Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Aide à l'administration de systèmes distribués

Hasoppe, Etienne

Award date:
1994

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 03. Jul. 2025

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

Aide à l'administration de systèmes distribués

Description et développement

par Etienne HASOPPE

Promoteur:
Professeur J. Ramaekers

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

Année académique 1993-1994

Facultés Universitaires Notre-Dame de la Paix

Institut d'Informatique

Rue Grandgagnage, 21B

B-5000 NAMUR

Aide à l'administration de systèmes distribués
Description et développement

Etienne HASOPPE

Résumé

Les Systèmes informatiques ont bien évolué. Malheureusement, la complexité de la gestion de ces systèmes n'a fait que croître. Là où il n'y avait qu'une seule machine, en voilà vingt qui fleurissent. S'il veut pouvoir gérer efficacement son domaine, l'administrateur du système doit se faire aider par des outils qui automatiseront certaines tâches et lui laisseront l'initiative dans d'autres cas. Notre étude consiste en une tentative de formalisation de ces outils, inscrite dans la tendance actuelle de l'intégration.

Abstract

Computer system has turn topsy-turvy since the system has become strongly complicated. Hence management has followed. So if he wants to manage efficiently, say, his network, one has to get some help from the very information system in order to avoid running from east to west all along the day. One response may be to build a software solution that allow the manager to do his job from his own machine. Our study tries to apply previous issue concerning formal description of a distributed application to such a management plateform.

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

Septembre 1994

Promoteur: Professeur J. Ramaekers

Table des matières

Table des matières

Introduction I

Chapitre premier: Cadre général 1

I.1.Introduction 1

I.2.Des systèmes d'information 1

I.3.De l'histoire des systèmes 3
I.3.1.Introduction 3
I.3.2.Le moment centralisé 3
I.3.3.Le moment déconcentré 3
I.3.4.Le moment intégré 3

I.4.De la distribution et de l'évaluation des systèmes distribués 5
I.4.1.Des systèmes informatiques 5
I.4.1.1.Les mémoires 5
I.4.1.2.Les réseaux 6

I.4.2.Des systèmes distribués 10
I.4.3.Des objectifs poursuivis dans la distribution 11
I.4.4.Des caractéristiques de la distribution 11
I.4.5.Observations 12

Chapitre deuxième: Administration des systèmes 16

II.1.Introduction 16

II.2.Des fonctions de l'administrateur 16

II.3.Partitionnement du système administrateur 18

II.4.Des plates-formes d'aides à l'administration 19
II.4.1.Caractéristiques 19
II.4.2.Conception d'une plate-forme 20
II.4.2.1.Architecture fonctionnelle 20
II.4.2.2.Contrôle de l'application 22
II.4.2.3.De la distribution de l'application 22
II.4.2.4.Architecture concrète 23
II.4.2.5.Autre perspective 24
II.4.2.6.Remarques 25

Chapitre troisième: Aux confins de la Data Logic..... 26

III.1.Introduction..... 26

III.2.La gestion Internet..... 27
III.2.1.Le modèle de gestion Internet..... 27
III.2.2.Le protocole de gestion..... 27
III.2.3.Le modèle d'information de gestion..... 28
III.2.4.Les extensions..... 29
III.2.5.Exemple..... 32

III.3.La gestion OSI..... 40
III.3.1.Le modèle de gestion ISO..... 40
III.3.2.Le protocole de gestion..... 41
III.3.3.Le modèle d'information de gestion..... 44

III.4.Comparaison..... 45
III.4.1.L'efficacité..... 45
III.4.2.La robustesse..... 47
III.4.3.La flexibilité..... 47
III.4.4.La sécurité..... 48
III.4.5.Les fonctions d'application..... 48
III.4.6.Les considéérations de coûts..... 48
III.4.7.Les domaines d'application des deux approches..... 48
III.4.8.La cohabitation..... 49
III.4.8.1.Les piles mixtes..... 49
III.4.8.2.Les piles duales de protocole..... 50
III.4.8.3.Les interfaces communes de programmation d'application..... 51
III.4.8.4.L'intégration Pass-Through..... 52

Chapitre quatrième: Evaluation des systèmes d'aide à l'administration..53

IV.1.Introduction..... 53

IV.2.Des critères fonctionnels..... 53

IV.3.Des critères opérationnels..... 53

IV.4.Autres critères..... 55

Chapitre cinquième: Etude de cas..... 57

V.1.Présentation de l'application distribuée.....57
V.1.1.ISM Manager.....58
V.1.2.MIB ISM.....59
V.I.2.1.Structure.....60
V.1.2.2.Opérations.....60
V.1.2.3.Dénomination d'objet et hiérarchie.....60

V.1.3.Les applications.....61
V.1.4.Les services.....62
V.1.5.Les agents intégrateurs.....63
V.1.6.L'interface Supra-Manager.....64
V.1.7.CMIS Request Broker.....65

V.2.Presentation de la boîte à outils.....67
V.2.1.ISM Monitor.....67
V.2.2.ISM Script.....68

Introduction

Introduction

La première idée de ce travail consistait à décrire les fonctions de l'administrateur d'un système distribué et d'imaginer des outils permettant de l'aider, avant de nous plonger dans la réalité des produits actuellement sur le marché pour rechercher les fonctions prises en compte et leur réalisation.

Une première lecture de la littérature et des annonces publicitaires (non reprises dans la bibliographie) nous a révélé combien la notion même de système distribué était utilisée de manière confuse pour couvrir des domaines tels que les systèmes ouverts, les systèmes décentralisés, etc.

Aussi avons-nous décidé de mieux cerner cette notion de distribution tant au niveau du système d'exploitation qu'au niveau de l'application en constituant une vue panoramique dans laquelle il serait possible de saisir le contexte plutôt qu'un fait particulier. C'est, nous le pensons, ce qui manque le plus actuellement, qu'il s'agisse d'informatique ou d'autres choses.

Dans ce contexte, les outils d'aide à l'administration pouvaient se concevoir de deux manières. Il était possible de décrire les fonctions d'aide à l'administrateur offertes par les plates-formes considérées comme des supports, mais nous pouvions également voir ces plates-formes sous l'angle des applications distribuées et tenter de leur appliquer une formalisation de conception et de préciser quelques approches de réalisations existantes.

Bien sûr nous pourrons illustrer nos développements par le cas vécu durant notre stage dans la firme BULL S.A. (France).

Ce mémoire est composé de cinq chapitres.

Le premier est consacré à une histoire et une caractérisation des systèmes d'information et parmi eux, les systèmes informatiques. Nous introduirons ensuite la distribution en tentant de lui adjoindre des caractéristiques. Il est intéressant de voir que les principes sous-tendant les systèmes distribués (délocalisation, etc.) sont généraux et ont d'autres applications que la seule informatique. Nous détaillerons ensuite les moyens techniques que comprennent les systèmes distribués.

Après une présentation de ce que nous entendons par administration et des tâches relevant de cette fonction, le second chapitre décrit une **architecture générale** des traitements de toute application distribuée, indépendamment de son infrastructure. Nous particulariserons cette application en considérant la plate-forme d'aide à l'administration.

Dans le troisième chapitre, nous entreprendrons la description de la réalisation d'une certaine catégorie de traitement que nous estimons à la base de toute plate-forme. Ayant isolé deux courants, nous les comparerons pour mieux cerner leurs spécificités.

Revenant un instant aux plates-formes, nous proposerons dans le quatrième chapitre des critères qui devraient permettre d'évaluer les qualités et les défauts d'une plate-forme.

Nous terminons, dans le cinquième chapitre, par une **étude de cas** dans laquelle nous tentons de reprendre tous les principes théoriques dégagés précédemment au travers d'une plate-forme d'administration développée par la société française Bull.

Chapitre 1

Chapitre premier: Cadre général

I.1 Introduction

L'administration des systèmes informatiques s'inscrit dans le contexte plus général des systèmes d'information. Dès lors, avant de nous focaliser plus particulièrement sur l'administration, nous aimerions prendre, ici, un peu de champ et présenter les différentes parties en présence.

I.2 Des systèmes d'information

Les organisations élaborent et utilisent des systèmes d'information pour satisfaire leur besoin d'information. Aussi considérerons-nous la définition donnée par [Bodart,1989].

> « Par système d'information d'une organisation, nous entendons une construction formée d'ensembles:
>
> - d'informations (...)
> - de traitements (...)
> - de règles d'organisation (...)
> - de ressources humaines et techniques requises pour le fonctionnement du S.I. »

Ainsi le système d'information ne peut être réduit au seul système automatisé mais comprend également des traitements interactifs et manuels. En effet, cette construction n'est pas aveugle; elle comprend et modélise les multiples facettes de la réalité de l'organisation qu'elle est censée servir. Les comportements organisationnels de cette réalité peuvent être perçus comme des combinaisons spécifiques de trois dimensions tour à tour dominantes:

- production ou exécution;
- communication;
- décision.

En raison de leur nature, ces trois dimensions revêtent une possibilité de formalisation qui va du plus structuré au plus informel.

Par ailleurs, chaque comportement peut être mené à différents niveaux « logiques » au sein de l'organisation : le niveau stratégique, où se définissent les objectifs à long terme ou généraux; le niveau de coordination, où les objectifs généraux sont traduits en termes d'objectifs à moyen et court terme; et le niveau opérationnel, où s'effectuent les tâches quotidiennes d'exécution. A cette certaine diversité de comportements correspond une diversité de systèmes mais nous en retiendrons deux types en particulier :

- les systèmes transactionnels pour des applications se situant au niveau opérationnel et basées sur des plans d'action parfaitement structurés;

- les systèmes analytiques pour des applications stratégiques ou de coordination et basées sur des plans semi structurés ou carrément informels (c'est à dire des outils d'analyse).

Tels sont les deux axes, dimensions et niveaux, d'une typologie classique des activités organisationnelles, brièvement exposées, représentée à la figure 1. Et celle-ci de nous donner une idée des domaines informatisés et vraisemblablement informatisables.

Les différentes notions que nous venons d'évoquer nous seront utiles lorsque nous parlerons d'un système d'information particulier: le système d'administration de système (Chap. II.). Mais pour l'heure, nous nous intéresserons à l'histoire des systèmes d'informations.

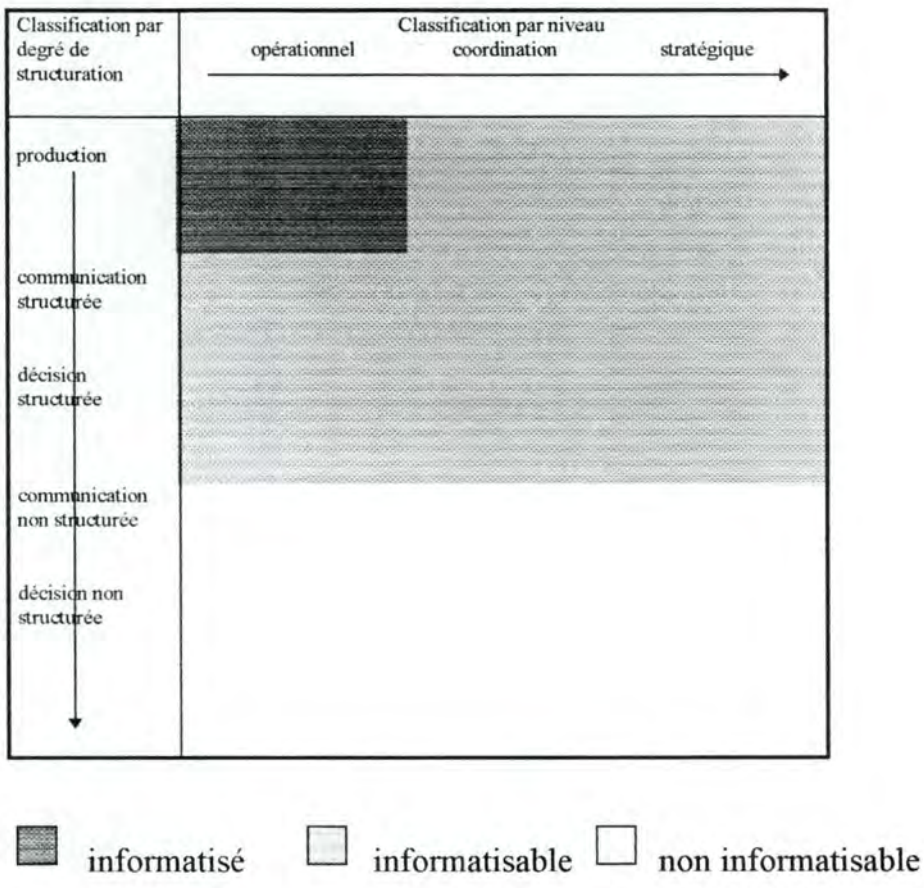


Fig. 1. - schémas d'analyse des activités organisationnelles.

I.3 De l'histoire des systèmes

I.3.1 Introduction

Dans cette présentation, nous ne nous limiterons pas à la découpe classique de l'évolution des ordinateurs. Nous utiliserons plutôt la notion de moment telle que présentée dans [Pichault,1990] afin de mieux percevoir les interactions qui existent entre l'évolution des organisations et l'évolution technique.

Ce terme désigne une étape logique au sein d'un processus qui contient en elle-même la puissance de conduire d'une idée à son contraire. Autrement dit, nous considérerons le moment tant dans son sens physique c'est à dire la puissance de mouvoir, la cause du mouvement et sa force, que dans son sens plus commun de repère dans le déroulement d'un processus.

Il s'agit donc autant de l'instant que des tensions qui affectent le mouvement d'informatisation.

Nous repérons ainsi trois étapes: le moment centralisé, déconcentré et intégré. Chacune présente une cohérence logique provisoire entre une base technique et un mode d'organisation du travail.

I.3.2 Le moment centralisé

Nous plaçant dans le cadre de la taylorisation des services, cette phase concerne les activités qui se prêtent d'emblée à la parcellisation, celles qui présentent un caractère cyclique ou répétitif: comptabilité, gestion, paie ... En les regroupant, on obtient un flux continu, ou plutôt un flux dont la densité est telle que le caractère discontinu des activités qu'il intègre s'estompe. De plus, les cycles étant différents et complémentaires, leur centralisation permet de mieux répartir le travail à effectuer et d'optimiser l'occupation des agents en vue de réduire la marge de liberté en prédéterminant l'exécution des tâches.

Toutefois se pose le problème de la capacité de traitement. Cette centralisation nécessite en effet des équipements capables de traiter une quantité importante d'informations en un minimum de temps.

Il apparaît donc indispensable d'ajuster la base technique en introduisant les systèmes mécanographiques remplacés bien vite par les unités informatiques centralisées.

I.3.3 Le moment déconcentré

Mais bientôt, la parcellisation de la chaîne de production conduisant à une prolifération des postes de travail, porte finalement préjudice à l'efficacité de l'ensemble tant sur le plan économique que sur le plan organisationnel. L'entreprise ne peut plus se permettre de fonctionner sur un mode centralisé unique et rigide, basé sur une idée d'optimalité très peu réaliste dans le nouveau contexte économique. Aussi va-t-on introduire une plus grande flexibilité dans le processus de production ce qui permettra

- une adaptation plus rapide aux changements imprévisibles du volume de travail;

- un meilleur suivi de la diversification des produits, dans la mesure où la concurrence ne porte plus tellement sur la capacité d'augmenter le volume de production mais plutôt sur la qualité du service offert au consommateur, personnalisation , conseils, informations;

Cette introduction se traduit sur le plan technique par l'apparition d'un mouvement de déconcentration aux dépends du modèle informatique traditionnel que l'hyper-centralisation rendait trop vulnérable aux pannes, aux problèmes d'engorgement et *sabotages*. Se basant sur les progrès techniques (généralisation des applications de la micro-informatique, miniaturisation des équipements, ...) et la baisse des coûts des composants, les nouvelles solutions informatiques n'impliquent plus nécessairement la centralisation des traitements. L'informatique, étant elle-même un produit, n'échappe pas aux évolutions contextuelles: diversification des applications, adaptation à la multiplicité des situations locales (personnalisation) et convivialité accrue.

Et cette déconcentration-même, envisagée initialement dans l'optique d'un désengorgement des unités centrales par le transfert d'activités secondaires au niveau local, va conduire à une rupture par rapport au moment centralisé. La tendance est désormais à la spécialisation des postes de travail et à la prolifération des équipements et du savoir-faire technique; les réalisations isolées et applications difficilement exportables abondent et les procédures d'apprentissage et d'installation sont souvent menées en parallèle, sans concertation.

D'autre part, l'impératif de flexibilité impose également l'adoption de nouvelles formes d'organisation. Ici aussi apparaît la tendance à la délocalisation:

- multiplication des lieux de décision;
- émergence d'une relative autonomie de gestion pour les unités locales permettant aux opérateurs d'intervenir partiellement dans la mise au point de leurs conditions de travail (élargissement de la sphère de responsabilité et diversification des tâches);
- externalisation d'un certain nombre de fonctions par un recours à la sous-traitance et au travail intérimaire;

I.3.4 Le moment intégré

Devant la prolifération incohérente des bases techniques et les nombreux problèmes qu'elle pose, les responsables vont tenter de rétablir une certaine coordination par l'intégration progressive des différents équipements locaux en une collection d'ordinateurs interconnectés. Dans un tel contexte, les maîtres mots sont compatibilité et communication.

La tendance est maintenant au poste de travail multifonctionnel, non plus limité à l'exécution d'un seul type de tâches mais offrant à l'utilisateur la possibilité de réaliser les applications les plus diverses et à la communication entre utilisateurs.

D'un point de vue organisationnel, les formules de flexibilité du travail ne font que s'étendre au cours du moment intégré tout en changeant de forme:

- la Culture d'entreprise, processus d'identification du travailleur aux intérêts supérieurs de l'entreprise, qui remet à l'honneur les valeurs de compétitivité et de rendement;
- les Cercles de Qualité qui permettent de saisir le savoir-faire collectif et de faire remonter les informations pertinentes.
- la Polyvalence.

Il est bien évident que c'est dans ce cadre que se développent les systèmes distribués dont nous allons aborder la présentation.

I.4 De la distribution et de l'évaluation des systèmes distribués.

Avant tout palabre sur quelque système informatique que ce soit, nous devrions peut-être préciser ce que nous entendons, pour l'instant, par les termes «*systèmes informatiques*».

I.4.1. Des systèmes informatiques

Il s'agit, en fait, d'un sous-ensemble du système d'informations, l'ensemble des informations, des traitements, des règles organisationnels, des ressources humaines en relation avec les ressources informatiques (processeurs, mémoires, imprimantes et des outils de communication).

Nous aimerions à présent détailler quelques classes de ressources informatiques afin de teinter cette présentation d'un soupçon de matérialité...

I.4.1.1 Les mémoires.

Ce sont elles les détentrices de l'information traitée par les autres ressources informatiques[1]. Nous pouvons établir entre elles une hiérarchie [Cardinael,1992] (fig.2).

Nous noterons que les caractéristiques de temps d'accès et de capacité évoluent parallèlement dans la même direction. En effet, plus la capacité augmente, plus le temps d'accès augmente. C'est pourquoi nous regrouperons les mémoires selon le critère unique de la vitesse en ne considérant que deux groupes : celui des mémoires attachées à des périphériques lents et celui des mémoires attachées à des périphériques rapides.

[1] Il nous faut être prudent dans la qualification. Nous avons vu en effet que les utilisateurs et autres opérateurs humains recelaient par ailleurs d'informations extérieures aux machines.

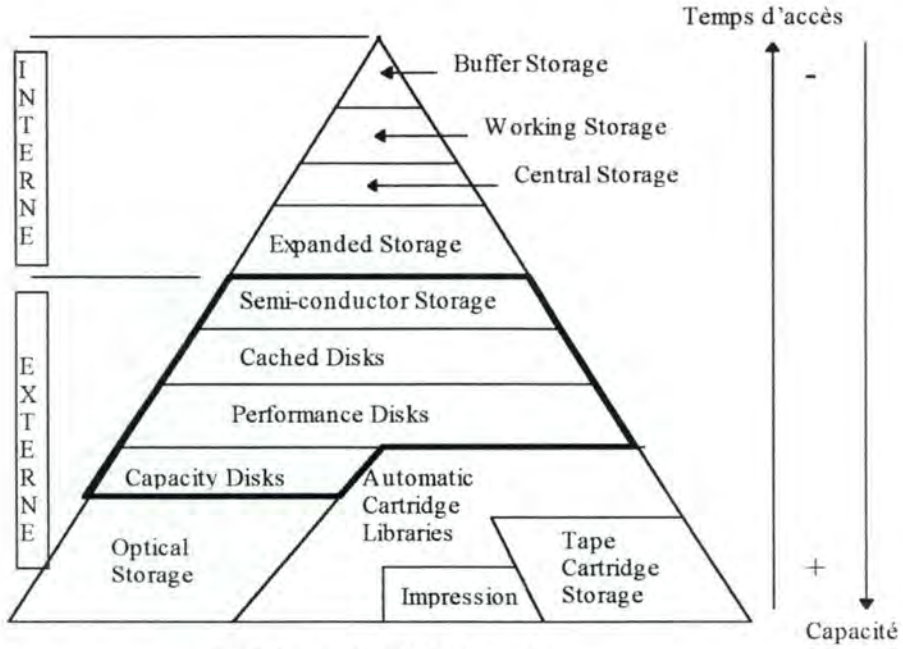


Fig.2 Le mystère de la Pyramide

I.4.1.2. Les réseaux.

Nous envisagerons dans un premier temps les types de réseaux pour ensuite envisager les architectures standards proposées pour la communication entre systèmes.

Les types de réseaux.

Nous retrouvons dans la littérature trois types de réseaux. Les réseaux à longue distance (WAN), les réseaux locaux (LAN) et les réseaux intermédiaires (MAN).

