Gestion de réseaux et services

Une introduction

Daniel Ranc

Résumé

La gestion de réseaux est un domaine qui, auparavant, se focalisait principalement sur des activités techniques de maintenance du réseau.

Aujourd'hui, la gestion des réseaux ne peut plus être envisagée sans son insertion dans l'ensemble des activités de l'opérateur. C'est ce que tente de montrer cet ouvrage d'introduction qui se propose d'emprunter un chemin passant par les fondamentaux conceptuels du TMN (Telecommunications Management Network), l'étude de l'architecture TINA et la contribution de CORBA.

TINA a représenté l'avènement des architectures de services réseaux. Une vision intégratrice supplémentaire nous est maintenant proposée par l'architecture NGOSS du consortium TeleManagement Forum[1]. Il s'agit d'une analyse basée sur les processus opératoires (*business processes*) de l'opérateur proposant une architecture globale incluant la libéralisation économique du secteur des TIC.

La gestion des réseaux et services met en jeu des systèmes informatiques parmi les plus sophistiqués jamais réalisés. L'ambition de ce document est de sensibiliser le lecteur à la problématique des systèmes informatiques complexes ainsi qu'à leur architecture.

Daniel Ranc

[1] Dont l'INT fait partie.

Sur ce document

L'auteur

Daniel Ranc s'intéresse au domaine de la gestion des réseaux et services depuis les années 90. Il est enseignant-chercheur à l'INT depuis 1998. Auparavant, au sein d'Alcatel Research & Development, il a contribué à de nombreux projets ESPRIT et IST.

Ces activités ont continué de plus belle à l'INT où une équipe compétente a été constituée, laquelle a livré des résultats remarquables notamment dans le domaine des modèles d'information au niveau gestion des Services.

Contributeurs

Ce document n'aurait pu exister sans le travail de fond de l'étudiant Anas Kabbaj de l'INPT de Rabat, dont le trop bref séjour à l'INT a néanmoins permis à ce document de prendre racine.

Versions

- 1.0: version initiale
- 1.2: remodelage, ajout frame et TINA
- 2.0: remodelage, ajout TMF
- 2.01: finitions
- 2.03: référencement INT

Références du document

Interne: INT-LOR-GRS-v2.03
ISBN: en cours

Remerciements

Les figures du chapitre 8 par courtoisie du TeleManagement FORUM

Références

Documents concernés ou cités

[1] ITU-T Recommendation M.3010 (2000): "Principles for a telecommunications management network".
[2] 3GPP TS 22.101: "Service aspects; Service Principles".
[3] 3GPP TS 32.111-1: " Telecommunication management; Fault Management; Part 1: 3G fault management requirements".
[4] IETF RFC 959: "File Transfer Protocol (FTP)"; October 1985, J. Postel, J. Reynolds, ISI.
(Status: Standard).
[5] IETF RFC 783: "Trivial File Transfer Protocol (TFTP)"; rev. 2, June 1981, K.R. Sollins MIT.
(Status: Unknown).
[6] IETF RFC 1157: "Simple Network Management Protocol (SNMP)": May 1990, J. Case, SNMP Research, M. Fedor, Performance Systems International, M. Schoffstall, Performance Systems International, J. Davin, MIT Laboratory for Computer Science. (Status: Standard).
[7] IETF RFC 2401: "Security Architecture for the Internet Protocol"; November 1998. (Status: Proposed Standard).
[8] The Object Management Group (OMG) "The Common Object Request Broker: Architecture and Specification", Revision 2.3, June 1999.
http://www.omg.org/technology/documents/vault.htm#CORBA_IIOP
[9] ITU-T Recommendation Q.811 (1997): "Lower Layer Protocol Profiles for the Q3 Interface and X interfaces".
[10] ITU-T Recommendation Q.812 (1997): "Upper Layer Protocol Profiles for the Q3 Interface and X interfaces".
[11] ITU-T Recommendation X.650 (1996): "Information Technology - Open Systems Interconnection – Basic Reference Model: Naming and Addressing".
[12] ITU-T Recommendation X.700 (1992): "Management Framework for Open Systems Interconnection (OSI) for CCITT applications".
[13] ISO 8571-1 (1988): "Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 1: General Introduction".
[14] ISO 8571-2 (1988): "Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 2: Virtual Filestore Definition".
[15] ISO 8571-3 (1988): "Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 3: File Service Definition".
[16] ISO 8571-4 (1988): "Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 4: File Protocol Specification".
[17] ISO/IEC ISP 10607-1 (1995): "Information technology - International Standardized Profiles
AFTnn - File Transfer, Access and Management - Part 1: Specification of ACSE, Presentation and Session Protocols for the use by FTAM".
[18] ISO/IEC ISP 10607-2 (1995): "Information technology - International Standardized Profiles AFTnn - File Transfer, Access and Management - Part 2: Definition of Document Types, Constraint sets and Syntaxes".

[19] ISO/IEC ISP 10607-3 (1995): "Information technology - International Standardized Profiles AFTnn - File Transfer, Access and Management - Part 3: AFT 11 - Simple File Transfer Service (Unstructured)".
[20] ITU-T Recommendation X.710 (1997): "Information Technology – Open Systems
Interconnection - Common Management Information Service".
[21] ITU-T Recommendation X.711 (1997): "Managed objects for diagnostic information of public switched telephone network connected V-series modem DCE's".
[22] ITU-T Recommendation X.25 (1996): "Interface between Data Terminal Equipment (DTE) and Data Circuit Terminating Equipment (DCE) for Terminals operating in the Packet Mode and connected to Public Data Networks by Dedicated Circuit".
[23] ISO/IEC ISP 11183-1 (1992): "Information technology - International Standardized Profiles AOM1n.OSI Management - Management Communications - Part 1: Specification of ACSE, presentation and session protocols for the use by ROSE and CMISE".
[24] ISO/IEC 9545:1994: "Information technology - Open Systems Interconnection - Application Layer Structure".
[25] ITU-T Recommendation X.200 (1994): "Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model".
[26] ITU-T Recommendation X.208 (1988): "Specification of Abstract Syntax Notation One (ASN.1)".
[27] ITU-T Recommendation X.209 (1988): "Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)".
[28] ITU-T Recommendation X.210 (1993): "Information Technology - Open Systems
Interconnection - Basic Reference Model: Conventions for the definition of OSI Services".
[29] ITU-T Recommendation X.211 (1995): "Information Technology - Open Systems
Interconnection - Physical Service Definition".
[30] ITU-T Recommendation X.212 (1995): "Information Technology - Open Systems
Interconnection - Data link Service Definition".
[31] ITU-T Recommendation X.213 (1995): "Information Technology - Open Systems
Interconnection - Network Service Definition".
[32] ITU-T Recommendation X.223 (1993): "Use of X.25 to provide the OSI Connection-mode network service for ITU-T applications".
[33] ITU-T Recommendation X.214 (1995): "Information Technology - Open Systems
Interconnection - Transport Service Definition".
[34] ITU-T Recommendation X.224 (1995): "Information Technology - Open Systems
Interconnection - Protocol for providing the connection-mode transport service".
[35] ITU-T Recommendation X.215 (1995): "Information Technology - Open Systems
Interconnection - Session Service Definition".
[36] ITU-T Recommendation X.225 (1995): "Information Technology - Open Systems
Interconnection - Connection-oriented session protocol: Protocol specification".
[37] ITU-T Recommendation X.216 (1994): "Information Technology - Open Systems
Interconnection - Presentation Service Definition".
[38] ITU-T Recommendation X.226 (1994): "Information Technology - Open Systems
Interconnection - Connection-oriented presentation protocol: Protocol specification".

[39] ITU-T Recommendation X.217 (1995): "Information Technology - Open Systems
Interconnection - Service definition for the association control service element".
[40] ITU-T Recommendation X.227 (1995): "Information Technology - Open Systems
Interconnection - Connection-oriented protocol for the association control service element: Protocol specification".
[41] ITU-T Recommendation X.219 (1988): "Remote Operations: Model, Notation and Service Definition".
[42] ITU-T Recommendation X.229 (1988): "Remote Operations: Protocol Specification".
[43] ISO/IEC 7776 (1995): "Information technology - Telecommunications and information exchange between systems - High-level data link control procedures - Description of the X.25
LAPB-compatible DTE data link procedures".
[44] ISO/IEC 8208 (2000): "Information technology - Data communications - X.25 Packet Layer Protocol for Data Terminal Equipment".
[45] ISO/IEC 8878 (1992): "Information technology - Telecommunications and information exchange between systems - Use of X.25 to provide the OSI Connection-mode Network Service".
[46] IETF RFC 1006: "ISO Transport on top of the TCP", Marshall T. Rose, Dwight E. Cass, Northrop Research and Technology Center, May 1987. Status: Standard.
[47] IETF RFC 793: "Transmission Control Protocol (TCP)", September 1981. Status: Standard.
[48] IETF RFC 791: "Internet Protocol (IP)", September 1981. Status: Standard.
[49] ITU-T Recommendation X.680 (2002): "Information Technology-Abstract Syntax Notation One (ASN.1): Specification of Basic Notation".
[50] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".
[51] 3GPP TS 22.115: "Service aspects; Charging and Billing".
[52] The Object Management Group (OMG) "The Common Object Request Broker: Architecture and Specification", Revision 2.1, August 1997.
http://www.omg.org/technology/documents/vault.htm#CORBA_IIOP
[53] 3GPP TS 32.400-series: "Telecommunication management; Performance Management (PM); Concept and requirements".
[54] 3GPP TS 32.600: "Telecommunication management; Configuration Management (CM); Concept and high-level requirements".
[55] 3GPP TS 32.200: "Telecommunication management; Charging management; Charging principles".
[100] TMF GB910: "Telecom Operations Map"; Approved Version 2.1 March 2000, (may be downloaded from http://www.tmforum.org.).
[101] 3GPP TS 32.102: "3G Telecom Management Architecture".
[102] ITU-T Recommendation M.3013 (2000): "Considerations for a telecommunications management network".
[104] 3GPP TR 21.801: "Specification Drafting Rules".
[105] TMF GB910B: "Telecom Operations Map Application Note-Mobile Services: Performance Management and Mobile Network Fraud and Roaming Agreement Management"; Public Evaluation Version 1.1, September 2000. (May be downloaded free from http://www.tmforum.org.)
[CAJB]: Jean-Marie CHAUVET "Corba ActiveX et Java Beans" - Paris: *Eyrolles,* septembre 1997.
[FUE 95] : L. A. de la fuente, J. Pavon, M. Wakano, "The TINA-C Management Architecture", Octobre 1995
[G.803]: ITU-T Rec. G.803, Architecture des réseaux de transport à hiérarchie numérique synchrone, Mars 1993.
[GRA 95]: P. Graubmann TINA-C Deliverable, "Engineering Modeling Concepts (DPE Architecure)", Dec 95.
[IMTCDE]: Manuel-Juan Simon de la Horra, "Implémentation of a management TMN agent in a Corba distributed environment".
[ISO 9596]: ISO/IEC 9596: Common Management Information Protocol Definition, 1991.

[ISO 10165-1]: ISO 10165-1/ITU-T X.720: Management Information Model, 1993.
[ISO 10165-4]: ISO 10165-4/ITU-T X.722: Guidelines for the Definition of Managed Objects, 1993.
[M.3010]: ITU-T Rec. M.3010, Principes pour un réseau de gestion des télécommunications, 1993.
[M.3100]: ITU-T Rec. M.3100, Modèle informationnel générique de réseau de gestion des télécommunications, 1993.
[ODP 96]: ISO/IEC IS 10746-1 - ITU-T Rec. X901, *ODP Reference Model Part 1. Overview and Guide to Use,* May 1996.
[OMG]: http://www.omg.org
[PRI 93]: RACE Project 2041 PRISM. Service Management Reference Configuration, RACE, Sept 1993.
[Q.811]: ITU-T Rec. Q.811, Profils des protocoles de couche inférieure pour l'interface Q3, 1993.
[Q.812]: ITU-T Rec. Q.812, Profils des protocoles de couche supérieure pour l'interface Q3, 1993.
[Q.12xx]: ITU-T Rec. Q120x, Q121x, Q122x: The Intelligent Network, 1993.
[RUB 94]: H. Rubin, N. Natarajan, "A Distributed Software Architecture for Telecommunication Networks", IEEE Network, Jan-Fév 1994.
[RUM 91]: J.Rumbaugh Object Oriented Modeling and Design. Prentice Hall: Engelwood Cliffs, N.J. 1991.
[Waldbusser 91]: S Waldbusser."Remote Network Monitoring Management Information Base", RFC 1271, novembre 1991.
[X730]: ITU-T Rec.X.730/ISO 10164-1 Gestion des objets, 1993.
[X731]: ITU-T Rec.X.731/ISO 10164-2 Gestion des attributs d'états, 1993.
[X732]: ITU-T Rec.X.732/ISO 10164-3 Gestion des attributs de relations, 1993.
[X733]: ITU-T Rec.X.733/ISO 10164-4 Gestion des rapports d'alarme, 1993.
[X734]: ITU-T Rec.X.734/ISO 10164-5 Gestion des rapports d'événements, 1993.
[X735]: ITU-T Rec.X.735/ISO 10164-6 Gestion des journaux, 1993.
[X736]: ITU-T Rec.X.736/ISO 10164-7 Gestion des rapports d'alarmes de sécurité, 1993.
[X737]: ITU-T Rec.X.737/ISO 10164-14 Catégories de test de diagnostic et de confiance, 1994.
[X738]: ITU-T Rec.X.738/ISO 10164-13 Gestion des synthèses, 1994.
[X739]: ITU-T Rec.X.739/ISO 10164-11 supervision de la charge , 1994.
[X740]: ITU-T Rec.X.740/ISO 10164-8 Gestion des traces de vérification de la sécurité, 1993.
[X741]: ITU-T Rec.X.741/ISO 10164-9 Objets et attributs pour le contrôle d'accès , 1993.
[X742]: ITU-T Rec.X.742/ISO 10164-10 Gestion des mesures de coût, 1993.
[X745]: ITU-T Rec.X.745/ISO 10164-12 Gestion des tests, 1993.

Table des matières

Figures et tableaux

Abréviations

API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
ATM	Asynchronous Transfer Mode
B2B	Business to Business
B-ISDN	Broadband ISDN
BOOTP	Boot protocol
CLI	Command Line Interface
CMIP	Common Management Information Protocol
CMIP/GDMO	Common Management Information Protocol/Guidelines for the Definition of Managed Objects
COPS	Common Open Policy Service
COPS-PR	COPS Usage for Policy Provisioning
CORBA IIOP Protocol	Common Object Request Broker Architecture Internet Inter-ORB
CORBA	Common Object Request Broker Architecture
CORBA/IDL Language	Common Object Request Broker Architecture/Interface Definition
DCN	Data Communications Network
DECT	Digital Enhanced Cordless Telecommunications
DHCP	Dynamic Host Configuration Protocol
DNS	Directory Name Service
DSS1	Digital Subscriber System 1
EM	Element Manager
EMS	Element Management System
FFS	For Further Study
FTAM	File Transfer Access and Management
FTP	File Transfer Protocol
ftp	FTP
GDMO	Guidelines for the Definition of Managed Objects
GGSN	Gateway GPRS Support Node
Go interface (PDF)	The interface between the GGSN and the Policy Decision Function
GSM	Global System for Mobile communications
HLR	Home Location Register
HSS	Home Subscriber Server
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IIOP	Internet Inter-ORB Protocol
IN	Intelligent Network
INAP	Intelligent Network Application Part

IRP	Integration Reference Point
IS	Information Service
ISDN	Integrated Services Digital Network
LDAP	Lightweight Directory Access Protocol
LDUP	LDAP Duplication/Replication/Update Protocols
LLA	Logical Layered Architecture
MAP	Mobile Application Part
MExE	Mobile Execution Environment
MIB	Management Information Base
MMI	Man-Machine Interface
NM	Network Manager
NMS	Network Management System
NRM	Network Resource Model
OS	Operations System
OSI	Open Systems Interconnection
OSS	Operations Support System
PDF	Policy Decision Function
PDH	Plesiochronous Digital Hierarchy
PDP	Policy Decision Point
PIB	Policy Information Base
PKI	Public Key Infrastructure
PLMN	Public Land Mobile Network
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RNC	Radio Network Controller
RSVP	Resource ReserVation Protocol
SDH	Synchronous Digital Hierarchy
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol (IETF)
SNMP/SMI	SNMP/Structure of Management Information
SOM	Service Operations Management
SS	Solution Set
SS7	Signalling System No. 7
TCP/IP	Transmission Control Protocol/ Internet Protocol
tftp	trivial ftp
TM	Telecom Management
TMF	TeleManagement Forum
TMN	Telecommunications Management Network (ITU-T)
TOM	Telecom Operations Map (TMF)
UML	Unified Modelling Language
UPT	Universal Personal Telecommunication
USIM	Universal Subscriber Identity Module
UTRA	Universal Terrestrial Radio Access
VHE	Virtual Home Environment

Chapitre 1 – Gestion des réseaux

Présentation générale

Administrer, c'est vouloir tirer le meilleur profit de la structure que l'on gère. Cependant, ce système est dual, car la conception d'une gestion dépend étroitement de la structure administrée. Inversement, le comportement futur de cette structure dépendra fortement de sa gestion. Les services et les équipements de télécommunication sont de plus en plus complexes et nombreux, cette évolution de la technologie met en évidence la nécessité de disposer d'architectures pouvant gérer ces services et contrôler ces ressources dans des environnements hétérogènes.

