fournissent des fonctions indépendantes d'un protocole de gestion. Les services sont des Objects Managers, ce qui signifie que leurs fonctionnalités sont accessibles via la MIB.

Les services disponibles au sein d'ISM/OpenMaster sont les suivants :

- **MIB Template Service** qui représente la base de données des classes d'objets gérés.
- **Alarm Log Service** qui permet de créer un journal d'alarmes
- **Event Log Services** qui permet de créer un journal de notifications d'événements
- **CMIS Database** est une base de données contenant les objets gérés accessible via CMIS.

Les SMI permettent à un ISM/Manager de communiquer avec d'autres managers. Ces composantes permettent à un ISM/Manager d'agir en tant qu'agent contrôlé par un manager, appelé *supra-manager*. Ce concept se retrouve sous l'ISM/Concentrator. L'ISM/Concentrator filtre les informations de gestion qui devraient être envoyées des agents vers le manager central (qui, dans notre cas, n'est autre que le supra-manager), n'envoyant ainsi que ce qui est réellement nécessaire. L'ISM/Concentrator prend à sa charge l'interrogation des agents, la surveillance des paramètres et la notification d'alarmes vers le supra-manager lorsque cela est nécessaire. Ce que nous retiendrons des Supra

Management Interfaces, c'est qu'elle permettent de hiérarchiser la centralisation des applications de gestion.

Les AI's communiquent avec les agents d'un sous-système de gestion spécifique en utilisant le protocole de gestion de l'agent. Ils communiquent avec les autres composantes d'ISM/OpenMaster à l'aide du service CMIS. La version actuelle d'ISM/OpenMaster comprend un AI par protocole de gestion supporté. Nous retrouvons ainsi un AI SNMP ; un AI CMIP , un AI DSAC/AEP et un AI GMP qui est un protocole propriétaire de Bull. Chaque AI peut supporter plusieurs agents et agit comme un multiplexeur. Comme nous le verrons (cfr. Les outils de développement d'ISM/OpenMaster.83) d'autres intégrateurs d'agents peuvent être ajoutés selon les besoins du client permettant ainsi d'utiliser des protocoles aussi divers que SNA d'IBM, X25, etc.

ISM/OpenMaster Agent

Sur chaque machine du système distribué tourne au moins un agent responsable du contrôle de cette machine. Les agents, qui constituent en fait les représentants de l'ISM/Manager sur la machine gérée, doivent rendre les objets gérés visibles pour l'ISM/Manager. Ce sont eux qui gèrent, localement, les différents objets composant la MIB. Ils doivent également effectuer toutes les opérations de gestion demandées par l'ISM/Manager. Il existe donc différents types d'agents implémentés dans ISM/OpenMaster :

- les agents SNMP
- les agents CMIP
- les agents DSAC/AEP
- les agents GMP
- les agents développés par client ...

La MIB d'ISM/OpenMaster.

Toutes les classes d'objets gérés de la MIB d'ISM/OpenMaster sont définies selon le standard contenant les spécifications des objets de l'architecture d'ISM/OpenMaster, ISM/GDMO. Cette architecture est basée sur la norme [ISO10165-4].
La MIB ISM/OpenMaster contient toutes les classes d'objets nécessaires à la gestion de l'ensemble du système distribué. Les classes d'objets vont donc représenter :

- des composantes locales (logiciels, périphériques, etc.)
- des utilisateurs
- des composantes de communication (interfaces, circuits, etc.)
- des abstractions du système distribué (vendeurs, localisation de composants ; etc.)
- des objets de gestion (agents, fichiers de journalisation , etc.)

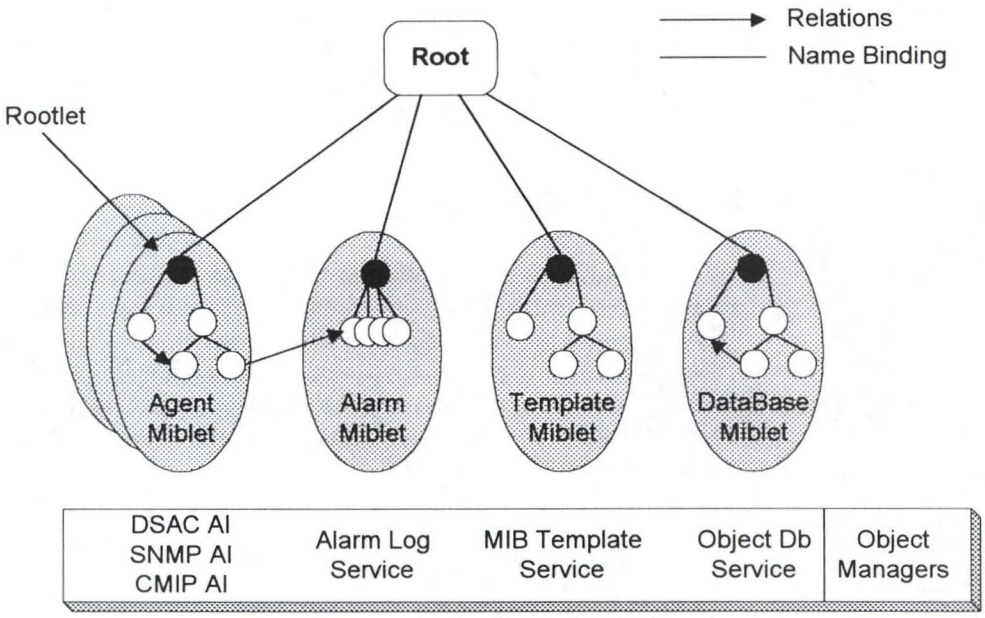
Comme nous l'avons vu précédemment, ISM/OpenMaster supporte plusieurs protocoles de gestion. La MIB ISM/OpenMaster devra dès lors contenir les classes d'objets gérés conformes aux protocoles SNMP, DSAC, CMIP, etc.

La MIB ISM/OpenMaster est représentée par un ensemble de sous-MIB appelées *MIBlets*. Chacune des MIBlets est un sous-arbre de la MIB ISM/OpenMaster et est attachée directement à la racine (comme illustré à la Figure 5-3). L'instance de l'objet géré attachée à la racine porte le nom de ***Rootlet***.

Chaque MIBlet est instanciée auprès de l'ISM/Manager afin qu'elle soit connue par le système. Les éléments d'ISM/OpenMaster qui se chargent de cette instanciation

s 'appellent les ***gestionnaires d'objets***[33]. Les intégrateurs d'agents, qui sont des gestionnaires d'objets, ont pour fonction de rendre visibles les éléments avec lesquels ils communiquent. Ainsi, l'intégrateur d'agent SNMP va instancier la MIBlet d'un agent SNMP qui contient les classes d'objets dérivées des objets SNMP (par exemple : SNMP System, TCP group, etc.).

Figure 5-3 : Représentation de la MIB ISM/OpenMaster



Malgré cette division de la MIB en MIBlets, la MIB d'ISM/OpenMaster est vue comme une seule entité où toutes les MIBlets partagent la même racine. Une application peut effectuer des opérations sur des instances d'objets gérés au sein de la MIB sans en connaître le gestionnaire d'objet qui s'en occupe. L'application mentionne simplement le DN de l'instance d'objet géré et c'est le CMIS Request Broker qui se charge d'orienter la requête vers le gestionnaire d'objet adéquat.

Nous retrouvons deux types d'objets au sein de la MIB. Les objets statiques et les objets dynamiques.
Les objets *statiques* sont crées et supprimés peu fréquemment. Comme ils sont présents plus longtemps dans le système, les utilisateurs et les opérateurs en ont connaissance. Ils représentent :

- des composantes matérielles : une imprimante, un modem, ...
- des configurations logicielles : une version d'un système d'exploitation par exemple.

Les objets *dynamiques* sont des objets qui seront créés et supprimés de manière régulière. Comme ils ne sont pas tout le temps présents dans le système, ils n'est pas utile de leur attribuer un nom explicite ou de les représenter sous forme d'icône. Les objets qui répondent à cette caractéristique sont par exemple :

- des connexions : des connexions TCP, ...
- des enregistrements d'événements dans des journaux : les alarmes, ...

Notons que la MIB d'ISM/OpenMaster est extensible et ce pour plusieurs raisons.

---

[33] Les services ainsi que les intégrateurs d'agents sont des gestionnaires d'objets.

- pour prendre en considération des nouveaux objets qui sont gérés par les agents
- pour le développement de nouveaux agents
- pour ajouter de nouvelles classes d'objets à la base de données CMIS (CMIS-DB)

Les outils de développement d'ISM/OpenMaster.

Les administrateurs, soucieux de trouver une solution intégrée pour la gestion de leur système distribué, sont bien souvent contraints d'utiliser les seules applications offertes par le système de gestion qu'ils vont choisir.

En plus des applications intégrées dans ISM/OpenMaster, Bull offre aux administrateurs des outils de développement leur permettant de développer leurs propres applications de gestion. Ce développement se fera grâce au langage propriétaire ISM Management Language, mieux connu sous le nom de SML. Il s'agit d'un langage interprété de haut niveau, proche du LISP. SML permet par exemple d'afficher une valeur représentée graphiquement en une seule ligne de code. Il réagit à des événements dans la même philosophie que X-Windows.

En plus de ce langage de développement, Bull offre une série de boîtes à outils. Les deux principales boîtes à outils sont l'*AI Toolkit*[34] et le *SNMP Agent Toolkit* (SAT). La première permet à chaque administrateur de créer son propre AI, au cas où il utiliserait un protocole autre que CMIP, AEP ou SNMP. Quant à la seconde boîte à outils, elle facilite le développement de nouveaux agents SNMP. Le SNMP Agent Toolkit comprend des librairies de fonctions C qui s'occupent de l'implémentation des opérations SNMP. Il existe en outre des supports à la compilation (*scripts*, *makefiles*), au debugging et aux tests de fonctionnalités de l'agent.

Conclusion

Qu'il s'agisse de gérer des serveurs, des systèmes Unix propriétaires ou ouverts voire même des réseaux d'ordinateurs individuels, ISM/OpenMaster est sans conteste la solution intégrée la plus adéquate. Elle agit en effet dans un milieu tout à fait hétérogène, respecte la plupart des standards émis par l'ISO et est compatible avec de nouvelles technologies telles que CORBA de l'OMG ou encore XMP de X/open. Pour satisfaire de telles contraintes, cette plate-forme logicielle fournit aux utilisateurs des applications intégrées de gestion.

Tout comme nous l'avons vu, ISM/OpenMaster possède également un ensemble d'outils de développement permettant à ses utilisateurs de construire leurs propres solutions de gestion.