Les réseaux à longue portée peuvent courir sur une distance de plusieurs milliers de kilomètres. Ainsi sont-ils utilisés à un niveau mondial. Ils reposent principalement sur le mécanisme de commutation selon lequel chaque noeud du réseau enregistre les messages qu'il reçoit avant de les faire suivre sur une ligne libre. Cette commutation peut se faire en conservant le chemin pendant toute la durée de la communication auquel cas elle sera dite Commutation de circuit. Elle peut également être appelée Commutation par messages si les informations à transférer sont envoyées en bloc, en un seul message de longueur variable. Enfin, plus couramment, elle peut être réputée « par paquets » lorsque les informations à échanger sont découpées en paquets de longueur fixe.

Les réseaux locaux sont limités à quelques kilomètres mais possèdent généralement des débits de transmission très élevés. Ils reposent souvent sur le principe de diffusion par lequel chaque noeud reçoit les informations et vérifie si elles lui sont destinées.

Les réseaux intermédiaires ou métropolitains utilisant les techniques des réseaux locaux servent à faire le lien entre réseaux locaux distants.
Tout ceci nous permet représenter la hiérarchie de ces réseaux en utilisant le schéma donné par [Deghorain,1992] de la figure3.

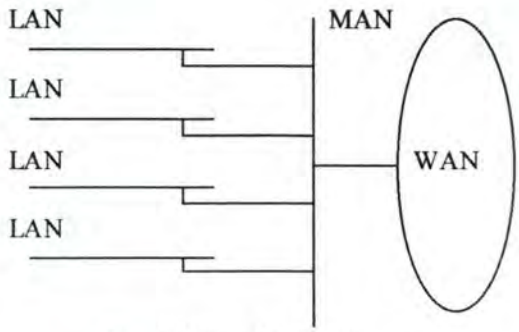


Fig. 3 Hiérarchie des réseaux

Les modèles.

Dans le but de promouvoir la communication et la coopération entre systèmes, certaines organisations se sont penchées sur une définition d'un modèle de référence de communication. Parmi elles, se trouvent l'ISO (International Standards Organisation) et le DOD (Department of Defense). Tandis que l'ISO développait l'architecture d'une solution complète et générale, le DOD travaillait dans le cadre des demandes du marché.

Les modèles qui en ont découlé sont basés sur le principe de répartition des fonctions en couches et sur la notion de protocole qui définit les conventions relatives à l'établissement de la communication, aux échanges de données et à la fermeture de la communication. Une découpe en couches, par le recours à une série limitée d'interfaces, permet la modification des fonctions au sein d'une couche sans que ces modifications aient des répercutions dans les couches supérieures.

Le modèle OSI.

Ce modèle introduit une structure en sept couches, chacune utilisant les services de la couche inférieure (sauf la première) et offrant des services à la couche supérieure (sauf la dernière). Nous nous proposons d'en faire une brève présentation.

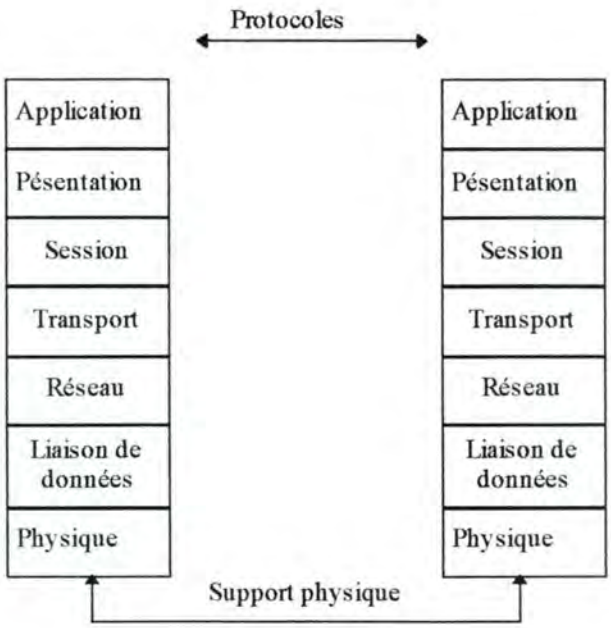


Fig. 4 Le monde selon OSI

- Niveau 1: la couche physique.

  Prenant ses fonctions juste au-dessus du support d'interconnexion (câbles, ondes), elle responsable de l'activation, du maintien et de la désactivation du circuit physique entre deux appareils à l'aide de moyens électriques et mécaniques qu'elle fournit, et de la transmission bit à bit à travers le circuit.

- Niveau 2: la couche Liaison de données

  Cette couche a en charge le transfert de données à travers la liaison et leur reconstitution en unités de donnée. A ce titre, elle doit généralement vérifier que les données arrivent correctement à destination, que le récepteur n'est à aucun moment submergé par un flot trop abondant de données. Elle doit de même opérer une détection d'erreur de transmission et fournir des mécanismes pour pallier à la perte, à la duplication et à la falsification des données transmises.

- Niveau 3: la couche Réseau

  Elle a pour mission de fournir un transfert transparent des données entre deux utilisateurs des services réseau. Ceci comprend le routage entre réseaux homogènes ou hétérogènes.

- Niveau 4: la couche Transport

  Cette couche marque la séparation entre les trois premières couches, orientées vers la communication proprement dite, et les autres. En plus de cette fonction d'interface, elle met à la disposition de ces utilisateurs des options permettant de négocier un certain niveau de qualité dans le transfert.

- Niveau 5: la couche Session

  Elle est chargée des communications logiques entre utilisateurs. Elle fournit des moyens d'organiser les échanges de données entre utilisateurs, comme les

transmissions simultanées, alternatives, points de contrôle et autre mécanisme de resynchronisation des flux de données. En somme, elle active, entretient et désactive les sessions de communication entre les participants en coordonnant et synchronisant les échanges entre applications.

- Niveau 6: la couche Présentation

  La couche présentation offre les services de description et de représentation des structures de données. Si elle n'est en rien concernée par la signification même des données, elle n'en doit pas moins la conserver. Elle accepte plusieurs types de donnée, négocie et convertit les représentations, qu'il s'agisse de celle utilisée par l'utilisateur émetteur, par l'utilisateur récepteur ou encore de celle utilisée dans l'échange entre entités de présentation.
  L'ISO a développé une syntaxe abstraite convenant à la présentation et au transfert, utilisée par les couches applications utilisatrices. Il s'agit de l'Abstract Syntax Notation One ou ASN.1 [ISO,8824]

- Niveau 7: la couche Application

  Cette ultime couche contient les services d'application tel la gestion, le transfert de fichiers, le courrier électronique,... Contrairement à la couche précédente, Elle connaît de la sémantique des informations échangées.

Le modèle DOD.

Le second modèle en couches ne provient pas d'un organisme de standardisation mais provient de recherches qui ont abouti au *TCP/IP protocol suite* et devenu un standard *de facto*.

Un logiciel dans cet environnement est organisé autour de quatre couches distinctes (Fig.5).

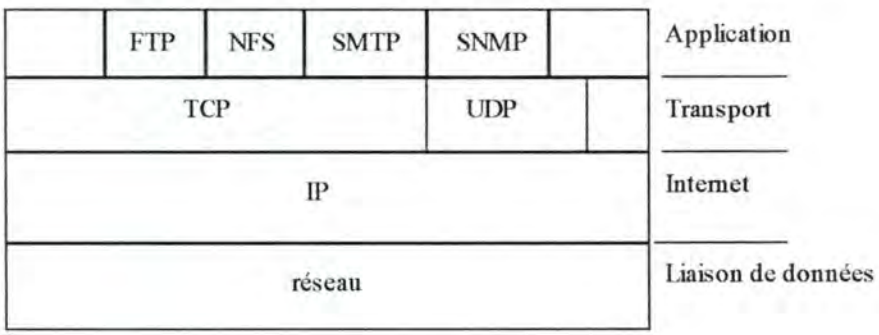


Fig.5 Le modèle DOD

- Niveau 1: la couche d'interfaces réseau ou Liaison de donnée.

  Cette couche correspond en fait aux deux premières du modèle OSI. Comme elles, celle-ci permet de masquer les détails du réseau. Cela se fait par la définition des interfaces entre les couches supérieures et ce réseau.

- Niveau 2: la couche Internet.
  Comprise comme la troisième couche du modèle OSI, elle prend en charge le routage des informations en faisant appel à l'interface réseau adéquat.

- Niveau 3. la couche Transport.
  Le premier devoir de cette couche est de fournir une communication entre deux applications. Elle doit en outre réguler le flot de données transférées.
  IL existe deux possibilités de protocole. Soit le TCP, Transport Control Protocol, qui doit en plus assurer un transport fiable c'est-à-dire un transport sans erreur qui conserve la séquence d'émission. Soit un UDP, User Data Protocol, qui renonce à la fiabilité.

- Niveau 4: la couche Application
  Cette couche offre un ensemble de protocoles de services aux utilisateurs en interagissant avec la couche de transport. Sont compris dans cet ensemble, FTP (File Transfer Protocol - transfert de fichiers), SMTP (Simple Mail Transfer Protocol - courrier électronique), SNMP (Simple Network Management Protocol - échanges d'informations de gestion), NFS (Network File System, partage de fichier transparent et intégré).

I.4.2. Des systèmes distribués.

Les systèmes distribués forment un champ nouveau de recherches. Dans la panoplie des systèmes centralisés, décentralisés et autres agrégats de systèmes autonomes reliés par des réseaux, nous nous intéresserons à ceux possédant le plus haut degré d'intégration d'ordinateurs autonomes.

L'idée de distribution peut se comprendre selon quatre pôles: la distribution du matériel, du contrôle, des traitements et celle des informations.

- Matériel
  Un système distribué doit comprendre plusieurs ordinateurs (processeur et mémoire) interconnectés via un réseau. Cette distribution est le reflet de la distribution physique des applications ou de la décomposition fonctionnelle du système.

- Contrôle
  Qu'il s'agisse de ressources physiques telles les processeurs, terminaux,... ou des ressources logiques comme les fichiers,..., il doit y avoir un contrôle permettant de gérer les ressources et de coordonner les activités. La stratégie utilisée peut être centralisée, hiérarchique ou déléguée aux entités autonomes.

- Traitements
  Les traitements peuvent être répartis de manière statique ou idéalement dynamique permettant une réaffectation selon les disponibilités du moment.

- Informations
  Les informations peuvent être dupliquées ou partitionnées dans le système.

I.4.3. Des objectifs poursuivis dans la distribution

Autrement dit « Quels sont les problèmes rencontrés dans les systèmes antérieurs auxquels la distribution entend donner une solution unifiée ?» Ces objectifs sont donc les bénéfices attendus de l'instauration des systèmes distribués.

Partage des informations.
Il s'agit de permettre la communication et le partage de l'information. En effet, souvent les informations sont générées en un lieu et utilisées en d'autres endroits.

Partage des ressources informatiques pour un meilleur usage des ressources telles que les imprimantes,...

Amélioration du rapport Coût/Performance.
Le coût d'un ordinateur dépend de ses performances en termes de vitesse de processeur et de capacité mémoire et ce coût va décroissant; le coût des moyens de communication, quant à lui, dépend généralement de leurs capacités et de leurs envergures mais diminue bien moins rapidement que le précédent.
Nous pouvons observer d'autre part que les exigences en matière d'interfaces homme/machine se sont durcies avec l'amélioration des environnements graphiques et autres. Ainsi les systèmes distribués sont-ils perçus non pas seulement comme économiquement souhaitables mais encore comme nécessaires pour une meilleure répartition des charges (*rightsizing)*.

Fiabilité.
Il ne s'agit pas seulement de disponibilité d'informations. Encore faut-il que ce que le système prétend avoir fait, ait été fait correctement. En cas de problème, il faut appliquer une redirection des activités vers une autre machine.

Modularité.
Une approche modulaire du système et des applications dans l'optique d'une réutilisation et d'un accroissement de la puissance de calcul.

Facilité d'évolution.
Cet objectif concerne tant la facilité de remplacement des éléments que l'aptitude à repousser les limites imposées par la capacité forcément finie des composants du système.

I.4.4. Des caractéristiques de la distribution

Nous venons de donner les principaux objectifs de la constitution de systèmes distribués. Bien sûr, les solutions y afférent constituent autant de propriétés de ces

systèmes. Mais il est un groupe parmi ces propriétés que l'on peut retenir et dont la présence trahit l'existence d'un système distribué.

Des systèmes distribués, Tanenbaum nous donne la définition suivante:

> Un système distribué est un système qui apparaît aux yeux de ses utilisateurs comme un système centralisé ordinaire s'exécutant toutefois sur plusieurs processeurs indépendants. Le concept-clé en est la transparence ou, en d'autres mots, le fait que l'usage de plusieurs processeurs reste invisible à l'utilisateur. En d'autres termes, l'utilisateur voit le système comme un monoprocesseur virtuel et non comme une collection de machines distinctes.

Cette transparence recouvre en fait plusieurs idées distinctes [Deghorain,1992].

- **Transparence de localisation :** l'utilisateur peut tout ignorer de la localisation de la ressource qu'il veut atteindre;
- **Transparence d'accès :** uniformité des procédures d'accès aux ressources;
- **Transparence de concurrence :** partage invisible et insensible des ressources;
- **Transparence de duplication et de partition :** l'utilisateur ignore tout de la distribution des informations;
- **Transparence des pannes :** quasi-occultation des pannes survenues dans le système;
- **Transparence de migration :** le déplacement de ressources au sein du système n'affecte en rien le bon déroulement des opérations en cours;
- **Transparence de performance :** adaptation dynamique de la configuration du système;
- **Transparence d'échelle :** elle concerne la facilité d'évolution évoquée plus haut.

Toutefois la transparence ne constitue tout au plus qu'une condition nécessaire à l'obtention d'un système distribué. Nous pouvons lui adjoindre d'autres conditions :

- la présence de plusieurs éléments de traitement (processeur et mémoire);
- la présence d'équipements d'interconnexion;
- l'indépendance de panne, c'est-à-dire qu'en aucun cas le mauvais fonctionnement d'un composant peut entraîner la mise hors service des autres éléments;
- l'existence d'un état commun de sorte que la chute d'un élément n'entraîne pas une perte de cohérence dans l'état du système.

Nous devons cependant préciser que ces propriétés ne constitue pas un ensemble suffisant pour reconnaître de manière infaillible un système distribué.

I.4.5. Observations

Cette distribution n'apparait donc pas à l'utilisateur. Ce qui nous permet de considérer le système distribué non plus comme un ensemble de machines mais comme une seule machine **centralisée virtuellement** (Fig.6). Du point de vue du programmeur, rares sont les systèmes d'exploitation distribués. Pratiquement, ces systèmes sont

concrétisés par l'intégration de différents systèmes non-distribués. Aussi la prise en charge de la réalisation des conditions que nous avons vues ne pourra se faire au niveau de base mais devra être opérée par les niveaux supérieurs. Ainsi le programmeur d'applications devra-t-il faire appel à des artefacts concentrés dans une couche sise entre le système d'exploitation et le niveau applicatif appelée couche de Middleware (Fig.7) qui a comme rôle d'assurer la transparence des communications (Voir [Karemera,1992]). Cet état des choses n'est pas sans rappeler l'évolution de la prise en charge des communications par les logiciels. Au début, une application espérant communiquer avec une autre entité se devait de gérer elle-même ses échanges. Peu à peu, les concepteurs ont introduit l'usage de boite à outils qui les déchargeait de cette tâche. Pour finir, le système d'exploitation a avalé la boite à outils en la transformant en module de télécommunication. Nous ajouterons encore qu'actuellement, la couche de middleware est loin d'être complète.

Nous observons par ailleurs que parmi les paradigmes de la programmation, **l'orienté objet** est le plus répandu en matière de distribution.

Les principaux avantages du paradigme dde l'orienté objet en matière de distribution d'un système d'information sont entre autres:

- Décomposition plus adéquate

Les objets offrent une variabilité dans la décomposition beaucoup plus grande que le concept de processus habituel. C'est à dire qu'il peut y avoir des objets de tailles différentes du plus léger au plus gros tout en gardant une grande cohésion dans la découpe.

- Concurrence

Les objets peuvent exister et agir en parallèle, permettant ainsi d'exploiter au mieux la concurrence. Même les objets internes à une processus peuvent utiliser la concurrence par le mécanisme des *Threads*. Cette concurrence implique l'idée de transaction, d'action *atomique* entre objets.

- Messages de haut niveau

Les objets sont unis entre eux par un protocole de messages plutôt que par une zone de mémoire partagée, par exemple. Cela donne un moyen d'*encapsulation* qui permet de modifier un élément sans avoir à répercuter les modifications dans tout le système. Cette caractéristique est par exemple intéressante dans la gestion de l'intégration de techniques différentes au sein d'un même système.

- Reconfiguration dynamique

Les interconnexions basées sur les messages permettent également des changements dynamiques du système. Les objets peuvent être remplacés par de nouvelles versions ou multipliés et déplacés. En effet, le mécanisme de message est un élément autonome contrôlant les échanges de sorte que les messages peuvent être redirigés sans que l'émetteur n'en ait connaissance. Aucune modification d'architecture n'est nécessaire lorsque le système croît. Cette caractéristique permet de construire des systèmes de gestion qui répondent au critère de maintenance, de robustesse, de disponibilité et de flexibilité vis à vis des besoins des utilisateurs.

- Transparence de dénomination et de localisation
Le mécanisme clé de la reconfiguration dynamique est le mécanisme de dénomination, c'est à dire de la traduction d'un nom symbolique en une adresse effective d'objet. Le mécanisme d'adressage peut retrouver lui-même la trace des objets de sorte que l'adressage est teinté de transparence...

- Décentralisation des services.
La décentralisation est à la base de la distribution. Aussi, alors que dans un système centralisé les opérations étaient prises en charge par un gros noyau, dans les systèmes distribués, les objets permettent une répartition des opération entre eux et ainsi sur différentes machines.

- Persistence
Les objets doivent souvent avoir une certaine durée de vie.

- Objets actifs
Certains objets sont passifs c'est à dire qu'ils attendent que d'autres prennent l'initiative de les réveiller. Les objets actifs possèdent leur propre séquence de contrôle leur donnant un certain esprit d'initiative.

- la médiation
Il s'agit d'intégrer différents systèmes considérés a priori comme incompatibles. Une solution est de voir ces systèmes comme des objets parlant chacun son dialecte dont la traduction sera assurée par d'autres objets (nous y reviendrons plus tard). Une autre méthode consiste à revoir les systèmes en leur apprenant le bon protocole...

Aux avantages du paradigme de l'orienté objet lors du développement d'un système distribué, s'opposent les quelques contraintes suivantes:

- Le langage dans lequel est programmer le système doit vérifier certaines propriétés que nous évoquerons en II.4.4

- Les objets résolvent quelques problèmes mais en posent d'autres comme la surchage de communication due à l'usage de messages, le mécanisme d'adressage,...

- L'utilisation d'informations disséminées posent le problème de performance (Index,...) mais aussi de sécurité et d'intégrité. Problème résolu dans l'usage des transactions.

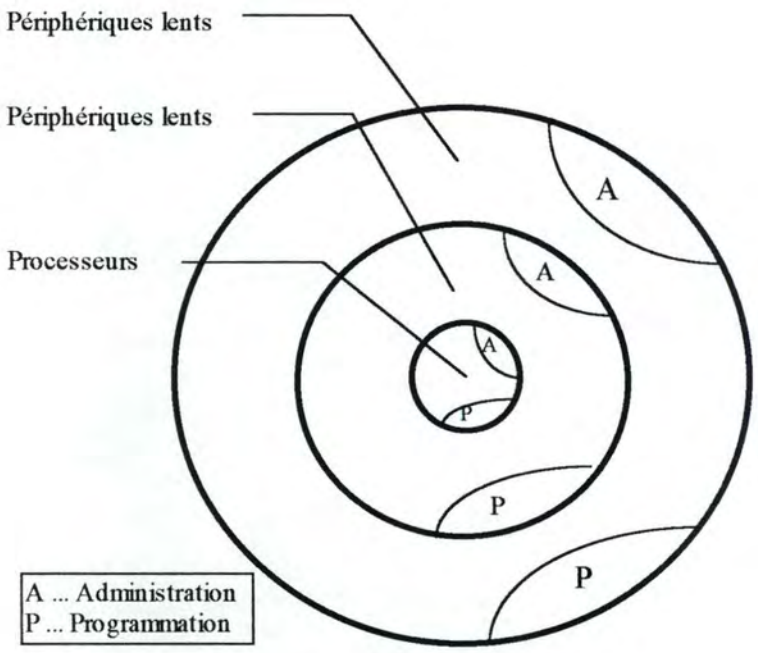


Fig. 6 La machine abstraite

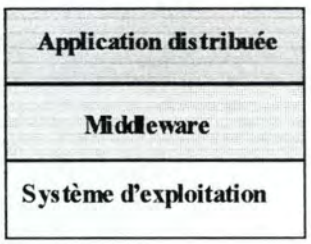


Fig. 7 Architecture des logiciels

Chapitre 2

Chapitre deuxième: Administration des systèmes.

II.1. Introduction.

Voilà donc le décor dans lequel doit agir l'administrateur d'un système informatique, un milieu fait de réseaux et de leurs composants, de stations de travail, de procédures de travail, de logiciels, d'utilisateurs et d'autres administrateurs, et de règles d'organisation. En fin de compte il n'est qu'un acteur au sein d'un système d'information particulier. Son action, l'administration d'un système distribué. D'une manière fort simple, il s'agit de l'ensemble des activités permettant de maintenir le système dans un état tel qu'il puisse assurer ses fonctions à la satisfaction des utilisateurs et des propriétaires. Ainsi les moyens d'administration seront autant d'actions concourant à la réalisation des plans établis par l'organisation dans l'espoir d'atteindre les objectifs qu'elle s'est fixée et

- garantissant une qualité de service aux utilisateurs telle que, par exemple, un temps de réponse convenable, une disponibilité des équipements, une sécurité et une fiabilité quels que soient les changements apportés au système;
- permettant une évolution du système;
- permettant une meilleure coopération avec d'autres systèmes.