De façon générale, une administration de réseaux a pour objectif d'englober un ensemble de techniques de gestion mises en œuvre pour:

- Offrir aux utilisateurs une certaine qualité de service;
- Permettre l'évolution du système en incluant de nouvelles fonctionnalités;
- Rendre opérationnel un système.

Les disciplines de l'administration d'un système

L'administration de réseaux ne peut pas être isolée de son environnement. En effet il s'agit d'une partie intégrante d'un système plus général constitué de trois parties fondamentales qui sont:

- Les utilisateurs ou consommateurs de services;
- Les serveurs d'applications ou fournisseurs de services;
- La machine de transport reliant les utilisateurs aux fournisseurs.

Ces trois groupes déterminent les trois disciplines d'administration d'un système:

L'administration des utilisateurs

Qui fournit l'ensemble des mécanismes pour:

- L'accessibilité et la connexion aux applications. En effet, l'utilisateur doit pouvoir se connecter aux différentes applications et, pour cela, il doit disposer d'un ensemble d'outils qui lui assurent la transparence des méthodes d'accès et de connexion aux différentes applications;
- L'accès aux serveurs de noms, afin de permettre la localisation des ressources et d'assurer à l'utilisateur l'existence et l'utilisation de ces ressources.
- La confidentialité et la sécurité: le système doit fournir l'ensemble des mécanismes qui permettent de garantir la confidentialité des informations de l'utilisateur, de sécuriser son environnement et de prévenir toute perte ou altération des échanges effectués par l'utilisateur;

- la qualité de service, qui est la perception du réseau ressentie par l'utilisateur. Son exigence majeure concerne la disponibilité et les performances du système ainsi que sa capacité à assurer un service convenable.

L'administration des serveurs

Qui recouvre l'ensemble des mécanismes à mettre en place pour:

- la connexion et la distribution des applications, cela afin de permettre l'interrelation des services;
- la gestion et la distribution des données, qui, comme pour les utilisateurs, doivent garantir la fiabilité de transmission des informations sur le réseau et offrir un certain nombre d'outils permettant le transfert de ces informations. C'est typiquement le rôle des outils de transfert de fichiers, qui permettent le partage des capacités de stockage entre plusieurs systèmes;
- la gestion des applications, correspond essentiellement au contrôle et la protection des accès à ces applications par la distribution de droits, ainsi qu'à la fourniture de mécanismes de contrôle d'utilisation de ressources concernant l'application.

L'administration de la machine de transport

Qui consiste à fournir des mécanismes sur:

- des opérations de réseau, dont le rôle est de permettre l'intervention sur le fonctionnement du réseau et la modification si nécessaire;
- les incidents réseau par la mise en place de mécanismes de détection et de correction. Lorsqu'une alerte est déclenchée, des actions doivent être prises pour résoudre rapidement l'incident et réduire son influence sur la qualité de service;
- les performances, dont le but est d'évaluer le système par un ensemble de paramètres comme le temps de réponse ou la charge du système. L'intérêt de cette évaluation est de permettre un jugement sur le fonctionnement du système;
- les coûts, afin de pouvoir les mesurer. En effet, dans un réseau, les coûts d'utilisation sont complexes à évaluer puisqu'ils s'appliquent à un ensemble de composants distribués;
- la configuration, dont le but est de déterminer la meilleure configuration du réseau améliorant ainsi les performances du système, et par la même, sa qualité de service. Cette détermination se fait généralement à l'aide de graphes;
- l'inventaire, qui a pour rôle de tenir à jour en permanence la liste des éléments (logiciels et matériels) qui constituent un système réseau;
- l'évolution et les changements, dont l'objectif est de fournir un certain nombre d'informations permettant de déterminer les nouveaux besoins à prendre en compte et les parties du système concernées.

Les architectures de l'administration des réseaux

Lors de la conception d'un système de gestion de réseau, trois architectures peuvent être considérées (voir figure 1.1).

Le choix de l'une ou l'autre dépend des besoins du gestionnaire de réseau. Ces architectures sont:

Architecture centralisée

C'est l'organisation la plus classique de l'administration, dans laquelle un seul manager (gestionnaire) contrôle toutes les ressources du réseau et les équipements distribués dans un réseau de télécommunication. Cette architecture présente l'avantage d'être facile à concevoir, mais en contrepartie elle s'avère inefficace dans le cas de réseaux étendus. La figure 1.2 montre ce type d'organisation.

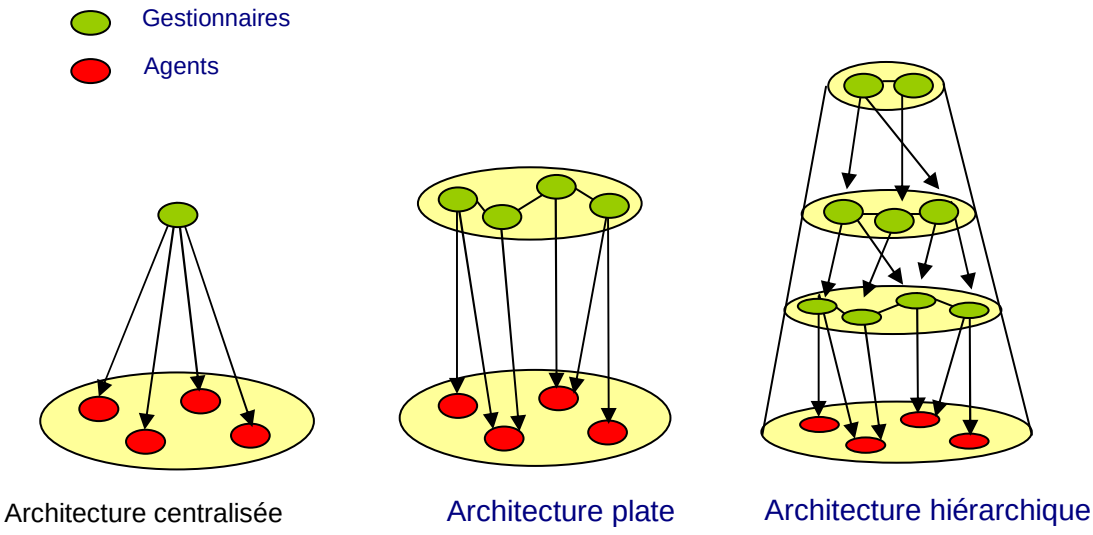


Fig. 1.1: les modèles d'organisation

Architecture plate

Dans cette organisation, plusieurs gestionnaires contrôlent différents aspects et portions du réseau de télécommunication. Cette conception implique une interaction entre les managers qui sont au même niveau de l'organisation.

Architecture hiérarchique

Dans ce type d'architecture, une certaine abstraction a lieu. En effet, il existe plusieurs niveaux dans lesquels les entités sont considérées. La notion de gestionnaire-agent change dans cette organisation puisque qu'un agent dans le niveau n devient un gestionnaire pour le niveau inférieur n-1.

La figure 1.2 montre une vue typique d'un environnement de gestion centralisé. On remarque qu'il y a un seul gestionnaire qui surveille tout le réseau, les agents ont chacun une visions d'un champ donné, et enregistrent tous les changements d'état dans les MIBs (bases de données). Les agents distants qui contrôlent donc une certaine partie des ressources et équipements réseau. Les constructeurs munissent leurs équipements d'agents logiciels qui dépendent de la plate forme et du protocole utilisé. Ces agents permettent de superviser et collecter des informations de gestion pour les enregistrer dans des bases de données locales. Les protocoles de gestion de réseaux sont variés, mais uniquement deux d'entre eux sont standardisés : SNMP (*Simple Network Management Protocol*) et CMIP (*Common Management Information Protocol*). Ces protocoles sont génériques et indépendants des équipementiers. Ils sont utilisés partout dans le monde, mais leur

utilisation dépend de l'environnement et de la complexité des services à fournir.

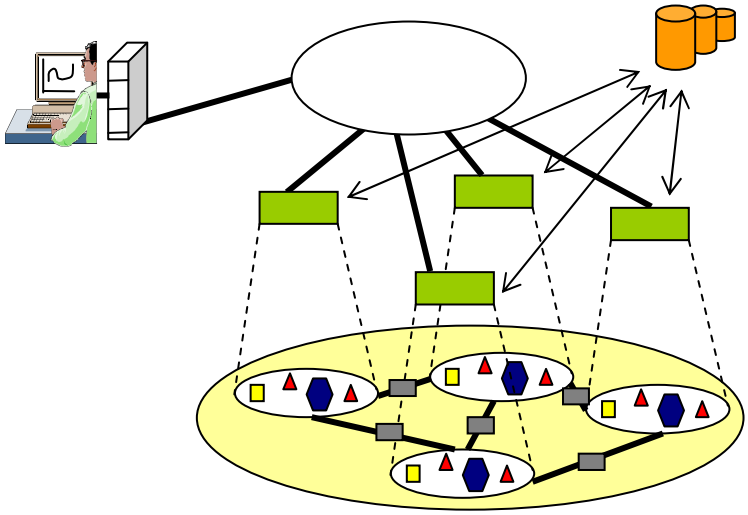


Fig. 1.2: architecture centralisée

Les fonctionnalités de l'administration des réseaux

Cinq groupes de fonctions sont définis pour satisfaire tous les besoins fonctionnels; ces groupes seront détaillés dans le paragraphe 2.4.1 du chapitre 2:

La gestion des fautes

Concerne la fonction de surveillance (monitoring), la localisation et la détermination des pannes.

La gestion des configurations et des noms

Concerne la fonction d'installation de composants, la fonction de contrôle et surveillance puis la fonction de gestion des noms.

La gestion des performances

Concerne la fonction de collecte de données, la fonction de gestion de trafic et la fonction d'observation de la qualité de service.

La gestion de la facturation

Concerne la fonction de surveillance de la charge des ressources, la fonction de calcul des coûts des ressources, la fonction de facturation et la fonction de gestion des limites utilisateur.

La gestion de la sécurité

Concerne la fonction de gestion de la confidentialité, la fonction d'audit et la fonction d'enregistrement et gestion d'abonnés.

Les critères pour une organisation logique

Afin de pouvoir définir une organisation logique d'un système de gestion, il est nécessaire de tenir compte de certains critères:

Le critère informationnel

Les moyens de gestion se basent tous sur un ensemble d'informations qu'ils doivent traiter. Ces informations proviennent de sources qui peuvent

être de deux types: les informations issues directement du système géré et les informations représentant le système géré. Le premier type d'informations provient essentiellement des sources suivantes: les composants du réseau, les outils de tests et les utilisateurs. Le deuxième type, informations décrivant le système géré sont stockées dans bases de données.

Le critère fonctionnel

Pour réaliser pleinement les différentes décisions administratives, il faut mettre en œuvre différents domaines de gestion plus spécifiques et dont les résultats fourniront la réalisation d'une décision. Ces domaines communs représentent des fonctionnalités partagées utilisables à différents niveaux de responsabilité de gestion. Un certain nombre de ces fonctions est nécessaire pour réaliser une organisation. Ce sont celles recensées par les organismes de normalisation, à savoir les fonctionnalités de configuration, de performances, de fautes, de sécurité et de comptabilité.

Le critère temporel

Les aspects temporels déterminent les contraintes de réaction d'un élément de gestion, ainsi que ses conséquences sur le système géré. Il existe trois grands types de contraintes temporelles:

- Le court terme de l'ordre de la minute à la journée, typiquement, ce sont les contraintes liées aux aspects opérationnels;
- Le moyen terme, de l'ordre de la journée à plusieurs mois, typiquement, ce sont les contraintes liées aux aspects d'environnement du système;
- Le long terme, de l'ordre de l'année, typiquement, ce sont les contraintes liées aux aspects évolutifs du système.

Le critère de discipline

La structure actuelle des réseaux implique de plus en plus une intégration des phénomènes périphériques à la communication. Ainsi, on ne peut plus, aujourd'hui, dans le cadre d'un réseau à intégration de services par exemple, dissocier ou tout au moins isoler, l'administration de réseaux de l'administration des utilisateurs et des services. Il y a trois disciplines d'administration qui doivent interagir étroitement pour mener à bien une administration globale d'un système:

- L'administration des utilisateurs: actuellement ne constitue pas une part importante, mais elle grandira avec l'émergence des réseaux à très large bande. En effet, ces réseaux fourniront aux utilisateurs un point d'accès unique vers des services aussi variés que la télévision, la téléphonie sur IP…
- L'administration des fournisseurs de services: ceux-ci auront de plus en plus besoin de moyens leur permettant de construire leurs services, d'en suivre la qualité et d'effectuer leur propre gestion.
- L'administration système: fournit les mécanismes pour surveiller, contrôler et coordonner les ressources dans l'environnement en question et les protocoles utilisés pour la communication d'informations sur ces ressources.

Conclusion provisoire

Une administration est donc une stratégie permettant de prendre plusieurs décisions selon des objectifs fixés au préalable par l'administrateur. Ces décisions sont implémentées par des actions à entreprendre sur plusieurs plans, à savoir l'administration des utilisateurs, des fournisseurs de services et la gestion système.

Une bonne politique d'administration consiste à définir pour un environnement donné les critères que l'on souhaite atteindre. Elle permet d'assurer une qualité de service donnée, de promouvoir l'évolution du système vers de nouvelles fonctionnalités et le fonctionnement normal du système.

Chapitre 2 - Comparaison gestion TMN vs. gestion SNMP

Le but de ce chapitre est de comparer deux approches différentes de la gestion des réseaux: la gestion SNMP (*Simple Network Management Protocol*) et la gestion TMN se basant sur le protocole CMIP (*Common Management Information Protocol*) et CMIS (*Common Management Information Service*). Tout au long de ce chapitre la comparaison sera faite entre les modèles TMN et IETF (*Internet Engineering Task Force*) puisque SNMP est né dans la communauté Internet. Les modèles informationnels seront abordés dans le paragraphe 2.1, les modèles architecturaux dans le paragraphe 2.2, les modèles de communication dans le paragraphe 2.3 et enfin les modèles fonctionnels du TMN et de l'IETF seront présentés dans le paragraphe 2.4.

Le système d'information du réseau est l'outil majeur de l'administrateur, il conditionne la plupart des activités et des potentialités de fourniture de services. La maîtrise du système d'information passera par la compréhension de sa sémantique et par la puissance de sa modélisation.

Comparaison des modèles informationnels TMN et IETF

Le modèle informationnel du TMN

Le modèle informationnel du TMN est orienté objet. Les concepts de ce modèle sont définis dans la norme relative au MIM (*Management Information Model*) [ISO 10165-1]. Ce modèle est caractérisé par la définition des objets avec des propriétés spécifiques, ces objets étant les objets de gestion. Ces derniers sont la représentation abstraite des ressources de communication, ressources physiques ou logiques tels qu'un PABX, une connexion, etc.

En d'autres termes, les objets gérés sont "la vue de gestion des ressources" à gérer. Le fonctionnement interne de la ressource et les relations entre objet géré ne sont pas connus de l'administration. Seules les caractéristiques définissant la ressource en tant qu'un objet administré sont accessibles par l'administration à travers son interface de gestion.

Les informations de gestion sont documentées à l'aide d'un ensemble de formulaires (*templates*), appelé GDMO (*Guidelines for the Definition of Managed Objects*) [ISO 10165-4].

Modélisation des classes d'objets gérés

Un objet géré ou MO (*Managed Object*) représente une ressource de communication dans un but de gestion. Un objet géré est une instance d'une classe d'objet de gestion ou MOC (*Managed Object Class*).

L'ensemble des objets gérés forme la vue de gestion OSI des ressources en question. La définition d'une classe d'objets de gestion consiste en:

- La position de la classe dans un arbre d'héritage;
- Un ensemble de paquetages obligatoires. Un paquetage obligatoire contient les caractéristiques qui sont primordiales et que l'on doit retrouver dans chaque instance de la classe d'objet;
- Un ensemble de paquetages conditionnels et la condition sous laquelle chaque paquetage est présent.
- La structure d'un paquetage est caractérisée par:
- Les attributs dont la classe d'objets dispose;
- Les notifications pouvant être émises par l'objet;
- Les opérations de gestion qui affectent les attributs de cet objet ou l'objet dans son ensemble;
- Le comportement que l'objet a en réponse aux sollicitations externes.

GDMO

GDMO est une notation semi-formelle qui permet de spécifier les MOCs. Les directives GDMO procurent un mécanisme normalisé pour définir de manière semi-formelle la syntaxe, la sémantique et les aspects comportementaux des informations de gestion, ce qui permet une compréhension commune des fonctions de gestion en limitant la place laissée à l'interprétation.

Des formulaires ont été définis pour les types suivants: classe, paquetage, attribut, groupe d'attributs, notification, comportement, action, paramètre et lien de nommage.