[34] Cette boîte à outils est appelée *ADK CMIP* et permet de développer à la fois des agents des intégrateurs d'agents et des services.

Chapitre

6

Présentation d'ISM Performance

L'objectif de ce chapitre est de présenter les différentes fonctionnalités ainsi que l'interface de l'application sur laquelle nous avons travaillé. Nous discernons ces deux aspects car ils font l'objet de deux parties séparées chez Bull. L'interface est représentée par ISM Performance Control et l'application proprement dite est appelée ISM Performance[35].

Nous montrerons également les modifications que nous avons effectuées sur le programme ISM Performance durant notre stage chez Bull.

Cadre général

ISM Performance et ISM Performance Control sont deux composantes d'ISM/OpenMaster. L'objectif d'ISM Performance Control est de contrôler la configuration d'ISM Performance via une interface graphique. ISM Performance est un *service*[36] qui peut être utilisé par différentes applications pour effectuer des interrogations (polling) de la MIB avec les buts suivants :

- Un rassemblement de statistiques (off-line)
- La génération automatique d'alarmes lorsque la valeur d'attributs surveillés dépasse le seuil fixé.
- La génération automatique de notifications lors du changement de valeur d'un attribut.

Notons que ISM Performance ne contient pas de fonctions qui permettent d'exploiter les statistiques[37] , alarmes ou les notifications.

Voyons maintenant la position qu'occupe ISM Performance au sein d'ISM/OpenMaster. Nous allons nous baser sur le schéma illustré à la Figure 5-1 et en expliquer les différentes relations entre les composantes.

La partie ISM Performance Control représente une interface-utilisateur pour l'utilisation d'ISM Performance. Au moyen de cette interface, l'utilisateur peut :

- Configurer la session de surveillance en termes d'éléments d'interrogation. Un tel élément décrit la fréquence des interrogations, la requête CMIS à exécuter et une instance de l'objet géré.
- Définir si le journal doit être activé ou pas pour les valeurs interrogées
- Définir les attributs à surveiller
- Définir les différents seuils attachés aux attributs

---

[35] Nous parlerons de service de performance ou encore d'ISM Performance, ces deux termes signifiant la même chose.

[36] Rappelons que les services font partie des gestionnaires d'objets tout comme les intégrateurs d'agents.

[37] Notons néanmoins que ces fonctions sont présentes dans d'autres applications au sein d'ISM/OpenMaster et qu'elles peuvent utiliser les résultats fournis par PMS.

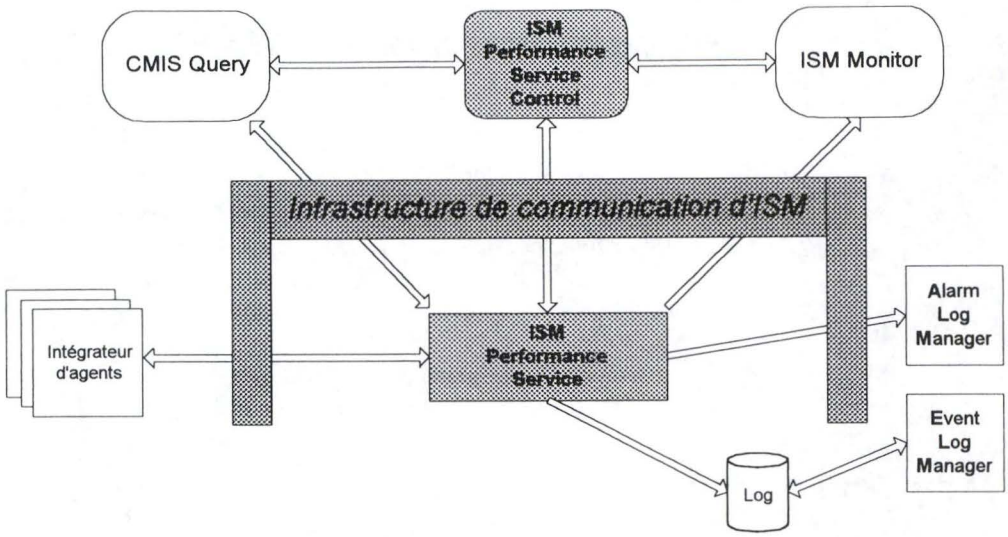


Figure 5-1 : ISM Performance au sein d'ISM/OpenMaster

La partie la plus importante, et celle qui nous intéresse le plus dans le carde de ce mémoire, est ISM Performance.
ISM Performance est un gestionnaire d'objet et il a donc pour fonction de gérer les objets de la MIB[38] relatifs au service de performances. Citons comme exemples les sessions, les requêtes, les seuils ...
Le service de performances va interroger à des intervalles réguliers les objets de la MIB qui sont définis dans les sessions de surveillance en 'exécutant' des requêtes prédéfinies qui vont envoyer des requêtes CMIS aux intégrateurs d'agents. Lors de la réception des réponses aux requêtes CMIS, ISM Performance va exécuter une des actions suivantes :

- Enregistrer les valeurs des attributs reçues en tant qu'événement dans des fichiers compatibles avec l'application Event Log Manager
- Comparer les valeurs des attributs, reçues suite aux requêtes, avec les seuils prédéfinis et ensuite va envoyer des alarmes à l'Alarm Log Manager si des seuils ont été franchis.
- Surveiller des changements éventuels des valeurs des attributs et si ces changements de valeurs dépassent un certain pourcentage de variation, ISM Performance va envoyer de notifications qui pourront être interceptées par les applications intéressées, par exemple ISM Monitor

Le MIB d'ISM Performance

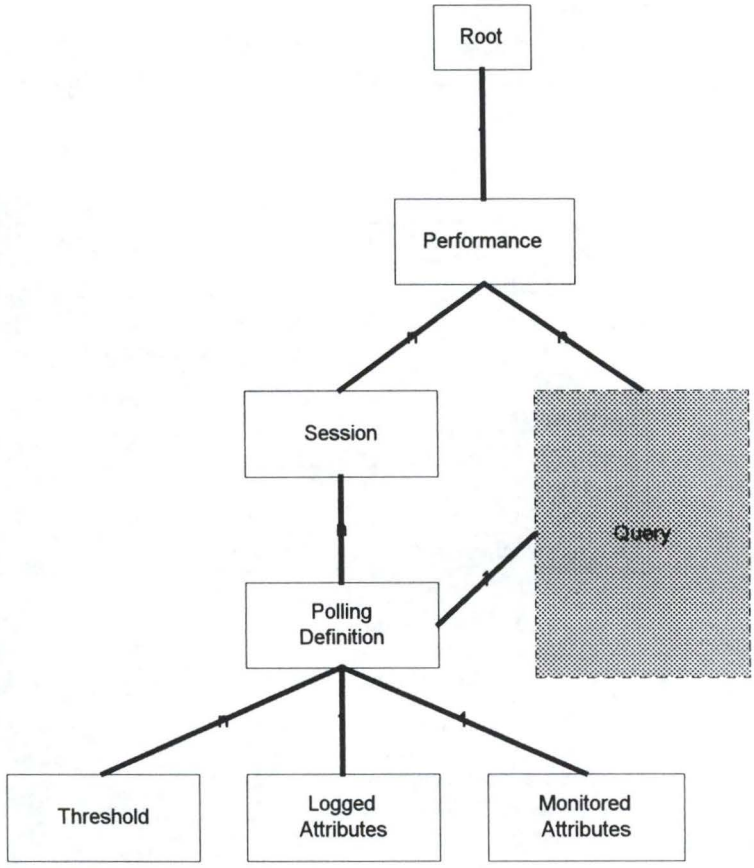
Comme nous l'avons dit auparavant, ISM Performance est un gestionnaire d'objet et il est responsable de la gestion de la MIBlet relative aux objets de performance. Nous allons décrire au cours de cette section la MIBlet de ISM Performance.

La MIBlet de ISM Performance contient deux types de classe d'objets gérés. D'une part les objets spécifiques à ISM Performance et ceux qui sont utilisés par ISM Performance mais qui sont sous la gestion de CMIS QUERY. Au total, il y a six classes d'objets

[38] La MIB de ISM Performance sera détaillée un peu plus loin

gérés par ISM Performance (cfr. Figure 5-2) et nous allons les décrire dans les lignes qui suivent.

Figure 5-2 : La MIB de ISM Performance



La classe d'objet Performance

L'instance de cette classe d'objet est la *rootlet*[39] de la MIBlet ISM Performance. Il ne peut en exister qu'une seule instance et elle est créée automatique lors de l'installation de ISM Performance. Le Tableau 5 représente la classe d'objet `Performance`.

Tableau 5 : La classe d'objet Performance

Nom de l'attribut	*Type de l'attribut*	*Type d'accès*	*Valeur par défaut*
perfId	GraphicString	GET	hostname : « ISM-perf »
maxActiveSessNb	Integer	GET	50

Cette classe d'objet a deux objets inférieurs dans l'arbre de contenance : `Session` et `Query`

[39] Nous en avons donné la définition dans ***La MIB d'ISM/OpenMaster.*** à la page 81

La classe d'objet Session

Cette classe d'objet décrit une action de surveillance prédéfinie.

Le Tableau 6 décrit les différents attributs de la classe d'objet `Session`. Dans ce tableau apparaît un type nouveau, il s'agit du type *administrativeStatus*. C'est un type énuméré qui peut prendre deux valeurs : *active* ou *inactive*. Le type SwitchTime permet quant à lui d'indiquer une heure de départ ou de choisir un départ manuel avec l'option NULL.

Tableau 6 : La classe d'objet `Session`

Nom de l'attribut	*Type de l'attribut*	*Type d'accès*	*Valeur par défaut*
sessionId	GraphicString	GET	None
description	GraphicString	GET-REPLACE	« »
adminStatus	administrativeStatus	GET-REPLACE	inactive
startTime	SwitchTime	GET-REPLACE	NULL
duration	Integer	GET-REPLACE	0
logAdminStatus	administrativeStatus	GET-REPLACE	inactive
maxLogSize	Integer	GET	100Kbytes
notifChangeAdminStatus	administrativeStatus	GET-REPLACE	inactive
changeThreshold	Percentage	GET-REPLACE	10%

L'identifiant de la session (***sessionId***) et la taille du fichier d'enregistrements (***maxLogSize***) sont des attributs qui doivent être fixés au moment de la création. C'est la raison pour laquelle on ne peut exécuter que des primitives GET sur ces deux attributs.

Chaque session peut être lancée ou stoppée indépendamment en changeant l'attribut *adminstatus*.