Nous comprenons donc la maintenance dans un sens très large puisqu'elle vise tant à garder le système en bon état de marche qu'à garder les utilisateurs satisfaits dans la mesure du possible.

Pour l'aider dans sa tâche, l'administrateur trouvera une série d'outils répondant plus ou moins à ses besoins. Mais avant d'envisager le cas des outils, il nous semble judicieux de présenter les différentes facettes de l'administration. Après quoi il sera plus simple d'envisager les fonctions que l'on peut attendre d'une plate-forme d'aide à l'administration de systèmes distribués.

II.2. Des fonctions de l'administrateur.

On considère bien souvent, à tort, que seuls quatre groupes de tâches (configurations, sauvegardes, résolution des problèmes, gestion) sont du ressort de l'administrateur, et encore avec un certain nombre de restrictions. Et l'on oublie qu'il est aussi chargé de dépanner et d'aider les utilisateurs, de prévoir, d'organiser, d'automatiser, de former et de se former. En outre, l'administrateur a pour mission implicite la gestion même des activités de gestion.

Nous basant sur [Cardinael,1992], nous répertorions les fonctions de l'administrateur de la manière suivante:

- Fonction de préparation

  Cette fonction concerne le partage des ressources pour les travaux planifiables caractérisés par le fait que la réponse ne doit pas venir par terminal et peut prendre quelques instants. Elle peut se décomposer en trois temps.

Nous avons tout d'abord la documentation ou le contexte d'exécution qui définit les ressources utilisées et la mesure dans laquelle elles sont utilisées, les opérations interdites, la place du travail dans la file d'attente, le remède à appliquer au travail même en cas de problème (abandon, redémarrage,...) et le travail de dépannage à exécuter, et bien sur l'heure de démarrage (« au plus tard ») et la durée estimée du travail.
Vient ensuite l'exécution proprement dite pour terminer par la remise en état du système si besoin est.

- Fonctions d'exploitation.
  Elles ont pour objet l'organisation des ressources et la comptabilité de leur utilisation.

- Fonctions de gestion d'espace.
  Nous y logeons les fonctions de gestion d'espace proprement dite, de gestion des médias et de la disponibilité.

  Gestion d'espace:
  Régulièrement, l'administrateur procède à un nettoyage du système. Pour ce faire, il opère soit une suppression de fichiers inutiles et encombrant; soit un abandon d'autres fichiers sachant qu'il est possible de les récupérer par ailleurs; soit encore une migration vers des supports moins critiques que les disques rapides et autres.

  Gestion de la disponibilité:
  Il s'agit de pouvoir rendre disponible des données qu'une panne aurait endommagées. A cet effet, l'administrateur doit en réaliser des sauvegardes. Il s'agit donc de la constitution des sauvegardes et de leur utilisation.

  Gestion des médias:
  Une fois les sauvegardes effectuées, il faut encore gérer convenablement leur support afin de pouvoir les localiser et les restituer le moment venu. Cette activité porte donc aussi sur la gestion des étiquettes.

- Fonction de sécurité/d'audit interne
  Elle porte sur les attaques physiques, le contrôle des accès physiques (locaux, terminaux,...) et les accès logiques (informations, applications,...) ainsi que sur le respect des procédures.

- Fonction de gestion des informations.
  Par cette fonction, toute information circulant dans le système est identifiée et mise dans un format commun.

- Fonction de gestion de base de données.
  Cette fonction couvre également le contrôle des règles endogènes ou exogènes en matière de manipulation et de stockage des données (directives européennes,...).

- Fonction de gestion du personnel.
  Cette fonction n'est évidemment pas propre à l'administration de systèmes distribués mais elle n'en demeure pas moins nécessaire.

- Fonctions système.
  Cette fonction opère sur les cotés matériel et logiciel du système informatique. Du point de vue matériel, elle comprend la planification d'installation de nouveaux équipements et la gestion de l'existant et de ses performances. Par la lorgnette du logiciel, elle vise à l'installation, la paramétrisation, la correction des erreurs (*patchs*) et le suivi des versions des applications.

- Fonction d'aide aux utilisateurs
  Cela consiste en la résolution des problèmes de l'utilisateur, qu'il s'agisse d'un problème de configuration, d'application ou simplement d'une question. La formation de ces utilisateurs aux principes guidant la vie du système relève également de cette fonction.

- Fonction de développement
  Deux domaines sont couverts: l'entretien et le développement.
  Sont incluses dans l'entretien les modifications ponctuelles immédiates telles les ajouts de fonctionnalités, la mise à jour d'une application en fonction des lois, etc, tandis que les autres modifications sont du ressort d'un projet de développement.

Voilà donc un aperçu des tâches pour le moins complexes qui attendent notre infortuné administrateur. Il peut toutefois déléguer ses pouvoirs à d'autres personnes, opérant ainsi une découpe qui peut entraîner la formation d'une partition de ses activités.

II.3 Partionnement du système d'administration

Cette partition que nous allons proposer, reprise dans [Deghorain,1992], est basée sur une distribution des responsabilités qui peut prendre quatre formes: la structure organisationnelle, l'infrastructure physique, la structure des services et la structure liée à la sécurité.

- La structure organisationnelle
  La découpe est dans ce cas basée sur la hiérarchie sévissant dans l'organisation et, dans une certaine mesure, sur une hiérarchie de directives, terme par lequel nous désignons les plans de l'organisation pour parvenir à ses fins[1]. Grosso modo, cette hiérarchie va du niveau stratégique au niveau opérationnel et comme nous l'avons vu au premier chapitre, d'un état de formalisation embryonnaire voire inexistant à une formalisation totale. Cette précision aura son rôle dans la conception de la plate-forme d'aide à l'administration.

[1] Ce terme correspond au terme anglais *Policy*.

- L'infrastructure.

  La répartition se fait cette fois sur base de la recherche des groupes quasi-autonomes de composants au sein du ou des réseaux (composantes simplement connexes). Et le partitionnement du système se fait sur bases de ces composantes. Ceci restreint la zone de surveillance et permet une localisation des fautes et une intervention plus rapide.
  C'est ce modèle qui préside les décompositions de réseaux en *Network Operations Center* (ou NOC).

- La structure des services

  Une des propriétés des systèmes distribués est la mise en oeuvre du principe de modularité. Les services sont par là regroupés en modules à partir d'un critère de *cohésion logique et fonctionnelle* et de *couplage faible*. Ainsi pour chaque module, nous aurons une spécialisation de l'administrateur.

- La structure liée à la sécurité

  Le critère d'agrégation n'est plus ici fonctionnel mais concerne les autorisations d'accès. Le partitionnement des ressources se fera selon la classe d'utilisateurs qui y ont accès.

Quelle que soit la partition adoptée, il sera nécessaire de permettre les échanges entre les différents administrateurs.

Cette notion de partition rappelle la notion de domaines que l'on trouve dans [SLOMA89]. Un domaine est un groupe d'objets (dans le cas présent, des ressources) qui partagent certaines opérations de gestion ou qui présentent une cohésion que l'on veut exploiter, et constitué à la faveur d'une politique de gestion commune (*Policy*). Ces opérations ne concernent plus les objets individuellement mais le domaine, collectivement. Notons tout de même qu'il ne s'agit plus d'un partitionnement puisqu'il peut y avoir recouvrement de domaines.

II.4. Des plates-formes d'aide à l'administration.

II.4.1 Caractéristiques

Les caractéristiques principales d'une plate-forme d'administration peuvent être désignées en quelques mots clé précisant ce que l'architecture doit offrir:

- l'intégration de systèmes, services et réseaux hétérogènes pour une interoperabilité et une présentation unifiée

Une plate-forme d'aide à l'administration intégrée et distribuée (par la suite, nous simplifierons l'appellation au seul terme de *plate-forme*) est intéressante pour l'organisation dans la mesure où elle permet d'harmoniser la cacophonie de ses multiples applications de gestion. La solution pour mettre de l'ordre dans cette diversité est l'intégration, mécanisme peu docile qui peut être apprivoiser par trois approches [Deghorain,1992]:

- L'intégration par intégrateur

Elle consiste à ajouter un système englobant les autres outils de gestion.

- L'intégration par traducteurs
Cette méthode préconise la traduction des fonctions et des informations de gestion dans un format compréhensible par le système intégrateur

- L'intégration par standards
Elle se base l'utilisation de standards de sorte que les différentes applications de gestion n'ont ni à englober ni à traduire les autres applications mais peuvent coopérer en toute quiétude (Fig.II.1).

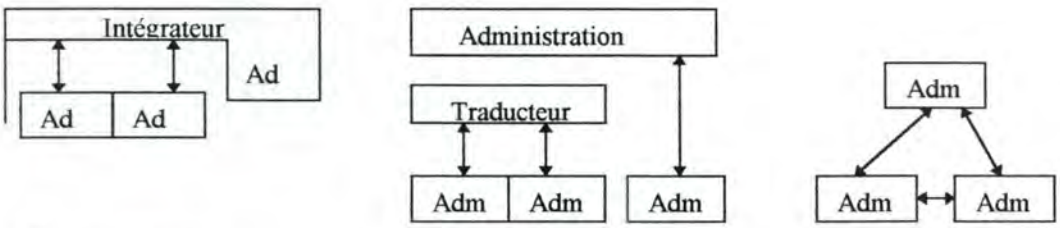


Fig. II.1 Les différentes approches d'intégration

- distribution des applications et des contrôles;
- dynamisme tant en termes de changement et de croissance qu'en termes de réponses aux besoins des administrateurs;
- l'ouverture pour communiquer avec d'autres systèmes et l'incorporation de composants externes;
- la conformité aux standards pour parfaire l'ouverture et l'intégration [IFIP;1991]

II.4.2. Conception d'une plate-forme

Une plate-forme d'administration est, dans un certain sens, un système d'aide à la décision c'est-à-dire un système composite[2] dont les agents sont des êtres humains ou des automates.

II.4.2.1 Architecture fonctionnelle

Comme nous le propose [Karemera,1993], son architecture regroupe les traitements en trois catégories: les modules de *Presentation logic*, de *Data logic* et de *Business logic*.

[2] [Dubois,1993]: un système fait de la combinaison de composants souvent hétérogènes, applelés agents, agissant en parallèle et coopérant vers un ou des buts assignés au système.

Presentation logic

- Service: gestion des entrées/sorties et du dialogue avec l'utilisateur.
  En matière d'interface homme-machine, une distinction peut être faite entre les interfaces construites sur la métaphore de la conversation et celles bâties sur la métaphore du mini-monde.
  Par la métaphore de la conversation, nous voyons l'utilisateur et le système échangeant des considérations à propos d'un monde non explicitement représenté dans l'interface, via un langage de commandes et d'autres structures linguistiques telles que des menus par exemple (dialogue conversationnel).

  Par la métaphore du mini-monde, l'interface constitue un monde dans lequel peut agir l'utilisateur en manuipulant les éléments qu'il y rencontre à l'aide de la souris ou autre tablette (manipulation directe).

- Traitements:
  «- les actions nécessaires à la communication avec l'utilisateur;
  - la gestion des éléments de l'interface;
  - l'enregistrement d'événements extérieurs (réception de messages de l'interface graphique, réception d'instructions);
  - la visualisation de l'état d'avancement de l'application et de ses éventuels problèmes techniques;
  - le contrôle syntaxique;
  - la vérification de la cohérence entre l'affichage des données et leur enregistrement dans la base de données.»

Data logic

- Service: manipulation des données.
- Traitement:
  - les actions primitives (de consultation, d'ajout, modification et suppression) sur la base de données;
  - les fonctions de l'application qui peuvent être considérés comme traitements élémentaires et non interactifs qui manipulent les informations de la base de données.

Business logic

- Service: décider de l'enchaînement des traitements de Presentation et Data logic pour atteindre l'objectif assigné à l'application.
- Traitement:
  - l'enregistrement d'événements provenant des autres modules de l'application;
  - la mémorisation de l'état de l'application, c'est à dire le résultat des traitements de P.L. et D.L.;
  - le déclenchement d'un traitement de P.L. ou D.L.
  Le comportement de la B.L. est en fait un ensemble de conditions-actions formant une table de décision.

La découpe modulaire entend répondre au principe double d'autonomie des fonctions d'application à l'égard de la conception et de la gestion de la base de donnée, et vis à vis de la conception et de l'exécution de l'interface homme-machine. Ceci met

en évidence la nécessité d'un dialogue entre modules que nous représentons dans l'architecture (Fig.II.2).

Les communications entre modules se font sur base de l'utilisation des services d'un module par un autre (relation d'utilisation). Une hiérarchie est ainsi définie dans laquelle

- les modules de niveau supérieur , c'est à dire de B.L., ne peuvent être appelés par les modules de P.L et de D.L.,
- les modules de P.L. et de D.L. utilisés par les B.L. peuvent se rendre service,
- les services peuvent être fournis entre modules d'un même niveau.

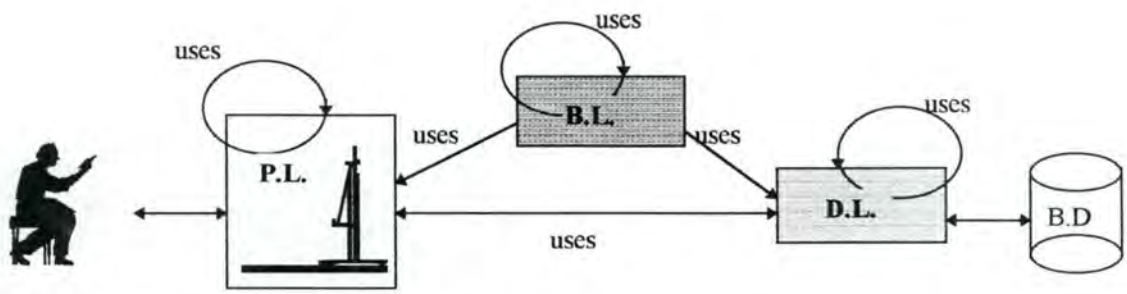


Fig. II.2 Architecture et Dialogue interne

II.4.2.2 Contrôle de l'application

Par ailleurs, le contrôle de l'application, pris en charge par la B.L., se caractérise par deux paramètres:

- son degré d'automatisation,
- le séquencement des opérations de l'application.

Le degré d'automatisation

Le contrôle d'une application peut être entièrement automatisé par la B.L., si l'ordonnancement des traitements de P.L. et D.L. est parfaitement planifiables.

Le séquencement des opérations

Par séquencement, nous entendons

- le contrôle séquentiel, s'il n'existe qu'une seule séquence possible des traitements.
- le contrôle multi-fils si plusieurs séquences sont possibles mais une seule exécutée à la fois non sans pouvoir être interrompue.
- le contrôle concurrent si l'exécution simultanée de plusieurs schémas est possible.

II.4.2.3 De la distribution de l'application

La distribution des catégories de module permet de définir trois classes d'architectures distribuées :

- Client/server processing caractérisé par la séparation des unités de Data logic du reste de l'application,
- Cooperative processing qui voit l'application divisée fonctionnellement en deux programmes exécutés sur deux stations différentes,
- Remote presentation processing qui consiste simplement à séparer les modules de Presentation Logic du reste de l'application.

Par cette distribution, il nous faut prendre en compte le dynamisme qu'implique le dialogue interne de l'application en créant une nouvelle catégorie de composants: les modules de communication. Il s'agit de traitements nécessaires à la communication entre agents matériels, et qui doivent être pris en charge au niveau de l'infrastructure du système par le biais de la couche de middleware. Ces liens entre application et infrastructure sont ceux qu'appelle explicitement le concepteur de l'application et non ceux destinés à la transparence de communication évoqués en point I.4.4. Ils se traduiront dans la réalité par les protocoles tels que X-Window, SNMP, CMIP,... (détaillés au chapitre III).

Par la suite, au chapitre V, nous appliquerons cette architecture à la plateforme d'administration ISM développée par le constructeur français Bull.

II.4.2.4 Architecture concrète.

Nous l'avons dit, les systèmes distribués sont rendus plus simple par l'adoption de l'approche orientée objet. Aussi ne nous étonnons pas que l'administration soit envisagée dans la même optique et passe dès lors par une gestion d'objets que nous définirons comme étant des entités formées de la conjonction d'informations et des fonctions opérant sur ces données. Par information, nous comprenons également des références à d'autres objets pour avoir la possibilité de gérer des domaines par exemple. Outre le fait d'une découpe cohérente, les objets permettent généralement une correspondance immédiate dans la programmation de l'application, qui ne dispense pas d'une découpe fonctionnelle préalable.

Notre découpe fonctionnelle peut alors être prolongée de la constitution de ces objets. Autrement dit, nous retrouverons les fonctions ayant trait aux différents modules dans chaque objet.

Modèle orienté objet

Nous proposons un modèle composé des trois éléments, un système administrateur qui échange des messages avec un agent en vue d'accéder aux informations que détient cet agent (Fig II.3).

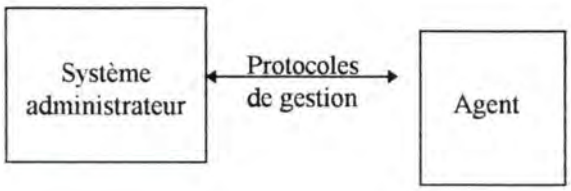


Fig. II.3 Modèle de gestion

Nous consacrerons le troisième chapitre à la présentation de protocoles de gestion. C'est sur eux en effet que repose la propagation efficace de l'information dans le système.

II.4.2.5 Autre perspective

Une plate-forme peut être vue de deux manières. Jusqu'à maintenant nous l'avons considérée comme un ensemble d'applications permettant d'aider un administrateur de système informatique. Mais elle constitue aussi une boîte à outils permettant de construire des applications sans avoir à se soucier des représentations propres aux divers environnements. Par ce moyen, le programmeur manipule les informations dont il a besoin dans un formalisme indépendant de tout système sous-jacent; et ce sont les outils qu'intègre la plate-forme qui prendront en charge la réalisation des méthodes d'accès, des représentations, etc.

Dans cette optique, la plate-forme est vue non plus comme une application mais comme un outil de middleware qui assure la transparence de communication entre l'administrateur et les ressources du système.

L'ISO propose que cet outil mette à la disposition des développeurs d'application les fonctions classées de la manière suivante (niveau 2):

- la gestion d'objets,
- la gestion des états qui permet de définir un état administratif (*shutting down*, *locked*, *unlocked*) et un état opérationnel (*disabled*, *enabled*, *active*, *busy*),
- la gestion des relations (Fig. III..6),
- la gestion des erreurs (*indeterminate*, *critical*, *major*, *minor*, *warning*),
- le contrôle des services de gestion,
- la gestion du contrôle des enregistrements d'événement,
- la gestion du schéma d'objets,

sur lesquelles s'appuierait d'autres développements répartis dans cinq aires (définies également par l'ISO, Niveau 1):

- la gestion des la configurations,
- la gestion de fautes,
- la gestion des performances,
- la gestion de la sécurité,
- la gestion de la comptabilité,

L'application (niveau 0) pourra, alors, utilisé une certaine combinaison de fonctions de gestion fournies par la plate-forme et une combinaison d'informations rendues accessibles (niveau 3).

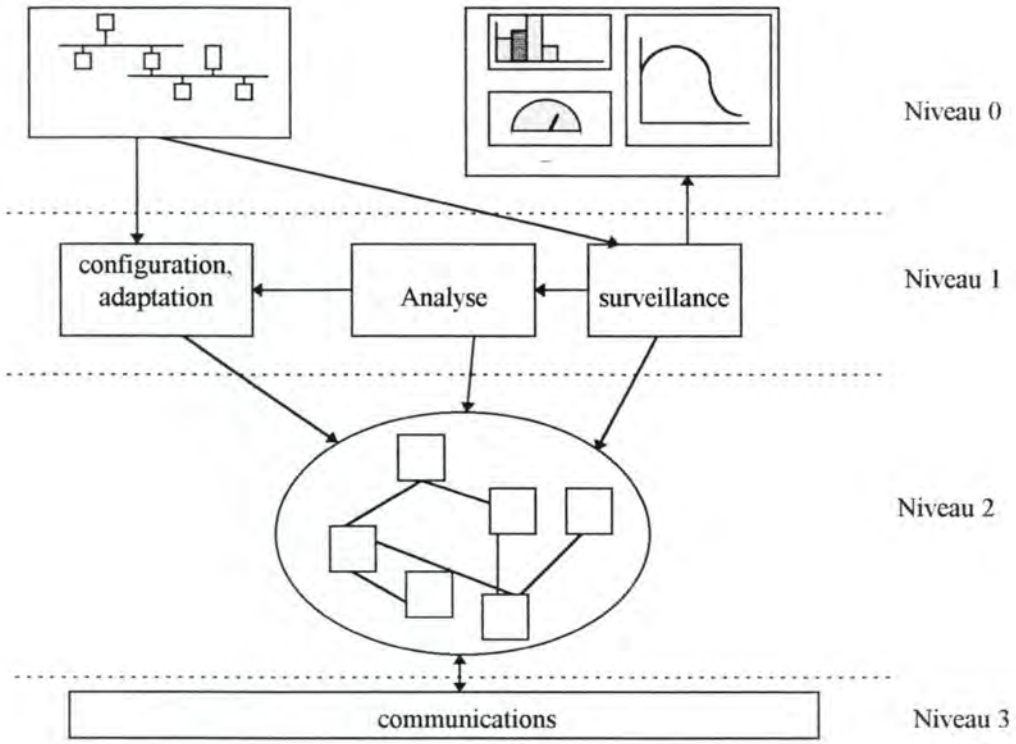
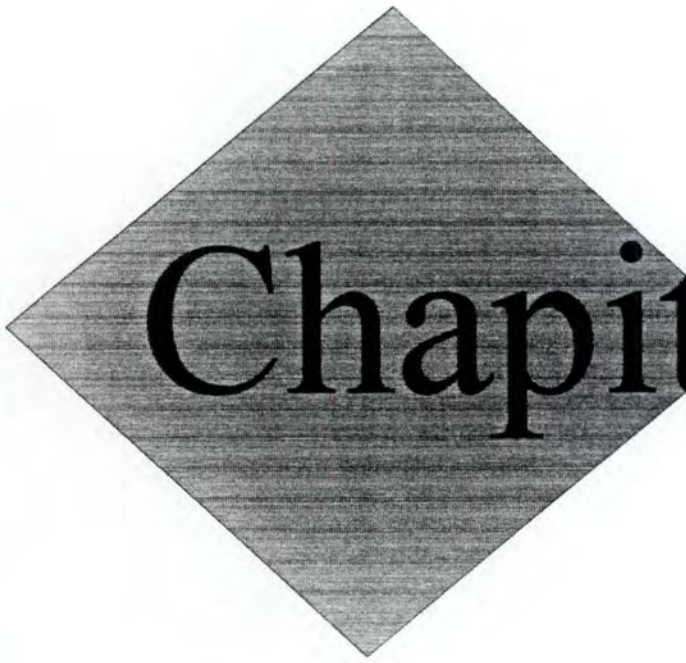


Fig. II.4

II.4.5. Remarques

Nous avons beaucoup parler des fonctions que la plate-forme pouvait offrir à l'administrateur. Mais nous avons vu qu'une des tâches d'administration est le développement du système qui comprend évidemment la plate-forme même. Aussi l'administrateur doit-il disposer d'un langage permettant par exemple d'ajouter une fonctionnalité ou d'étendre un agent.