• Classe:

L'exemple suivant déclare une classe d'objet de gestion concernant le type d'objet Entité_protocole:

```
Entité_protocole MANAGED OBJECT CLASS DERIVED FROM    Entité;
        <Super-classe dont la définition est héritée par la classe entité_protocole>
CHARACTERIZED BY Paquetage_entité_protocole;
        <Ce paquetage obligatoire est décrit ci-dessous>
REGISTERED AS {exemple_Classe_1}
        <Enregistrement de la classe dans l'arbre d'enregistrement ISO>
```

• Paquetage:

Un paquetage est une construction modulaire contenue dans un objet de gestion. Un paquetage possède un ensemble d'attributs, de notifications, d'opérations et de comportements. Le concept de caractéristiques d'un paquetage obligatoire est commun à toutes les instances d'une classe d'objet donnée. A la création (resp. la suppression) d'un objet, tous les paquetages obligatoires sont créés (resp. supprimés) simultanément. Les paquetages conditionnels sont présents si les conditions explicites qui leur sont associées sont satisfaites. Si tel est le cas, alors le paquetage conditionnel est créé avec l'objet duquel il dépend. Une fois instantié, le paquetage conditionnel ne pourra être supprimé que lors de la suppression de l'objet le contenant.

Un paquetage est décrit à l'aide d'un formulaire. Si le paquetage peut être réutilisé par plusieurs classes d'objets de gestion; alors celui-ci doit être spécifié séparément, sinon sa déclaration peut être contenue dans la classe d'objets qui le possède. Le concept de paquetage est illustré ci-dessous.

Paquetage_entité_protocole	**PACKAGE**	
BEHAVIOUR Comportement_Classe_Entité;		
ATTRIBUTE	Taille_Max_PDU	GET
	<nom d'attribut et statut (lecture, écriture)>	
	Nb_PDUs_Entrée	GET;
	Nb_PDUs_Sortie	GET;
	Nb_PDUs_Entrée/s	GET;
	Nb_PDUs_Sortie/s	GET;
	Nb_PDUs_Perte	GET;
	Nb_PDUs_Erreur	GET;
	Nb_PDUs_Perte/s	GET;
	Taux_Perte	GET;
	Taux_Erreur	GET;
ATTRIBUTE GROUPS	groupe_compteur;	
	groupe_compteurs;	

La syntaxe d'un attribut est un type ASN.1 qui décrit la façon dont les instances de valeur d'attribut sont transportées dans le protocole de gestion. A chaque attribut sont associés des droits d'accès GET, REPLACE ou GET-REPLACE signifiant que l'attribut est accessible en lecture, écriture, ou lecture-écriture. Un attribut peut être multivalué, c'est à dire posséder un ensemble de valeurs. Dans ce cas les deux droits

Taille_Max_PDU **ATTRIBUTE** <Nom de l'attribut>
WITH ATTRIBUTE SYNTAX
MATCHES FOR EQUALITY, ORDERING <Opération pouvant être réalisées sur l'attribut>
REGISTERED AS {exemple_attribut_1} <Nom d'enregistrement>

d'accès ADD et REMOVE permettent d'ajouter ou de retirer une valeur à/de la liste de valeurs de l'attribut.

- **Groupe d'attributs:**

Un groupe d'attributs est un moyen de référencer un ensemble d'attributs ayant la même sémantique.

Groupe_compteur **ATTRIBUTE GROUP** <Nom du groupe d'attribut>	
GROUP ELEMENTS	Nb_PDUs_Entrée, Nb_PDUs_Sortie, Nb_PDUs_Perte, Nb_PDUs_Erreur
DESCRIPTION	<Groupe d'attributs contenant tous les attributs relatifs à des compteur de PDUs traitées par l'entité protocolaire. Ces compteurs sont incrémentés depuis l'initialisation du protocole>

• **Notifications:**

Les objets de gestion peuvent émettre des notifications à chaque occurrence d'événements détectés. Elles contiennent des informations relatives à ces événements survenus de façon soit interne soit externe.

```
seuil_perte_dépassé NOTIFICATION
BEHAVIOUR Comportement_seuil_perte_dépassé
WITH INFORMATION SYNTAX <ModuleNotification.InfoPerte>
AND ATTRIBUTE IDS          Taux_Perte/s, Taux_Perte
REGISTERED AS {exemple_Notification_1}
```

• **Opérations:**

Afin de manipuler les objets, deux types d'opérations ont été définies: les premières applicables aux attributs de l'objet, les deuxièmes applicables à l'objet lui-même.

Les opérations sur les attributs sont émises aux objets de gestion, afin de lire ou de modifier les valeurs d'attribut. Selon leur structure, les attributs peuvent être soit des attributs simples, soit des attributs multi-valués, soit des attributs de groupe. Un attribut simple possède une valeur unique alors qu'un attribut multi-valué est composé d'un ensemble de valeurs qui peuvent être manipulées individuellement. Un attribut de groupe se réfère à un groupe d'attributs à l'intérieur d'une classe d'objets. Une opération sur un groupe d'attributs est réalisée sur chaque attribut de manière individuelle dans ce groupe. Différentes opérations sont possibles en fonction du type d'attribut à manipuler:

- Get attribute value;
- REPLACE attribute value;
- REPLACE-WITH-DEFAULT value;
- ADD member;
- REMOVE member.

Les deux dernières opérations ne sont applicables qu'à des attributs multivalués, alors que les deux premières sont les seules à pouvoir s'appliquer à des attributs de groupe.

Une opération GET est translatée en un service CMIS M-GET. Les opérations REPLACE, REPLACE-WITH-DEFAULT, ADD, et REMOVE sont traduites en un service CMIS M-SET.

Les opérations sur les objets incluent la création d'un objet (CREATE), la suppression d'un objet (DELETE), l'exécution d'une action par un objet (ACTION).

L'opération de création demande la création et l'initialisation d'un objet de gestion. Cette opération est unique puisqu'elle s'applique à un objet qui n'existe pas encore. Il est de la responsabilité du processus de gestion système dans le système créé, de créer cet objet. L'équivalent CMIS est M-CREATE.

L'opération de suppression s'applique à tous les objets de gestion qui peuvent être supprimés par une opération de gestion. A l'exécution de

cette opération, si l'objet de gestion contient d'autres objets de gestion, soit cet objet détruit l'ensemble des objets contenus pour garantir l'intégrité du nommage, soit il refuse la suppression tant que les objets contenus n'ont pas été préalablement supprimés. Le formulaire lien de tous les objets contenus définit le comportement de l'objet lors d'une suppression. L'opération DELETE est translatée en un service CMIS M-DELETE. Le mode des opérations CREATE et DELETE est confirmé. Quant à l'opération ACTION, elle est utilisée afin de définir des opérations arbitraires. La définition de ces actions et l'information nécessaire afin de les exécuter, font partie de la spécification de la classe d'objets correspondante. Ce mécanisme est prévu afin de réaliser des activités de gestion complexes. Le service CMIS équivalent est M-ACTION.

- **Comportement:**

La description du comportement d'un objet de gestion spécifie les caractéristiques dynamiques d'un objet et de ses attributs, les circonstances pour lesquelles les notifications doivent être émeises et les actions. GDMO fournit un formulaire pour la spécification du comportement d'un objet de gestion, en langage naturel.

Contenance et nommage

Pour faciliter la supervision et le contrôle des ressources à travers les MOs qui les représentent, il est nécessaire de hiérarchiser les MOs. Ces derniers sont structurés selon une relation de contenance (voir figure 2.1).

Un MO est contenu dans un autre MO tel qu'un sous-répertoire est contenu dans un répertoire. Une structure appelée arbre de contenance est déduite à partir de cette relation entre objets. Les MOs sont nommés sur la base de cet arbre: un objet doit être nommé de façon non ambiguë pour l'identifier et le référencer. Le nommage se fait en fonction de la position du MO dans l'arbre de contenance, grâce à une séquence de noms appelée « nom de distribution relatif » ou RDN (*Relative Distinguished Name*). L'arbre de contenance permet aussi au gestionnaire d'exercer un contrôle hiérarchique sur les ressources à gérer.

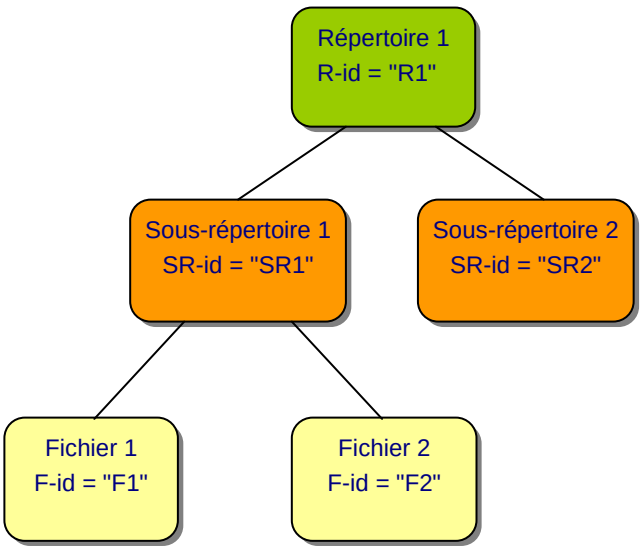


Fig. 2.2: l'arbre de nommage

L'enregistrement

Le nom du formulaire est utilisé pour désigner de manière unique un type d'information au sein du document dans lequel il est déclaré. L'identifiant du type d'information ou `OBJECT IDENTIFIER` est attribué lors de l'enregistrement du type d'information. Il est unique au monde, il est véhiculé dans les protocoles de communication et il permet de reconnaître un type donné parmi tous les types enregistrés. Les identifiants sont délivrés par des autorités d'enregistrement, organisations habilitées à enregistrer des types d'information comme l'ISO ou l'ITU-T. Les identifiants correspondent à une position dans un arbre d'enregistrement (*Registration Tree*).

Modèle informationnel de la communauté Internet

Pour la gestion de la communauté Internet, l'IETF (*Internet Engineering Task Force*) a défini trois RFCs :

- RFC 1155: décrit la structure et le nommage de l'information de gestion, à savoir le SMI, avec une extension définie dans le RFC 1212,
- RFC 1157: définit le protocole SNMPv1 utilisé pour accéder via le réseau aux objets gérés,
- RFC 1213: décrit la base d'information de gestion (MIB II), ce RFC enrichit le RFC 1066 qui décrit une première version de la MIB, MIB1.

Le modèle informationnel défini par la communauté Internet est simple afin de fournir rapidement une infrastructure opérationnelle de gestion de réseaux. Cette simplicité entraîne évidemment des faiblesses conceptuelles dans le modèle.

Structure de l'information de gestion (SMI)

La SMI (*Structure of Management Information*) définit comment chacun des éléments d'information, qui concernent les équipements gérés, est représenté dans une base d'information d'administration, la MIB (*Management Information Base*). Les objets gérés sont accessibles grâce à la MIB. Les objets contenus par la MIB sont définis en utilisant le langage ASN.1 (*Abstract Syntax Notation One*). Chaque type d'objet a son nom, sa syntaxe et son encodage.

- **Nom d'objet:**

Les noms sont utilisés pour identifier les objets gérés. Chaque objet possède un identificateur d'objet qui lui est propre. Ce dernier a une structure hiérarchique, arborescente. C'est une séquence de nombres entiers qui parcourent un arbre (voir figure 2.2). L'IETF utilise l'arbre d'enregistrement de l'OSI afin de nommer l'information de gestion. Cet arbre est composé d'une racine à laquelle sont liés tous les nœuds. Chaque nœud est identifié de manière unique. Le nœud-racine de l'arbre n'est pas indiqué, mais il possède trois fils, à savoir l'ITU-T marqué ccitt(0), le second administré par l'ISO et marqué iso(1) et iso-ccitt(2) administré par les deux à la fois.

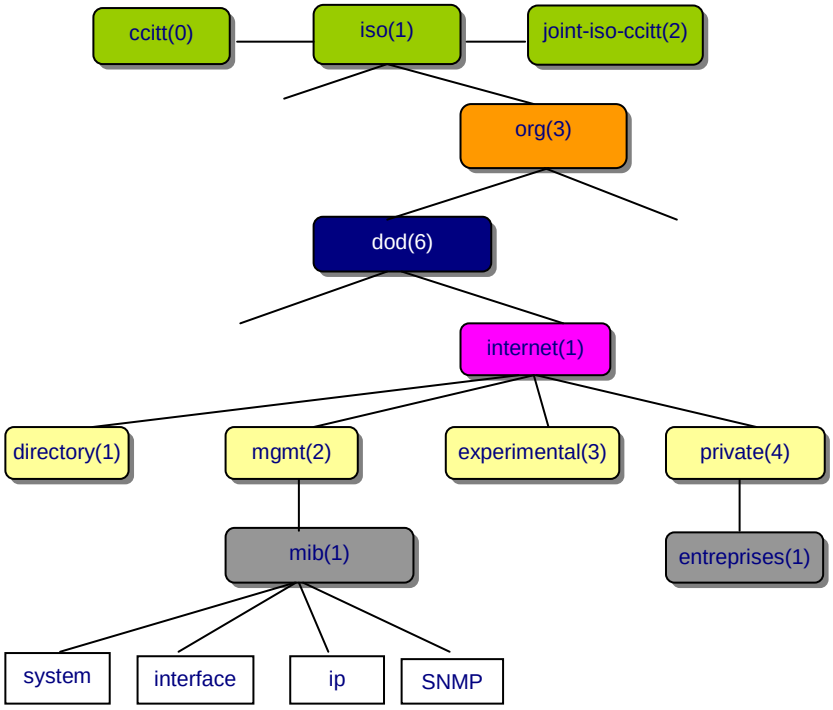


Fig. 2.3: la hiérarchie d'enregistrement dans l'internet

Ainsi l'identificateur d'Internet est: `internet OBJECT IDENTIFIER = {iso(1) org(3) dod(6) 1}`. Chacun des objets dans le sous-arbre `internet` possède un identificateur dont le préfixe est: 1.3.6.1.

- **Syntaxe:**

La syntaxe est utilisée pour définir la structure de donnée qui correspond au type d'objet. Un type d'objet correspond grossièrement à champ type d'une classe d'objets. Un ensemble limité de constructeurs ASN.1 sont utilisés afin de définir cette structure. Les types primitifs ASN.1 considérés sont "`Integer`", "`Octet String`", "`Object Identifier`", et "`Null`". Les types composés sont "`Sequence`" et "`Sequence Of`". De plus de nouveaux types ont été définis: "`Network Adress`" afin de représenter des adresses de protocole, "`IP Adress`" pour représenter des adresses Internet 32 bits, "`Counter`" pour définir des compteurs de type entier appartenant à l'intervalle[0, 2^{32}-1], "`Gauge`" pour caractériser un entier positif ne dépassant pas une taille maximale, "`Timeticks`" afin de compter le temps en centièmes de seconde, et finalement "`Opaque`" qui représente une syntaxe quelconque encodée sous forme d'un `OCTET STRING`.

- **Encodage:**

Lorsque l'instance d'un type d'objet a été identifiée, sa valeur peut être transmise en appliquant les simples règles d'encodage d'ASN.1 à la syntaxe de type d'objet. Pour décrire un objet, le SMI définit le formulaire suivant:

Object: un nom textuel pour nommer l'objet.

Syntax: la syntaxe abstraite pour le type d'objet, présentée en utilisant ASN.1.

Definition: une description textuelle de la sémantique du type d'objet.

Access: *read-only*, *read-write*,*write-only* ou *not accessible*.

Status: *mandatory*, *optional* ou *obselete*.

La base d'information de gestion (MIB)

SNMP définit une collection d'objets "standard" à administrer à travers la spécification des MIB-I et MIB-II. Les objets définis par SNMP ont une structure particulièrement simple. En effet, la définition d'un objet est limitée à un type simple ou à une table d'objets de types simples.

La MIB-I correspond au premier lot de définitions d'objets SNMP. Elle contient une centaine d'objets, rangés par groupes fonctionnels au nombre de huit: *System* décrit le nœud géré, *Adress Translation* définit la mise en correspondance entre adresses réseaux (par exemple, IP) et adresses physiques (par exemple, MAC), *Interface* contient un ensemble d'informations utiles pour la gestion des ports/interfaces du nœud géré, *IP* fournit un ensemble de données nécessaires pour suivre l'état de l'entité IP, ce groupe propose deux tables: IPAddrTable qui contient des informations d'adressage relatives aux adresses IP du nœud courant et IPRoutingTable qui contient des informations sur les adresses IP destinataires telle que l'adresse du nœud suivant, le groupe *TCP* offre des informations sur les connexions TCP, *ICMP* fournit un ensemble de compteurs ou statistiques sur les messages ICMP, *UDP* traite les datagrammes UDP, *EGP* contient des informations sur les entités voisines EGP et sur l'état d'EGP à travers des compteurs. Ces huit groupes permettent de gérer uniquement un réseau TCP/IP. Certains de ces groupes sont obligatoires (les cinq premiers), d'autres sont considérés comme obligatoires si l'équipement à gérer fonctionne avec un protocole particulier. Par exemple si un routeur utilise EGP, alors le groupe EGP est obligatoire. Afin de descendre plus finement dans les fonctionnalités d'un équipement, de nouveaux objets ont été ajoutés progressivement aux groupes existants, définissant ainsi la MIB-II.