L'attribut ***startTime*** est défini en vue de développements futurs. Il permettra de lancer le commencement automatique de sessions. Pour l'instant, un session doit être lancée manuellement.

ISM Performance permet également de définir la durée pendant laquelle on va interroger les objets de la MIB au moyen de l'attribut ***duration***. Une session sera inhibée lorsque cette période sera écoulée. A ce moment, une notification sera émise afin de signaler la fin de la session.
Une session qui a une valeur « 0 » pour son attribut *duration* signifie qu'il s'agit d'une session à durée indéterminée et qu'elle devra être arrêtée manuellement.

L'enregistrement de la survenance de changements dans les valeurs d'attributs peut être activé ou inhibé en changeant la valeur de l'attribut ***logAdminStatus.***

L'attribut ***maxLogSize*** indique la taille maximale du fichier d'enregistrements.

La notification des changements concernant les valeurs des attributs peut être activée ou inhibée en changeant la valeur de l'attribut ***notifChangeAdminStatus.***

Afin d'éviter que des notifications soient émises lors de changements mineurs de la valeur des attributs. Un attribut ***changeThreshold*** a été défini afin de ne notifier les changements de valeur uniquement si ils dépassent un pourcentage contenu dans cet attribut.

La classe d'objet session a un seul objet inférieur, il s'agit de la classe `Polling Definition`.

La classe d'objet Polling Definition.

Cette classe d'objet va décrire l'objet sur lequel la requête devra être exécutée. Le Tableau 7 donne les attributs des objets surveillés.

Tableau 7 : La classe d'objet `Polling Definition`

Nom de l'attribut	*Type de l'attribut*	*Type d'accès*	*Valeur par défaut*
pollingDefId	GraphicString	GET	None
pollRate	Integer	GET-REPLACE	None
baseObjectInstance	DistinguishedName	GET-REPLACE	None
query	DistinguishedName	GET-REPLACE	None

L'identifiant de la définition de l'interrogation (***pollingDefId***) doit être défini au moment de la création de l'objet et ne pourra plus être changé par la suite.

L'attribut ***baseObjectInstance*** contient le DN de l'objet sur lequel la requête sera exécutée.

L'attribut ***pollRate*** définit (en secondes) la fréquence des interrogations de l'objet sondé.

On connaît la requête à exécuter grâce au DN de l'objet query défini dans ***query***.

La classe d'objet `Polling Definition` possède trois objets inférieurs, il s'agit des classes d'objets `threshold`, `loggedAttributes` et `monitored Attributes`

La classe d'objet threshold

Cette classe d'objet détermine les attributs[40] à surveiller et définit les valeurs des seuils qui y sont associés. Ces attributs sont détaillés dans le Tableau 8.

Tableau 8 : La classe d'objet `threshold`

Nom de l'attribut	*Type de l'attribut*	*Type d'accès*	*Valeur par défaut*
thresholdAttribute	CompAttrId	GET-REPLACE	None
thresholdAttrType	enumére	GET-REPLACE	None
severityIndThreshold	SeverityIndThreshold	GET-REPLACE	None
eventType	Object Identifier	GET-REPLACE	serviceAlarm
problemType	Int8	GET-REPLACE	thresholdCrossed
problemCode	Int32 ou String32	GET-REPLACE	ISM PERFORMANCE
problemText	Octet String 255	GET-REPLACE	« »

L'attribut ***thresholdAttribute*** doit être crée à la création de l'objet et il ne peut être changé par la suite. Le type de cet attribut définit comme suit[41] :

```
CompAttrId          : :=CHOICE
{
        name        DisplayString,
```

[40] Nous avons pris la liberté de ne pas expliquer tous les types d'attributs qui apparaissent les tableaux qui suivent car il nous a semblé que cela ne relevait pas de la plus grande importance.

[41] Il s'agit d'une définition en ASN1 tirée de CMIP-1

attrId CMIP-1.AttributeId
}

L'attribut ***severityIndThreshold*** définit les différents niveaux de seuil et les sévérités d'alarmes qui y sont associées.

Les trois derniers attributs ***eventType***, ***problemType***, ***problemCode*** et ***ProblemText*** sont utilisés pour les alarmes générées.

La classe d'objet loggedAttributes

Cette classe d'objet décrit les attributs (calculés ou non) qui sont enregistrés dans des fichiers d'enregistrements.

Le Tableau 9 nous montre les différents attributs de cette classe d'objets. Il n'y a aucune valeur par défaut.

Tableau 9 : La classe d'objet `loggedAttributes`

Nom de l'attribut	*Type de l'attribut*	*Type d'accès*	*Valeur par défaut*
loggedAttrId	GraphicString	GET	None
loggedAttrLst	SET OF CompAttrId	GET-REPLACE	None

La classe d'objet monitoredAttributes

Cette classe d'objets définit les attributs (calculés ou non) qui sont surveillés, c'est à dire les attributs pour lesquels une notification de changement de valeur sera générée.

Le Tableau 10 nous montre les différents attributs de cette classe d'objets. Il n'y a aucune valeur par défaut.

Tableau 10 : La classe d'objet `monitoredAttributes`

Nom de l'attribut	*Type de l'attribut*	*Type d'accès*	*Valeur par défaut*
monitoredAttrId	GraphicString	GET	None
monitoredAttrLst	SET OF CompAttrId	GET-REPLACE	None

Le module de traçage de ISM Performance

L'objectif de notre stage était de créer des modules d'affichage de traces lors de l'activité d'ISM Performance. En effet, il n'est pas toujours évident pour l'utilisateur d'ISM Performance de savoir a priori dans quelle intervalle évolue les valeurs d'un attribut. C'est pour cette raison que nous avons choisi d'offrir à l'utilisateur de nouvelles options dans ISM Performance qui lui permettront de se rendre compte des valeurs possibles des attributs interrogés.

Les modules réalisés sont au nombre de deux. :

- Le premier module permet d'afficher sur le standard output[42] ***toutes*** les informations reçues par ISM Performance lors de l'activation d'une session.

[42] Lorsque l'on travaille dans un environnement graphique, le standard output represente une fenêtre texte.

- Le second module permet d'afficher dans une fenêtre graphique les valeurs des attributs ***d'un*** poll d'une session.

La connaissance de ces valeurs permettra à l'utilisateur du service de performances de mettre au point de manière moins approximative qu'auparavant des sessions de surveillance de performances, surtout en ce qui concerne la valeur des seuils de certains attributs.

La trace d'une session

Enoncé du problème.

Le but de ce module consiste à tracer les attributs standards et les attributs calculés d'une session. Les noms et les valeurs respectives de ces attributs seront affichés au niveau du standard output. Nous veillerons également à afficher les caractéristiques de l'attribut, c'est-à-dire que le nom de l'attribut sera précédé de `LOGGED` si il s'agit d'un attribut qui est doit être enregistré dans un journal, de `NOTIFIED` si il s'agit d'un attribut qui est notifié et de `THRESHOLDED` si il s'agit d'un attribut qui est comparé à certains seuils.

Spécifications du problème.

Vu que les l'exécutions d'ISM Performance et de PMSC se font au niveau de deux interpréteurs distincts, nous avons choisi d'utiliser la m-action comme moyen de communication entre ces deux processus. La m-action contiendra un argument qui signalera si il s'agit de l'activation ou de la désactivation de la trace.

Pour indiquer que la session était à tracer, nous avons du choisir entre deux possibilités. D'une part, nous pouvions ajouter un attribut après l'idx (une sorte d'index) de la session pour indiquer que celle-ci était à tracer ou bien nous pouvions indiquer qu'une session était à tracer en indiquant que chacun des polls contenu dans cette session était à tracer. C'est cette dernière possibilité que nous avons choisi[43].

Soit une Session,

Sur-Activation ACTIVATE TRACE SESSION

Si Session Active
Alors
Griser item ACTIVATE TRACE SESSION
∧ Autoriser DEACTIVATE TRACE SESSION
∀ Poll ∈ Session, Mettre à jour TRACE-SESS
Afficher la trace sur le Standard output

Sur-Activation DEACTIVATE TRACE SESSION
Griser item DEACTIVATE TRACE SESSION
∧ Autoriser ACTIVATE TRACE SESSION
∀ Poll ∈ Session, Mettre à jour ***trace-sess***
Ne plus Afficher la trace sur le standard-output

Mettre à jour ***trace-sess***

---

[43] Les éléments qui sont imprimés en petites majuscules (par exemple : ACTIVATE TRACE SESSION) représentent des items de menu ou de boutons dans une fenêtre de dialogue.

Chaque poll a un attribut TRACE-SESS qui indique si le poll est à tracer

Si TRACE-SESS = 0
    Alors le poll n'est pas à tracer
        Mettre à jour structure
Si TRACE-SESS = 1
    Alors le poll doit être tracé
    Mettre à jour structure

La trace d'un poll

Enoncé du problème.

Ce module aura pour but de tracer les attributs d'un poll sujets à des notifications. Tous les changements de valeur seront affichés *et pas seulement les changements de valeur supérieurs à un certain pourcentage*. Il faut pour cela que la session soit active. La trace de ces attributs sera affichée dans une fenêtre X-Window qui contiendra une fenêtre 'ScolledText' afin de permettre à l'utilisateur de reprendre les traces en les sélectionnant et pour lui permettre également de consulter l'historique des traces en bougeant l'ascenseur situé à droite de la fenêtre contenant les traces.

Spécifications du problème.

Sur-Activation de l'item VISUALIZE du menu CONTROL
    Lancer trace de la session
    Mettre à jour TRACE-POLL...
        Afficher Fenêtre
            Insérer les éléments tracés dans la SCROLLED-TEXT
        Sur-Pression STOP
            Mettre à jour TRACE-POLL...
            Désactiver Session
        Sur-Pression QUIT
            Si Session Active
                Alors Mettre à jour TRACE-POLL...
                    Désactiver Session
                    Fermer Fenêtre
            Si Session Inactive
                Alors Fermer Fenêtre

Mettre à jour TRACE-POLL

Chaque poll a un attribut TRACE-POLL indiquant si le poll est à tracer

Si TRACE-POLL = 0
    Alors le poll n'est pas à tracer
    Mettre à jour structure
Si TRACE-POLL = 1
    Alors le poll doit être tracé
    Mettre à jour structure

Conclusion

ISM Performance a fait ses preuves au sein d'ISM/OpenMaster. Il est encore l'objet pour l'instant de quelques modifications mineures qui n'ont pas de grandes influences sur son fonctionnement global.

Vu l'importance de la gestion des performances au sein d'un environnement distribué, ISM Performance rendra toujours service aux administrateurs ou aux utilisateurs d'ISM/OpenMaster.