C'est pourquoi nous distinguons trois types de langage en fonction de leur utilisation. Le langage de développement d'application qui fournit les mécanismes de représentation abstraite des informations,etc., le langage de commandes, destiné à l'élaboration de commandes plus facile d'usage pour l'utilisateur du système, et le langage d'extension de la plate-forme considérée comme outils de midlleware pour la réalisation de nouveaux outils (de nouveaux agents,etc.)..



Chapitre 3

Chapitre troisième : Aux confins de la Data Logic

III.1. Introduction

L'automatisation de l'administration est fortement liée au progrès en matière de modélisation, de structure et de représentation de l'information en général et à la description et l'implémentation des bases d'information en particulier. On ne peut en effet rien gérer sans un maximum d'informations. C'est pourquoi il nous faut définir la base sur laquelle pourront s'appuyer les tentatives de gestion.

Il existe bien entendu une très grande variété de protocoles de gestion mais beaucoup sont l'oeuvre et l'usage des seuls propriétaires. Deux standards *ouverts* ont toutefois émergé:

- La gestion Internet et
- La gestion OSI.

Ces deux standards spécifient deux protocoles pour transmettre des informations de gestion à travers une interface de communication. Les approches utilisées diffèrent quelque peu. La gestion OSI fut définie dans une optique de généralité tandis que Internet développait la sienne par essais jusqu'à l'obtention d'une administration de réseau efficace.

Ils furent tous deux imaginés pour permettre de possibles extensions sans que les utilisateurs du protocole aient à tout comprendre. De plus, chaque approche est bâtie sur un même concept: l'objet. Dans une telle approche, les opérations associées à l'objet sont exercées sur des informations qui représentent un élément du système à gérer. Chaque requête contient une liste de paramètres auxquels s'applique l'opération et peut entraîner une réponse.

Bien qu'ils utilisent le concept d'objet, ils ne le comprennent pas de la même manière. Voici un tableau qui résume la comparaison des deux sens.

Internet	ISO
MIB (Management information Base)	Bibliothèque de Définition des classes d'objets
Table	Classe d'objets
Elément de Table	Classe d'objets
Type d'objet	Attribut
Variables	Instanciation d'une classe d'objet

III.2 La gestion Internet

L'administration dans l'environnement TCP/IP est définie par trois composantes :
- la structure et l'identification des informations de gestion,
- le protocole d'échanges d'informations,
- la base d' informations de gestion.

Document	Sujet
RFC 1155	Structure et identification des informations de gestion
RFC 1157	SNMP
RFC1213	MIB-II

Parmi ces trois parties, le plus reconnu est SNMP parfois utilisé en référence de l'ensemble. Il faut avouer que c'est plus commode!

III.2.1 Le modèle de gestion Internet.

Le modèle est basé sur le paradigme du client/serveur où les serveurs, appelés Agents, possèdent les ressources, et les clients, appelés managers, les besoins. Les agents communiquent avec les managers en utilisant le protocole SNMP.

Les agents sont disposés à chaque noeud du réseau qu'il faut gérer comme les routeurs, les bridges, les serveurs de fichiers, etc. Tandis que le manager est une application qui fonctionne dans un centre de contrôle (NOC), par exemple.

Dans le modèle, toutes les opérations sont présentées comme modification (Set) ou consultation (Get) d'objets aussi appelés variables. La surveillance de l'état du système est principalement effectuée par *polling*, c'est à dire par envoi de requête d'interrogation aux agents bien que l'agent puisse, de manière fort limitée il est vrai, émettre des messages à l'adresse du manager (Traps). Aussi l'agent a-t-il le plus souvent un rôle passif.

III.2.2. Le protocole de gestion.

Il s'agit d'un protocole de Question/Réponse qui impose aux agents de se plier aux exigences des cinq messages de la figure 3.1.

Message	Description
Get (requête)	demande de consultation de variables
GetNext (requête)	consultation de variables dont on ne connaît pas le nom ou le nombre
Get (réponse)	réponse à un Get, Getnext et Set
Set (requête)	demande de modification de valeur de variable Cette opération n'aboutit que si toutes les variables ont pu être mises à jour

Trap	indication d'un changement: accès illicite,...

Un agent SNMP opère sur une communication sans connexion et non garanti: UDP/IP. Les agents interagissent avec le système sous-jacent pour obtenir et manipuler les informations. Lorsque l'agent émet une notification au manager, ce dernier a tout loisir de demander plus d'informations (Fig. III.3). SNMP décourage l'emploi des notifications mais n'y impose pas de restrictions. Il y a en effet plusieurs inconvénients à leur usage. Cela nécessite que les ressources informatiques produisent ces notifications qui peuvent parfois contenir beaucoup d'informations. Pendant ce temps, elles ne peuvent plus s'occuper de choses vraiment utiles! Le fait que le protocole soit non garanti limite la fiabilité des notifications. Et si par malheur il arrive qu'il y ait engorgement du réseau, une part non-négligeable de la capacité du réseau sera occupée par le convoyage de notifications. Il est possible de réduire la quantité de notifications en imposant l'usage de jauges et de limites de tolérance. Ce faisant, l'agent est obligé de maintenir ces informations supplémentaires et de vérifier que les limites ne sont pas atteintes ce qui se traduira, inévitablement, par un accroissement de la charge de la gestion dans le chef de la ressource gérée. D'autre part, le manager s'il veut avoir l'image la plus proche possible de l'état réelle du système doit utiliser le polling très régulièrement. Ce qui peut entraîner également une surcharge du réseau... La solution proposée est d'envoyer une notification toute simple et rapide lorsqu'un problème survient et de laisser le manager s'enquérir du problème.

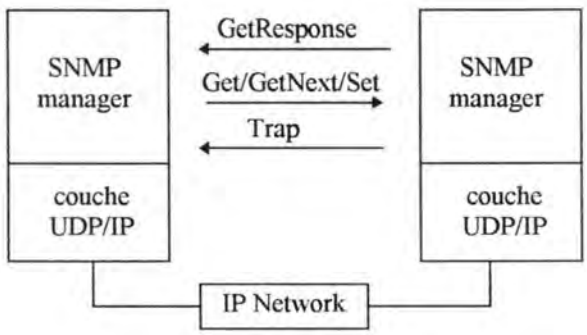


Fig. III.3 Les Rôles.

Nous signalions que l'usage d'un protocole non garanti entraînait une limitation de fiabilité. Cela est vrai dans la mesure où les applications ne prennent pas la gestion de la fiabilité du transport à leur compte.

III.2.3 Le modèle d'information de gestion.

La représentation et l'utilisation des informations ne sont pas du ressort du protocole mais plutôt du SMI (Structure of Management Information). Il fournit une méthode de définition des informations comme une base d'informations de gestion (MIB). Cette MIB est construite avec une partie des types de données ASN.1. Chaque élément de la MIB doit avoir un descripteur, un identificateur et une syntaxe. Les informations sont rassemblées en module par groupes d'intérêt.

Un autre standard connu sous le nom de MIB-II définit les informations pertinentes à la gestion. Les règles qui ont présidé à son élaboration sont le maintien de l'indépendance vis à vis du protocole, la définition d'un ensemble d'informations essentielles de gestion et la possibilité d'en ajouter au gré des entreprises. La séparation des informations d'avec le protocole permet de créer de nouvelles MIB sans avoir à modifier le protocole. Les groupes d'*objets*[1] de la MIB-II se présentent comme suit:

Système	Description, localisation, etc. des agents
Interfaces	Description, vitesse, etc. des interfaces (cfr. I.3)
Correspondance d'adresses	correspondance entre adresse réseau et adresse physique
IP	Datagrammes entrés, sortis et transmis.
ICMP (internet Control Message Protocol)	Statistiques concernant les entrées/sorties ICMP
TCP	Nombre de connexions maximum, ouvertes, comptabilité des entrées/sorties
UDP	Datagrammes entrés, sortis, erronés, etc.
Exterior Gateway Protocol (EGP)	Messages EGP reçus, erronés, etc.
Transmission	FDDI, Token-ring,etc
SNMP	Statistiques et comptabilité de l'agent SNMP

III.2.4 Les extensions.

SNMP, deuxième du nom (SNMPv2), réaménage le modèle administratif en trois composantes. Dans la première version, l'échange administratif était identifié par la seule communauté (*community*). Elle impliquait trois aspects:

- l'identification des entités autorisées à envoyer des requêtes,
- l'identification des opérations autorisées,
- l'identification des informations accessibles par ces opérations.

Aujourd'hui, la relation se base sur trois composants:

- Le concept de *Party*

  Cet environnement d'exécution, résident dans un agent, est associé à trois séries d'attributs: les attributs de transport, d'authentification et de secret. Ces attributs viennent sertir les informations qui font l'objet de la communication, de sorte que le message SNMPv2 se compose de trois structures imbriquées: SnmpPrivMsg, SnmpAuthMsg et SnmpMgmtCom suivant la figure 3.1.

[1] Le terme objet est ici utilisé pour désigner l'ensemble des informations qui ont trait à un objet réel. (voir exemple).

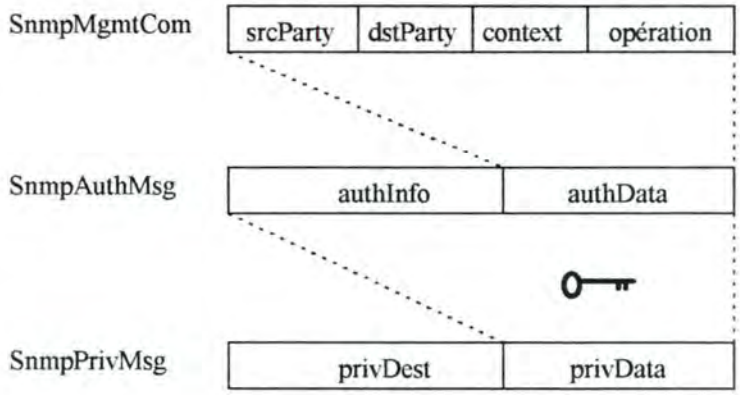


Fig. 3.1 Structures imbriquées

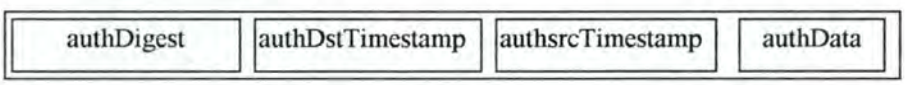
Les attributs se regroupent ainsi:

- les attributs de transport qui identifie le service de transport utilisé (normalement UDP), le protocole d'administration (SNMPv2), l'adresse du service de transport et la taille maximum des messages qui peuvent être reçus;

- les attributs d'authentification qui portent sur l' identification de l'origine du message, l'intégrité du message et la protection contre le *rejeu*.

partyAuthProtocol	identifie l'algorithme utilisé pour *sécuriser* le message
partyAuthClock	permet d'attache un timestamp aux messages
partyAuthPrivate	représente la clé privée utilisée par l'algorithme d'authentification
partyAuthPublic	représente la clé publique utilisée par l'algorithme d'authentification
partyAuthLifetime	précise la durée de validité du message

Le protocole d'authentification standard est le v2md5AuthProtocol. Le message SnmpAuthMsg généré est de la forme



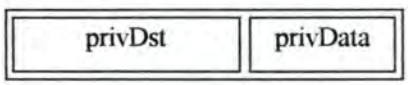
où le champ authData est le SnmpMgmtCom, contenant toutes les informations nécessaires à l'opération et l'opération elle-même, et où le champ authDigest est calculé avec l'algorithme MD5.

- les attributs de secret qui identifie l'algorithme utilisé pour chiffrer et déchiffrer les messages.

partyPrivProtocol	identifie l'algorithme utilisé par l'entité pour chiffrer et

	déchiffrer et les messages
partyPrivPrivate	la clé sécrète utilisée dans l'algorithme partyPrivProtocol
partyPrivPublic	la clé publique utilisée dans l'algorithme partyPrivProtocol

Le protocole de secret standard est le desPrivProtocol. Le message SnmpPrivMsg généré est de la forme



où le champ privData est le SnmpAutMsg éventuellement chiffré et où le champ privDst est le destinataire.

- du concept de *context* désignant une collection d'informations locales ou éloignées accessibles par l'entité. Selon qu'il faille pour les atteindre un mécanisme d'accès local ou non, on parlera respectivement de MIBview ou de relation proxi. (*proxi relationship*). La relation proxi permet par exemple de décharger une entité de la gestion des accès en n'autorisant les échanges qu'avec un certain proxi-agent. Et c'est cet agent qui s'occupent de la recevabilité des requêtes. Elle permet aussi d'opérer une conversion de protocole lors d'un échange entre deux membres de systèmes différents. Cette fonction fait partie la couche de Middleware dont nous avons évoqué l'usage au chapitre II.

- et du concept de *access policy* pour déterminer les opérations qui peuvent être exécutées dans un contexte donné. Ces *polices* comptent dans leur structure quatre champs principaux:
  - aclTarget, identifiant la partie destinataire,,
  - aclSubject, identifiant l'émetteur,
  - aclRessources, identifiant le *context*,
  - aclPrivileges, identifiant les opérations permises.

D'autres aménagements ont été réalisés dans le modèle opérationnel. SNMPv2 permet la communication entre managers par l'envoi d'une notification de type Inform, le type Trap étant réservé aux échanges agents-managers. Cette relation a pour but d'informer une autre partie de certains faits.

Il existe deux types d'avertissement: le groupe des alarmes et le groupe des événements.

Une alarme est peut-être caractérisée par
- snmpAlarmVariable: identifie l'*objet* qui doit être surveillé par échantillonnage
- snmpAlarmInterval: le nombre de secondes entre les prélèvements
- snmpAlarmSampleType: indique si les repères (Threshold) sont exprimés en valeur absolue ou relative (delta)
- snmpAlarmValue: donne la dernière valeur d'échantillon comparée aux repères
- snmpAlarmStartupAlarm: indique l'action à accomplir lorsqu'une valeur d'échantillon dépasse un repère
- snmpAlarmRisingThreshold: situe le seuil supérieur

- snmpAlarmFallingThreshold: situe le seuil inférieur
- snmpAlarmRisingEventIndex: indique une ligne dans la table snmpEventTable qui correspond à l'événement généré lorsque le repère supérieur est dépassé
- snmpAlarmFallingEventIndex: indique une ligne dans la table snmpEventTable qui correspond à l'événement généré lorsque la valeur d'échantillon est plus petite que le repère inférieur.
- snmpAlarmUnavailableEventIndex: identifie une ligne dans la table snmpEventTable qui correspond à l'événement généré lorsque la variable d'échantillonnage est devenue inaccessible.

Les trois dernières propriétés définissent l'objet des notifications entre managers.

Quant aux événements, leurs propriétés intéressantes, mémorisées dans la table snmpEventTable, sont les suivantes. Pour chaque événement,
- snmpEventID: identifie l'événement
- snmpEventDescription: donne une description de l'objet de la notification
- snmpEventEvents: précise le nombre de fois que cet événement à été déclenché
- snmpEventLastTimeSent: indique la valeur de sysUpTime (horloge du système) lorsque l'événement à été déclenché pour la dernière fois
-

Une deuxième table précise les applications de gestion susceptibles de recevoir une notification. Les colonnes intéressantes sont
- snmpEventNotifyIntervalRequested: l'intervalle de temps exprimé en secondes entre deux retransmissions
- snmpEventNotifyRetransmissionsRequested: le nombre de retransmission avant abandon
- snmpEventNotifyLifetime: le temps (en secondes) que cette entrée de la table restera active.

En outre, SNMPv2 ajoute la possibilité de rechercher un ensemble d'informations en une seule requête: l'instruction GetBulk.

Enfin, il met à la disposition de l'agent les moyens de nuancer ses réponses par l'ajout d'un code de retour. Les réponses sont alors de quatre types: sans erreur et sans exception, avec exception(s), avec erreur(s), avec TimeOut.

Un petit problème: les unités de données générées par les versions de SNMP sont incompatibles entre elles!

III.2.5 Exemple

Imaginons une application permettant la gestion de la configuration d'un système informatique c'est-à-dire une application permettant,entre autres, à l'administrateur d'interroger à partir de sa machine n'importe quelle autre machine du système distribué pour en connaître ses caractéristiques physiques (type de processeur, nombre de périphériques, etc.). Nous pouvons répartir les fonctions qu'elle comporte dans nos trois catégories de modules à savoir les modules de Business Logic(enchaînements des requêtes, des écrans,...), de Presentation Logic (la visualisation du réseau) et de Data logic. Quelle que soit la découpe que l'on opère, nous retrouverons les fonctions chargées des opérations sur les informations (disques, mémoires, processeurs,etc.)

relatives aux ressources dans la D.L. .Les développements que nous venons de faire, vont nous permettre de les raffiner quelque peu.

Nous prendrons l'exemple des disques. Chaque machine peut avoir aucun, un ou plusieurs disques durs dans sa configuration. Chaque disque possède un certain nombre de propriétés dont le nombre de têtes logiques ou physiques, de cylindres, de secteurs, d'octets par secteur, le nombre de pistes réservées à la landing zone, de secteurs défectueux, sa capacité maximale, sa capacité effective et par un nombre de partitions physiques. Celles-ci sont caractérisées par une taille, un type, un statut (active ou non).

Nous pensions donc organiser la structure de donnée de l'objet disque de la manière suivante.

Configuration
- Périphérique
  - Disques
    - Nombre maximum de disques
    - Nombre effectif de disques
    - Disque(NumDisk)
      - Nombre de têtes
      - Nombre de cylindres
      - Nombre de secteurs
      - Nombre d'octets par secteur
      - Nombre maximum de partitions
      - Nombre effectif de partitions
      - Partition(NumPartition)
        - Nom de l'OS gérant la partition
        - Statut de la partition
        - Taille de la partition
        - Offset de la partition
        - Nombre de cylindres réservés par le système
        - Nombre de divisions au sein de la partition
        - Division(NumDivision)
          - Adresse de début de la division
          - Longueur de la division
        - Nombre d'entrées maximum dans la table des secteurs défectueux
        - Nombre effectif de secteurs défectueux

Malheureusement, le schéma de structuration des informations prévu par SNMP ne reconnaît pas les tables de structures englobant d'autres tables. Aussi, avons-nous dû aplanir notre structure. De ce fait, l'accès au données se fait de manière bien moins naturelle. De plus, il faut prévoir l'ajout d'un index dans la table de partitions et de deux index dans la table des divisions. Notre bout de MIB devient alors la *chose* suivante.

Configuration
- Périphérique
  - Disques
    - Nombre maximum de disques
    - Nombre effectif de disques
    - Disque

Numero du Disque
Nombre de têtes
Nombre de cylindres
Nombre de secteurs
Nombre d'octets par secteur
Nombre maximum de partitions
Nombre effectif de partitions

Partition
Numero du disque
Numero de la partition
Nom de l'OS gérant la partition
Status de la partition
Taille de la partition
Offset de la partition
Nombre de cylindres réservés par le système
Nombre de divisions au sein de la partition
Nombre d'entrées maximum dans la table des secteurs défectueux
Nombre effectif de secteurs défectueux

Division
Numero du disque
Numero de la partition
Numero de la division
Adresse de début de la division
Longueur de la division

Cette limitation du nombre de niveau d'imbrication des tables a pour conséquence un allégement de l'agent qui, nous le rappelons, doit exister sur tous les noeuds du réseau mais qui ne peut pas gêner le fonctionnement de l'équipement. Ce qui donne la notation ASN.1 reprises aux pages 10 à 14 de ce chapitre.

Nous pouvons, ici, illustrer les réserves que nous émettions à l'égard de l'appellation « objet ». Lorsque nous présentons l'objet-disque, il faut entendre deux choses: les informations qui viennent d'être citées et les fonctions qui s'y rapportent. Ces dernières sont concentrées dans l'agent qui pourrait n'avoir en charge que la seule structure de disque mais qui, pour des raisons d'économie, traitent d'autres informations rassemblées au sein d'un seul groupe: la configuration. C'est pourquoi, s'il est conceptuellement possible de parler d'objets-disques, il n'est plus possible au stade de la réalisation d'en faire autant. Le seul objet qui reste est l'objet-configuration.