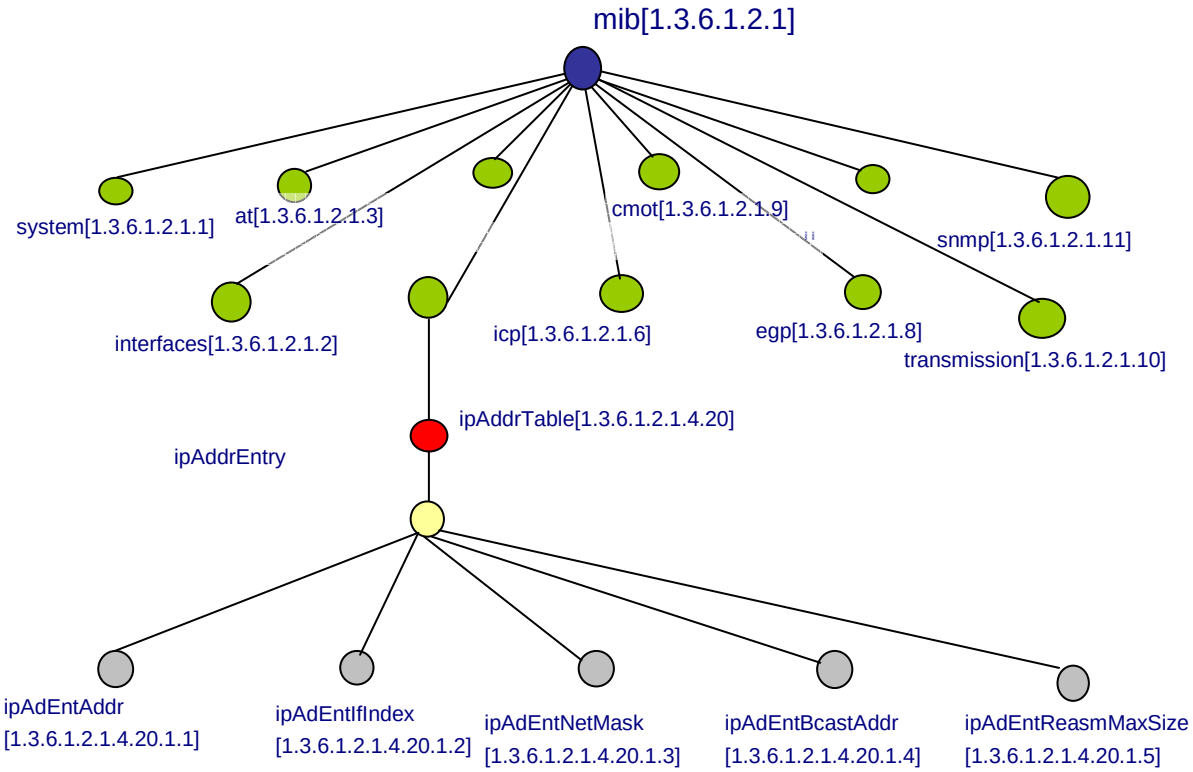


Fig. 2.4: l'arbre d'enregistrement de la MIB 2

De plus de nouveaux groupes ont été ajoutés tels que SNMP, qui possède trente objets de types simples utilisés pour gérer les performances du protocole de gestion. La figure 2.4 illustre l'arbre d'enregistrement de la MIB-II. Par ailleurs il faut des outils pour rendre les MIBs conviviales. Comme toute base de données, elle requiert des outils pour la rendre accessible. Généralement, ces outils sont proposés sur les plates-formes. Ces dernières peuvent offrir un compilateur de MIB pour avoir le bon format, un navigateur qui affiche l'arbre de la MIB de manière graphique et qui permet de rechercher les objets et un générateur d'état de MIB également graphique.

La MIB RMON

L'horizon des MIB-I et MIB-II a été largement agrandi par l'introduction de la MIB RMON [Waldbusser 91]. Cette dernière apporte de nouvelles fonctionnalités telles que statistiques, historiques, détection de seuils d'alarme, gestion des hosts, estimation de flux, filtrage de capture de paquets, gestion des notifications d'événements… qui donnent un rôle plus important à l'agent qui exécute des tâches plus complètes pour décharger le travail du gestionnaire.

La MIB RMON réalise deux tâches: tout d'abord elle fournit une vue d'ensemble pour la gestion d'un réseau à travers une surveillance à distance. Les MIBs précédentes se focalisaient sur la gestion d'équipements (routeurs, ponts, stations, etc.) avec une emphase minimale sur la gestion du réseau dans son ensemble. C'est ce dernier point que traite RMON (voir figure 2.5). En second, elle spécifie les particularités de la gestion d'un réseau Ethernet. Les concepteurs de la MIB RMON ont choisi au départ ce réseau, mais depuis, des MIBs RMON pour les sous réseaux Token Ring et FDDI ont vu le jour.

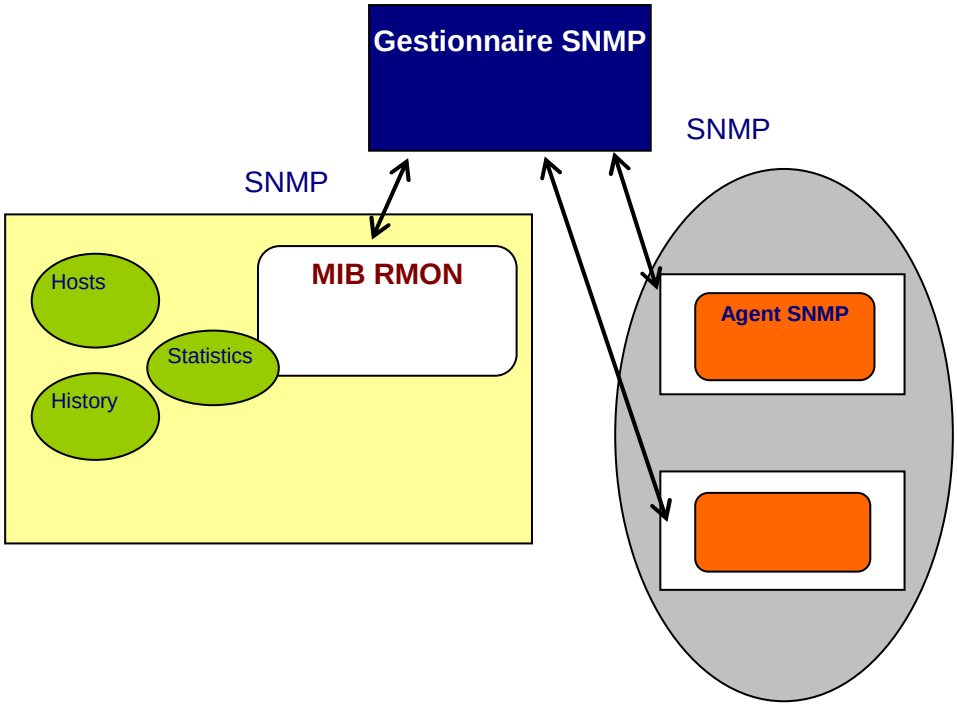


Fig. 2.5: MIB RMON

La MIB RMON définit les neuf groupes suivants:

- le groupe"*Statistics*" met à jour des statistiques mesurées par l'agent pour chaque interface supervisée. Ces statistiques comprennent les paquets, les octets, les diffusions, ainsi que les collisions sur le segment local, aussi bien que le nombre de messages rejetées par l'agent;
- le groupe "*History*" comprend des objets utilisés pour enregistrer des informations à des fins d'analyse par le gestionnaire;
- le groupe "*Alarm*" fournit un mécanisme pour définir des seuils;
- le groupe"*Host*" contient des statistiques sur chaque host générant du trafic sur le réseau;
- le groupe "*HostTopN*" étend le groupe *Host* en fournissant des statistiques triées;
- le groupe "*Matrix*" contient une matrice de trafic au niveau de la couche MAC du sous-réseau Ethernet. Cette matrice indique le trafic et le nombre d'erreurs par paire de nœuds du réseau, une paire étant représentée par un nœud émetteur et un nœud destinataire;
- le groupe "*Filter*" permet de définir des filtres de capture de paquets;
- le groupe "*Packet capture*" permet la capture de paquets si adéquation avec le filtre;
- le groupe "*Event*" contrôle la génération et la transmission d'événements.

Pour plus de détails sur ces groupes, se rapporter à [Waldbusser 91].

SNMPv2

SNMPv2 a vu le jour en 1993, mais il ne fait pas l'unanimité de la communauté Internet, qui trouve certains aspects trop complexes à mettre en œuvre. Néanmoins, SNMPv2 se veut être une amélioration de SNMP. On y trouve un nouveau protocole de sécurité, le chiffrement (optionnel), des communications de gestionnaire à gestionnaire, un transfert d'informations de masse, de nouveaux objets MIB, la capacité

d'ajouter ou de supprimer des lignes de tables et de nouveaux types de données SMI.

Le SMI de SNMPv2, décrit au sein du RFC 1442 est divisé en trois parties: définition des modules, définition des objets et définition des traps. Les définitions de modules sont utilisées afin de décrire des modules d'information. Une macro ASN.1, `MODULE-IDENTITY` est utilisée pour préciser la sémantique d'un objet géré. Les définitions d'objets sont utilisées afin de décrire les objets de gestion. Les définitions de notifications (des traps) sont utilisées afin de décrire des émissions non sollicitées d'informations de gestion. Une macro ASN.1, `NOTIFICATIONS-TYPE` est prévue à cet effet.

En résumé, les différences entre les deux approches OSI et SNMP sont profondes. Alors que le modèle informationnel proposé par l'ISO se fonde sur l'approche orientée objet, les objets définis par l'IETF dans le monde Internet sont décrits sous forme de variables simples mises dans un arbre d'enregistrement. Les avantages et les inconvénients de chacune des solutions sont que:

- Le modèle informationnel de l'ISO est plus complexe et difficile à appliquer. Des outils appropriés doivent être utilisés afin de créer des librairies d'objets de gestion OSI;
- Les propriétés d'héritage et d'abstraction associées à l'orienté objet, peuvent être utilisées afin de créer une structure d'objet bien définie;
- Les nombreuses librairies d'objets associées aux MIBs créées par l'IETF montrent clairement qu'elles ne fournissent pas de structuration suffisante de l'information. En particulier l'absence de mécanisme d'héritage pour la réutilisation signifie que des informations dérivées les unes des autres peuvent être contenues dans des sous-arbres différents de l'arbre d'enregistrement.

Modèles architecturaux de gestion TMN - SNMP

Une architecture est une structure d'ensemble, c'est à dire une définition des règles de composition du système et de la coopération des composants à des fins de transparence pour l'utilisateur du service fournit par cette architecture. Une cohérence parfaite des différentes parties du réseau et une grande modularité sont nécessaires, car chaque partie doit pouvoir être modifiée ou remplacée sans affecter les autres. L'ensemble matériel et logiciel s'analyse à travers une architecture physique du réseau et une architecture logique du réseau. Les trois grandes fonctions à travers lesquelles un réseau rend un service global, délimitent des plans, à savoir:

- Plan usager pour le transfert des informations des applications usagers;
- Plan de contrôle pour la signalisation;
- Plan de gestion pour toutes les applications d'administration.

Cadre architectural de la gestion TMN

Les normes de gestion TMN traduisent une approche système de la gestion de réseau. Elles définissent les éléments nécessaires à la

spécification de la gestion des ressources de communication d'un système.

Du point de vue architectural, la figure 2.6 explicite la couche application pour le service de gestion. La couche application où se trouve la gestion système est la couche la plus haute dans l'architecture OSI. Les processus de communication qui se trouvent dans cette septième couche sont d'une telle diversité que la structure de la couche application a été normalisée (ISO 9545) de façon modulaire. Le concept de base de la couche application est le processus d'application qui regroupe l'ensemble des fonctions nécessaires à la réalisation d'une application donnée.

La gestion système est assurée par les processus d'administration de système SMAP (*Systems Management Application Process*) qui communiquent entre eux grâce à des entités d'application de gestion système SMAE (*Systems Management Application Entity*).

Ces dernières se composent de domaines de gestion (SMFA, *System Management Functional Area*), ce sont des composants du niveau applicatif qui s'appuient, comme toutes les applications, sur les éléments de service d'application (ASE, *Application Service Element*) de base. Ce sont ACSE (*Association Control Service Element*) qui gère l'association à travers laquelle se déroule l'échange de données applicatives et ROSE (*Remote Operations Service Element*) qui est l'élément de service d'opérations distantes. Pour les applications de gestion on a recours à CMISE (*Common Manangement Information Service Element*) pour effectuer les échanges entre entités de gestion système.

La gestion de système OSI apparaît donc comme une application pour laquelle il faut exprimer le profil fonctionnel et la QoS désirée pour le flux (lien) de gestion, afin de choisir le réseau de communication le plus adéquat. En effet, CMIP exécute les échanges réels entre un gestionnaire (un manager) et un agent de système géré. Il utilise une MIB pour connaître les possibilités de l'agent géré et en avoir une image.

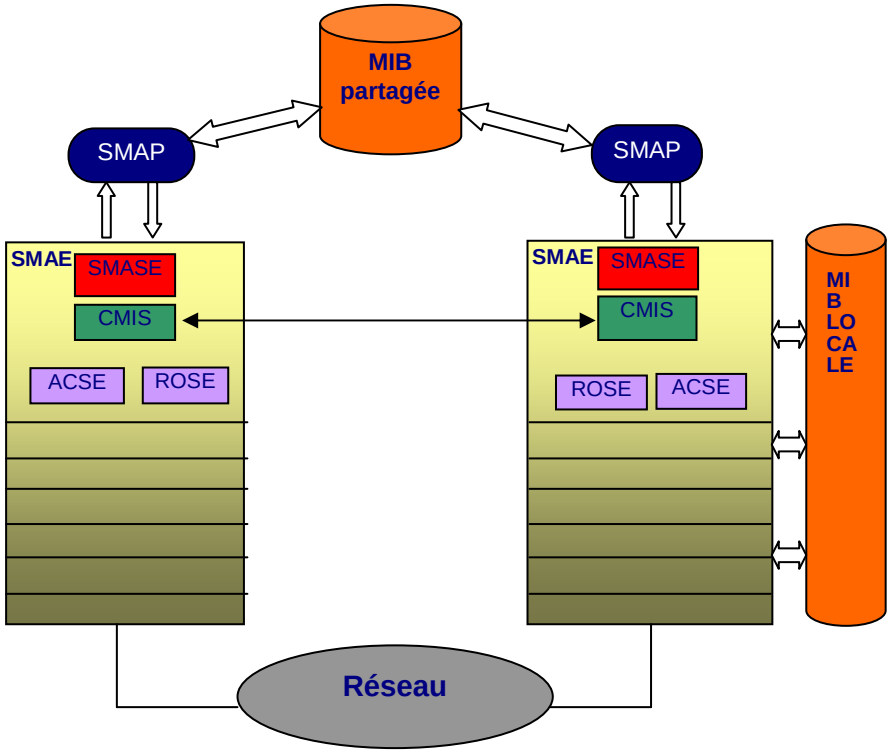


Fig. 2.6: Cadre architectural et gestion de composants réseau préconisée par l'OSI

Cadre architectural de la gestion SNMP

Par abus de langage, le protocole SNMP (*Simple Network Management Protocol*) couvre l'ensemble des standards de gestion préconisés par la communauté Internet (IETF). En général, ces standards sont surtout utilisés dans un contexte LAN alors que le cadre architectural de l'ISO s'applique dans des contextes WAN.

La communauté Internet recommande SNMP pour mettre en œuvre rapidement des moyens de gestion. Les standards SNMP, établis depuis 1990, sont axés autour de deux concepts:

- la définition d'un protocole d'échange SNMP qu'on détaillera dans le paragraphe 4.3;
- la définition des informations d'administration des MIBs qu'on a déjà développées.

Le protocole SNMP définit essentiellement les règles permettant l'envoi de messages d'administration entre des "gestionnaires" et des "agents". Un agent est un logiciel opérant à l'intérieur d'un équipement à gérer (terminal, serveur de terminaux, passerelle, pont, routeur, unité centrale, etc.) alors qu'un gestionnaire est un logiciel résidant dans une station de gestion de réseaux NMS (*Network Management Station*), et a la possibilité d'adresser des requêtes vers des agents. Le protocole SNMP fonctionne en mode non connecté et ne nécessite pas de couche de transport fiable, d'où l'utilisation d'UDP (*User Datagram Protocol*). Ceci permet de définir le cadre architectural (voir figure 2.6) représentatif des standards de faits de la communauté Internet.