Néanmoins, il dispose actuellement d'un certain potentiel qui pourrait être considérablement accru en implémentant de manière plus fine les standards de l'ISO, c'est ce dont il est question dans le chapitre qui suit.

Chapitre

7

Implémentation des standards ISO de performance au sein d'ISM Performance.

Au cours de cette partie, nous allons analyser dans quelle mesure la version actuelle d'ISM Performance respecte les standards de l'ISO relatifs aux mesures de performances. Nous nous baserons principalement sur les standards [ISO10164-11] et [ISO10165-4].

Avant d'entamer l'analyse proprement dite, nous présenterons les éléments essentiels qui doivent apparaître dans tout progiciel de gestion de performances. Ensuite, nous envisagerons les modifications qu'il faudrait apporter à ISM Performance afin qu'il devienne en quelque sorte un outil de gestion de performance '*modèle*'. Nous terminerons ce chapitre par des perspectives futures sur la gestion des performances.

Les éléments caractéristiques d'un progiciel de gestion de performances.

Le but d'un progiciel de gestion de performances est de donner les moyens nécessaires à l'administrateur lui permettant d'optimiser au maximum l'utilisation de toutes les ressources du centre informatique sur lequel il travaille. Cet objectif doit être atteint dans les moindres coûts.

Un outil de gestion de performances doit apporter une vision globale du système. Pour être efficace, il doit répondre principalement aux critères suivants :

- Avoir accès aux bases de données en utilisant le minimum de ressources du système ; en particulier, il doit pouvoir sélectionner les accès à celles-ci. Il faut noter qu'il fonde la majeure partie de son travail sur les bases de données. Par conséquent, celles-ci doivent être fiables (ceci est essentiel).
  En effet, chaque centre informatique a une population d'utilisateurs différente et des tâches à accomplir qui lui sont propres. De ce fait, ces éléments doivent être pris en compte par le gestionnaire de performances.

De plus, les accès aux bases de données effectués lui permettent de faire des calculs de performances liées directement au nombre d'utilisateurs et au nombre et à la nature des applications utilisées.

- **Traitement de la remontée des alarmes** de façon continue en jouant un rôle de surveillant sur le système. En effet, la détection des divers problèmes qui peuvent survenir est indispensable au gestionnaire de performances et chacun de ces problèmes doit être répertorié dans une base de donnée pour être étudié immédiatement ou ultérieurement.
- **Possibilité de faire des analyses sur des intervalles de temps prédéfinis par l'administrateur**. En effet, chaque centre informatique a des horaires propres pour l'exécution des travaux qu'il doit effectuer. De ce fait, il doit être possible de sélectionner des intervalles de temps où les conditions sont défavorables (exemple : période de surcharge, instant où se déclenche une alerte,....) pour bien voir ce qui se passe.
- **Analyses en temps réel** : il doit pouvoir effectuer des analyses en temps réel pour connaître à tout instant l'activité du système et l'évolution de celle-ci. Ceci est fondamental pour diagnostiquer les problèmes de performances.

- **Convivialité du support graphique** où l'on peut observer les analyses effectuées en temps réel, et l'historique complet (mesures enregistrées lors des analyses). Ce support graphique permet à l'administrateur de se rendre compte rapidement des origines d'un problème de performance. C'est un outil de diagnostic.
- Plusieurs types d'analyses doivent être possibles tels que :
  - ⇒ analyses par utilisateur, groupe d'utilisateurs ou applications
  - ⇒ corrélation d'informations (association de courbes de domaines différents)
  - ⇒ comparaison des serveurs et stations ...
- **Détection des interactions entre les différents composants systèmes**. L'ignorance de ces paramètres nuit fortement à la résolution d'un problème de performances.
- **Détermination de manière spécifique des ressources utilisées par chaque processus**. Cela est très utile lors d'une alerte, l'administrateur se rend compte très rapidement de l'origine du problème puisqu'il sait exactement l'environnement de travail du processus qui a provoqué cette alerte. (Cela reprend en partie les analyses faites par l'application évoquée précédemment).
- **Historisation** : il doit faire des sauvegardes des données titrées des analyses qui lui serviront lors d'autres analyses ultérieures.
- Sachant que dans les entreprises, un système UNIX peut tourner sans interruption 24 heures par jour, le progiciel de gestion de performances doit pouvoir **faire ses analyses sans interrompre l'activité du système**.

En plus des points évoqués précédemment, il doit fournir des rapports (fait de façon automatique ou non) sur ces analyses qui soient les plus clairs possibles pour l'administrateur.

Les caractéristiques qui ont été évoquées ci-dessus donnent une vision de la gestion de performances en terme d'analyses effectuées au ***passé*** et au ***présent***. Aussi, certains produits amplifient cette approche en intégrant le ***futur*** du système, On parlera alors de ***Capacity Planning***.

On se préoccupe alors de l'utilisation des ressources futures en terme de planification. C'est l'objet du paragraphe suivant[44].

La planification permet d'avoir une vision anticipée des besoins dus aux accroissements des charge ou à la mise en production de nouvelles applications dans un centre informatique.

Les deux objectifs principaux de la planification sont :

- de garantir à l'utilisateur un service global de qualité avec un capital minimum
- de maîtriser les coûts

La planification se justifie parfaitement pour un système distribué à grande échelle.

La seule solution acceptable permettant de trouver un compromis est de connaître exactement quels sont les besoins présents et futurs. Pour parvenir à cela, les entreprises qui se trouvent confrontées à ce problème font appel à des outils qui possèdent cette fonction de planification.

Donnons alors les principales caractéristiques de cet outil qui offre la possibilité de faire de la gestion de performances prévisionnelle.

---

[44] Les standards ne prennent pas en considération cet aspect de la gestion des performances mais il nous semble néanmoins utile le citer pour mémoire.

Il doit être capable de déterminer avec exactitude les ressources nécessaires pour un nombre d'utilisateurs donné. Il pourra ainsi prévoir les effets sur le système provoqués part une augmentation du nombre utilisateurs.

- Il doit offrir une notion de flexibilité dans le sens où il est voué à être confronté à des changements fréquents d'utilisateurs et d'applications.
- Il doit pouvoir consulter et utiliser l'historique du gestionnaire de performances pour déceler les tendances relatives à chaque type de processus déjà présent. Ces données lui sont utiles pour sa 'planification'.

Cette application de planification permet donc d'avoir une identification anticipée des besoins afin d'assister les responsables ayant à prendre des options financières, stratégiques ou techniques engageant la vie de l'entreprise autant en interne qu'en externe.

Analyse d'ISM Performance

Avant de commencer l'analyse d'ISM Performance, nous tenons à avertir le lecteur qu'il ne s'agit que d'une étude ... et qu'aucune implémentation ne sera proposée dans les réflexions qui suivent.

Le standard qui fait l'objet de notre curiosité a été présenté au chapitre 4 (Les Objets Métriques) . Si nous consultons les outils de gestion de performances couramment utilisés, seul BULL supporte CMIP sur sa plate-forme ISM/OpenMaster[45].

Le statut d'ISM Performance

Le lecteur pourrait s'interroger sur la raison de notre étude car la version actuelle d'ISM Performance est fonctionnelle et satisfait les besoins des clients ?

Nous motiverons cette réflexion en affirmant que les progiciels, tout comme les êtres humains ne sont jamais *parfaits*, et il y a donc toujours moyen de les améliorer. C'est ce que nous allons essayer de montrer dans les pages qui suivent.

Le service de performances d'ISM/OpenMaster s'inspire du standard [ISO10164-11]. D'après les informations que nous avons pu rassembler durant notre stage, les concepteurs du service de performance n'ont retenu du standard que quelques leçons.
En effet, lors d'un examen plus précis du service de performances, on remarque que certaines notions définies par la norme se retrouvent au sein d'ISM Performance.

La première étape de cette réflexion sera d'identifier ces notions. Nous tenons à signaler que la présence des notions au sein du service de performances n'implique pas qu'il y a respect du standard... il s'agit simplement de l'adéquation du service de performances aux concepts généraux devant figurer dans tout programme de gestion de performance.

La norme [ISO10164-11] se base sur ***un*** objet principal qui est le ***scanner***. Ensuite la norme définit une panoplie de métriques qui utiliseront les valeurs récoltées par cet objet scanner.

Le point de vue adopté par ISM Performance est différent. ISM Performance va interagir avec d'autres composantes[46] de la plate-forme ISM/OpenMaster afin de mener à bien son activité.

---

[45] [Jander,1994]

Contrairement à l'objet scanner dans le standard, le service de performance n'interroge pas les objets lui même. Cette interrogation est effectuée via le CMIS Query qui est une application permettant d'interroger des objets de la MIB via des requêtes CMIS.

La vue d'ISM Performance qu'ont les autres applications, comme ISM Performance Control, est celle d'un gestionnaire d'objet et il est donc accessible via es requêtes CMIS.

Enumérons les notions qui sont à la fois présentes dans ISM Performance et dans la norme :

- L'interrogation des objets de la MIB (par ISM Query.)
- La définition d'intervalles d'interrogation
- La notion de métrique c'est-à-dire le calcul d'un attribut à partir de plusieurs autres valeur d'attributs observées.
- Un état d'administration pour activer, suspendre ou stopper un session de mesure de performances.
- Le principe de seuil (threshold) tel qu'il est défini dans [ISO10165-2]
- Le concept de jauge
- Les journaux d'alarmes et de notifications

Nous constatons que les éléments se trouvant dans le service de performances sont en quelques sortes le minimum requis assurer un processus de gestion de performances.

Une exploitation de statistiques off-line sur base de fichiers de log est intégrée dans d'autres composantes d'ISM/OpenMaster.

C'est donc la voie d'un produit minimum mais fonctionnel, qui été choisie par les concepteur du service de performances d'ISM/OpenMaster, par rapport à la possibilité de créer un produit plus complet mais qui aurait peut être requis une durée de mise en œuvre nettement plus importante[47].
Ce genre de réaction rentre dans la logique de lois du marché. Si BULL ne propose pas des produits présents chez le concurrents ... cela aura comme conséquence la perte de parts de marchés.

La logique suivie par ISM/OpenMaster est nécessaire mais il serait intéressant de penser à des évolutions possibles du produit, notamment un respect du standard qui demanderait à l'équipe en charge de la maintenance du service de performances l'apport de modifications majeures.

Dans la version actuelle de d'ISM Performance Service, aucun concept[48] concernant la MIB définie pour les performances n'est *implémenté*. La raison est simple ! Le principe

---

[46] L'Alarm Log Manager, Event Log Manager,CMIS Query et ISM Performance Service Control sont les principaux.