D'autre part, et nous le comprenons, c'est cet agent qui contiendra les fonctions de Data Logic expliquée au chapitre II tandis que les fonctions de communication, en fait les applications du protocole, qu'ajoute le *toolkit* utilisé pour confectionner l'agent, peuvent être rattachées aux modules de communication.

```
Bull        OBJECT IDENTIFIER ::= { entreprises 107 }
Bull_Mibs   OBJECT IDENTIFIER ::= { Bull 2 }
Bull_mibs_mib1 OBJECT IDENTIFIER ::= { Bull_mibs 1 }
Hw_Config   OBJECT IDENTIFIER ::= { Bull_mibs_mib1 1000 }
HwConfig    OBJECT-TYPE
        SYNTAX Hw_Config_Rec
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "configuration materielle"
        ::= { Hw_Config 1 }

Hw_Config_Rec  ::= SEQUENCE {
        CfMscSystem
          CfMsc_System,
        CfDevSystem
          CfDev_System,
        FileCfSystem
          CfFile_System
              }

CfDevSystem   OBJECT-TYPE
        SYNTAX CfDev_System,
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "configuration des devices"
        ::= { HwConfig 2 }

CfDev_System  ::= SEQUENCE {
        CfDiskDevice
          CfDisk_Device,
        CfFloppyDevice
          CfFloppy_Device,
        CfTapeDevice
          CfTape_Device,
        CfCommDevice
          CfComm_Device,
        CfVideoDevice
          CfVideo_Device,
        CfCdRomDevice
          CfCdRom_Device,
        CfCtrlDevice
          CfCtrl_Device
              }

CfDiskDevice  OBJECT-TYPE
        SYNTAX CfDisk_Device
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "ensemble de donnees concernant les disques"
        ::= { CfDevSystem 1 }

CfDisk_Device  ::= SEQUENCE {
        CfNbMaxDisk
          INTEGER,
        CfNbEffDisk
          INTEGER,
        CfDiskTable
          SEQUENCE OF CfDiskEntry
        CfDiskPart
          SEQUENCE OF CfPartEntry
        CfDiskDiv
          SEQUENCE OF CfDivEntry
              }


CfNbMaxDisk   OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre de disques actuellement supportes"
```

```
        ::= { CfDiskDevice 1 }

CfNbEffDisk    OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre de disques en usage actuellement"
        ::= { CfDiskDevice 2 }

CfDiskTable    OBJECT-TYPE
        SYNTAX SEQUENCE OF CfDiskEntry
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "description des disques durs : valeurs statiques / dynamiques"
        ::= { CfDiskDevice 3 }

CfDiskEntry    OBJECT-TYPE
        SYNTAX CfDisk_Entry
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "l'ensemble des donnees a propos d'un disque"
        INDEX { NumDisk }
        ::= { CfDiskTable 1 }

CfDisk_Entry  ::= SEQUENCE {
        CfNumDisk
          INTEGER,
        CfCylinders
          INTEGER,
        CfHeads
          INTEGER,
        CfSectors
          INTEGER,
        CfBytes
          INTEGER,
        CfMaxNbPart
          INTEGER,
        CfNbPart
          INTEGER
              }

CfDiskPart    OBJECT-TYPE
        SYNTAX SEQUENCE OF CfPartEntry
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "informations sur une partition d'un disque"
        ::= { CfDiskDevice 4 }

              }
CfDiskDiv     OBJECT-TYPE
        SYNTAX SEQUENCE OF CfDivEntry
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "informations sur une division d'une partition"
        ::= {CfDiskDevice 5 }

CfNumDisk      OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre de disques effectif"
        ::= { CfDiskEntry 1 }

CfCylinders    OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
```

```
        "nombre de cylindres sur le disque"
        ::= { CfDiskEntry 2 }

CfHeads      OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre de tetes pour le disque"
        ::= { CfDiskEntry 3 }

CfSectors    OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre de secteurs pour le disque"
        ::= { CfDiskEntry 4 }

CfBytes      OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre de bytes par secteur"
        ::= { CfDiskEntry 5 }

CfMaxNbPart  OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre maximum de partitions sur le disque"
        ::= { CfDiskEntry 6 }

CfNbPart     OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre effectif de partitions sur le disque"
        ::= { CfDiskEntry 7 }

CfPartEntry  OBJECT-TYPE
        SYNTAX partentry
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "description d'une partition"
        INDEX { CfPartDisk,CfPartNum }
        ::= { CfPartition 1 }

partentry    ::= SEQUENCE {
        CfPartDisk
          INTEGER,
        CfPartNum
          INTEGER,
        CfPartOs
          INTEGER,
        CfPartAct
          ENUMERATED { actif (1),
                passif (2) },
        CfPartSize
          INTEGER,
        CfPartOff
          INTEGER,
        CfPartRsv
          INTEGER,
        CfPartDivvyNb,
          INTEGER,
        CfPartMaxBadT
          INTEGER,
        CfPartNBadT
          INTEGER
```

```
                }

CfPartDisk    OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "numero du disque"
        ::= { CfPartEntry 1 }

CfPartNum     OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "numero de la partition"
        ::= { CfPartEntry 2 }

CfPartOs      OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "Type d'OS gerant cette partition"
        ::= { CfPartEntry 3 }

CfPartSize    OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "Taille de la partition"
        ::= { CfPartEntry 4 }

CfPartOff     OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "offset de la partition"
        ::= { CfPartEntry 5 }

CfPartRsv     OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre de cylindres reserves par le systeme"
        ::= { CfPartEntry 6 }

CfPartDivvyNb  OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre de divisions au sein de la partition"
        ::= { CfPartEntry 7 }

CfDivEntry    OBJECT-TYPE
        SYNTAX cfdiventry
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "ensemble des donnees relatives aux divisions"
        INDEX { CfDivDisk, CfDivPart, CfNumDivvy }
        ::= { CfDiskDiv 8 }

cfdiventry    ::= SEQUENCE {
        CfDivDisk
          INTEGER,
        CfDivPart
          INTEGER,
        CfNumDivvy
          INTEGER,
```

```
        CfDivvyOff
          INTEGER,
        CfDivvySize
          INTEGER,
              }

CfDivDisk     OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "numero du disque"
        ::= { CfDivEntry 1 }

CfDivPart     OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "numero de la partition"
        ::= { CfDivEntry 2 }

CfNumDivvy    OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "adresse de debut de division"
        ::= { CfDivvyEntry 3 }

CfDivvyOff    OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "adresse de debut de division"
        ::= { CfDivvyEntry 4 }

CfDivvySize   OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "longueur de la division"
        ::= { CfDivvyEntry 5 }

CfPartMaxBadT OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre d'entrees maximum dans la table des badtracks"
        ::= { CfPartEntry 8 }

CfPartNBadT   OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
        "nombre de mauvais cylindres"
        ::= { CfPartEntry 9 }
```

III.3. La Gestion OSI

La gestion OSI est basée sur les documents suivants:

Document	Sujet (donné en anglais pour en préserver le sens)
7498	Management Framework
10040	Systems Management Overviews
9595	Common Management Information Services (CMIS)
9596-1	Common Management Information Protocol (CMIP)
10165-1	Structure of Management Information (SMI)
10165-2	Definition of Management Information(DMI)
10165-4	Guidelines for the Definition of Managed Objects (GDMO)
10164-1	Object Management Function
10164-2	State Management Function
10164-3	Attributes for Representing Relationship
10164-4	Alarm Reporting Function
10164-5	Event Report Management Function
10164-6	Log Control Function
10164-7	Security Alarm Reporting Function
10164-8	Security Audit Trail Function

III.3.1. Le modèle de gestion ISO

Ce modèle utilise également le paradigme du client/serveur où le client est appelé système administrant (*managing system*) et le serveur, système administré (Managed system). Le système géré prend le rôle de l'agent, recevant des opérations destinées à administrer les *objets* et envoyant des notifications. Le système gestionnaire prend le rôle du manager , demandant l'exécution d'opérations de gestion et recevant les notifications. Le même système peut jouer tant au manager qu'à l'agent de sorte que les communications entre managers sont possibles par un simple changement de casquette.

Dans ce modèle également, les opérations sont exécutées sur des instanciations d'objets gérés (*Managed object instances*) et leurs propriétés. A l'instar d'Internet, la valeur des attributs sont consultés ou modifiés, et des messages d'avertissement peuvent arriver.

En général, les opérations Get, Set, Create, Delete, Action et CancelGet sont générées par le système « gérant » dans le rôle du manager et les réponses, par le système « géré » dans le rôle de l'agent. La notification d'événement est aussi envoyée par le système géré.

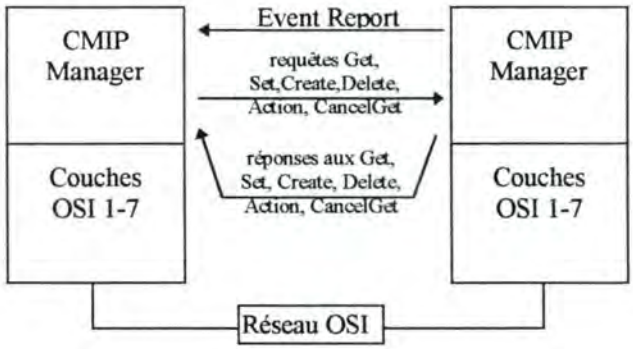


Fig. III.5 Les rôles

Un *Event Report* est une indication non demandée d'un événement important survenu dans le système géré. Il contient l'identité de l'événement et les informations nécessaires à la compréhension du message et peut être confirmé ou non. Le modèle conçoit l'administration dirigée par les événements. C'est pourquoi il fait un usage intensif de ces notifications. D'autres normes traitent des mécanismes de contrôle qui peuvent être utilisés pour détruire, enregistrer et faire suivre un événement à la discrétion du manager.

Le ciblage et le filtre permettent d'appliquer les opérations CMIP sur l'entièreté de la MIB. Ainsi tous les attributs qu'un objet géré présente au manager peuvent être saisis en une seule fois. Le filtre définit les conditions sur les attributs qui doivent être réalisées pour que l'opération soit exécutée. Le ciblage détermine, dans la hiérarchie des instances (Contemnent Hierarchy), la position de départ de l'opération. Lorsqu'une requête porte sur plusieurs objets gérés, l'opération peut engendrer plusieurs réponses. Par exemple, un Get d'attributs sur plusieurs objets gérés génère autant de réponses que d'objets gérés, chacune contenant les attributs d'une seule entité.

CMIP fournit aussi la possibilité d'une synchronisation rudimentaire en proposant une exécution de type Atomic (soit tout le monde peut répondre et répondent effectivement, soit un objet géré ou plus ne peuvent répondre et personne ne répond) et une exécution « Best-effort » (répondent les objets qui le peuvent).

Les spécifications de l'ISO comprennent également quelques fonctions de gestion de systèmes (SMF) s'appuyant sur les services offert par CMIS et CMIP:

- la gestion d'objets,
- la gestion des états qui permet de définir un état administratif (*shutting down*, *locked*, *unlocked*) et un état opérationnel (*disabled*, *enabled*, *active*, *busy*),
- la gestion des relations (Fig. III.6),
- la gestion des erreurs (*indeterminate*, *critical*, *major*, *minor*, *warning*),
- le contrôle des services de gestion,

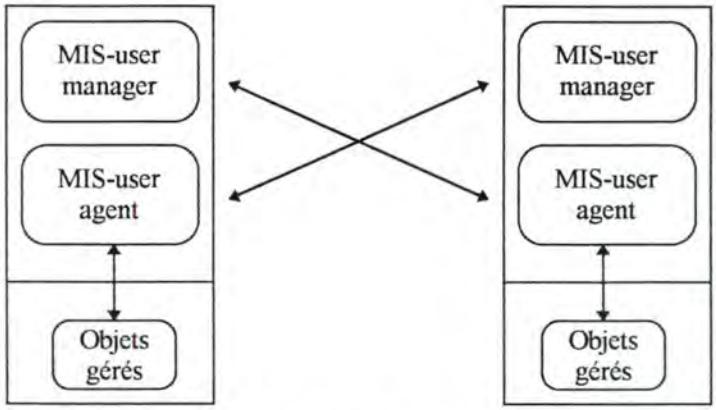


Fig. III.4. Les casquettes

III.3.2. Le protocole de gestion

Le protocole CMIP (Common Management Information Protocol) définit un plus grand éventail de messages que SNMP, et fournit des mécanismes de distribution de la charge entre agent et manager, pertmettant ainsi de réguler le trafic généré par l'administration. De plus, CMIP fait la différence entre les accès aux *objets* et aux attributs. Une norme associée, CMIS (Common Management Information Service) décrit les services abstraits qui sont réalisés à travers l'échange CMIP. La syntaxe des messages et des informations est définie en notation ASN.1. Voici les messages autorisés par CMIP.

Message	Description
Event Report	rend compte de la survenance d'un évenement à propos d'une ressource géré
Get	récupère l'information
Set	demande la modification d'informations
Action	demande l'exécution d'une action
Create	demande la création d'un nouvel objet géré
Delete	demande la destruction d'un objet géré
CancelGet	ordonner l'annulation d'un Get

La plupart des messages contiennent les paramètres suivants
- le type d'opération ou de notification,
- l'identité de la cible ou de la source (objets gérés),
- un *estampillage* facultatif (*timestamp*),
- de l'information concernant un objet ou une opération.

L'identification des objets gérés cibles se fait à partir de leur type, de leur nom, de leur position dans la hiérarchie et par un filtre faisant référence à leurs attributs.

Ces caractéristiques permettent au système administrateur de modifier une série d'objets en une seule opération et de sélectionner les objets à partir de leur état dynamique. Les instances d'objets sont organisés au sein d'une hiérarchie de contenance (*contenment hierarchy*) selon leur nom.

- la gestion de la configuration
    1. modifier les paramètres qi contrôle les opérations de routine du système,
    2. associer un nom à un objet géré,
    3. récolter, sur demande, des informations sur l'état du système,
    4. obtenir des notifications de changement d'état du système,
    5. changer la configuration du système,
- la gestion de la distribution d'application,
- la gestion de fautes
    1. enregistrer et entretenir les rapports d'erreur,
    2. générer des notifications d'erreur sur base de compteurs ou de jauges,
    3. agir à la moindre erreur,
    4. retrouver et identifier les erreurs par des tests de diagnostics, par exemple,
    5. prévenir les pannes en agissant sur base d'alertes,
- la gestion des performances
    1. rassembler les informations statistiques,
    2. examiner et entretenir les enregistrements concernant l'histoire du système,
    3. déterminer les performances dans des conditions réelles et artificielles
    4. modifier le mode de fonctionnement du système en vue de mener des tests,
    5. permettre la particularisation des rapports et des informations retenues,
- la gestion de la sécurité
    1. distribuer les informations utiles à la sécurité,
    2. notificier d'événements ayant trait à la sécurité,
    3. autoriser l'accès au système ou exclure selons des directives,
    4. authentifier et identifier les requêtes de gestion, en ce compris les accès aux informations sensibles par les utilisateurs,
- la gestion de la comptabilité
    1. informer les utilisateurs des ressources consommées et des coûts encourus,
    2. permettre l'élaboration et la mise en oeuvre de tarifs echelonnés en fonction de l'utilisation des ressources (tarif dégressif, etc.),
    3. permettre de centraliser les frais engendrés en différents endroits consécutivement à la réalisation d'un objectif commun,
    4. permettre d'effectuer une mise à jour des tarifs,
- la gestion du contrôle des enregistrements d'événement,
- la gestion du schéma d'objets.

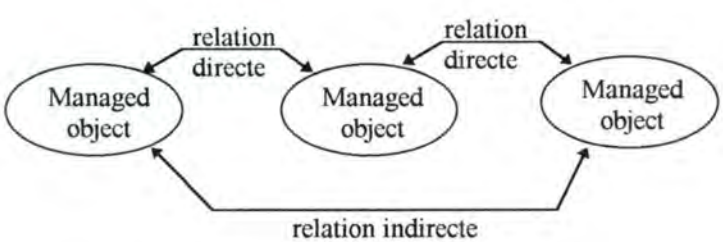


Fig. III.6 les types de relation

Ces fonctions utilisent le modèle d'information de gestion pour représenter les objets gérés. Ils offrent une vue standard de certaines tâches communes à plusieurs applications.

III.3.3. Le modèle d'information de gestion

Le modèle proposé représente le monde réel et les ressources comme des objets gérés caractérisés par

- les opérations qu'ils acceptent,
- les notifications qu'ils émettent,
- les attributs dont ils se composent,
- le comportement qu'ils ont adopté.

le GDMO (Guidelines for the Definition of Managed Objects) définit la notation utile à l'expression des propriétés observées. Les objets qui partagent des spécifications identiques sont regroupés en classes d'objet. De nouvelles classes peuvent être créées à partir de classes existantes en usant du mécanisme d'héritage. La description orientée objet fournit quatre fonctions clés.

Encapsulation	permet des accès aux données par des *méthodes* et cache la programmation interne
Structure de classe d'objet	offre la possibilité d'élargir et de combiner les classes existantes pour en former de nouvelles
héritage de classe	permet de définir une classe comme un raffinement d'une autre classe
Comportement allomorphique	procure un mécanisme de migration et de coexistence entre de multiples versions d'une définition de classe

La propriété d'allomorphisme permet à une même opération d'être appliquée à des classes d'objets différents. Chaque objet répondra de la façon déterminée dans une définition de *compatible class*. Ainsi, l'utilisation de fonctions communes sur de nouvelles classes d'objets ne nécessitera pas une nouvelle programmation.

III.4. Comparaison

Puisque l'utilisation et la conformité aux standards nous parait la solution la plus adéquate pour une plate-forme d'intégration, parce qu'elle permet une interopérabilité plus grande, il nous parait intéressant de procéder à une petite comparaison entre les deux environnements présentés plus haut. Et c'est en nous basant sur les travaux de l'OSI/NM-FORUM que nous tenterons de mettre en lumière leurs faiblesses et leurs points forts.

Les différents points abordés sont
- les conséquences en termes d'efficacité et de performances,
- la robustesse des solutions d'administration basées sur ces standards,
- la flexibilité et l'extensibilité inhérentes à chaque modèle,
- les avancées en matière de sécurité,
- les fonctions de base accessibles aux applications,
- des considérations de prix,
- les zones privilégiées d'application des approches.

III.4.1 L'efficacité

L'efficacité et les performances d'un système d'administration sont le résultat d'une combinaison de facteurs tels que
- la part de la capacité du réseau utilisée pour le trafic d'administration,
- la consommation en temps de calcul et en capacité de mémoire,
- la complexité et la taille du produit.

Ces facteurs sont influencés par certains aspects de la gestion Internet et ISO parmi lesquels l'impact du protocole sous-jacent, l'usage du polling contre une gestion dirigée par les événements, les opérations affectant de multiple objets.

Orienté polling ou Orienté événement

Tant SNMP que CMIP comportent des messages pour signaler des événements. La différence se fait au niveau de leur utilisation. Pour SNMP, la tendance est à l'utilisation du Polling. CMIP propose un mode de travail orienté événement.

Dans le modèle du polling, le manager est responsable des demandes d'information. Cette approche consomme une bonne partie de la capacité du réseau pour être efficace. Alors que le polling peut être insignifiant à petite échelle, la surchage qu'il impose augmente avec la taille du réseau, pour donner finalement des temps de réaction assez important sur les grands systèmes. Mais le polling est une méthode simple à programmer particulièrement du côté de l'agent. Généralement, cette méthode sera utilisée pour détecter non pas le problème mais plutôt le type de problème.

Le paradigme de l'orienté-événement, ce sont les agents qui sont responsables de faire connaître les informations pertinentes, que ce soit régulièrement ou ponctuellement. La surcharge de trafic peut être considérée moindre que celle générée par le polling dans de vastes environnements. Cette approche peut se révéler plus efficace dans un réseau plus qu'incertain. Mais des problèmes peuvent se poser, par exemple, lorsqu'une entité s'écrase lamentablement sans laisser de traces (notifications). De plus, elle implique une plus grande complexité du côté agent que le polling.

Les fonctions de gestion de l'ISO permettent de suspendre, de faire suivre et d'enregistrer les événements à la convenance du manager alors qu'il n'y a pas de mécanismes directs semblables dans SNMP.

Entre les deux approches, nous trouvons une méthode que nous avons déjà décrite: le polling dirigé par les notifications (Trap-directed polling). L'agent génère une notification à l'attention du manager qui se chargera d'en apprendre plus long par des opérations d'interrogation.

Opérations sur de multiples objets

Parfois certaines informations n'ont de sens que considérées au sein d'un ensemble. Pour récupérer cet ensemble nous pouvons procéder à une multiplication des requêtes portant chacune sur une information mais cela a pour conséquence un accroissement de trafic et une grande consommation de la capacité de traitement. Un remède à cela est de regrouper les demandes en une seule, contenue dans un message unique et portant sur plusieurs informations.

Parmi les situations dans lesquelles les opérations sur plusieurs informations peuvent être utiles sont

- demande concernant certains attributs sur le même objet,
- demande concernant des attributs d'objets différents mais très proches,
- demande concernant des attributs communs à différents objets.

Tant SNMP que CMIP fournissent ces mécanismes avec pour CMIP des options de filtre et de ciblage qui réduisent les informations nécessaires au strict minimum mais qui augmente la complexité de l'agent.

Impact du protocole sous-jacent

CMIP est conçu pour fonctionner sur une connexion tandis que SNMP utilise un service sans connexion. L'orienté-connexion nécessite une confirmation pour signaler la réception des messages dont l'ordre est préservé. La surcharge de trafic engendrée par l'établissement de la connexion est quelque peu compensée par une détection plus rapide de la non-réception du message ce qui permet une réaction plus efficace. C'est pourquoi certaines implémentations de SNMP comprennent des algorithmes d'optimisation de ces délais.