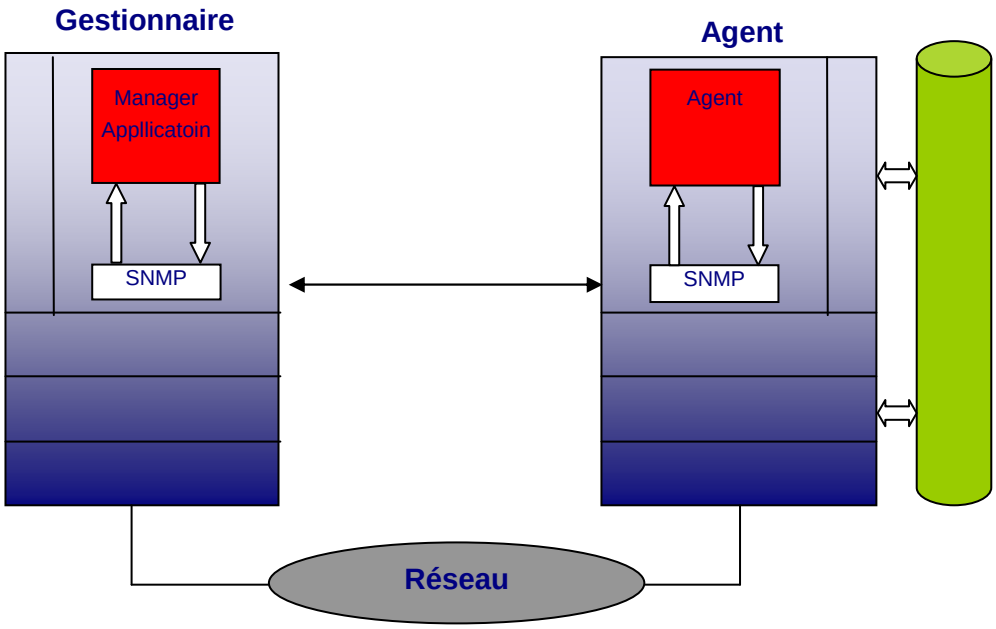


Fig. 2.7: Architecture pour la gestion de composants réseau préconisée par l'IETF

Comparaison des modèles de communication TMN et IETF

Le modèle de communication permet d'appréhender les échanges entre les composants d'un système, dans leur dimension qualitative et quantitative. Plus que dans tout autre domaine, les flux, les échanges, sont la base, le cœur des réseaux. C'est l'abstraction de ce lien qui a conduit à la modélisation générique "nœud, lien, réseau" de notre contexte réseau global (système distribué). C'est par l'étude des échanges de messages que les structures, les architectures se conçoivent.

Le modèle de communication du TMN

Le modèle de communication TMN identifie, comme tous les modèles de communication, les échanges entre les composants architecturaux. Les dialogues entre deux entités fonctionnellement différentes dans un même système est de type client/serveur. Ce dialogue entre deux entités adjacentes se dénomme service en général, et plus exactement CMIS pour la gestion. Quant au dialogue entre deux entités fonctionnellement équivalentes entre deux systèmes supporte les règles régissant la mise en relation, ce dialogue c'est le protocole. Celui associé aux entités CMIS est la spécification CMIP. L'administration est une application, elle se situe donc en couche 7 du modèle de référence. Le flux applicatif s'appuiera sur les services rendus par les couches inférieures et par les autres services de la couche 7.

Le service CMIS

CMIS est l'ensemble des fonctions communes de la gestion système permettant de réaliser le transfert des opérations et des notifications de gestion. Il est offert par l'entité de service CMISE de la couche application. L'élément CMISE se compose de quatre entités distinctes:

- cœur de CMISE, la machine protocolaire CMIPM (*Common Management Information Protocol Machine*) qui met en œuvre le protocole et gère les unités de données du protocole CMIP;
- le fournisseur de service CMISE qui est chargé de recevoir les primitives CMISE et de transmettre les unités de données de service correspondantes (*Service Data Unit, SDU*) à CMIPM;
- l'utilisateur de service ROSE qui gère les requêtes faites à ROSE. Elles permettent d'échanger les PDUs entre deux protocoles CMIP;

Enfin, un gestionnaire de d'entité qui gère les primitives non normalisées de communication avec la fonction de contrôle d'association simple (SACF).

L'entité CMISE distingue les six services d'opérations (qu'elle peut traiter sans aucune connaissance du contenu, laquelle est laissée à SMASE) et un service de notification, qui sont:

- `M_CREATE` pour demander la création d'un objet dans la MIB de l'agent (Mode confirmé);
- `M_DELETE` pour supprimer des objets dans la MIB de l'agent (Mode confirmé);
- `M_ACTION` pour demander l'exécution d'une action sur un ou plusieurs objets (confirmé ou non);
- `M_SET` pour modifier les valeurs d'attributs d'objets (confirmé ou non);
- `M_GET` pour consulter les valeurs associées aux attributs d'objets (confirmé ou non);
- `M_CANCEL_GET` pour demander de ne pas recevoir les résultats d'un GET précédent (Mode confirmé);
- `M_EVENT_REPORT` pour transmettre un rapport d'événement relatif à un objet (confirmé ou non).

A ces sept primitives s'ajoute la possibilité de disposer d'une réponse multiple, sans création de nouvelle primitive, et de moyens de sélection d'objets de gestion par un mécanisme de filtrage en profondeur. Ces extension sont disponibles pour les primitives `M_DELETE`, `M_ACTION`, `M_SET` et `M_GET`:

- La sélection de l'objet géré se fait en deux phases: la réduction de l'espace de choix (*scoping*) et le filtrage (*filtering*). Les noms des instances d'objets gérés sont organisés hiérarchiquement dans un arbre d'information de gestion, le MIT (*Management Information Tree*).
- Le scoping consiste à identifier un ensemble d'objets: permet de définir un objet de base, correspondant à la racine d'un sous-arbre de l'arbre d'information de gestion et spécifie le niveau de *scoping* (objet de base seul, les descendants de l'énième niveau…).
- Le *filtering* consiste à appliquer des tests à chacun des objets de l'ensemble sélectionnés par le *scoping* pour extraire un sous-ensemble: teste la présence ou la valeur d'attributs de l'objet, à l'aide d'une ou de plusieurs assertions groupées par des opérateurs logiques.

Un paramètre de synchronisation permet à un utilisateur du service de spécifier comment les opérations sur les instances d'objets gérés doivent être synchronisées, quand plusieurs objets ont été sélectionnés par le *scoping* et le *filtering*. Ce paramètre peut prendre deux valeurs:

- *atomic:* vérification de la possibilité d'exécution sur tous les objets, s'il existe une ou des impossibilités, aucune n'est exécutée, sinon toutes sont exécutées.
- *Best effort*: exécution sur tous les objets où elle est possible, c'est la valeur par défaut.

Le protocole CMIP

Pour présenter le protocole CMIP qui se base sur les états de fonctionnement de la machine protocolaire CMIPM (*CMIP Machine*), il faut connaître la structure de CMISE (voir figure 2.7). Une instance de ce dernier est composée d'une machine protocolaire CMIPM, d'un fournisseur de service CMISE (*CMISE-provider*), d'un utilisateur de service ROSE (*ROSE-user*), d'un gestionnaire et de trois SAPs, *SAP-cmise-smase, SAP-cmise-rose, SAP-cmise-sacf.*

La machine CMIPM met en œuvre les éléments de procédure du protocole CMIP et gère les unités de protocole CMIP (CMIP-PDUs). *CMISE-provider* gère les primitives de service CMISE. Il reçoit les primitives CMISE de requête et de réponse provenant du *CMISE-user* de SMASE et en transmet les unités de service CMISE (CMISE-SDUs) à CMIPM. Il émet des primitives CMISE d'indication et de confirmation à l'intention du *CMISE-user* de SMASE lorsque CMIPM lui demande de transférer des CMISE-SDUs.

ROSE-user gère les primitives de service ROSE. Il reçoit des primitives ROSE d'indication provenant du *ROSE-provider* de ROSE et en transmet les unités de service ROSE (ROSE-SDUs) à CMIPM. Il émet des primitives ROSE de requête à l'intention du *ROSE-provider* de ROSE lorsque CMIPM lui demande de transférer des ROSE-SDUs.

Le gestionnaire gère les primitives non normalisées d'interaction avec SACF (*Single Association Control Function*) chargé de coordonner le travail des ASEs (*Application Service Element). SAP-cmise-smase* représente le point d'accès aux services de CMISE. C'est l'une des extrémités de la connexion qui relie SMASE à CMISE. Par cette connexion transitent les primitives CMISE entre ces deux éléments. *SAP-cmise-rose* représente le point d'accès aux services de ROSE. Par cette connexion transitent les primitives ROSE entre ces deux éléments. *SAP-cmise-sacf* représente le point d'accès aux services non normalisés offerts par CMISE à la fonction SACF.

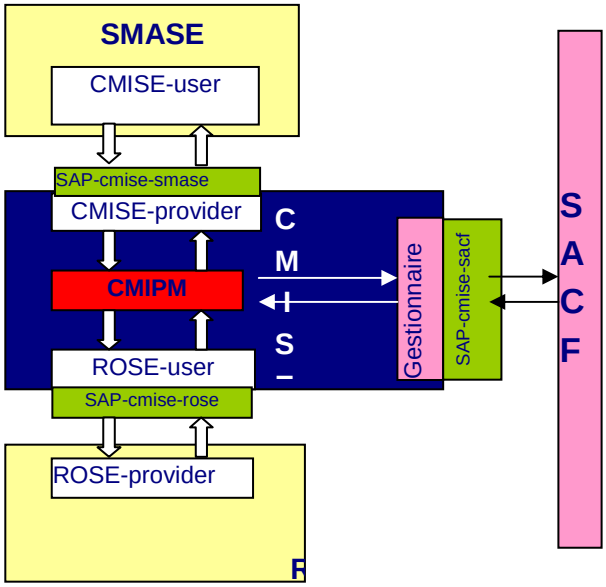


Fig. 2.8: Composition de CMISE

- **Etats de la machine CMIP:**

Une machine protocolaire est décrite sous la forme d'un automate. Celui-ci se présente comme une boite noire sollicitée par des événements extérieurs, qui déclenche un traitement interne spécifique pour chaque événement reçu en fonction de l'état dans lequel elle se trouvait au début du traitement, il génère des événements sortants et change éventuellement d'état à la suite de chaque traitement déclenché. La description du protocole CMIP par la norme [ISO 9596] permet d'identifier six états de fonctionnement de CMIPM:

- IDLE: c'est l'état de repos de CMIPM. Dans cet état, le protocole est inactif;
- ASSOCIATED: c'est l'état de fonctionnement normal de CMIPM lorsque l'association d'application est établie. Il est prêt à accepter des primitives CMISE;
- WCFESTA: CMIPM est en attente d'une confirmation d'établissement d'association que doit rendre l'entité SMAE distante;
- WRPESTA: CMIPM est en attente d'une réponse d'établissement d'association que doit rendre son utilisateur, c'est à dire SMASE;
- WCFTERM: CMIPM est en attente d'une confirmation de terminaison d'association que doit rendre l'entité SMAE distante;
- WRPTERM: CMIPM est en attente d'une réponse de terminaison d'association que doit rendre son utilisateur SMASE.

Le protocole CMOT

Le protocole CMOT est la migration du système de gestion de réseaux TCP/IP vers la normalisation. Le groupe qui travaille sur CMOT (*CMIP over TCP/IP*) s'est intégré à l'OSI sous le nom de OIM (*OSI Internet Management*). Son principal rôle est de produire un ensemble de spécifications qui offrent les services CMIS et le protocole CMIP sur des réseaux de type TCP/IP. L'architecture envisagée pour CMOT consiste à rajouter au-dessus des protocoles TCP et UDP une couche présentation simplifiée LPP (*Lightweight Presentation Protocol*), les services ACSE, ROSE, CMIP, et CMIS. Avec CMOT, la relation gestionnaire agent fonctionne en mode connecté et la MIB est constituée d'un ensemble

d'objets comme pour la gestion OSI. Cependant, bien que la démarche semble fort intéressante, il n'y a pas à l'heure actuelle de véritable implémentation de ce protocole sur les diverses plates-formes commerciales.

Le modèle de communication de l'IETF

SNMP (*Simple Network Management Protocol*) est né dans la communauté Internet des milieux de la recherche et de la défense américaines. Recommandé depuis avril 1988 comme Internet Standard (RFC 1098), SNMP s'est imposé comme standard de "fait" pour les réseaux TCP/IP. Aujourd'hui, il est implanté sur les produits de nombreux constructeurs comme Cisco, Sun, Digital, IBM, Proteon, Hewlett-Pachard, Unisys, 3COM, etc.

SNMP repose sur une collecte et un échange de données entre, d'un côté, des stations d'administration de réseau et, de l'autre, des éléments de réseau gérés ou agents. Ces derniers sont des équipements de réseau tels que les concentrateurs Ethernet, ponts, routeurs, contrôleurs de terminaux, machines serveurs, dans l'environnement TCP/IP. Chaque élément géré contient une base d'informations de gestion MIB, maintenue en temps réel. Le protocole SNMP est utilisé pour lire, modifier ces informations, intervenir sur les équipements à distance ou signaler certains événements en temps réel. Les informations recueillies par la station d'administration sont stockées généralement dans une base de données relationnelle. Ces informations peuvent ensuite être analysées par des outils tels que les systèmes experts ou des outils de représentation graphique.

Le protocole SNMP

Les opérations de gestion de réseau sont architecturées en pseudo transactions "question-réponse" entre la station d'administration et les agents SNMP. La station de gestion visualise et contrôle les équipement par l'interrogation des agents SNMP. Pour l'obtention d'information de gestion. Les agents répondent simplement aux interrogations en renvoyant ou en changeant les données actuelles contenues dans les MIBs de leurs équipements. Le protocole SNMP possède cinq types de messages:

- trois requêtes :
  - **GetRequest (liste d'objets)**: permet de lire les objets de la MIB. Emise par le gestionnaire, cette commande est ensuite analysée par l'agent qui consulte dans la MIB les objets en argument de la primitive GetRequest. L'agent répond au gestionnaire par l'envoi d'une primitive GetResponse contenant la valeur des objets demandés.
  - **GetNextRequest (liste d'objets):** permet de faire une lecture séquentielle des informations dans la MIB. Cette commande est particulièrement utilisée pour la lecture des tables dans la MIB. Après avoir lu un premier enregistrement de la table avec la requête Get ou GetNext, les autres enregistrements de la table sont lus de manière séquentielle par une série de GetNext.

• **SetRequest (liste d'objets et leurs valeurs):** permet de modifier des objets dans la MIB. A la réception de cette commande l'agent met à jour les variables de la MIB à partir des valeurs en argument de SetRequest. Chacune des variables doit être précisément indiquée, et la valeur doit être en accord avec la syntaxe de la variable à modifier, sinon l'agent signale une erreur.

- une réponse:

• **GetResponse (liste d'objets et leurs valeurs)**: c'est la réponse de l'agent aux primitives GetRequest, GetNextRequest et SetRequest. Sur chaque requête du gestionnaire, l'agent répond en utilisant GetResponse. Cela peut être une réponse positive (exécutant ou confirmant l'accomplissement de l'opération demandée), ou négative dans le cas d'erreur.

- un message d'événements:

• **Trap:** c'est une commande spéciale non sollicitée. Elle est émise par l'agent vers le gestionnaire sur un événement particulier spécifié à priori.

Le protocole SNMPv2

Les opérations du protocole SNMPv2 sont au nombre de cinq: GetRequest, GetNextRequest, GetBulkRequest, SetRequest et InformRequest. Cette dernière a la particularité d'être générée d'un gestionnaire en direction d'un autre gestionnaire pour permettre une gestion hiérarchique ou distribuée. Les opérations GetRequest, GetNextRequest et SetRequest sont identiques à celles de même nom définies dans SNMP version 1. L'opération GetBulkRequest permet de récupérer les valeurs d'une suite de successeurs lexicographiques d'identificateurs d'objets. Elle a le même effet qu'une suite de message GetNextRequest, mais la bande passante utilisée est fortement réduite.

Dans SNMP version 1, la sécurité est réalisée par un seul mécanisme qui présente certaines faiblesses. Le SNMPv2 a enrichi l'aspect sécurité et fournit des messages SNMPv2 authentifiés et encryptés. L'information de sécurité est présente hors des messages SNMPv2.

En résumé de ce paragraphe, on constate que SNMP est un standard de fait pour les systèmes basés sur les protocoles TCP/IP, alors que CMIP est standardisé par l'ISO. On remarque aussi que sur beaucoup de points, SNMP et CMIP semblent être très proches, mais en fait, ils sont fondamentalement différents (voir tableau 2.1). En effet, on peut dire:

- Qu'ils ont le même objectif: transmettre des informations de gestion du réseau;
- Qu'ils utilisent une base de données (la MIB);
- Qu'ils possèdent les mêmes primitives de base (get, set) et la signalisation d'événements (*Trap* pour SNMP, *Event* pour CMIP).

CMIP/CMIS		SNMP	
Mode connecté		Mode non-connecté	
Interface-Objet Recherche d'une VALEUR Modification d'une VALEUR		Interface-Attribut Recherche d'une VALEUR Modification d'une VALEUR	
Synchronisation - atomic - Best effort		Synchronisation - Atomic seulement	
Possibilité de *scoping* et *filtering*		Non	
Gestion par événement NOTIFICATION		Gestion par polling (fort volume de trafic) Evénements possibles	
Actions particulières possibles sur les objets		Pas de possibilités d'actions particulières	
Requêtes et réponses peuvent inclure des objets complexes		Requêtes et réponses incluent des valeurs d'attributs élémentaires (compteur)	

Tableau 2.1: comparaison CMIP/SNMP

Les fonctionnalités de CMIP sont donc plus puissantes que celles de SNMP (ne pas oublier que le S de SNMP veut dire Simple) car elles peuvent s'appuyer sur une MIB où les données sont agrégées: la MIB est constituée d'un ensemble d'objets et non des attributs simples et des tables comme dans le cas des MIBs SNMP. En fait l'avantage de la simplicité de SNMP est amputé par son manque de puissance.