[47] En suivant cette voie, Bull réagit comme l'IAB, à qui on doit SNMP. C'est-à-dire que l'on préfère avoir un produit simple et fonctionnel endéans un temps assez court plutôt que de concocter un programme complet qui impliquera un délai de réalisation important.

[48] Il s'agit des objets métriques

des objets "scanners" veut que ce soit *l'agent* qui les implémente. Hors, il n'existe pas d'agent couramment disponible qui implémente ces objets au sein d'ISM/OpenMaster.

Il faut donc faire des interrogations périodiques pour pouvoir suivre l'évolution d'un indicateur. Ce qui a pour conséquence inévitable d'engendrer un trafic non négligeable sur le réseau.

Nous venons d'exposer le principal inconvénient de la version actuelle d'ISM Performance. L'idée de notre réflexion est donc de s'orienter vers un agent intelligent pour implémenter la norme de l'ISO.

Vers un agent intelligent ...

Pour faire le lien entre la MIB performance définie par l'ISO et le service de performance d'ISM/OpenMaster, nous distinguons deux approches :

La première approche considère le service de performance comme un agent, qui implémente les objets métriques correspondant aux sessions définies actuellement. Cette approche obligerait une refonte complète du progiciel car il nous faudrait repenser tout le service de performance pour qu'il puisse respecter le standard.
L'interface avec les composantes d'ISM/OpenMaster serait donc changée, nous entendons par là, un changement de la vue offerte aux autres applications comme ISM Monitor, ISM Performance Service Control, etc. Opter pour cette approche passe par une implémentation 'pure et dure' de la MIB définie dans le standard.

La seconde approche est d'étendre le service de performance afin qu'il soit capable de prendre en compte cette MIB performance présente chez un agent. Ceci est intéressant car alors l'agent est capable de notifier périodiquement (ou d'envoyer sur requête une suite de valeurs collectées) au lieu d'être interrogé sans arrêt.
Ici, ce que nous voulons changer, c'est l'interface entre ISM Performance et les objets à scruter. On considère que des agents au sein du système distribué implémentent les objets métriques contenus dans le standard et l'objectif d'ISM Performance serait de traiter les messages notifiés par les agents répartis dans le réseau. Une illustration de cette approche est donnée à la Figure 7-1.

Ce concept d'agent est de manière générale repris par la majorité programmes de gestion de performances. Mais cette approche n'est pas sans risques car les agents devront, à terme, devenir de plus en plus intelligent en raison de la quantité d'informations[49] qu'ils auront à traiter.

Rappelons que les deux techniques utilisées pour remonter les informations collectées par les agents au niveau du gestionnaire sont le *polling* et *l'event-reporting*.
Brièvement le *polling* est une interaction continue via des demandes/réponses entre le manager et l'agent.
Dans *l'event-reporting*, l'initiative appartient à l'agent, et c'est cet agent qui va envoyer des informations ou des rapports. Le manager est 'à l'écoute' de ce que pourrait lui envoyer l'agent.

---

[49] En effet, vu l'intérêt grandissant apporté à la gestion des performances, les gestionnaires voudront obtenir les informations sur pratiquement toutes les composantes de leur système.

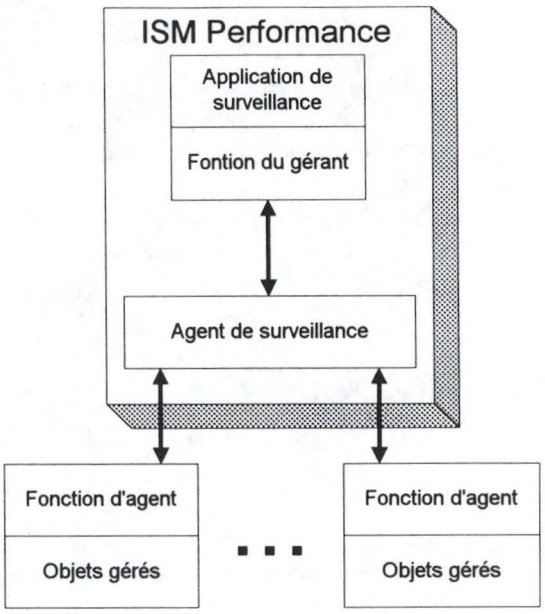


Figure 7-1 : Une modélisation de surveillance de performances

Les deux techniques se défendent, l'OSI ne privilégie pas une technique par rapport à l'autre mais elle essaye de trouver un compromis entre les deux solutions[50].

Le choix d'une approche

Si nous étions aux phases préliminaires de conception d'ISM Performance, nous opterions pour la première approche, quitte à impliquer plus de personnes dans la mise en œuvre du programme[51].

Mais comme un produit est déjà existant, il serait plus logique de penser à faire une étude de l'existant et d'envisager des améliorations progressives du produit. C'est donc cette démarche que nous allons suivre dans la suite de ce chapitre.

Une implémentation en douceur ...

Nous allons proposer maintenant une étude d'implémentation pas à pas. Nous allons essayer de proposer les changements successifs qu'il faudrait apporter à la MIB du service de performances ainsi que les différentes étapes qui nous conduiront vers une version du service de performances respectant le standard.

Etape 1

Nous avons d'une part un programme de mesures de performances qui dispose de sa MIB et il y a d'autre part un standard qui propose une MIB différente.

Il nous semble primordial que le progiciel de mesure de performances dispose de sa propre MIB *et* qu'il puisse gérer tous les classes d'objets contenues dans sa MIB. C'est donc ici qu'interviendra la première modification qu'il faudra apporter à ISM Perfor-

---

[50] SNMP quant à lui apporte un intérêt particulier au polling.

[51] Il faut savoir qu'ISM Performance représente un an et demi de travail pour *une seule* personne.

mance. Comme nous l'avions expliqué précédemment[52] la MIB d'ISM/OpenMaster contient des classes d'objets qui lui sont spécifiques et elle contient une classe d'objet qui est utilisée par ISM Performance mais qui est gérée par CMIS Query.
C'est de cette dernière classe d'objet dont il est question.

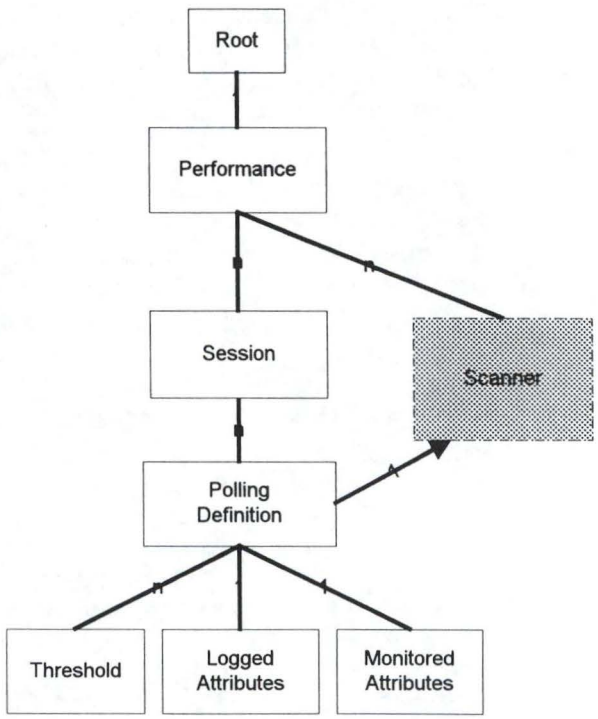
Vu que l'interrogation des objets représente la pilier de toute la gestion des performances, nous proposons de remplacer ce module CMIS Query, qui n'est pas géré par ISM Performance, par un nouvelle classe d'objet conforme à la classe d'objet 'scanner' définie dans [ISO10164-11].

Pourquoi ?

De cette manière, ISM Performance disposera non seulement de sa propre MIB mais il sera également apte à gérer tous les classes d'objets contenues dans sa MIB. Il serait ainsi totalement indépendant. Une fois cette classe d'objet 'scanner' implémentée, il ne reste plus qu'à la mettre en œuvre sur les différents agents.

De plus, c'est sur cette classe d'objet scanner que repose toute la norme. Il serait donc ridicule d'implémenter des fragments de la norme et de pouvoir les exploiter en raison de l'absence de la classe d'objet principale...

Figure 7-2 :Une première version de la MIB modifiée



Sa fonction ?

Cette classe d'objet scanner a le comportement suivant :

Elle permet de collecter les valeurs de attributs des objets gérés et de produire des synthèses à partir des informations qui ont été extraites.

Les valeurs des attributs sont retirés lors d'une interrogation qui est lancée périodiquement à la fin de chaque intervalle d'interrogation.

---

[52] Voyez à ce propos **La MIB d'ISM/OpenMaster.**à la page81

Les effets de bords

Pratiquement aucun, excepté que le scanner contient un attribut *granularity period* (intervalle d'interrogation) qui lui permettra d'effectuer toutes ses interrogations de manière autonome.

Nous ne passons donc plus par l'infrastructure de communication pour effectuer les interrogations. Tout est réalisé de manière locale.

Etape 2

Maintenant que nous avons un instrument adéquat pour interroger les objets, nous allons standardiser le mécanisme de détermination de l'objet à analyser ainsi que la référence aux attributs de cet objet qui sont à examiner.

Il s'agit d'implémenter la classe d'objet *monitor metric*. Cette classe d'objet va avoir pour fonction de surveiller les valeurs d'un attribut au sein d'un objet à des intervalles spécifiés par la granularity period. Cette nouvelle classe d'objet contiendra également un attribut avec la valeur de la jauge dérivée.

Sans l'apport de paquets conditionnels, la valeur de la jauge dérivée contiendra la valeur de la jauge lors de l'interrogation précédente Par contre, on peut greffer à cette classe d'objet toute une série de paquets conditionnels qui permettront de calculer la valeur de la jauge dans tous les cas de figure.

Etape 3 et suivantes ...

A ce niveau de l'implémentation, nous disposons de l'essentiel pour un programme de mesures de performances. Au cours des autres étapes, nous ne ferons qu'ajouter des nouvelles classes d'objets contenant des informations supplémentaires relatives à la jauge dérivée.

Il s'agira d'informations sur la valeur moyenne, la variance, etc.