Par ailleurs, CMIP s'appuie sur une pile ISO complète ce qui entraîne, entre autres, une taille plus grande des unités de message passant par le réseau.

III.4.2. La robustesse

La livraison fiable

Nous avons dit dans le point précédent que CMIP avait été conçu pour utiliser une connexion qui assurait la fiabilité tandis que SNMP utilisait UDP pour l'envoi des requêtes, présumant que la seule réponse suffit comme signal de bonne arrivée.

Les auteurs de SNMP croient qu'en utilisant une communication sans connexion, ils parviendront à passer dans n'importe quel champ de mines que constitue les problèmes de réseau; tandis que les concepteurs de CMIP estiment qu'une connexion est le meilleur moyen d'accéder à une livraison fiable.

En fait, si une application ne nécessite pas une surveillance draconnienne de la fiabilité, autant qu'elle utilise SNMP, moins encombrant et plus efficace dans ce cas. Si par contre la fiabilité est primordiale, qu'elle utilise CMIP.

La synchronisation et l'atomicité

SNMP et CMIP ne fournissent pas de moyens de vraie synchronisation (Two-phase commit). Pas plus qu'il n'y a de synchronisation pour les opérations nécessitant l'envoi de plusieurs messages. De plus, lors d'opérations portant sur plusieurs objets, rien ne garantit que, durant la manoeuvre, il n'y aura pas de modifications extérieures sur les objets visés. Ce qui nous faudrait ici est le mode transactionnel entrevu dans le chapitre II.

Le niveau de décomposition des fonctions

CMIP définit des opérations telles que Create, Delete et Action. Elles peuvent introduites dans SNMP via l'instruction Set. En effet il suffit d'attacher le déclenchement d'une action à la modification d'un certain attribut. Ceci comporte le risque d'un déclenchement involontaire.

III.4.3. Flexibilité et extensibilité

Modélisation de l'information

les différences principales peuvent être résumées dans le tableau suivant.

Sujets	ISO/CCITT	Internet
type de syntaxe d'attribut/variable	constructions simples ou complexes	constructions simples
réutilisabilité de la syntaxe	syntaxe réutilisable entre attributs	syntaxe unique
Option de variable/attribut	les attributs optionnels et obligatoires peuvent coexister au sein d'un objet. De plus l'option est dynamqie et statique	les attributs obligatoires et optionnels ne peuvent cohabiter une même table et l'option est statique seulement
Opérations Create,Delete,Action	définies comme des propriétés des objets gérés.	ces concept n'existent pas.

Opérations Trap/Notification	définies comme des propriétés des objets gérés.	intégrées depuis SNMPv2 dans la MIB
Relation d'héritage	l'héritage est utilisé pour raffiner et étendre la définition d'objet	ces concept n'existent pas.
Containment Relationship	Le containment est utilisé pour organiser les objets gérés par une relation de dépendance.	ces concept n'existent pas.
Style de structure de MIB	Utilise l'orienté objet et est hiérarchique par nature	Utilise un ordre lexicographique et n'admet que les structures plates

III.4.4. Sécurité

Nous avons déjà abordé le cas de SNMP. Quant à CMIP, il fait appel au service de la couche application pour l'authentification des entités communicantes.

III.4.5. Fonctions d'application

Tant l'ISO que l'Internet ont fourni des fonctions pouvant servir de base aux applications. Toutefois, Internet, désireux de garder ses agents les plus simples possibles, en a délégué beaucoup aux programmeurs d'application.

III.4.6. Les considérations de coût

Les coûts de développement

Bien sûr, la conception des agents SNMP est relativement simple. Mais cette simplicité se paie par une plus grande difficulté dans l'élaboration des applications basées sur ce protocole. Elles doivent en effet prendre en charge tout ce dont SNMP n'a pas voulu si elle veulent arriver à un niveau égal aux applications développées sur CMIP.

Les coûts de déploiement

Il existe évidemment beaucoup de produits développés pour SNMP (première version) tandis que ceux construits sur base de CMIP sont beaucoup plus rares. Dès lors les coûts d'acquisition d'une base SNMP seront-ils moindre que ceux d'une base CMIP.

III.4.7 Les domaines d'application des deux approches

Voici quelques exemples d'applications.

Les applications qui travaillent sur base du polling et qui n'exigent pas une notification d'erreur, utiliseront plus efficacement SNMP. Nous pourrions imaginer une application qui réaliserait une simulation de l'évolution du système, de résistance à la charge, etc. sur base des informations réelles: taux d'erreurs du réseau, etc.;

Les applications comme la surveillance et la répartition de la charge, la gestion des tests, etc. seraient sans doute plus avantageusement programmées sur les services de CMIP;

Les applications de visualisation de la topographie et de la topologie du réseau pourraient utiliser indifféremment les deux approches.

D'un point de vue purement pratique, il est évident que des agents basés sur SNMP seront mieux gérés en utilisant les standards Internet, comme les agents CMIP selon les normes ISO. Ceci pour mettre en évidence le problème de coexistence des deux standards et de tout autre d'ailleurs.

III.4.8 La cohabitation

Cette coexistence est prise en compte par plusieurs méthodes.

III.4.8.1 Les piles mixtes

Les piles de protocole mixtes parmi lesquelles CMIP For The Internet, CMIP Over LLC, SNMP OVER OSI.

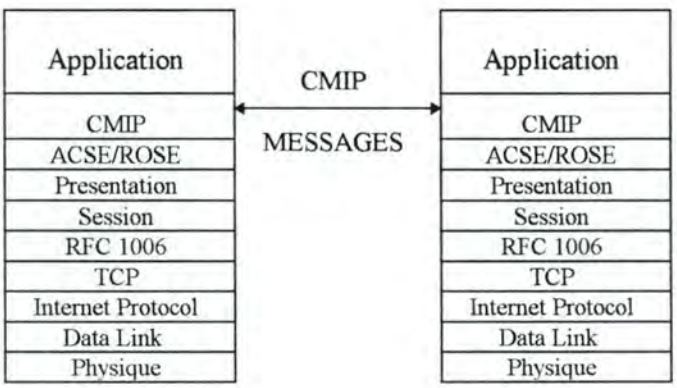


Fig. III.7 CMIP For The Intenet

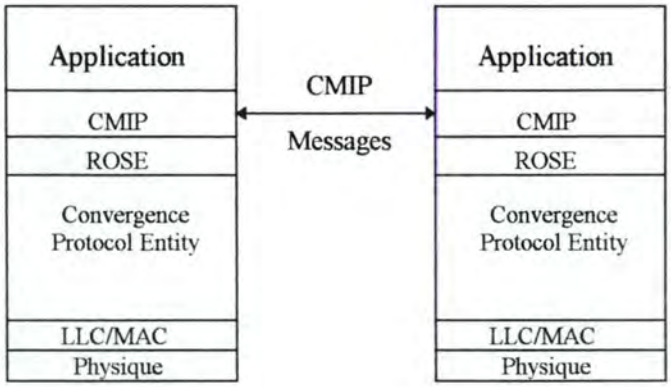


Fig. III.8 CMIP Over LLC

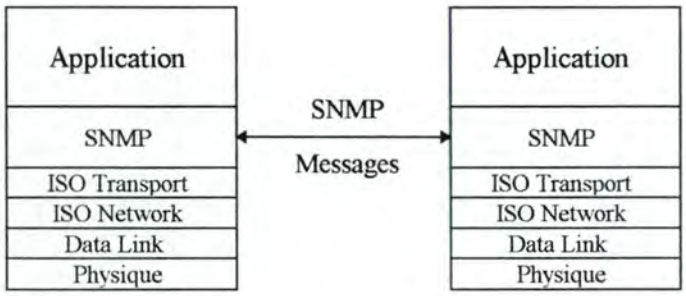


Fig. III.9 SNMP Over OSI

III.4.8.2 les piles duales de protocole

Cette approche demande l'existence côte à côte des piles OSI et TCP/IP dans la même entité afin de permettre l'interconnexion entre deux processus de gestion. Les deux piles partagent les deux premières couches qui porteront aussi bien les messages réseau d'OSI que les messages IP d'Internet.

Une stratégie serait de concevoir le manager sur une telle pile alors que l'agent évoluerait dans son environnement *natif*.

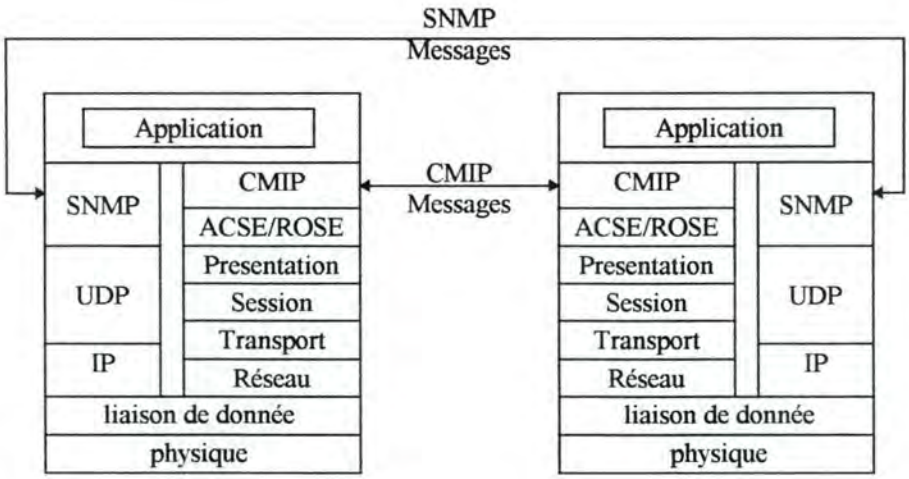


Fig. III.9 Piles duales de protocole

III.4.8.3 Les Interfaces communes de programmation d'application (API)

Une extension aux piles duales de protocole est l'utilisation d'API pour aider les concepteurs d'application dans leur développement dans cet environnement. Ces API fournissent un ensemble consistant de structures de données et de verbes remplaçant les interfaces propriètaires de chaque pile.

Plusieurs niveaux dans le développement de cette optique sont observés.

Par exemple, XMP propose une interface commune à SNMP et à CMIP en définissant un ensemble de fonctions qu'une application peut utiliser pour envoyer ou recevoir des requêtes des deux bords. Ce sont les mêmes fonctions qui sont utilisées pour les services communs aux deux protocoles, dont les paramètres sont de manières standards (X/Open OSI-Abstract Data Manipulation ou XOM). Mais ces APIs communes de ce niveau n'assure pas la transparence de protocole ni une véritable intégration de service. Les développeurs doivent toujours avoir à l'esprit le modèle d'information utilisé et les différences de service, nommage et gestion d'erreur. Une solution est de limiter l'usage des spécificités de chaque standard ce qui entraîne une réduction des fonctionnalités offertes à un dénominateur commun.

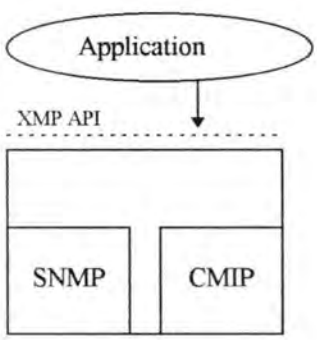


Fig. III.10

De plus en plus, sont étudiés des interfaces abstraites qui permettent une plus grande indépendance de l'application par rapport aux protocoles de gestion. Tel est le cas de OMG/CORBA, sur lequel se base les environnements de gestion tels que OSF-DME. CORBA définit un *instrumentation request broker* par lequel les clients ont la possibilité d'envoyer des requêtes à un serveur modélisé par un objet qui se charge de la répartition. L'interface pour chaque objet est spécifiée dans une notation appelée IDL (*Interface Definition Language*). Simplifiée, l'interface peut être offerte par les objets pour cacher la complexité des procédures auxquelles elle fait appel. Ces API abstraites ont aussi à établir un équilibre entre fonctionnalité et indépendance vis à vis du protocole.

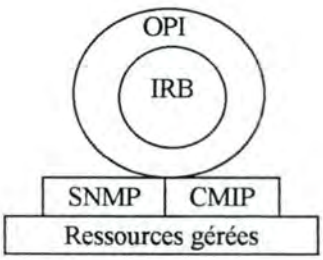
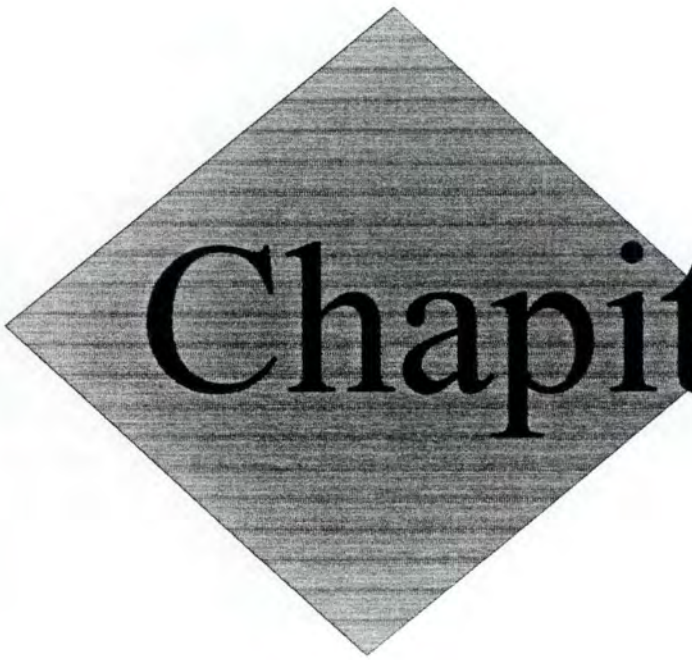


Fig. III.11

III.4.8.4 L'intégration *Pass-Through*

L'intégration est rejetée au niveau application. et peut se faire selon les différentes approches citées au chapitre II.



Chapitre 4

Chapitre quatrième: Evaluation des systèmes d'aide à l'administration.

IV.1. Introduction.

Le concept de système intégré d'aide à l'administration étant encore dans sa genèse, il n'existe que peu de méthodologies d'évaluation des possibilités de ces systèmes. C'est pourquoi nous nous sommes limités à la définition d'une liste de critères d'analyse. Ce ne peut être qu'une série de lignes directrices puisqu'en définitive, la décision d'opter pour telle ou telle plate-forme reposera sur les caractéristiques du système à administrer. A cette fin, il est impératif d'y avoir mené une véritable étude de l'existant.

La liste est divisée en trois parties: les critères fonctionnels qui concernent le support des fonctions développées au chapitre II, les critères opérationnels axés sur les caractéristiques facilitant l'utilisation du système par les opérateurs, et divers autres critères ayant trait à l'utilisation d'une application en général.

IV.2. Des critères fonctionnels

Parce qu'ils jouent un rôle important dans la réalisation de la transparence, de l'interopérabilité et de l'intégration des systèmes différents, il est naturel que les standards apparaissent dans cette partie. En outre les plates-formes devraient couvrir les aires fonctionnelles envisagées également au chapitre II.
Ces critères ont pour objet de **rendre compte de l'adéquation des possibilités des plates-formes avec l'environnement de leurs utilisateurs**.

- Support des standards de représentation et de manipulation des informations de gestion, qu'il s'agisse de standards de gestion selon OSI ou Internet.
- Prise en compte des fonctions de gestion des configurations
- Prise en compte des fonctions de gestion des performances
- Prise en compte des fonctions de gestion des fautes
- Prise en compte des fonctions de gestion sécurité
- Prise en compte des fonctions de gestion de la comptabilité
- Prise en compte des fonctions de gestion de l'aide
- Prise en compte des fonctions de gestion de développement
- Possibilité d'intégrer ou d'utiliser d'autres outils d'aide à l'administration

IV.3. Des critères opérationnels.

Ces critères se concentrent principalement **sur l'installation et l'utilisation des plates-formes**. Ils fournissent un moyen de juger de leur bienséance dans l'environnement de leurs utilisateurs.

- Aisance d'installation.

Dès l'instant où les plates-formes doivent être installées par leurs futurs utilisateurs et non plus par la firme qui les distribue, ce qui se traduit par le fait que l'utilisateur est le principal responsable de l'installation, l'aisance d'installation revêt une grande importance.
Les moyens de rendre plus facile l'installation sont par exemple la présence d'un logiciel d'installation automatique permettant une paramétrisation par l'opérateur, une optimisation automatique, etc. sans que l'on puisse se passer définitivement d'un manuel cohérent et structuré d'installation.

- Aisance d'utilisation.
  Ce critère prend en compte la facilité avec laquelle l'opérateur peut passer d'une fonction à une autre et peut les exécuter ainsi que l'existence d'un manuel complet et cohérent d'utilisation. Il connaît également du fait qu'il faut des accès privilégiés au système pour faire usage de la plate-forme, en tout ou en partie. Ce dernier point permet ou au contraire interdit la délégation de fonctions, parfois intéressantes, aux utilisateurs au système.

- Compatibilité avec système existant
  La plate-forme doit pouvoir s'intégrer à l'environnement préexistant sans grands changements dans ce dernier. Notons que l'importance de l'administration n'est pas très ancienne. Aussi ne sera-t-il pas étonnant de trouver des équipements ne permettant pas de gestion. Cet état de chose évolue, cependant; avec les nécessités et de plus en plus le matériel offre les moyens de les administrer (carte SNMP,...).

- Interface Homme/machine.
  Dans ce point de vue, nous prenons en compte l'intelligibilité des informations fournies et des choix proposés que ce soit dans leurs présentations ou dans leurs représentations.
  Ainsi la présentation des informations de la MIB peut-elle s'accompagner d'une explication sur leurs significations, sur l'unité dans laquelle elles s'expriment, etc. Peut aussi être précisé le fait qu'elles sont calculées sur un intervalle ou en temps absolu.
  La représentation graphique du système est aussi un point important. Nous pouvons en attendre les caractéristiques suivantes:
    - ◊ la possibilité d'avoir une vue hiérarchique du système administré;
    - ◊ la possibilité de représenter les ressources non-administrables;
    - ◊ la possibilité d'obtenir une carte logique (position au sein d'un réseau,...) et physique (localisation des ressources) du système;
    - ◊ la possibilité de personnaliser la représentation (icônes,...),
    - ◊ la possiibilité d'obtenir une aide pertinente pour chaque fenêtre,
    - ◊ une limitation du nombre d'opérations clavier et souris
    - ◊ une conformité avec les conventions spécifiéees par OSF/Motif.

  Les choix, quant à eux, peuvent être proposés sous forme de listes, menu déroulants,...
  En tout cas, les propriétés que l'on pourrait en attendre sont:

- Performances.
  Ces performances peuvent être définies comme la capacité du système à traiter un grand nombre de questions et de réponses de manière acceptable. Ce qui veut dire que la plate-forme doit pouvoir opérer sans dégrader les performances du système pour les autres utilisateurs (dégradation du temps de réponse, ...). Elles traduisent également le fait que la plate-forme doit utiliser la part du système la plus petite possible pour les activités de gestion.

- Fiabilité.
  Puisqu'une plate-forme d'administration doit fournir les moyens de contrôler les opérations du système, il est impératif que la fiabilité soit établie comme un critère à part entière. La fiabilité concerne aussi bien le produit lui-même que l'effet produit sur le système dans son ensemble.

- Sécurité.
  Le critère opérationnel de sécurité est centré sur le contrôle d'accès et l'identification dans l'usage de la plate-forme et des informations de gestion, et sur la sécurité des communications entre opérateurs (humains ou logiciels).
  Différents niveaux d'accès sont souhaitables en fonction des différentes classes d'utilisateurs qui pourraient avoir besoin des informations de gestion.

- Adaptation à la taille du système.
  Ce critère fait référence aux possibilités d'utilisation dans des systèmes de différentes tailles. Cela implique l'aptitude à manipuler une très grande quantité d'informations, à représenter un système plus grand, à fournir à l'administrateur la possibilité de gérer de multiples domaines d'administration qu'ils soient définis sur base de la structure géographique, organisationnelle ou fonctionnelles. De cette dernière aptitude découle la prise en charge des interactions administrateur-à-administrateur.
  Par ce critère, nous en appelons à une rationalisation de la plate-forme. En effet il est peu probable que pour un système donné, toutes les fonctions assurées par la plate-forme soient nécessaires.

IV.4. Autres critères.

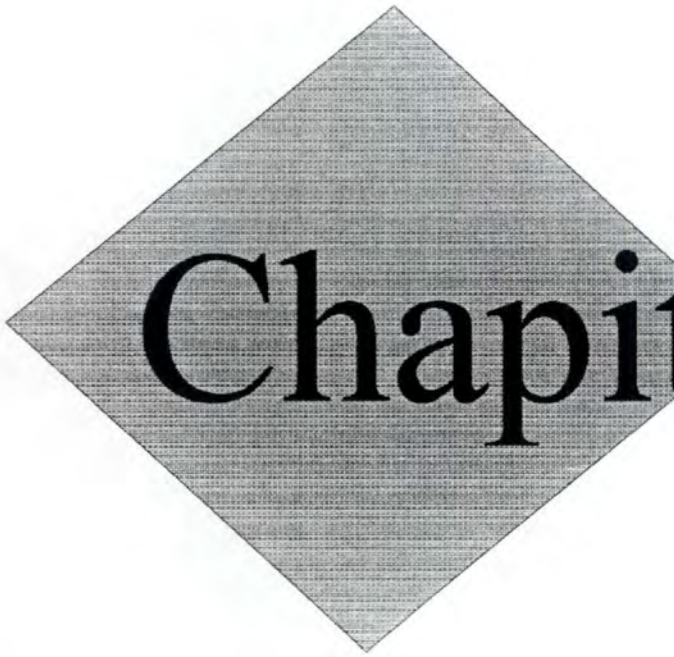
- Documentation.
  Comme nous l'avons dit plus haut, une documentation d'installation et d'utilisation est essentielle. A ces deux types de documentation peut s'ajouter une description de l'architecture de la plate-forme.