Chapitre 3 - Le modèle fonctionnel de la gestion TMN

La dimension fonctionnelle de la gestion de réseau et de service est la partie centrale des réponses que doit maîtriser l'administrateur, car elle représente l'ensemble des traitements à mettre en œuvre pour satisfaire les demandes de transfert des utilisateurs quelles que soient l'heure, la journée ou la période de l'année. Le réseau doit donc effectuer un transfert fiable des informations dans les temps de réponse demandés. C'est ce que l'on a défini comme étant la qualité de service (QoS) dont la charge revient au système de gestion. En d'autres termes, pour qu'un réseau soit opérationnel, il doit se doter d'un système de gestion pour contrôler ses opérations, surveiller ses performances, gérer ses pannes, ses modifications et ses configurations. Pour ce faire, une structuration des traitements est nécessaire, un modèle fonctionnel doit être ajouté aux trois autres modèles.

Les aires fonctionelles

Gestion de la configuration

Elle recouvre l'ensemble des fonctions de contrôle, d'identification, de collection et de fourniture d'informations sur les objets administrés. Elle a pour but d'aider à la réalisation d'un fonctionnement continu des services d'interconnexion. La gestion de la configuration permet en effet de démarrer, initialiser et arrêter le système, aussi positionner les paramètres du système ouvert, recueillir les informations sur l'état du système et agir sur ces états, enfin modifier la configuration du système ouvert et associer les noms aux objets administrés.

Gestion des fautes

La gestion des fautes recouvre l'ensemble des fonctionnalités qui permettent la détection, l'isolation et la correction d'une opération anormale dans un système. Les fautes proviennent de pannes de composants matériels ou logiciels. Elles se manifestent par des erreurs dans le fonctionnement du système. Ces erreurs peuvent être passagères ou persistantes. Lorsqu'une erreur est détectée, sa localisation et sa cause doivent être diagnostiquées par une analyse des informations de l'état du système. Après le diagnostic, une opération curative est menée pour permettre une reprise du fonctionnement normal du système. La gestion des fautes est une fonction administrative vitale pour assurer aux utilisateurs un niveau de services satisfaisant du système.

Gestion des performances

La gestion des performances consiste à effectuer des mesures sur le système et faire des calculs afin d'évaluer le comportement des objets

gérés et l'efficacité des activités de communications. Dans le but de planifier et analyser le système, elle offre en plus des fonctionnalités permettant de collecter des données statistiques, de maintenir et d'examiner des journaux sur l'historique de l'état du système.

Gestion de la sécurité

La gestion de la sécurité se compose des fonctions d'administration de réseaux ayant rapport avec la sécurité dans les réseaux de télécommunication. Elle permet de faire de l'authentification, le contrôle et la maintenance des autorisations et des contrôles d'accès, la gestion des clés, la maintenance et l'examen des fichiers de sécurité.

Gestion de la comptabilité

Elle a pour but d'offrir l'ensemble des fonctionnalités qui permettent d'établir les charges et d'identifier les coûts relatifs à l'utilisation des ressources gérées. Elle permet d'informer les utilisateurs des coûts encourus ou des ressources consommées. Elle fixe des limites comptables pour l'utilisation des ressources, et combine les coûts lorsque plusieurs ressources sont utilisées pour réaliser une communication de données.

Comme on l'a indiqué précédemment, CMISE est un service générique

Les Fonctions de gestion système

pour la communication d'information de gestion, mais il ne couvre pas toutes les fonctionnalités. Un ensemble de fonctions de gestion système ou SMF (*Systems Management Functions*) ont été définis par les normes de gestion système. Chaque SMF est définie par un modèle et une définition exprimée en termes de services CMISE. Chaque SMF définit à l'aide de la notation GDMO des objets de gestion spécifiés.

Les applications de gestion à savoir: gestion des fautes, de la configuration, des performances, de la sécurité et de la comptabilité; utilisent toutes les services qu'offrent les SMFs (voir figure 4.8). ces dernières enrichissent les services de CMISE. Les SMF ont été publiées sous forme de norme à l'ISO [X730 à X750] et reprises par l'ITU-T [X730 à X746].

- **Fonction de gestion d'objets:** [X730], elle définit des services génériques de manipulation d'objet. Elle décrit les services de création et de suppression d'objets gérés, d'invocation sur ces objets, de lecture et de modification d'attribut d'objets. Elle spécifie aussi les notifications à émettre afin de reporter au système de gestion la création et la suppression d'objets, ou encore le changement de valeurs d'attributs d'objets dans le système géré.
- **Fonction de gestion des états**: [X731], elle définit les moyens de contrôler l'état d'une ressource. L'état de gestion d'un objet est en fait affecté par trois facteurs:
  - son interopérabilité (état opérationnel, *operational state*): indique si la ressource est ou n'est pas physiquement installée;
  - son utilisation (état d'utilisation, *usage state*): indique si la ressource est ou n'est pas utilisée à un instant donné;

- son administration (état administratif, *administrative state*): indique la permission ou l'interdiction d'utiliser la ressource de façon logique.

Pour le besoin de la gestion, l'état d'une ressource est qualifié par plusieurs attributs définis par leurs valeurs et transitions possibles (comportements), accessibles au moins en lecture. Un type de notification est défini pour informer le gestionnaire de ces transitions.

• **Fonction de gestion des relations:** [X732], elle identifie et modélise les types de relation qui peuvent exister entre objets représentant les différentes parties d'un système. Trois catégories de relations sont définies:

- relation de contenance;
- relation unilatérale: la relation n'est exprimée que dans un des deux objets liés;
- relation réciproque: la relation est exprimée dans les deux objets liés qui peuvent s'identifier mutuellement.

Plusieurs services ont été définis afin de gérer les relations entre objets, et donc d'observer les dépendances ou influences que les objets ont les uns sur les autres.

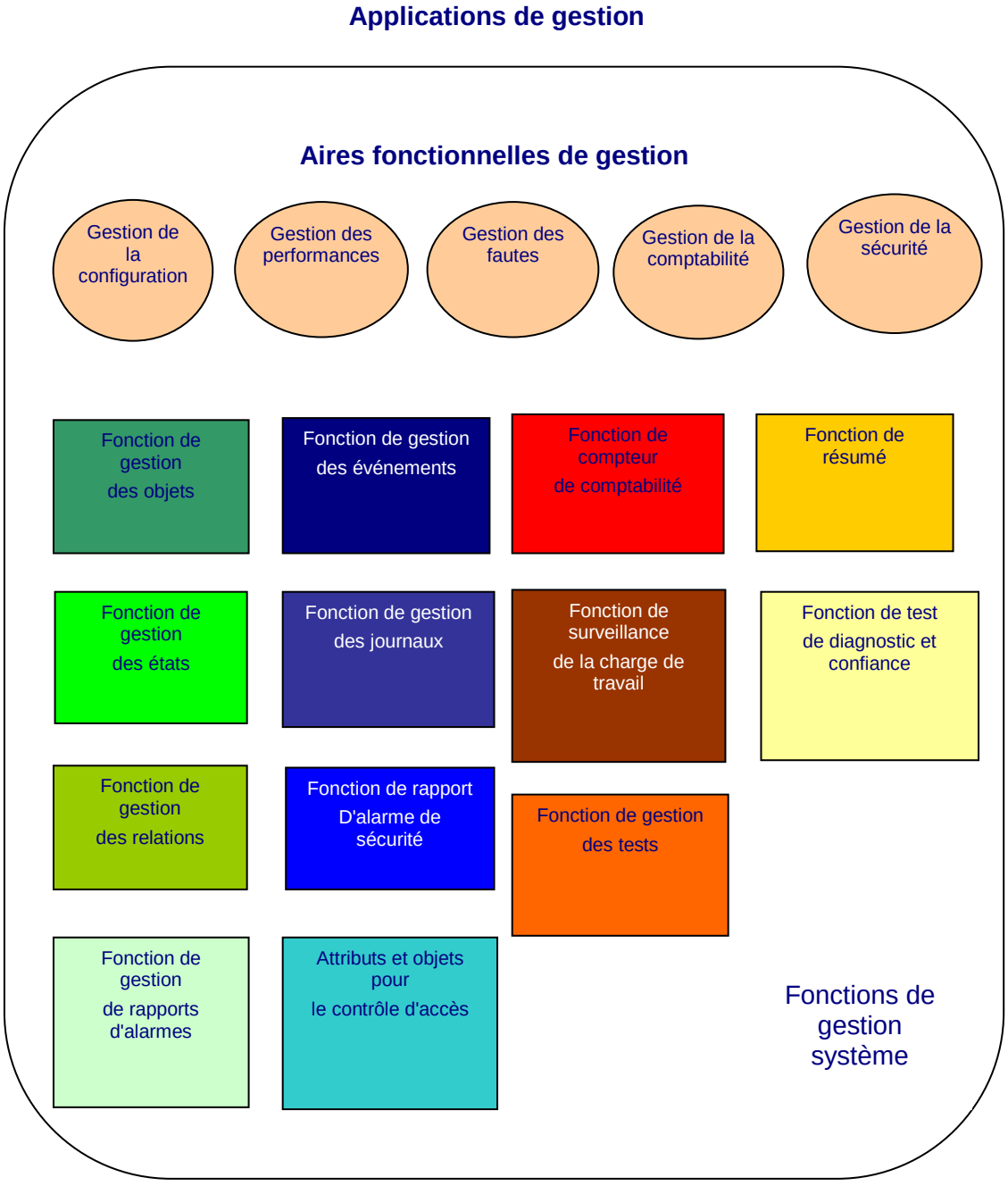


Fig. 3.1: Relations entre aires fonctionnelles et fonctions de gestion système

• **Fonction de gestion des rapports d'alarmes:** [X733], elle normalise des comptes rendus de notification d'un événement spécifique. Cinq types d'alarmes sont spécifiés par leur sémantique et leurs paramètres:

- alarme de communication concernant le transport d'information,(exemple: erreur d'établissement d'appel);
- alarme de qualité de service relative à la dégradation de la qualité de service, (exemple: temps de réponse trop long);
- alarme de traitement logiciel associée aux fautes de traitement logiciel, (exemple: erreur de version de logiciel);
- alarme d'équipement concernant un faute d'équipement (exemple: problème d'alimentation);

- alarme d'environnement relative à l'entourage de l'équipement (exemple: détection d'intrusion).

• **Fonction de gestion des rapports d'événements:** [X734], elle doit permettre aux systèmes de gestion de:

- sélectionner les rapports d'événements à notifier;
- spécifier le ou les gestionnaires destinataires;
- suspendre et reprendre l'activité d'envoi des rapports;
- modifier les modalités d'envoi des rapports.

Cette fonction s'appuie sur un objet particulier, le discriminant de rapport d'événement qui permet:

- la sélection des notifications locales qui doivent faire l'objet de rapports d'événement relatifs à un objet ou à un ensemble d'objets (rôle de filtre);
- la détermination des destinataires des rapports.

• **Fonction de gestion des journaux:** [X735], elle définit le service pour la mémorisation des notifications d'événement dans des enregistrements (*log record*) et journaux (*log*). Ce service permet de:

- sélectionner les enregistrements à insérer dans un journal;
- modifier les critères d'enregistrement;
- rechercher et supprimer des enregistrements;
- initialiser et supprimer des journaux, etc.
- La fonction de contrôle de journal s'appuie sur deux classes d'objets supports:
- la classe journal qui modélise les ressources de mémorisation;
- la classe enregistrement de journal qui modélise les enregistrements mémorisés dans un journal.

• **Fonction de rapport d'alarme de sécurité:** [X.736], elle offre des services de détection et de notification, identiques à ceux fournis par la fonction de gestion d'alarme, mais relatifs aux événements de sécurité. Cinq types d'alarmes sont spécifiés:

- violation d'intégrité due par exemple à une information perdue;
- violation opérationnelle produite, par exemple, à la suite d'un refus ou d'une indisponibilité de service;
- violation physique qui peut découler d'une détection d'intrusion;
- violation de service suite par exemple à un problème d'authentification;
- violation de plage horaire si une activité est effectuée hors plage horaire.

• **Autres fonctions de gestion:**

La fonction d'audit de sécurité [X.740] fournit un service de notification de rapports de sécurité. Elle permet l'enregistrement d'événements de sécurité.

La fonction attributs et objets pour le contrôle d'accès [X.741] permet à un système de gestion de domaine de contrôler les accès aux ressources de gestion.

La fonction compteur de comptabilité [X.742] définit un service de collecte de données relatives à l'utilisation des ressources OSI et un service de contrôle de cette collecte pendant et après utilisation de la ressource.

La fonction de surveillance de la charge de travail [X.739] fournit les outils permettant de détecter au plus vite les problèmes de manque de ressources, avant que les effets ne s'en fassent ressentir au niveau des utilisateurs.

La fonction de gestion des tests [X.745] permet de gérer les tests sur les systèmes réels et sur les ressources OSI en général. Chaque test nécessite la création d'un environnement de test, le contrôle et la surveillance du système durant le test, et le retour aux conditions normales.

La fonction de résumé [X.738] concerne la collecte et l'agrégation des informations (exemples: mesures de débit de temps de réponse, de disponibilité, etc.) pendant des intervalles de temps définis.

La fonction de test de diagnostic et de confiance [X.737] des catégories de tests dits de diagnostic et de confiance qui permettent d'étudier l'aptitude d'une ressource à rendre les services pour les quels elle est prévue.

Conclusion

Le modèle informationnel a préparé les données, le modèle architectural et celui de la communication ont délimité l'environnement de travail pour effectuer les traitements, quant au modèle fonctionnel, il a regroupé les applications que l'administrateur a en charge pour répondre aux besoins et pour maintenir le réseau dans un état opérationnel. Ceci pour les deux systèmes de gestion, à savoir celui du TMN (CMIP) et celui de l'IETF (SNMP). SNMP répond rapidement aux besoins mais manque de puissance; CMIP est plus puissant sémantiquement mais pose le problème de ne pouvoir être présent sur tous les équipements.

Chapitre 4 - L'architecture agent-manager

Introduction

Les systèmes de gestion de réseaux sont tous construits sur deux entités fondamentales: le manager et l'agent. Ce chapitre va tenter de clarifier leurs propriétés et leurs fonctions.

Selon l'architecture OSI, les managers et les agents sont des composants applicatifs, c'est-à-dire situés en couche 7. Comme on l'a vu dans le chapitre 2, ils s'appuient sur les services de la couche 7: ROSE/ACSE pour la session sécurisée, et sur les services SMASE (Systems Management Application Service Entities).

Dans le cadre IETF, le manager est une application utilisant UDP comme protocole de transport (c'est d'ailleurs un reproche fréquent, car la fiabilité est laissée à l'appréciation de l'application).

Un manager est un composant assumant un certain nombre de fonctions de gestion (des Systems Management Functions) adressant des besoins du système de gestion global. Pour ce faire, le manager procède à l'émission de requêtes à l'agent. Ces requêtes s'appuient sur un certain protocole qui est CMIS/P pour le TMN, et SNMP dans le cadre de l'IETF.

L'agent dispose de l'ensemble de l'information nécessaire pour répondre aux requêtes. Cette information est stockée dans une MIB (Management Information Base). Lorsque l'agent reçoit une requête, il consulte sa MIB et répond au manager avec la ou les valeurs demandées. Ce schéma fondamental est illustré par la figure 4.1:

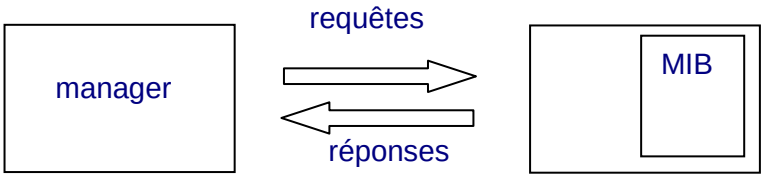


Fig. 4.1: les entités manager et agent

Les protocoles

Il est utile d'examiner d'un peu plus près les protocoles mis en œuvre par les managers et agents tant TMN qu'IETF.

La dynamicité

- En TMN, le protocole CMIP assure une communication totalement asynchrone: la requête est envoyée à la volée, et le manager n'attend

pas la réponse (requête non bloquante). L'agent et le manager gèrent des files de requêtes numérotées. Lorsque l'agent, plus tard, expédie sa réponse, le manager ré-établit l'association avec l'archive de la requête qui correspond. Le TMN fournit un concept unifié pour la communication de l'agent: les *notifications*. Ces dernières transportent tant les réponses à des requêtes, que des informations générées spontanément comme des alarmes. Les notifications confèrent aux systèmes TMN une grande dynamicité.

- Pour l'IETF, la requête SNMP est synchrone c'est-à-dire bloquante: le manager reste en attente de la réponse. Ceci est parfois risqué dans la mesure où le protocole sous-jacent est UDP. Si le paquet UDP se perd, le manager risque de se trouver bloqué. Par ailleurs SNMP ne prévoit pas de mécanisme de notification. Les alarmes envoyées par l'agent se servent d'un paquet SNMP particulier, le Trap.