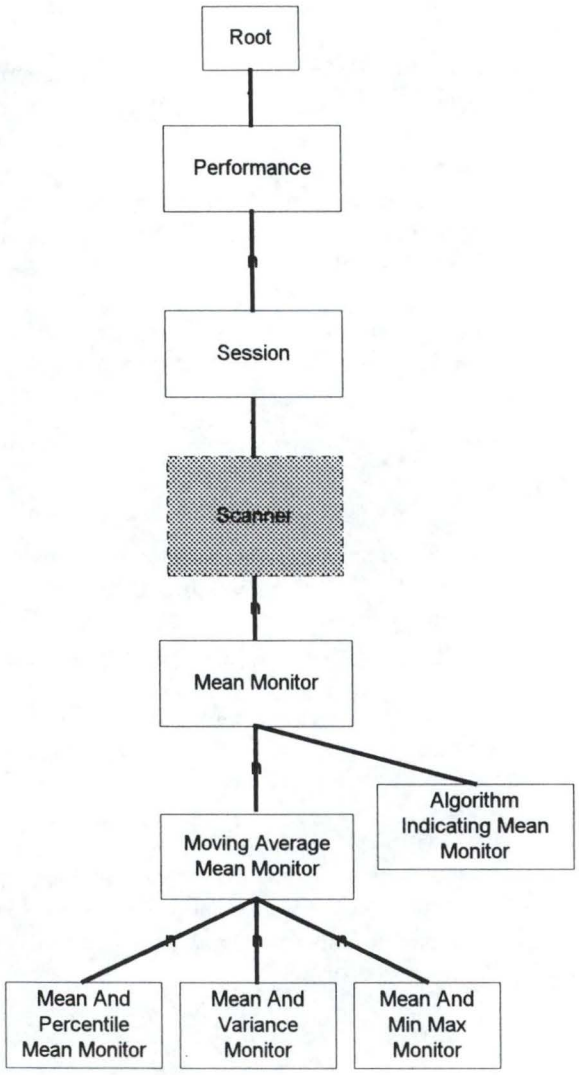
Toutes ces classes d'objets ont été présentées dans le chapitre 4 sur les objets métriques.

Ces classes d'objets ne contiennent que des 'calculs' et si certains objets métriques proposés dans la norme ne sont d'aucun intérêt pour ISM Performance, nous pouvons ne pas les implémenter.

Mais la plupart des objets métriques proposés par cette norme serviront à réaliser des statistiques sur les ressources et donc il est toujours utile d'avoir de telles données sous la main.

A terme, c'est-à-dire lorsque toutes les implémentations auront été réalisées, la MIB du service de performance sera conforme à la MIB proposée à la Figure7-3.

Figure7-3 : Représentation finale de la MIB d'ISM Performance



Agent vs non-agent

Les deux approches ont des avantages et des inconvénients. Nous avons déjà exposé quelques uns de ces avantages et inconvénients mais il ne nous semble pas inutile de les synthétiser.

L'approche 'non-agent'.

Comme nous l'avons déjà signalé, cette approche engendre un trafic supplémentaire sur le réseau, ce qui peut à l'échelle de réseaux importants entraîner des temps de réponses accrus, et donc une diminution des performances.

Par contre, si on considère ISM Performance comme un service, il garde une indépendance face aux protocoles sous-jacents, ce qui, avouons-le, représente un avantage de taille. De cette manière, le service de performance peut accéder n'importe quel objet (SNMP, CMIP, DSAC, X.25, SNA, ...) sans devoir apporter des modifications aux composantes d'ISM/OpenMaster.

Une telle approche demande moins de codage et centralise toutes les performances en un seul point. Une éventuelle coordination de certaines interrogations peut s'avérer plus difficile à exécuter dans des circonstances qui ne sont pas idéales.

L'approche 'agent'.

Nous pourrions dire que ce qui est un avantage pour l'approche 'non-agent' est un inconvénient pour l'approche 'agent' et inversement.

Néanmoins, les avantages de l'approche 'agent' se font ressentir au niveau du fonctionnement quotidien. Tenir compte des différents protocoles de communication, nous ne devons le faire que lors de la réalisation de l'agent et donc cela ne pose qu'un problème conceptuel.

Par contre, une surcharge du réseau se fera ressentir durant toute l'exploitation du système d'information et peut dépendre d'autres paramètres.

Evolution futures

Toute ce que nous venons de présenter, ne représente qu'une réflexion faite sur base de documents, de renseignements pris chez BULL et de choix personnels. Comme je l'avais souligné au début de ce chapitre il n'était nullement question de présenter une version fonctionnelle d'ISM Performance vérifiant le standard.

Nous pourrions donc considérer cette réflexion comme phase préliminaire du développement d'un 'nouveau' service de performance au sein d'ISM/OpenMaster. Nous pourrions envisager de réaliser une version définitive du programme , en accord avec le standard.

Nous pourrions ainsi l'utiliser et comparer 'réellement' les différences de performances par rapport à la version actuelle. Avec cette version *'orientée Agent'* d'ISM Performance, la plate-forme ISM/OpenMaster pourrait offrir un service semblable à celui offert par ses principaux concurrents.

Conclusion

Comme nous l'avons vu tout au long de ce mémoire, la gestion est un domaine très vaste et sur lequel nous n'avons pas encore une totale emprise. Il suffit pour s'en convaincre de voir le nombre d'outils qui sont actuellement développés pour faciliter la gestion des systèmes.

Nous nous sommes focalisés sur la gestion des performances et nous avons constaté l'intérêt particulier qu'il fallait y accorder. Ce n'est pas tellement le concept de gestion des performances qui est neuf, car nous avions déjà recours à ce type de gestion dans les mainframes, mais c'est plutôt la distribution de cette gestion.

De plus nous pouvons remarquer, que bien souvent on assimile la gestion des performances aux réseaux. Nous avons vu que les réseaux représentent quelque chose d'essentiel mais ils ne représentent pas la composante principale des systèmes distribués. Il faut considérer le système comme un tout ! D'où l'avènement d'outils de gestion intégrée de systèmes tel que celui proposé par BULL.

BULL propose une des seules architectures dont le fonctionnement repose sur des services CMIS. C'est la raison pour laquelle, nous nous sommes fixé l'objectif de présenter tout ce qui se rapportait à la fois à la gestion OSI et à la gestion des performances.

Mais il ne faut pas se voiler la face et force est de constater que le leader du marché est actuellement SNMP et que CMIP a tendance à battre de l'aile. La communauté informatique est partagée en ce qui concerne le sort de l'OSI. Certains voient une issue favorable, récompensant ainsi l'OSI des efforts qu'elle a du fournir. D'autres misent tout sur SNMP.

Maintenant, nous savons tous que SNMP se veut être très simple mais il est également très orienté vers les réseaux. Ce qui à terme aurait pu lui causer quelques soucis face à l'OSI qui offre un outil de gestion intégrée. Malheureusement pour l'OSI, un nouveau modèle architectural est né au sein du CMG. Il s'agit de l'architecture UMA.

UMA ,qui repose sur architecture simple, est déjà l'objet de nombreuses convoitises par les développeurs d'applications. Nous retiendrons de l'UMA qu'elle est très orientée vers l'administration des systèmes. Nous pourrions dès lors imaginer un mariage entre SNMP et UMA, l'un orienté gestion-réseau et l'autre orienté gestion-système, ce qui à partir de deux choses qui se voulaient simples au départ donnerait un outil *complet* de gestion intégrée. Si ce mariage a lieu, les chances de voir l'OSI s'affirmer un jour sur le marché seraient très réduites.

Cette guerre entre les organismes de normalisation est loin d'être terminée et nous nous serions tentés de dire 'Heureusement !'. Car quand nous voyons comment l'OSI a impressionné la communauté informatique avec sa décomposition en couches, nous retiendrons de tout ceci qu'il y a toujours des leçons à tirer et que ce n'est qu'en comparant deux choses que nous pouvons dire que l'une est meilleure que l'autre !

Liste des acronymes

ACSE	Association-control-service element
AE	Application entity
AEP	Administrative Exchange Protocol
AI	Agent Integrator (intégrateur d'agent)
AI	Agents Intégrateurs
API	Application Programmatic Interfaces
ASE	Application-service element
CCITT	International Consultative Committee on Telegraphy and Telephony
CMG	Computer Measurement Group
CMIP	Common management information protocol
CMIS	Common management information service
CMISE	Common management information service element
CMOT	CMIP Over TCP/IP
CORBA	Common Object Reqeuest Broker Architecture
DCI	Data Capture Interface
DCL	Data Capture Layer
DN	Distinguished name
DSL	Data Services Layer
EWMA	Exponentially Weighted Moving Average
GMP	GCOS Management Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet engineering task force
ISM	Integrated System Management
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union-Telecommunication Standardization Sector
MAL	Measurement Application Layer

MAP	Measurement Application Program
MCL	Measurement Control Layer
MIB	Management Information Base
MIT	Management information tree
MLI	Measurement Layer Interface
OMG	Object Management Group
OSI	Open systems interconnection
OSI/NMForum	OSI/Network Management Forum
OSIE	OSI Environment
PDU	Protocol data unit
PMS	Performance Monitoring Service
PMSC	Performance Monitoring Service Control
PMWG	Performance Management Working Group
RDN	Relative distinguished name
ROM	Root Object Manager
ROM	Root Object Manager
ROSE	Remote-operation-service element
SAT	SNMP Agent Toolkit
SMAE	Systems-management application entity
SMASE	Systems-management application-service element
SMF	System management function
SMI	Supra Management Interface
SNMP	Simple network-management protocol
SQL	Simple Query Language
TCP/IP	Transfer Control Protocol/Internet Protocol
UMA	Universal Measurement Architecture
XMP	X/Open Management Protocol

Bibliographie

[Deghor,1991] Deghorain Hugues, *L'administration des systèmes distribués : Approche de la gestion OSI*, Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, Namur, 1992

[EIS-PMS,1994] Doummar Raphaël, *External Interface Specification : ISM3.1 Performance Monitoring Service & Control Application*, Document interne Bull, révisé le 25.11.1994

[Fleisch,1994] Fleischmann Albert, *Distributed Systems : Software Design and Implementation,* Springer-Verlag,1994

[GHLDD,1994] Emsley Ian, *Global High Level Design Document,* Document interne Bull, draft du 01.04.1994

[Herman,1994] Herman James, *Service Management : A Focused Goal for Integrated Network and Systems Management*, Data Communications International, November 1994

[ISO10040] ISO/IEC 10040, *Information technology - Open Systems Interconnection - Systems Management Overview*, International Standard Organization

[ISO10164-1] ISO/IEC 10164-1, *Information technology - Open Systems Interconnection - Systems Management : Object Management Function*, International Standard Organization

[ISO10164-10] ISO/IEC 10164-10, *Information technology - Open Systems Interconnection - Systems Management - Part 10 : Accounting meter function*, International Standard Organization