- Niveau de qualification de l'opérateur.
  Le niveau de compétence des administrateurs varie d'organisation en organisation. C'est pourquoi le choix d'une plate-forme doit tenir compte des exigences de cette plate-forme en matière de qualifications de ses utilisateurs. Voici quelques questions que l'on devrait se poser: A quel type d'utilisateur la plate-forme est-elle destinée ? Quels sont les moyens existants pour adapter la plate-forme aux autres types d'utilisateurs (mise en place de filtres,...) ? Quelles sont les informations à donner à ou reçues de la plate-forme ?

- Coûts.
  Il y a bien sûr les coûts de logiciel de la plate-forme (managers et agents) mais aussi les coûts de maintenance et des licences.

- Le support-client du distributeur.
  Un point important est le service après-vente. Ce service devrait inclure assistance à l'installation, la correction des erreurs, le suivi des nouvelles versions,... En fait il s'agit de déterminer le *sérieux* du distributeur.

- L'expérimentation.
  L'expérimentation ou l'utilisation en d'autres sites et couvrant une variété plus ou moins grande de systèmes peuvent également être considéré comme un critère de sélection. Il prouverait par là son efficacité et sa faculté d'adaptation.



Chapitre 5

Chapitre cinquième: Etude de cas

Nous procéderons dans ce dernier chapitre à l'étude d'un cas particulier: la plate-forme ISM (Integrated System Management) de Bull. Nous la considérerons tout d'abord comme une application distribuée pour ensuite la voir comme une boîte à outils (couche de *middleware*), idée que nous avions évoquée au chapitre.

V.1 Présentation de l'application distribuée

De plus en plus, les constructeurs proposent des solutions pour administrer des systèmes distribués. Bull s'est placé dans cette mouvance par sa proposition d'une plate-forme intégrée d'aide à l'administration. Elle permet de gérer les différents éléments des systèmes (systèmes d'exploitation, utilisateurs, réseaux, applications, etc) de manière transparente.

L'architecture de ce produit est conforme aux normes édictées par l'ISO et aux travaux effectués par OSI/NM-FORUM.

Ainsi nous retrouvons les notions d'administrateur et d'agent concrétisées dans l'ISM Manager et les ISM agents. Leur dialogue se fait conformément aux protocoles de gestion.

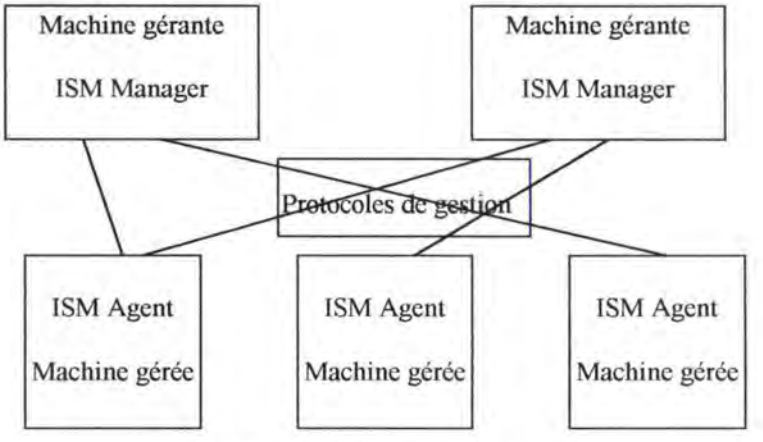


Fig. V.1 Architecture

V.1.1 ISM Manager

Nous l'avons vu, un système hétérogène peut contenir un grand nombre d'agents utilisant des protocoles différents (répartis par exemple dans des domaines différents). ISM doit donc pouvoir prendre en compte ces différences et les gérer tout en présentant une interface uniforme dans la mesure du possible.

Aussi, pour satisfaire à ces exigences, l'ISM Manager est décomposé en niveaux. Le plus bas, l'agent intégrateur ou AI, contient le programme spécifique à un protocole. Le plus élevé contient les applications construites le plus indépendamment des couches inférieures. Les protocoles actuellement intégrés sont SNMP, CMIP et DSAC (un protocole développé pour l'environnement propriétaire de Bull).

D'autre part, les applications d'ISM Manager voient tous les sous-systèmes à travers une seule base d'information (MIB). Dans le cas où le protocole sous-jacent utilise une approche orienté-objet, les objets appartenant à la MIB attachée au protocole seront visibles dans la MIB ISM. Celle-ci contiendra donc un ensemble formé par l'agglomération des ensembles de classes d'objets de chaque sous-système. Dans le cas contraire, ce sera à l'agent intégrateur du sous-système à créer les objets gérés pour représenter les ressources réelles.

La structure du Manager peut s'exprimer par le schéma suivant. C'est à lui que nous allons appliquer notre découpe en modules de Presentation Logic, Data Logic et Business Logic.

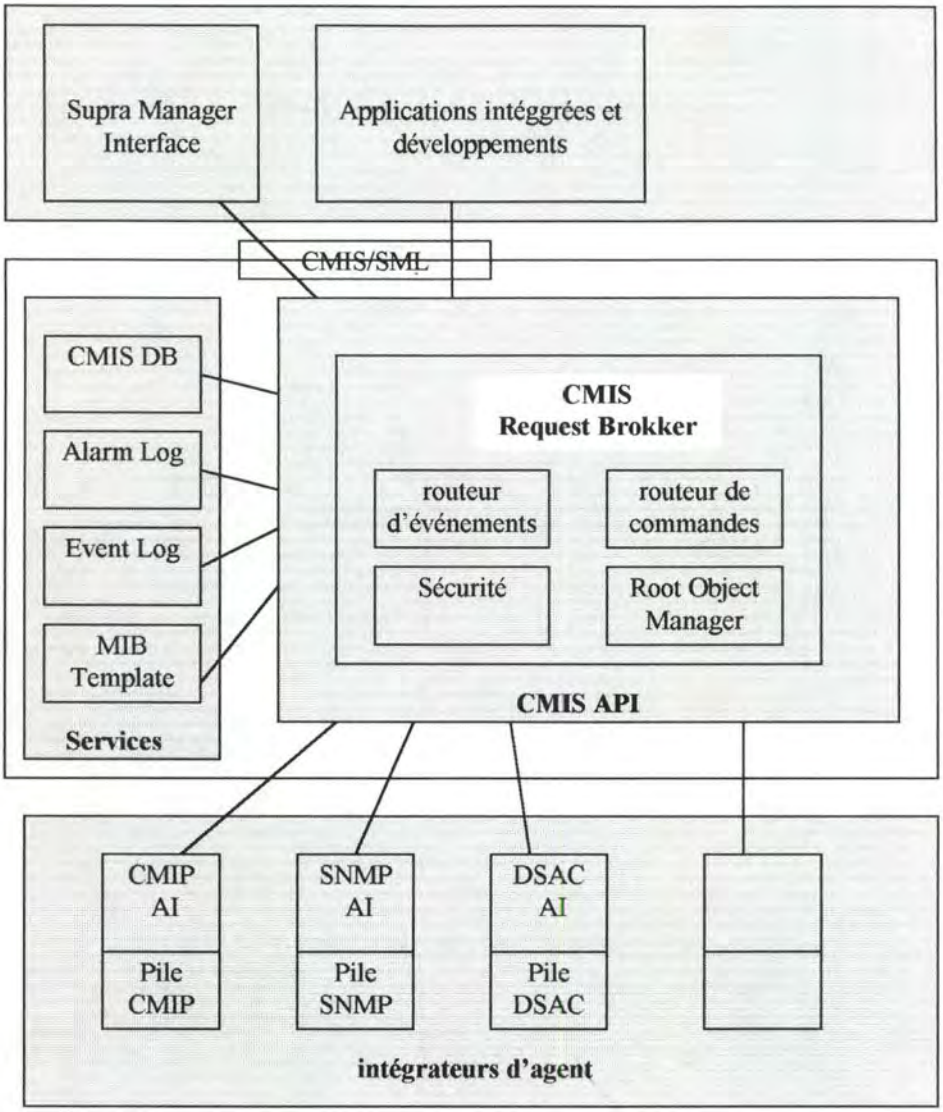


Fig. V.2 Architecture interne

Les différentes composantes du manager sont les applications, les agents intégrateurs, les distributeurs de requêtes (*CMIS Request Broker*), les services et l'interface de Supra-Manager. Nous passerons en revue tout ce joli monde après avoir vu la notion de MIB dans ISM.

V.1.2 MIB ISM

Le schéma de la MIB ISM (ISM MIB Template) contient toutes les classes d'objets gérés nécessaires à la gestion d'un système distribué complet. Ces classes sont définies selon le ISM-GDMO (ISM object specification framework) conforme à celui défini par l'ISO. Elles couvrent des domaines aussi divers que les composants locaux du système (comme les logiciels, les périphériques, etc.), les utilisateurs, les composants de communications (circuits,...), les objets de gestion (agents, rapports, alarmes, etc.) ou encore des abstractions telles que les vendeurs, les localisations,etc. Lorsque le système

distribué est composé de plusieurs sous-systèmes, si la syntaxe utilisée par un sous-système est différente de celle d'ISM, c'est à l'agent d'intégration qu'incombe la charge de la traduction.

V.1.2.1 Structure

La partition du système distribué trouve son reflet dans les eaux de la MIB. Aussi pouvons-nous voir la MIB comme un ensemble de sous-MIB, appelées MIBLETs. Chacune de ces MIBLETs est un sous-arbre rattaché directement à la racine. L'instance d'un objet (c'est à dire un membre d'une classe d'objet) de gestion rattachée à la racine se voit gratifiée du titre de RootLet et possède certaines propriétés que nous ne détaillerons pas.

Chaque MIBLET est gérée (c'est-à-dire rendues visibles) par un composant d'ISM appelé Object Manager ou gestionnaire d'objets. Les intégrateurs d'agent sont des object managers parce qu'ils rendent visibles les objets des agents avec lesquelles ils communiquent. Les services sont aussi des gestionnaires d'objets car comme nous le verrons dans la section qui leur est réservée, ils font apparaître quelques objets internes au manager ISM.

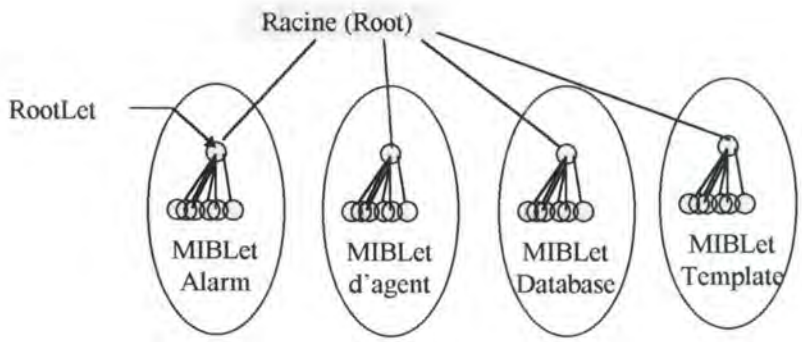


Fig. V.3 Structure

V.1.2.2 Opérations

Malgré la division en MIBLETs, l'ISM MIB est vue comme une unité, où toutes les parties partagent une même racine. Une application peut donc utiliser une instance d'objets où qu'elle se trouve dans la MIB, sans avoir à connaître le composant qui l'a créée. L'application fournit simplement le nom de l'instance et c'est au CMIS Request Broker que revient la responsabilité de faire suivre la requête au bon gestionnaire d'objet. De la même manière, le CRB transmet les notifications des services aux applications.

V.1.2.3 Dénomination d'objet et hiérarchie

La MIB a une hiérarchie d'arbre strict c'est à dire que chaque instance n'a qu'un seul supérieur direct. Les instances sont nommées par leur position dans cet arbre (*Containment tree*).

V.1.3. Les applications

Ce sont les composants qui ont une interface utilisateur. C'est pourquoi nous les considérerons comme faisant partie des modules de Presentation Logic. Bien sûr, ces fonctions peuvent être décomposées en modules de BL,DL et PL plus fins mais nous nous limiterons à ce qui suit en guise d'illustrations des fonctions de l'administrateur de systèmes distribués.

ISM utilise généralement une interface graphique conforme à OSF MOTIF. Plusieurs copies de chaque application peuvent être actives, une par fenêtre active.

Les applications peuvent être classées de la manière suivante.

	Base	*Compagnon*	*Local*
Générique	ISM Monitor ISM Alarm Remote Operation Data Export Trouble Ticket Script	tableurs, traitements de texte,etc.	Remote Login
Spécifique	Software Distribution DB Monitor OS Monitor PC Monitor etc.	DBA Expert Pilot Backup/Restore etc.	Unix Sysadm AIX SMIT NMF6 DNS ou CNS NOI DPS7

Les *applications de base* sont celles qui sont exécutées sur la machine d'ISM Manager et font usage de son infrastructure commune et des protocoles standards d'ISM. Les applications complètement intégrées à ISM Manager sont définies en SML, en utilisant l'environnement de développement ISM SML (voir plus loin).

Les *applications compagnons* sont exécutées sur la machine d'ISM Manager mais ne font pas partie intégrante d'ISM. Cette catégorie inclut les applications qui existent indépendamment d'ISM mais dont le personnel d'administration fait usage et celles qui seront prochainement intégrées à ISM. Les applications compagnons peuvent être développées dans un autre langage en utilisant les CMIS API directement..

Les *applications locales* ne sont pas exécutées sur la machine d'ISM Manager mais sur d'autres appartenant au domaine géré. Elles sont accessibles à l'administrateur en utilisant un protocole de terminal via l'application de base de *Remote Operation*. Cette technique est pour l'instant celle qui permet d'*intégrer* des outils de gestion d'un sous-système.

Les *applications génériques* sont celles qui s'appliquent à tous les composants du système c'est-à-dire celles qui peuvent gérer tous les objets de la MIB. Elles fournissent un ensemble de fonctions de base, indépendantes des sous-systèmes.

Les *applications spécifiques* sont celles qui gèrent certains éléments et fournissent des fonctions particulières. Contrairement aux précédentes qui ne connaissaient bien souvent que la syntaxe des classes d'objet, les applications spécifiques ont accès tant à la syntaxe qu'à la sémantique.

Nous reviendrons sur deux de ces applications au point V.2.

V.1.4 Les services

Ce sont les composants qui fournissent les services communs (et par là indépendant des protocoles de gestion). Ces services sont des *Object Managers*, ce qui signifie qu'ils réalisent leurs fonctions sur des informations propres à ces services et qui sont accessibles par la MIB. Ils se décomposent eux-mêmes en

- **CMIS Database (CMIS-DB)**

  Ce service constitue un répertoire des instances d'objet afin d'assurer leur persistance lorsque ISM Manager est éteint. Les classes d'objets contenues dans la CMIS-DB sont locales à ISM Manager et n'ont pas de contrepartie réelle. Elles permettent de définir la représentation que l'utilisateur a choisie pour son système distribué. Outre les fonctions de base, CMIS-DB fait appel à deux services secondaires (Fig. V.4):

  - Le service d'inventaire utilisé pour conserver les informations génériques de l'inventaire. Cet inventaire contient une partie statique permettant de décrire les objets standards (réseau, équipement, fonction, localisation, etc.);

  - Le service de configuration utilisé pour garder les détails de la configuration de ISM Manager.

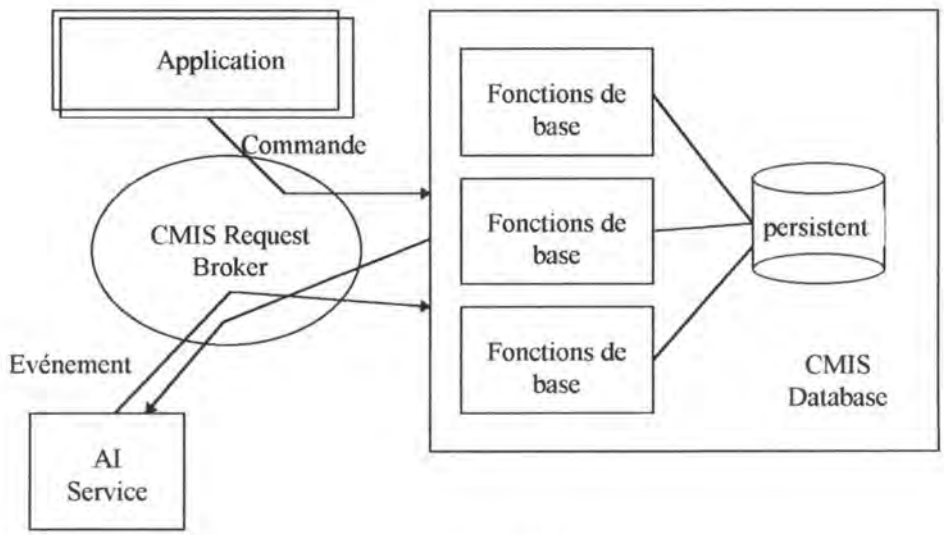


Fig. V.4

- **MIB Template Service**, fournissant une base de données contenant la description des classes d'objets gérés,

- **Service d'enregistrement d'alarme** qui entretient la MIBLET contenant les objets utilisés pour gérer les alarmes.

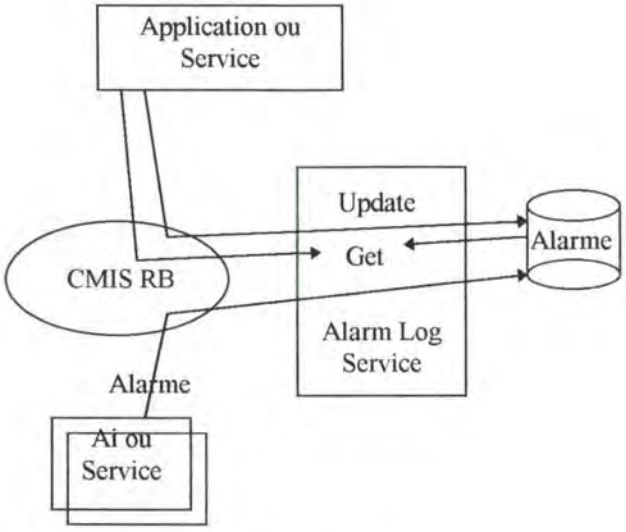


Fig. V.5

- **Service d'enregistrement d'événement** qui entretient la MIBLET contenant les objets utilisés pour gérer les événements.

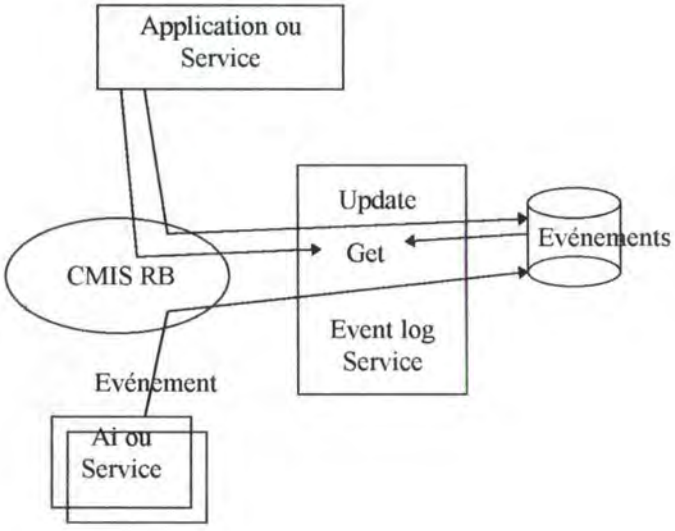


Fig. V.5

V.1.5. Les Agents Intégrateurs

L'Agent Intégrateur est responsable d'un sous-système géré. C'est lui qui communique avec les agents distribués. Il offre les fonctions standards de service ISM

(M_CREATE, M_DELETE, M_GET, M_SET, ALARM, ENROL) et réalise ces services en utilisant les fonctions offertes par un protocole particulier du sous-système.

L'AI contiendra les programmes réalisant les opérations suivantes:

- multiplexage pour permettre à un certain nombre d'agents de communiquer avec à un certain nombre d'applications ou de services;
- traduction de protocole pour convertir le protocole propre au sous-système dans le protocole d'ISM;
- conversion de MIB;
- génération d'alarme;
- découverte des Rootlet;
- extension de la MIB.

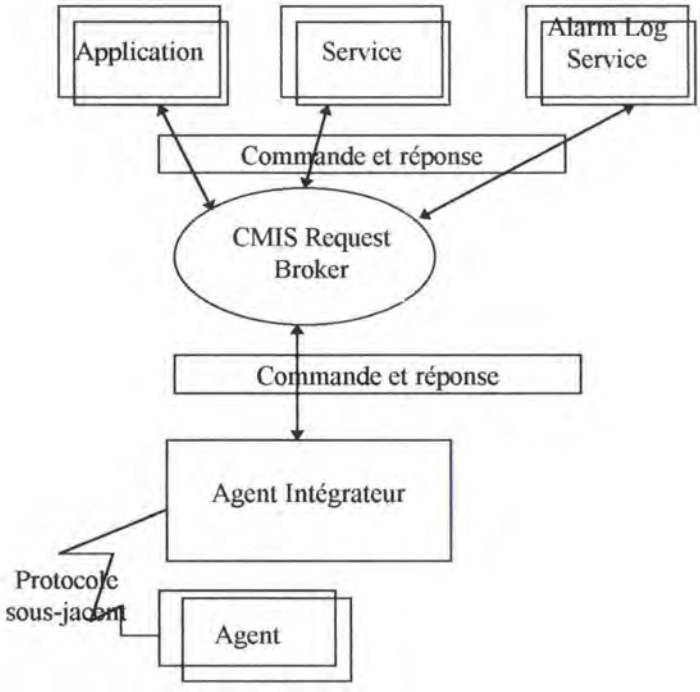


Fig. V.6

Ces agents intégrateurs peuvent être étendus en utilisant un des types de langages de programmation évoqués dans le chapitre II: le langage d'extension de la plate-forme.