Les requêtes

Cas de CMIP

En TMN, cinq types de requêtes existent: `get`, `set`, `create`, `delete`, `action`.

- `Get` permet de lire une valeur d'attribut;
- `Set` est l'opération d'écriture;
- `Create` est l'opération d'instantiation d'une MOC (Managed Object Class) pour obtenir un nouvel objet (Managed Object ou MO) peuplant la MIB;
- `Delete` efface une instance de MO;
- `Action` est un message de déclenchement d'une méthode de MO.

La forme générale de la requête est:

```
<request>(fdn, oid, type, scope, filter)
```

Le `fdn` est le Full Distinguished Name de l'objet à atteindre (voir la section 3.3) dans la MIB.

L'`OID` de l'objet à atteindre est l'Object Identifier tel que définit par l'ITU-T pour cet objet. En effet les modèles d'information standard étant déposés publiquement à l'ITU-T, cette dernière confère un identifiant unique à tout objet le désignant de façon universelle.

Le `type` de la requête peut être « best effort » ou « transactional ». Dans ce dernier cas l'agent assure des propriétés ACID (cf. cours de bases de données).

Le `scope` est la profondeur de la requête, à partir de l'objet désigné par le `fdn`. Un scope de `1` restreint la requête à l'objet lui-même. Un scope « `any` » amplifie la requête à tout le sous-arbre constitué par l'objet désigné.

Le `filter` permet de spécifier des conditions sur les objets. Ainsi, on peut vouloir lire uniquement les attributs d'objets ayant leur attribut `AdministrativeState` positionné à « `available` ».

Cas de SNMP

SNMP reprend les principes précédents en les simplifiant. Il n'y a que les requêtes `get` et `set`. La forme générale de l'invocation est:

```
<request>(oid, value)
```

On note l'absence de mécanismes de création/délétion: la MIB SNMP est totalement statique.

Par ailleurs l'agent possède une méthode d'envoi d'alarmes par le paquet spécial « Trap ».

Structure de l'information

La structure de la MIB diffère grandement selon l'architecture considérée, TMN ou SNMP.

Cas du TMN

Les MOs du TMN sont structurés selon le standard X.400, c'est-à-dire selon une arborescence. La MIB TMN est ainsi capable d'accueillir un très grand nombre de MOs (de l'ordre de 10^E6 pour des équipements SDH ou WDM) sans difficulté.

Le FDN (full distinguished name) d'un MO est le chemin d'accès de l'objet à partir de la racine de la MIB. Le FDN est constitué d'une liste de prédicats « attribut-valeur » permettant une grande flexibilité de nommage, similaire à ce que propose LDAP qui repose sur des standards identiques.

Cas de SNMP

Pour SNMP, l'information est statique et structurée de façon linéaire ou tabulaire. Le nommage de l'objet est effectué par l'OID. Les MIBs sont de structure totalement statique.

Le Frame

L'interaction entre le manager et l'agent mérite une analyse supplémentaire. En effet, il est utile de considérer non seulement la double entité manager-agent, mais également ce qui est sous-jacent à l'agent, à savoir la ressource gérée.

En effet, les spécifications du TMN ou de l'IETF mettent en œuvre des mécanismes de gestion abstraits dans le but d'être indépendant du vendeur de matériel. Le problème est que cette démarche induit une distance sémantique arbitraire entre le cycle de vie des objets de gestion, et le cycle de vie du matériel géré. Un exemple: la mise en œuvre d'une cross-connection dans une matrice SDH se fait en une seule opération CMIP alors qu'il s'agit probablement de nombreuses et délicates interactions au niveau des composants électroniques (CPU, registres de commande) du matériel SDH.

La figure 4.2 présente le frame, c'est-à-dire la triade associant le manager, l'agent et la ressource.

Cette représentation illustre le propos: en fait c'est *l'implémentation* des MOs, c'est-à-dire le logiciel qui réalise les fonctions liées aux attributs et aux actions du MO, qui prend en charge la distance sémantique entre l'interaction protocolaire CMIP ou SNMP, et le cycle de vie des ressources gérées i.e. les cartes électroniques du matériel de l'équipement.

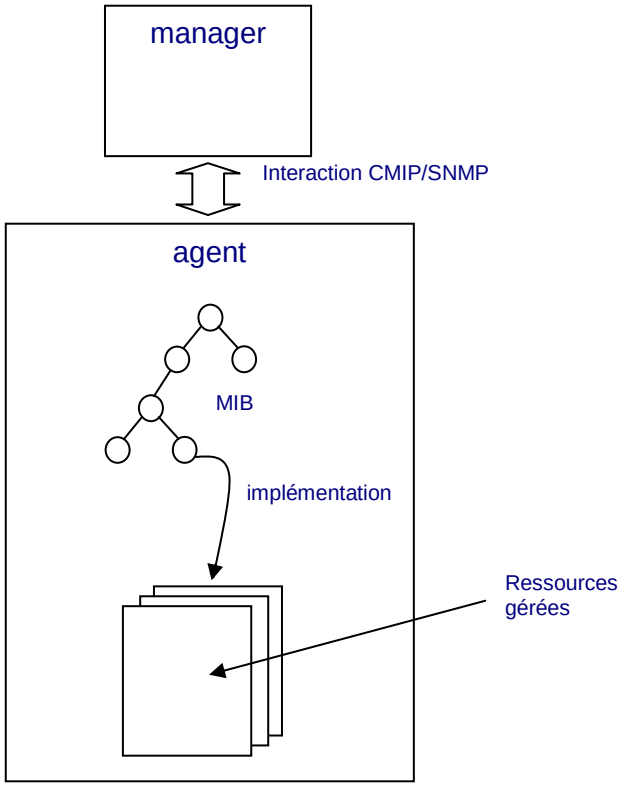


Fig. 4.2: le frame

Une difficulté supplémentaire, dans ce contexte, est que souvent le matériel est sujet à de multiples variations (améliorations, évolutions) qui rendent nécessaire l'écriture de nombreuses versions du logiciel de l'agent. On comprend ici pourquoi un agent, en particulier TMN (mais SNMP également, dans une moindre mesure) est une entité onéreuse et complexe tant dans son coût de développement, que dans sa maintenance.

Pour conclure cette section, considérons maintenant les caractéristiques de l'information de gestion en termes de répartition et de cycle de vie.

Par défaut, la répartition de l'information telle que présentée dans ce chapitre est telle que toute l'information est maintenue dans l'agent, le manager étant virtuellement vide d'information[2]. Mais cette répartition est-elle optimale ? Elle induit nécessairement le déclenchement systématique d'une requête à chaque besoin du manager, et donc une forte consommation de bande passante du réseau de gestion (DCN).

Une alternative consisterait à mettre en œuvre une mémoire tampon ou cache, dans le manager. Mais le cycle de vie de l'information est à considérer dans ce contexte. Classifions les MOs selon ce cycle de vie:

- Les objets *permanents* sont les MOs invariables dans le temps (c'est-à-dire d'une durée de vie supérieure à la durée de la session du système de gestion). Couramment il s'agit de la configuration système, de la version du logiciel, du nom du vendeur etc.
- Les objets *éphémères* sont des MOs n'ayant de signification qu'à l'instant où ils sont lus. Il s'agit par exemple des valeurs de compteurs de performance (nombre de collisions, de paquets perdus etc).

[2] L'information a minima vitale dont le manager a besoin est bien sûr l'adresse de l'agent.

- Une troisième catégorie est celle des objets ni permanents, ni éphémères. Ces objets ont une variabilité *intermédiaire*, sans que l'on puisse la caractériser de façon plus précise.

Une gestion habile d'un cache consisterait à allouer les objets permanents au cache dans tous les cas, et à ne jamais utiliser le cache pour les objets éphémères. Les objets intermédiaires quant à eux, doivent faire l'objet d'un compromis. Le savoir-faire de l'ingénieur mettra en œuvre une stratégie « prudente » en ne les plaçant jamais dans le cache, quitte à consommer davantage de bande passante en requêtes. Une stratégie « hardie » consistera à prendre un certain risque à lire ces objets à partir du cache, quitte à ce que leur valeur soit parfois fausse.

C'est dans la recherche de tels compromis que l'on obtient, au bout du compte, un système efficace malgré sa complexité.

Chapitre 5 - TMN, le réseau de gestion des télécommunications

Introduction

La nature des informations à véhiculer à travers les réseaux ne cesse pas de se diversifier. Des applications comme la téléphonie publique munie des services de réseau intelligent, la radiotéléphonie mobile, la transmission de données à faible et à haut débit, des images fixes ou animées, de la voix … amènent les exploitants à gérer une variété toujours plus grande de réseaux. Mais dans chaque cas, il s'agit d'exploiter et d'entretenir des équipements de commutation et de transmission, ainsi que d'améliorer les fonctions gérant les nombreux services évolués que les usagers et les fournisseurs de services réclament sans cesse.

Définition du TMN

Le TMN (*Telecommunication Management Network*), le réseau de gestion des télécommunications, a pour but de répondre aux exigences d'exploitation, d'administration et de maintenance des réseaux de télécommunications de tout type de technologie. Il présente l'avantage de se situer dans un contexte multi fournisseurs. En effet, le TMN tient compte de la diversité des équipements de télécommunication et de celle des centres informatiques qui les gèrent. Il permet également à l'exploitant du réseau de développer de nouvelles applications de gestion tout en s'adaptant à l'organisation de son réseau.

Ce chapitre présente le TMN et ses aires fonctionnelles ainsi que ses architectures informationelle, fonctionnelle et physique.

Une application de gestion de réseau est généralement composée de plusieurs agents répartis sur l'ensemble du réseau à surveiller. Pour coordonner leurs actions, ces agents communiquent à travers un réseau de transmission de données appelé le *Data Communication Network* (DCN). Dans le cas du TMN, le réseau de gestion des télécommunications (*Telecommunication Management Network*), il est fourni une structure de réseau pour interconnecter des systèmes d'exploitation et des équipements normalisés. Les relations et la situation du TMN vis-à-vis du réseau sous son contrôle sont illustrées par la figure 5.1:

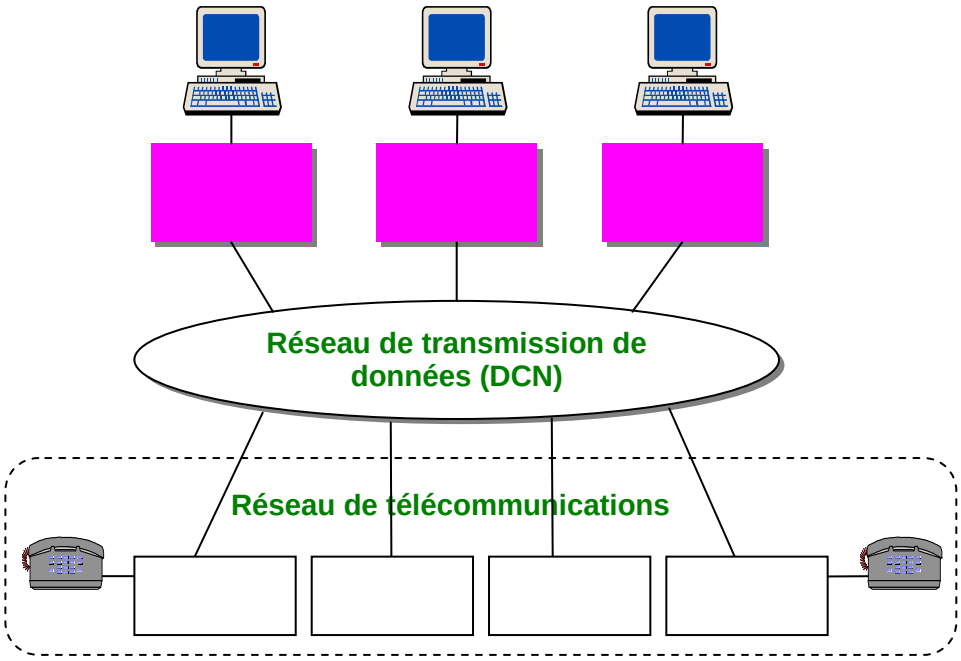


Fig. 5.1: relation entre réseau de gestion et réseau géré

Cette figure montre trois parties essentielles:

- Les éléments de réseau de télécommunication se composent essentiellement des commutateurs, qui sont des équipements à forte composante logicielle, et des équipements de transmission.
- Des postes de travail qui communiquent avec les systèmes de gestion et les éléments de réseau.
- Le réseau de transmission de données reliant les élément de réseau aux systèmes de gestion (Data Communication Network ou DCN).

Le TMN apparaît sur la figure 5.1 distinct du réseau qu'il contrôle, ce qui n'est vrai que sur le plan logique. En effet, les deux réseaux physiques ne sont pas totalement séparés. Ainsi, un réseau ATM se sert des canaux OAM pour la gestion, tandis que la SDH met à profit les octets de gestion réservés à cet effet dans les en-têtes des trames. A l'inverse, des équipements divers peuvent être amenés à être gérés par une voie IP distincte du réseau de transmission, et c'est le cas des infrastructures fixes des réseaux GSM.

Fonctions de gestion offertes par le TMN

Afin de fournir aux usagers des services avec un certain niveau de qualité et de coût, le TMN recouvre l'ensemble des activités de surveillance, d'analyse, de contrôle et de planification du fonctionnement des ressources d'un réseau de télécommunication. Dans cette optique, le TMN s'appuie sur cinq aires fonctionnelles (voir chapitre 2) appelées aussi SMFA (*Systems Management Functional Area*) ou classiquement les fonctions FCAPS:

- la gestion des fautes (F);
- la gestion de la configuration (C);
- la gestion de la comptabilité (A = accounting).
- la gestion des performances (P);
- la gestion de la sécurité (S).

Architecture du TMN

L'ITU-T découpe l'architecture générale du TMN selon trois vues [M.3010]. Elles peuvent être considérées séparément lors de la conception et de la planification d'un TMN, ce sont:

- l'architecture informationnelle du TMN,
- l'architecture fonctionnelle du TMN,
- l'architecture physique du TMN.

L'architecture informationnelle permet de représenter les ressources disponibles sur le réseau de télécommunications supervisé. Pour ce, elle utilise des données pour les besoins de surveillance, de contrôle et de gestion. L'architecture fonctionnelle décrit la mise en œuvre d'un TMN à partir des différentes catégories de blocs de fonctions et des différentes classes d'interconnexion entre ces blocs. Quant à l'architecture physique, elle décrit la réalisation des fonctions du TMN en termes de composants physiques et d'interfaces pour la communication.

Architecture informationnelle du TMN

Elle a pour rôle de modéliser les ressources du réseau. Elle utilise l'approche orientée objet pour mettre en œuvre le modèle d'information du réseau. L'architecture informationnelle définit aussi des couches de gestion qui correspondent à des niveaux où des décisions doivent être prises et où résident les informations de gestion à travers un modèle informationnel. Ces couches sont purement logiques car il n'y a pas de mise en correspondance au niveau d'une implantation physique du TMN. Des composants qui résident logiquement dans deux couches différentes peuvent se trouver intégrés dans le même composant physique.

Architecture logique en couches

Les couches définies dans la recommandation M.3010 sont:

- couche élément de réseau,
- couche gestion d'élément de réseau,
- couche gestion de réseau,
- couche gestion de service,
- couche commerciale.

La couche élément de réseau ou NEL (*Network Element Layer*) réalise les fonctions de gestion de base fournies par les équipements du réseau. Parmi ces fonctions on trouve la collecte d'informations de performance, la collecte d'alarmes et l'auto-diagnostic.

La couche gestion d'élément de réseau ou EML (*Element Management Layer*) assure la gestion individuelle de chaque élément du réseau. Elle fournit une abstraction des fonctions offertes par la couche NEL afin de garantir l'indépendance vis à vis des constructeurs.

La couche gestion de réseau ou NML (*Network Management Layer*) gère l'ensemble des éléments du réseau tels qu'ils lui sont présentés par la couche EML. Cette gestion peut être faite individuellement ou en tant qu'ensemble. Les fonctions de la couche NML ont pour but de supporter des applications de gestion qui requièrent une vue de bout en bout du réseau de télécommunications. En recevant les informations de la couche EML, la couche NML crée une vue globale de l'ensemble du réseau indépendante de toute technologie du réseau.

La couche gestion de service ou SML (*Service Management Layer*) se charge de la gestion des services offerts aux clients tout en utilisant le service de facturation et en assurant un certain niveau de qualité de service. Elle peut être décomposée en deux sous-couches: celle de la "gestion des services de base" qui sont fournis par le réseau et celle de la "gestion des services à valeur ajoutée" qui sont offerts par des fournisseurs de services autres que l'opérateur du réseau.

La couche gestion commerciale ou BML (*Business Management Layer*) assume la responsabilité qui se rapporte à la totalité de l'entreprise. Elle établit les accords entre opérateurs et supporte la planification stratégique.

Exemple: architecture logique en couche du service VPN dans un réseau ATM

Cet exemple décrit l'architecture logique répartie en couches pour la fourniture d'un service de réseau privé virtuel large bande ou BVPN (*Broadband Virtual Private Network*). Plus précisément, déployer un réseau privé virtuel sur un réseau ATM.