[ISO10164-11] ISO/IEC 10164-11, *Information technology - Open Systems Interconnection - Systems Management - Part 11 : Metric objects and attributes*, International Standard Organization

[ISO10164-12] ISO/IEC 10164-12, *Information technology - Open Systems Interconnection - Systems Management - Part 12 : Test management function*, International Standard Organization

[ISO10164-13] ISO/IEC 10164-13, *Information technology - Open Systems Interconnection - Systems Management - Part 13 : Measurement summarization function*, International Standard Organization

[ISO10164-2] ISO/IEC 10164-2, *Information technology - Open Systems Interconnection - Systems Management - Part 2 : State Management Function*, International Standard Organization

[ISO10164-3] ISO/IEC 10164-3, *Information technology - Open Systems Interconnection - Systems Management - Part 3 : Attributes for representing relationships*, International Standard Organization

[ISO10164-4] ISO/IEC 10164-4, *Information technology - Open Systems Interconnection - Systems Management - Part 4 : Alarm reporting function*, International Standard Organization

[ISO10164-5] ISO/IEC 10164-5, *Information technology - Open Systems Interconnection - Systems Management - Part 5 : Event reporting management function*, International Standard Organization

[ISO10164-6] ISO/IEC 10164-6, *Information technology - Open Systems Interconnection - Systems Management - Part 6 : Log control function*, International Standard Organization

[ISO10164-7] ISO/IEC 10164-7, *Information technology - Open Systems Interconnection - Systems Management - Part 7 : Security alarm reporting function*, International Standard Organization

[ISO10164-8] ISO/IEC 10164-8, *Information technology - Open Systems Interconnection - Systems Management - Part 8 : Security audit trail function*, International Standard Organization

[ISO10164-9] ISO/IEC 10164-9, *Information technology - Open Systems Interconnection - Systems Management - Part 9 : Objects and attributes for access control*, International Standard Organization

[ISO10165-1] ISO/IEC 10165-1, *Information technology - Open Systems Interconnection - Structure of management information - Part 1 : Management Information Model*, International Standard Organization

[ISO10165-2] ISO/IEC 10165-2, *Information technology - Open Systems Interconnection - Structure of management information - Part 2 : Definition of management information*, International Standard Organization

[ISO10165-4] ISO/IEC 10165-4, *Information technology - Open Systems Interconnection - Structure of management information - Part 4 : Guidelines for the definition of managed objects*, International Standard Organization

[ISO7498] ISO/CEI 7498, *Systèmes de traitement de l'information - Interconnexion de systèmes ouverts - Modèle de référence de base*, International Standard Organization

[ISO7498-4] ISO/CEI 7498-4, *Systèmes de traitement de l'information - Interconnexion de systèmes ouverts - Modèle de référence de base - Partie 4 : Cadre Général de Gestion*, International Standard Organization

[ISO9595] ISO/IEC 9595, *Information technology - Open Systems Interconnection - Common management information service definition*, International Standard Organization

[ISO9596] ISO/IEC 9596, *Information technology - Open Systems Interconnection - Common Management Informartion Protocol* , International Standard Organization

[ISO9596-1] ISO/IEC 9596-1, *Information technology - Open Systems Interconnection - Common Management Informartion Protocol - Part 1 : Specification*, International Standard Organization

[Jander,1994] Jander Mary, *Performance Management Keeps Check on Client-Server Systems,* Data Communications International, May 1994

[Jander-2,1994] Jander Mary, *Management Frameworks : Moving toward a Unified view of distributed Networks*, Data Communications International, February 1994

[Libert,1996] Libert Pascal, *Generix : un agent générique pour la gestion des systèmes distribués*, Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, Namur, 1996

[Mullen,1989] Mullender Sape, *Distributed Systems*, ACM Press - Addison Wesley, 1989

[Nosbus,1995] Nosbusch Marc, *Développement d'un Agent de Mesures de Performances pour la Gestion des Systèmes Distribués*, Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, Namur, 1995

[OSI,1993] *OSI : The Open Systems Networking Standard*, Computer Technology Research Corp, 1993.

[PMS,1995] *Integrated System Management - Guide de l'utilisateur : Surveillance des performances,* Manuel fourni avec le logiciel, Juillet 1995

[Sloman,1987] Sloman Morris, Kramer Jeff, *Distributed Systems and Computer Networks*, Prentice Hall,1987

[Sloman,1994] Sloman Morris, *Network and distributed systems management,* Addison-Wesley, 1994

[Stallings,1993] Stallings William, *SNMP, SNMPv2 and CMIP The practical Guide to Network-Management Standards*, Addison-Wesley, 1993

[Svobod,1987] Svobodova Liba, *The role of OSI in Distributed Computing*, The 7th International Conference on Distributed Computing Systems, September 1987

[Terplan,1992] Terplan Kornel, *Communication Network Management (Second Edition)*, Prentice Hall, 1992

[Umar,1993] Umar Amdjad, *Distributed Computing and Client-Server Systems,* Prentice Hall, 1993

[Xopen,1995-1] X/Open Guide, *Systems Management : Universal Measurement Architecture Guide,* X/Open Company Limited, March 1995

[Xopen,1995-2] X/Open Preliminary Specification, *Systems Management : UMA Data Pool Definition (DPS),* X/Open Company Limited, March 1995

[Xopen,1995-3] X/Open Preliminary Specification, *Systems Management : UMA Measurement Layer Interface (MLI),* X/Open Company Limited, March 1995

[Xopen,1995-4] X/Open Preliminary Specification, *Systems Management : UMA Data Capture Interface (DCI),* X/Open Company Limited, March 1995

Annexes

Annexe 1 : La gestion OSI

Utilisation des services CMISE par les fonctions de gestion-système

Nous remarquons dans ce tableau que certaines fonctions de gestion-système n'utilisent aucun service CMISE. Cela est tout simplement dû au fait que pour de nombreuses opérations, la fonction de gestion d'objets passe comme intermédiaire.

	M-EVENT-REPORT	M-GET	M-SET	M-ACTION	M-CREATE	M-DELETE
Gestion d'objet	✓	✓	✓	✓	✓	✓
Gestion d'état	✓					
Gestion des relations	✓					
Rapports d'alarmes	✓					
Gestion des rapports d'événements						
Contrôle des journaux						
Rapport des alarmes de sécurité	✓					
audit-trail	✓					
Contrôle d'accès						
Métriques de comptabilité	✓			✓		
Surveillance de la charge						
Gestion des tests	✓			✓		
Résumé	✓			✓		

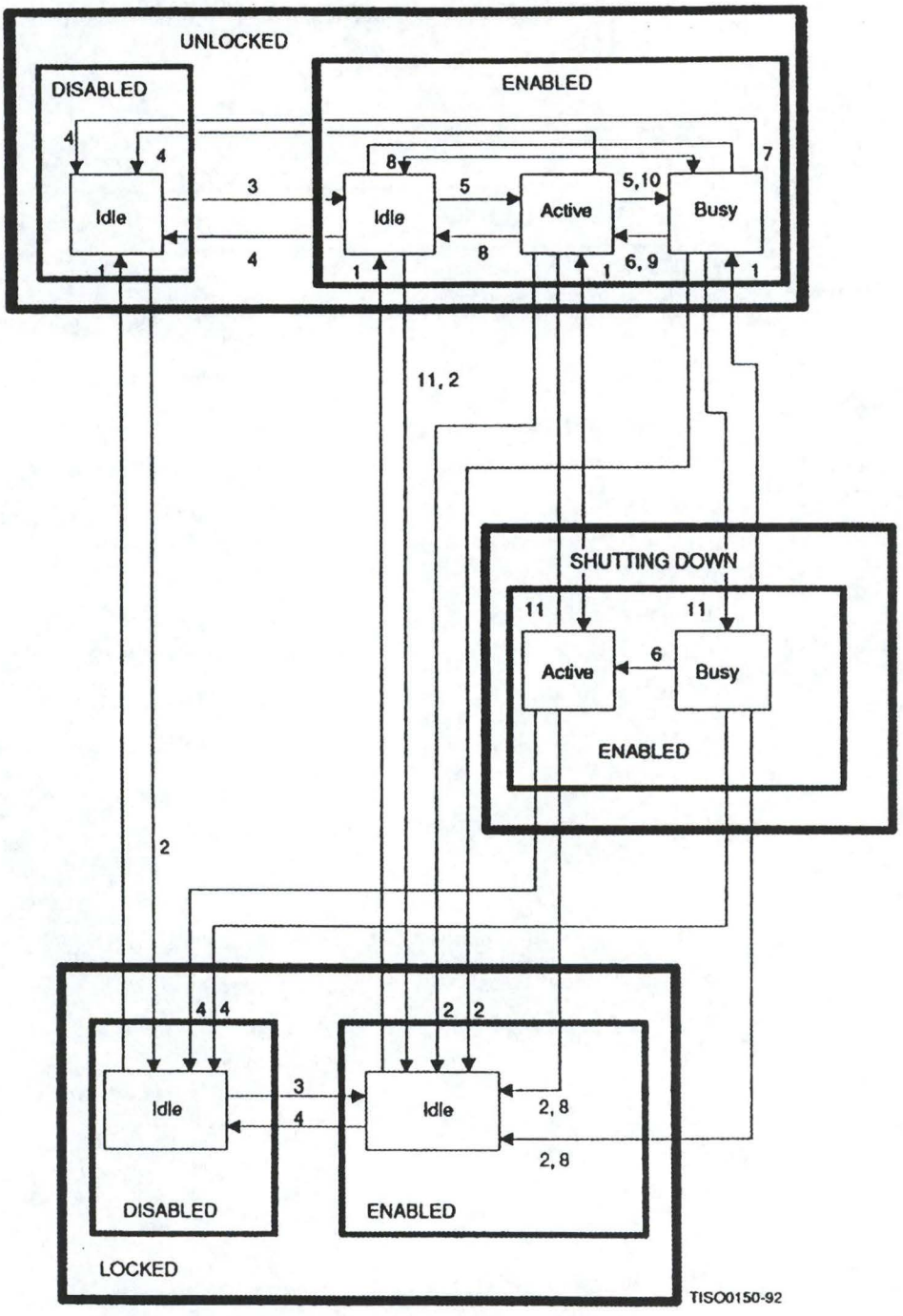
Intervention des fonctions de gestion-système dans les aires fonctionnelles

Le tableau ci-dessous nous donne un aperçu des fonctions de gestion-système utilisées par chacune des cinq aires fonctionnelles.