V.1.6. L'interface Supra-Managers

Cette interface permet à un ISM Manager de jouer le rôle d'agent pour communiquer avec d'autres managers en utilisant un des protocoles standards.

Elle admettra des demandes d'un Supra-Manager et les traduira en requêtes conformes au protocole interne d'ISM Manager. Les requêtes sont passées au gestionnaire d'objet adéquat pour l'exécution en utilisant les services du CMIS Request Broker.

Ce composant sera donc utilisé pour permettre les communications entre ISM Managers, au sein de la hiérarchie mais aussi dans le contexte d'une *régionalisation* du système, et pour offrir des moyens d'échange avec des Managers étrangers.

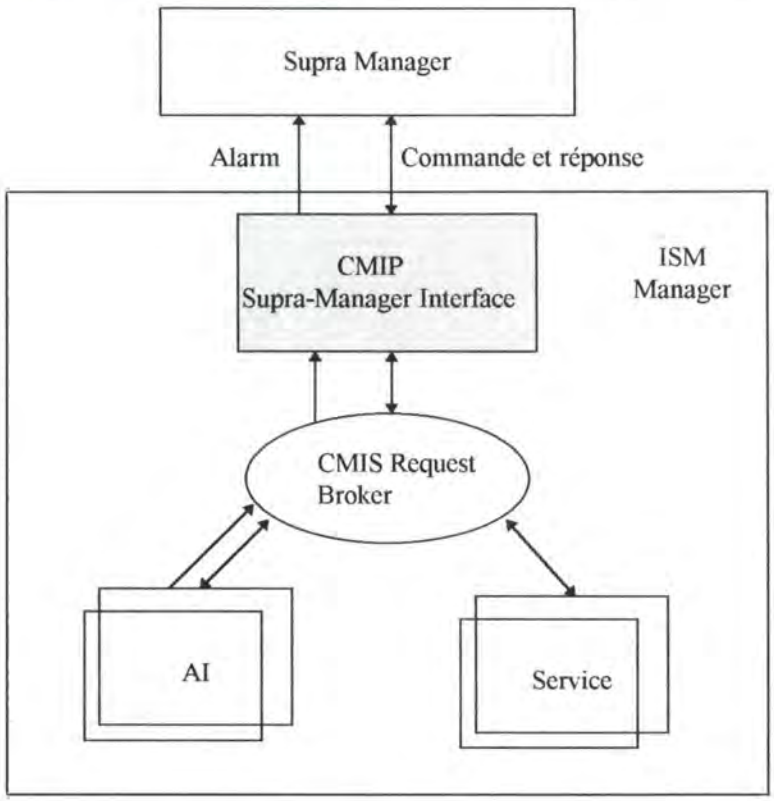


Fig. V.7

V.1.7. CMIS Request Broker

Tous les composants d'ISM Manager communiquent entre eux via le CMIS Request Broker qui fournit les fonctions de transfert de message. Les services offerts sont M_GET, M_DELETE, M_CREATE, M_SET, M_ACTION, M_EVENT_REPORT, M_CANCEL_GET. Le CRB assure la distribution des éléments à travers de multiples systèmes.

Il est composé principalement de deux parties:

- le Routeur de commande (Command Router)

Les applications ISM doivent voir la MIB comme un ensemble sans avoir à connaître l'identité du gestionnaire d'objet responsable de l'instance qu'elles veulent utiliser. Le routeur de commande connaît la localisation de chaque gestionnaire et est ainsi chargé de l'acheminement des commandes vers l'Object Manager responsable. Lorsqu'un demandeur envoie une requête au CRB, celui-ci utilise le répertoire (*Rootlet Directory*) pour déterminer le gestionnaire possédant l'objet spécifié par la requête.

- le Manager d'objet racine (Root Object Manager)

Le Root Object Manager est le gestionnaire d'objet racine. Il assure deux fonctions différentes: la maintenance des tables de routage et le ciblage dans les informations de la Racine.

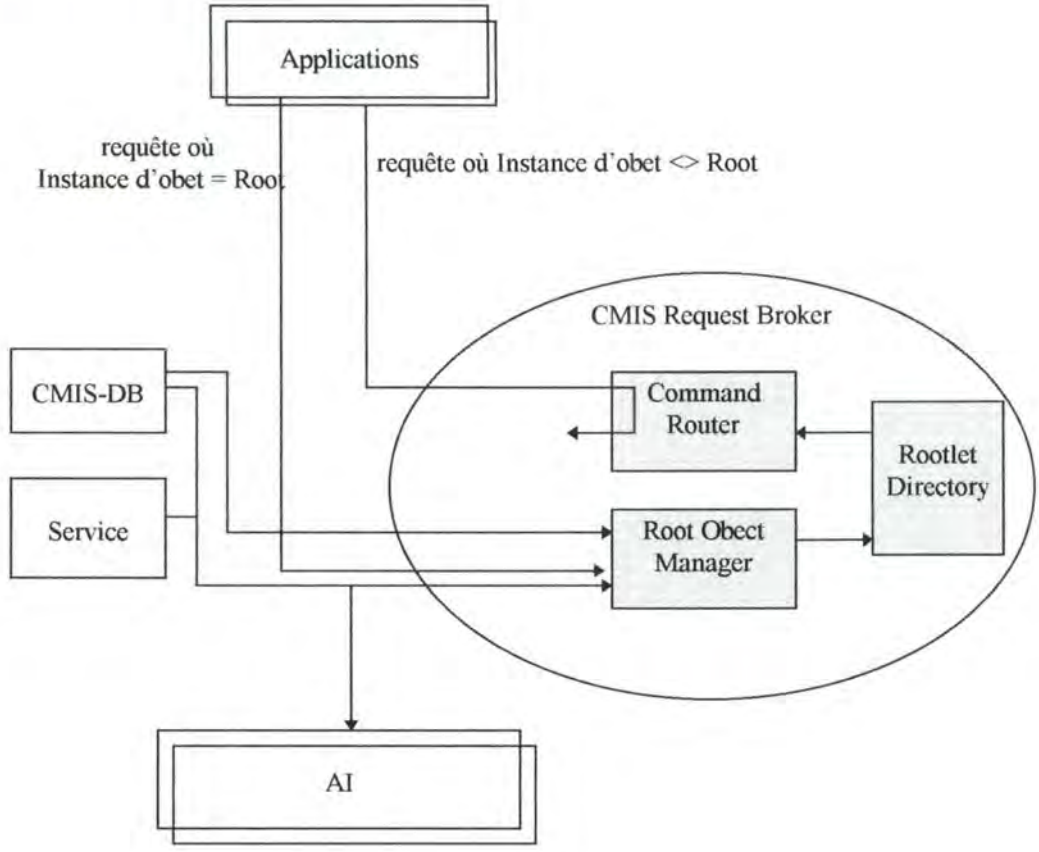


Fig. V.8

Tout ce petit monde étant présenté, il nous faut encore le situer par rapport aux trois catégories du chapitre II..

Nous basant sur la définition que nous avions donnée alors, le CMIS Request Broker est le seul composant que nous ne rangeons pas dans la Data Logic (à part bien sûr la MIB et les applications). En effet, Nous pouvons considérer que c'est lui qui gère l'enchaînement des autres objets qui finalement se résument à une série d'objets capables de mettre à jour ou de récupérer les informations qu'ils détiennent.

Les agents intégrateurs sont pour leur part des unités de Data Logic en communication avec d'autres Data logic distribuées (Agents) selon un modèle de distribution de *Client/serveur Processing.*

V.2. Présentation de la boîte à outils (Middleware)

Nous aborderons la plate-forme sous l'angle de la boîte à outils. L'application distribuée que nous considérions en première partie va laisser la place à un ensemble d'outils propres à définir des applications distribuées sans que le programmeur de l'application ait à prendre en compte les vicissitudes des sous-systèmes (protocoles de gestion différents, etc.). Pour cela, nous considérerons deux applications de base d'ISM Manager telles qu'elles sont décrites dans le Global High-Level Design Document: ISM Monitor et ISM Script..

L'architecture générale de ces applications est présentée par la figure V.9.

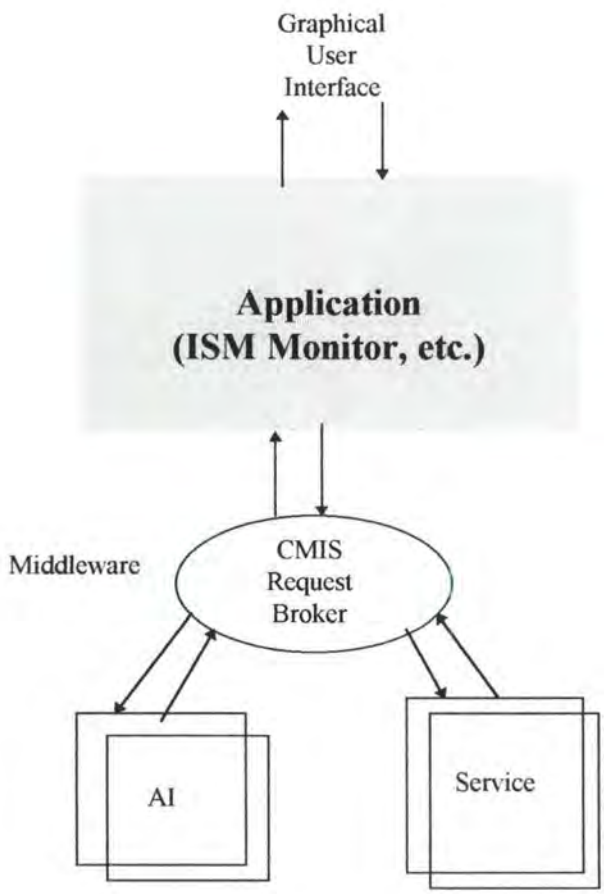


Fig. V.9

V.2.1 ISM Monitor

L'interface de l'application contient une image du réseau montrant les systèmes, les connexions, etc., et leurs attributs, par des symboles qui sont animés selon leur état. Une fenêtre séparée contient le *ISM Application Board* montrant toutes les applications d'ISM, permettant ainsi d'accéder à d'autres fonctions.

L'image du réseau peut être animée en choisissant certains attributs à surveiller en temps réel comme le taux d'occupation du CPU, statut opérationnel, niveau d'alarme, etc. De plus, ISM Monitor peut inclure dans sa représentation une image de fond donnant la situation des locaux, etc.

Une instance d'objet peut être sélectionnée, via son icône, et affichée en tableau ou matrice, ce qui permet à l'utilisateur de passer d'instance en instance à travers toute la MIB.

ISM Monitor permet également d'associer un panneau (*instrument panel*) visualisant sous forme graphique (jauges,etc.) la valeur des attributs d'une ou plusieurs instances d'objets.

En choisissant le symbole d'une instance d'objet et en appelant un menu, l'utilisateur peut être guidé dans les actions accessibles pour la classe d'objet à laquelle appartient l'instance sélectionnée.

ISM Monitor est une application générique de base.

Dans cet exemple, nous voyons clairement deux des trois catégories de modules définis dans le chapitre II. Les modules de Presentation Logic, qui gèrent toutes les représentations et toutes les actions de l'utilisateur sur l'interface, peuvent soit transmettre les informations directement à une Data Logic, soit remettre le contrôle dans les mains d'une Business Logic. Dans la présentation, nous apercevons des techniques propres tant au mode conversationnel (les menus, etc.) qu'au mode interactif (métaphores du mini-monde) comme la sélection des objets. Ces objets sont gérés par des agents, invisibles à l'utilisateur mais qui constituent les modules de Data Logic puisqu'ils permettent de récupérer les informations ou de les modifier. Nous pouvons par ailleurs reconnaître que dans le comportement du CMIS Request Broker se retrouvent les caractéristiques que nous avions relevées dans les modules de communication. Nous pouvons donc le considérer comme faisant partie de cette catégorie alors que tout à l'heure nous le classions dans les modules de Business Logic.

V.2.2 ISM Script

ISM Script est en fait une extension du langage d'élaboration de scripts (*shsml*) du système d'exploitation UNIX et qui offre la possibilité d'accéder au GUI (Graphical User Interface), d'accéder à la MIB via des requêtes CMIS soumises au CMIS Request Broker, d'accéder au MIB Template Service, de programmer des réactions aux événements par l'enregistrement de routines *script* de retour (callbacks).

ISM Script utilise un serveur qui peut manipuler les commandes additionnelles et est construit selon le schéma V.10.

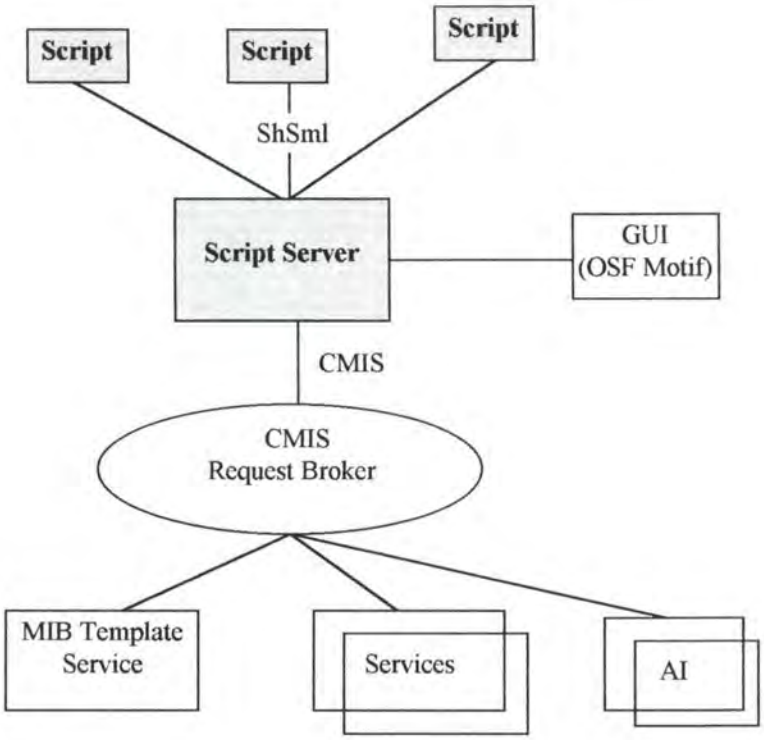
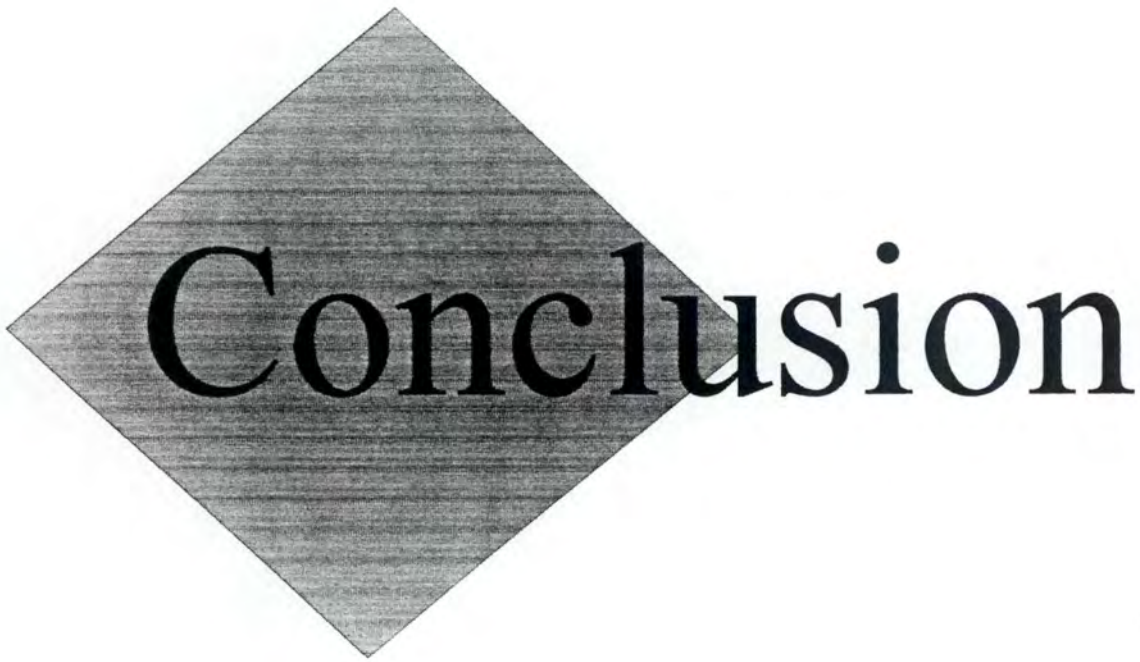


Fig. V.10

Nous retrouvons ici une autre application du chapitre II. Nous y avions abordé en effet trois catégories de langages. Parmi celles-ci, figurait la catégorie des langages destinés aux extensions du langage de commandes. ISM Script est un exemple de ce type de langage. Nous noterons encore une fois le rôle de transparence que joue le CMIS Request Broker.



Conclusion

Conclusion

En réponse aux objectifs fixés au début de ce travail, nous avons montré le contexte dans lequel évoluait l'administrateur de systèmes distribués. Les tâches relatives à cette fonction dans l'organisation ne peuvent se limiter à la seule gestion des machines.

Nous avons montré également l'intérêt de posséder une plate-forme d'administration, pour des systèmes conséquents, par la définition de quelques propriétés qu'il serait bon qu'elle vérifie.

Nous avons vu que leur conception relevait de différents aspects que nous avons répertoriés:

- au niveau de l'**application:** où toute plate-forme distribuée et intégrée peut être perçue comme une couche logicielle de middleware, assurant une transparence de communication et de programmation;

- au niveau de l'**architecture**: où, sur base d'une proposition générale d'architecture des systèmes de gestion, le concepteur peut envisager différentes répartitions des traitements de son application. A ce niveau, nous avons favorisé une approche par standards, qui permet une plus grande souplesse d'évolution et d'intégration.

Même si la présentation de ces différents éléments peut encore faire l'objet d'études plus approfondies, la manière dont nous avons abordé le domaine des systèmes distribués nous permet d'éliminer bon nombre d'ambiguïtés et confusions fréquemment relevées dans la littérature.

En l'absence actuelle d'une démarche systématique de développement de systèmes distribués, nous avons appliqué divers éléments qui peuvent s'avérer utiles dans l'élaboration d'une méthodologie de développement de plates-formes.

Dans ce travail, nous n'avons fait qu'appliquer une ébauche d'architecture à une plate-forme existante. Et nous avons finalement envisagé beaucoup plus le côté architecture des plates-formes que le côté construction d'applications à partir d'une plate-forme.

Aussi un prolongement pourrait être l'exploration de ce second pôle.

Nous n'avons vu en outre que quelques aspects de l'intégration de la distribution. Un autre prolongement de notre travail, pourrait être l'étude de l'intégration des différents aspects de distribution, dans une démarche classique de développement.

Malgré les progrès importants réalisés ces dernières années, notamment dans la portabilité des environnements middleware, la conception d'une application de gestion distribuée et le choix de son infrastructure reste un exercice difficile mais intéressant.

Ce travail ne s'arrête pas là. Ce qu'il nous manque, entre autres choses, est l'expérience des sujets abordés dans ce travail.

Bibliographie

Bibliographie

[Bodar89] Bodart F., Pigneur Y.,
Conception assistée des systèmes d'information - Méthodes, Modèles, Outils,
Masson, 2ème ed., 1989.

[Black92] Black U.
OSI. A model for computer Communications Standards
Prentice-Hall International Editions,1992.

[CA91] Computer Associates
Computing Architecture for the 90's,
Computer Association Inc., 2nd ed.,1990.

[Comer91] Comer D.E.
Internetworking with TCP/IP- Volume 1
Prentice-Hall International Editions,1991

[Coulo88] Coulouris G.F., Dollimore J.,
Distributed systems, Concepts and Design,
Addison-Wesley, 1988.

[Davies83] Davies D.W.,
Distibuted Systems-Architecture and Implementation,
Springer-Verlag, 1983.

[Degho92] Deghorain H.,
L'administration des systèmes distribués. Approche de la gestion OSI.
FUNDP, Mémoire, 1992.

[Duboi91] Dubois E.,
Méthodologie de développement de logiciels,
FUNDP, Notes de cours, 1991.

[Duce84] Duce D.A. et Jones G.P.
Distributed Computing
APIC Studies in Data processing N°20

[Décis93] (groupe),
les multiples facettes de l'administrations
Décision Micro,13 septembre 1993,pp36-89.

[Karem93] Karemera H.
Applications distribuées, description et principes de développement
FUNDP, Mémoire, 1993.

[Mulle89] Mullender S.
Distributred Systems
ACM Press,1989

[Osf91] - Introduction to OSF DCE,version 1.0
- Application development guide, OSF DCE version 1.0
- Administration development guide, OSF DCE version 1.0

[Sloma89] Sloman M.S., Moffeit J.D.
Managing distributed system, notes préparatives
Imperial College, Department of computing Londo, 10 août 1990

[Stall91] Stallings W.,
Data and Computer Communications,
Maxwell Macmillan International, 1991.

[Tanen90] Tanenbaum A.,
Réseaux, architectures, protocoles, applications,
InterEditions,Paris,1990.

[Tanen89] Tanenbaum A.,
Les systèmes d'exploitation, conception et mise en oeuvre
InterEdition, 1989

[Tanen85] Tanenbaum A.
Distributed operating system
in Computing Survey, vol 17, N°4,ACM, New-York, Décembre 1985