	Gestion de la configuration	**Gestion des anomalies**	**Gestion de la sécurité**	**Gestion des performances**	**Gestion des informations comptables**
Gestion d'objet	✓	✓		✓	
Gestion d'état	✓	✓			
Gestion des relations	✓				✓
Rapports d'alarmes		✓			
Gestion des rapports d'événements	✓			✓	
Contrôle des journaux		✓			✓
Rapport des alarmes de sécurité			✓		
audit-trail			✓		
Contrôle d'accès	✓		✓		
Métriques de comptabilité					✓
Surveillance de la charge				✓	
Gestion des tests		✓		✓	
Résumé				✓	

Diagramme combiné des états[53]

Cette illustration représente une utilisation combinée des différents états.



1 Unlock
2 Lock
3 Enable
4 Disable
5 New user
6 User quit
7 New user (nonsharable resource)
8 Last user quit
9 Capacity increase
10 Capacity decrease
11 Shut down

[53] Ce schéma est extrait du standard [ISO10164-2]. D'autres exemples y sont illustrés.

Annexe 2 : Les informations de gestion

Les informations se trouvant ci-dessous ne sont pas exhaustives, elles servent uniquement à montrer comment se concrétisent les définitions vues dans le chapitre sur les objets métriques[54].

Types de compteurs suggérés

Type de compteur	*Définition*
Corrupted PDUs Received	*Le nombre total de PDU erronés qui ont été reçus*
Incoming Connection Reject Error	*Le nombre total de demandes entrantes de connexion qui ont été reçues par l'objet géré mais qui ont été rejetées suite à des erreurs de protocoles*
Incoming Connection Requests	*Le nombre total de demandes entrantes de connexion*
Incoming Disconnect	*Le nombre total de demandes entrantes de déconnexion*
Incoming Disconnect Error	*Le nombre total de demandes entrantes de déconnexion qui ont été reçues par l'objet géré mais qui ont été rejetées suite à des erreurs de protocoles*
Incoming Protocol Error	*Le nombre total de PDUs (entrants) 'error report' ou 'reset' qui ont été envoyés par l'objet géré suite*
Octets Received	*Le nombre total d'octets de données 'utilisateurs' reçues*
Octets Retransmitted Error	*Le nombre total d'octets retransmis*
Octets Sent	*Le nombre total d'octets envoyés*
Outgoing Connection Reject Error	*Le nombre total de demandes sortantes de connexion qui ont été reçues par l'objet géré mais qui ont été rejetées suite à des erreurs de protocoles*
Outgoing Connection Requests	*Le nombre total de demandes sortantes de connexion*
Outgoing Disconnect	*Le nombre total de demandes sortantes de déconnexion*
Outgoing Disconnect Error	*Le nombre total de demandes sortantes de déconnexion qui ont été reçues par l'objet géré mais qui ont été rejetées suite à des erreurs de protocoles*
Outgoing Protocol Error	*Le nombre total de PDUs (sortants) 'error report' ou 'reset' qui ont été envoyés par l'objet géré suite à des erreurs de protocole*
PDUs Received	*Le nombre total de PDUs reçus*
PDUs Retransmitted Error	*Le nombre total de PDUs retransmis*
PDUs Sent	*Le nombre total de PDUs envoyés*

Définition de types d'attributs génériques

[54] Ces exemples sont tirés de [Stallings, 1993].

Attributs relatifs aux compteurs

Type d'attribut	*Type de valeur*	*Propriétés inhérentes*	*Opérations autorisées*	*Relations implicites*	*Spécification*
Non-settable Counter	Single value	• la valeur courante est un entier non négatif • il a une valeur maximale • il ne peut être qu'incrementé d'une unité • sa valeur retourne à 0 lorsqu'il atteint son maximum • la valeur initiale est 0	• Get	• est directement lié à un 'counter threshold' • peut déclencher un événement quand il retourne à 0	• l'événement interne qui est observé • la valeur maximale • estimation du temps avant la remise à 0
Settable Counter	Single value	• la valeur courante est un entier non négatif • il a une valeur maximale • il ne peut être qu'incrementé d'une unité • sa valeur retourne à 0 lorsqu'il atteint son maximum • la valeur initiale est 0	• Get • Set to (une valeur arbitraire)	• est directement lié à un 'counter threshold' • peut déclencher un événement quand il retourne à 0 ou quand il est modifié	• l'événement interne qui est observé • la valeur maximale • estimation du temps avant la remise à 0
Counter Threshold	Set-valued	• les niveaux de comparaison sont des entiers non négatifs • les valeurs des offsets sont des entiers non négatifs • l'interrupteur des notifications est sur *on* ou *off*.	• Set to (par défaut) • Get • Set • Add • Remove	• est directement lié à un compteur • est directement lié à une notification prédéfinie	• compteur auquel il s'applique • la notification qui doit être déclenchée

Attributs relatifs aux jauges

Type d'attribut	*Type de valeur*	*Propriétés inhérentes*	*Opérations autorisées*	*Relations implicites*	*Spécification*
Gauge	Single value	• la valeur courante est un entier ou un réel non négatif • elle a une valeur maximale et minimale • elle peut être incrementée ou décrémentée de valeurs arbitraires • sa valeur retourne à 0 lorsqu'il atteint son maximum • il n'y a pas de réinitialisation	• Get	• est directement lié à un 'gauge threshold' ou a un tide mark • seulement un tidemark maximun et minimum peuvent être appliqué • seulement un threshold (même multi-niveau) peut être appliqué • peut être utilisé pour mesurer d'autres informations de gestion	• la variable dynamique mesurée • la valeur maximale et minimale
Gauge Threshold	Set-valued	• la valeur courante est un entier non négatif • il a une valeur maximale • il ne peut être qu'incrementé d'une unité • sa valeur retourne à 0 lorsqu'il atteint son maximum • la valeur initiale est 0	• Get • Set • Add • Remove	• est directement lié à une jauge • directement lié à une notification prédéfinie	• la jauge à laquelle il s'applique • les notifications prédéfinies qui doivent être déclenchées • estimation du temps avant la remise à 0
Tide Mark	Set-valued	• les niveaux de comparaison sont des entiers non négatifs • les valeurs des offsets sont des entiers non négatifs • l'interrupteur des notifications est sur *on* ou *off*.	• Set to (par défaut) • Get	• est directement lié à une jauge • peut être directement lié à un événement qui est déclenché lorsque le valeur courante change	• la jauge à laquelle il s'applique • la direction (maximale ou minimale)

Types d'attributs spécifiques

Nous ne renseignerons pas tous les types d'attributs spécifiques qui sont définis dans [ISO10165-2], le lecteur intéressé par la liste complète des types d'attributs spécifiques définis par la norme s'y référera.

Liés aux événements	
Nom	**Description**
Additional information	*Informations supplémentaires sur les notifications*
Additional text	*Texte supplémentaire dans les notifications*
Attribute identifier list	*Liste des identificateurs d'attributs*
Event time	*Moment de la génération de l'événement*
Event type	*Type de l'événement*
Monitored attributes	*Un ensemble d'attributs et leur valeur au moment de l'alarme*
Perceived severity	*Niveau de sévérité de l'alarme*
Probable cause	*Cause probable de l'alarme*
Threshold info	*Information à propos du seuil qui a été franchi*

Utilisés pour la dénomination (naming)	
Nom	**Description**
Discriminator Id	*Nomme les instances de la classe d'objet discriminator*
Log Id	*Nomme les instances de la classe d'objet Log*
Log record Id	*Nomme les instances de la classe d'objet log record*
System Id	*Nomme les instances de la classe d'objet system*
System title	*Nomme les instances de la classe d'objet sytem*

Liés aux états	
Nom	**Description**
Administrative state	*Etat d'administration actuel*
Alarm status	*Ensemble de conditions pour les alarmes*
Availability status	*Ensemble de conditions sur le statut de disponibilité*
Control status	*Ensemble de conditions relatives au contrôle*
Operational state	*Etat opérationnel courant*

Liés aux relations	
Nom	**Description**
Member	*Ensemble des objets membres de l'objet en question*
Owner	*Ensemble des objets propriétaires de cet objet*

Relationships	*Relations composites de cet objet*
User	*Ensemble des objets qui utilisent les services de cet objet*

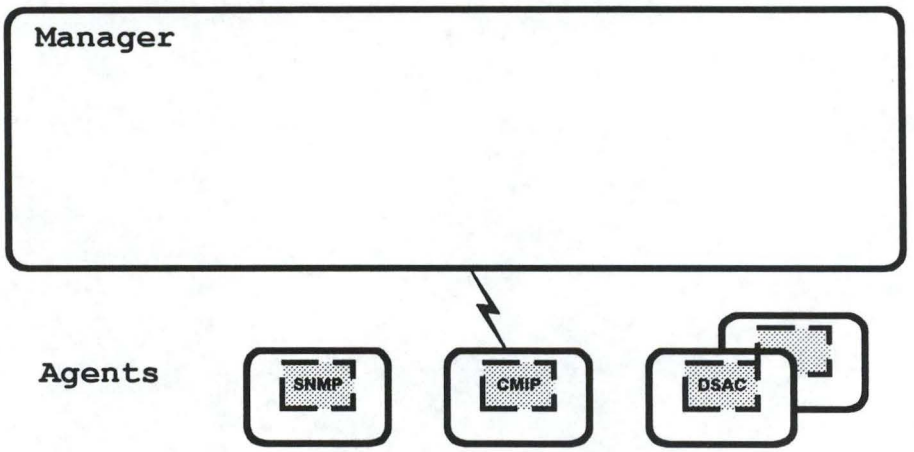
Types de notifications

Ce tableau donne une liste non-exhaustive des types de notifications et des types d'attributs que l'on peut retrouver au sein de celles-ci.

Type de notification	**Attributs**	**Rapport**
Attribute value change	sourceIndicator, attributeIdentifierList, attributeValueChangedefinition, notificationIdentifier, correlatedNotifications, additionalText, additionalInformation	Changements apportés aux attributs
Object creation	sourceIndicator, attributeList, notificationIdentifier, correlatedNotifications, additionalText, additionalInformation	Création d'un objet
Object deletion	sourceIndicator, attributeList, notificationIdentifier, correlatedNotifications, additionalText, additionalInformation	Suppression d'un objet
Environmental alarm	probableCause, specificProblems, perceivedSeverity, backedUpStatus, backUpObject, trendIndication, thresholdInfo, notificationIdentifier, correlatedNotifications, stateChangeDefinition, monitoredAttributes, proposedRepairActions, additionalText, AdditionalInformation	Détection d'un événement dans l'environnement

Annexe 3 : ISM/OpenMaster

Gérer le monde réel



Les applications de gestion