Un réseau privé virtuel est un réseau privé logique et non pas physique. Il se compose de ressources virtuelles qui représentent une abstraction des ressources physiques. Un VPN peut être vu simplement comme un réseau constitué de liens logiques entre les différents sites distants d'une même organisation en utilisant le RTC (Réseau Téléphonique Commuté) et l'ADSL, cf. figure 5.2. En plus du fait d'offrir aux entreprises des communications à longue distance performantes, les clients du VPN peuvent gérer leur réseau virtuel en changeant dynamiquement sa configuration, ils peuvent rajouter ou libérer des liens logiques entre leurs réseaux privés locaux se trouvant sur leurs sites.

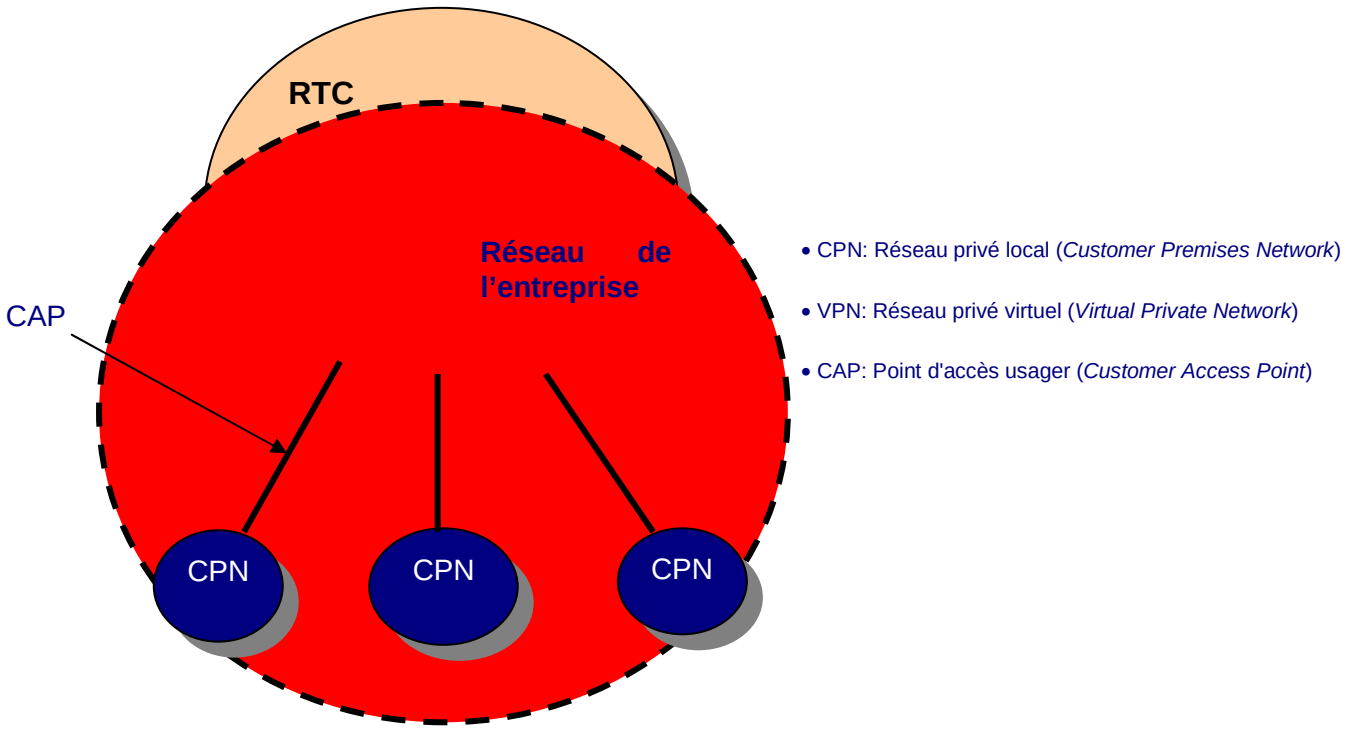


Fig. 5.2: configuration d'un VPN

Dans la répartition en couches on trouve les commutateurs du réseau ATM (*Asynchronous Transfer Mode*) à gérer dans la couche NEL (élément de réseau) comme décrit dans la *figure 5.3*. Ils fournissent un service d'établissement, de maintien et de libération de la connexion par la gestion.

La couche EML contient les gestionnaires d'éléments de réseaux qui fournissent une abstraction de la connectivité à l'intérieur des commutateurs ATM. Ils offrent également un service de gestion de cette connectivité.

La couche NML offre grâce au système de gestion de l'opérateur impliqué dans la fourniture du BVPN une vue abstraite de la connectivité de bout en bout à l'intérieur du RTC et le service de gestion associé (service de gestion des connexions).

La couche SML offre le service de configuration du VPN au client. Pour ce faire, la couche SML s'appuie sur les services de la couche NML, et cette dernière utilise le service de gestion des connexions offert par la couche EML, tout ceci en cascade.

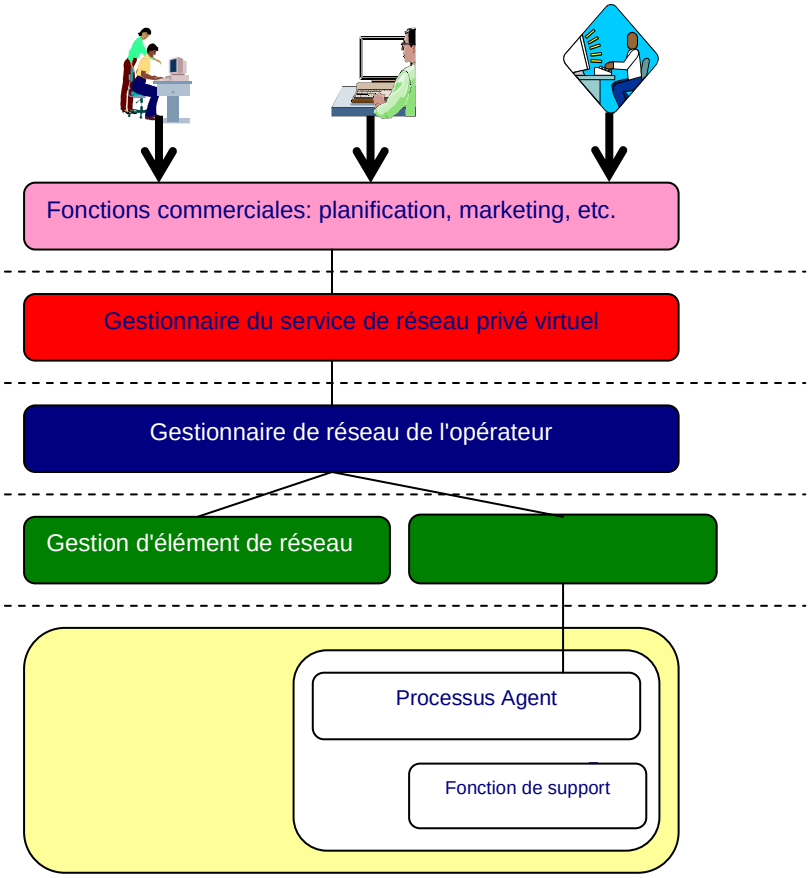


Fig. 5.3: Exemple d'architecture logique en couches

Dans chaque couche de gestion existe un modèle d'information qui fournit une abstraction des ressources. A ce propos, l'ITU-T a proposé dans la recommandation M.3100 un modèle informationnel générique de réseau de télécommunication appelé GNIM (*Generic Network Information Model*). Il offre grâce à sa généricité l'avantage d'être indépendant d'une technologie de réseau particulière; il peut donc être utilisé dans différentes technologies comme l'ATM, FrameRelay, SDH, PDH, RNIS, etc. Le modèle peut être vu comme un ensemble de classes réutilisables pour la gestion d'un réseau de télécommunication. Le modèle a pour but de couvrir toutes les couches logiques du TMN, mais dans son état actuel il est applicable principalement à la couche EML. Le GNIM s'appuie sur le modèle informationnel de gestion défini par le TMN (voir chapitre 4) et sur le modèle de référence de la recommandation G.803 pour le SDH défini par l'ITU-T et dont il reprend les concepts d'organisation en couches et de subdivision.

Architecture fonctionnelle du TMN

Pour gérer un réseau de télécommunication, les applications de gestion doivent résider sur des composants (équipements) réparties géographiquement. Il faut donc définir des procédures pour la communication entre ces applications réparties. Ceci est d'autant plus vrai que l'environnement des télécommunications est par nature varié. Il y a notamment des relations inter-opérateurs et des environnement multi-fournisseurs. Il est donc obligatoire de définir des procédures normalisées pour ces applications. Ceci est notamment l'objectif de la définition de

l'architecture fonctionnelle du TMN. A cet effet, elle se compose en un ensemble de blocs de fonctions et points de référence entre ces blocs.

Blocs de fonctions et points de référence

Un bloc de fonctions est un groupement de fonctions conçues pour des tâches bien spécifiques relatives au transport et au traitement des informations de gestion. Il en a six dans l'architecture fonctionnelle du TMN:

- fonction de système d'exploitation ou OSF (*Operation System Function*),
- fonction de médiation ou MF (*Mediation function*),
- fonction poste de travail ou WSF (*Work Station Function*),
- fonction d'élément de réseau ou NEF (*Network Element Function*).
- fonction d'adaptateur Q ou QAF (*Q Adaptor Function*),
- fonction réseau de communication ou DCF (*Data Communication Function*) (voir figure 5.4).

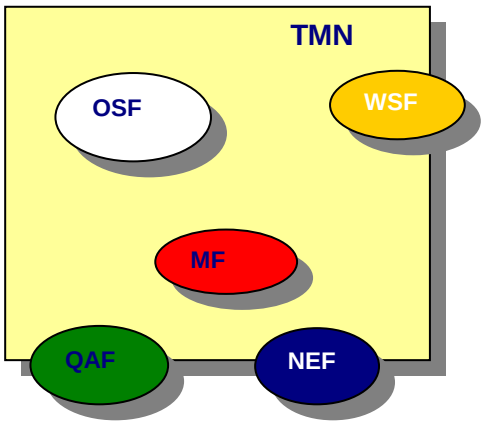


Fig. 5.4: les blocs fonctionnels du TMN

• Le bloc OSF représente l'application de gestion. Il gère les ressources de télécommunication en traitant les informations de gestion.

• Le bloc NEF représente pour chaque élément de réseau l'ensemble des fonctions disponibles sur cet élément et qui assurent sa gestion. Le NEF communique donc avec le TMN pour le gérer. La MIB et toute application de gestion associées au NEF sont considérées comme faisant partie du TMN, le reste des éléments de ce bloc fonctionnel ne le sont pas.

• Le bloc WSF offre les moyens d'interpréter et de présenter l'information du TMN au gestionnaire humain. Il établit donc le dialogue entre l'administrateur réseau et les blocs OSF. L'administrateur et l'interface utilisateur ne font pas partie du TMN.

• Le bloc MF facilite la communication en jouant un rôle de médiation comme son nom l'indique. En effet, il agit sur l'information passant entre une fonction OSF et une fonction NEF ou une fonction QAF. Pour ce, il comprend:

- une fonction de conversion d'information qui traduit un modèle de données en un autre et modifie ainsi le contenu des messages d'information de gestion.
- une fonction de conversion de protocole qui traduit un protocole en un autre.

et éventuellement des fonctions complémentaires comme: le stockage d'information, le filtrage, la condensation d'information, etc.

- Le bloc QAF permet de relier des entités non-TMN au TMN. Pour cela, il assure une traduction entre un point de référence du TMN et un point de référence non-TMN. Plus précisément, il connecte au sein d'un TMN des entités n'ayant pas des interfaces conformes aux interfaces normalisées par le TMN aux rôles des blocs OSF et NEF. Ces entités jouent généralement le rôle de système d'exploitation.
- Le bloc DSF permet d'échanger des informations de gestion entre les différents blocs fonctionnels. Ceci par la fourniture des moyens de communication de données qu'utilisent ces blocs. Il effectue des transferts d'informations entre les blocs de fonctions. Il correspond aux services offerts par les couches basses du modèle OSI. La fonction DCF peut par exemple être réalisée par un réseau X.25, c'est pour cela qu'elle n'apparaît pas sur la figure 4.4.

Entre les blocs fonctionnels existent des points de référence y assurant l'interaction fonctionnelle et/ou le passage d'informations. Ce sont des points conceptuels qui interconnectent les blocs fonctionnels. Ce sont en quelque sorte des interfaces entre applications puisqu'ils décrivent les frontières de services entre les différents blocs. Il existe trois types de points de référence TMN: q, f et x.

Les points de référence q connectent soit directement soit via le bloc de fonction DSF (fonction de communication de données) les blocs de fonctions NEF et OSF, NEF et MF, MF et MF, QAF et MF, MF et OSF, et OSF et OSF. Dans le type q on distingue deux sous-types, q_x et q_3 (voir figure 5.5).

Les points de référence qx se situent entre les blocs NEF, QAF ou MF et MF alors que les points de référence q3 se trouvent entre les blocs NEF, QAF ou MF et OSF.

Les points de référence f prennent place entre les OSFs les WSFs et WSF et MF. Ils établissent généralement l'interface X-windows à travers laquelle l'administrateur interagit avec les fonctions du TMN.

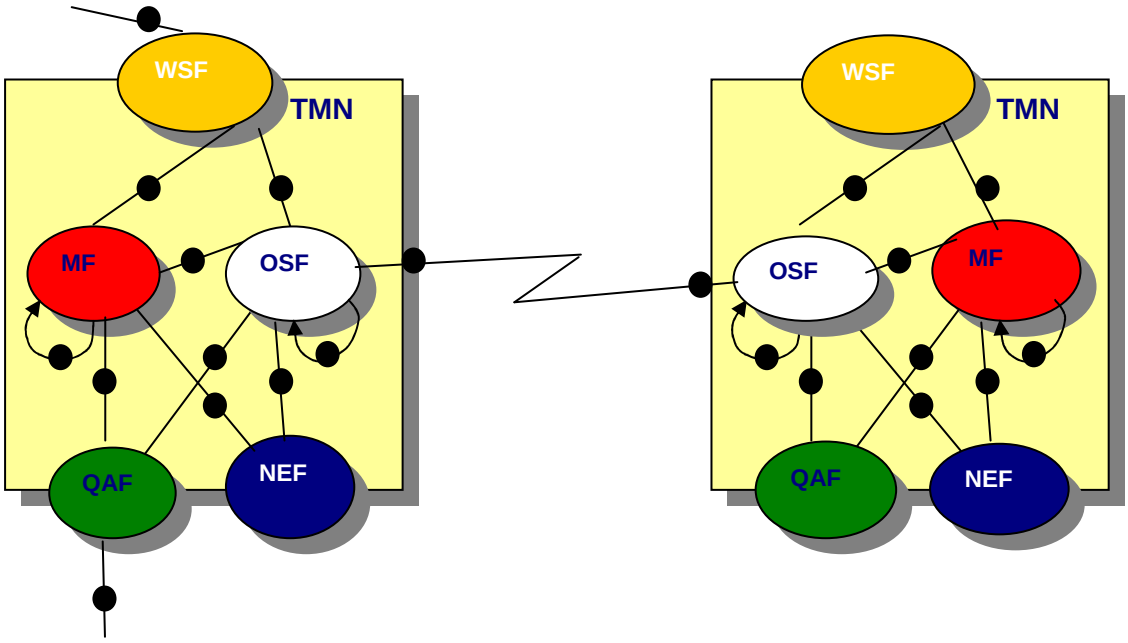


Fig. 5.5: Points de référence entre les blocs de fonctions TMN

Les points de référence x servent à relier deux OSFs de deux TMNs différents, en d'autres termes interconnecter deux opérateurs différents. Ils peuvent également relier un OSF d'un TMN à une fonctionnalité équivalente de type OSF d'un autre réseau. Vue la position critique de ces points, ils sont obligés de présenter des attributs de sécurité.

Les points g et m sont des points de référence externes au TMN. Les points de référence g se situent entre l'opérateur humain et le bloc WSF. Par contre, les points de référence m relient les QAFs à des entités non-TMN ou non conformes aux recommandations TMN.

L'ensemble des définitions de blocs de fonction et de points de référence constitue un modèle de référence du TMN. Les fonctionnalités logiques de tout TMN peuvent être décrites à travers un schéma de configuration de référence (voir figure 5.5) avec des arrangements de blocs de fonctions et de points de référence.

NB: Il faut noter que ce sont les interfaces qui sont normalisées et non les blocs fonctionnels du TMN.

Exemple du cas de la gestion du service VPN dans un réseau ATM

Dans cet exemple on verra l'architecture fonctionnelle de la gestion de la configuration d'un service réseau privé virtuel large bande (BVPN).

A la couche NEL (voir figure 5.6), les blocs NEF représentent l'ensemble des fonctions offertes par les commutateurs ATM afin d'assurer leur propre gestion.

La couche EML comporte des blocs OSF et des blocs MF. Les OSFs gèrent et prennent en charge l'abstraction de la fonction de connectivité fournie par la couche NEL. Les blocs de fonction MF jouent leur rôle de médiation en prenant leur emplacement logique dans la couche EML.

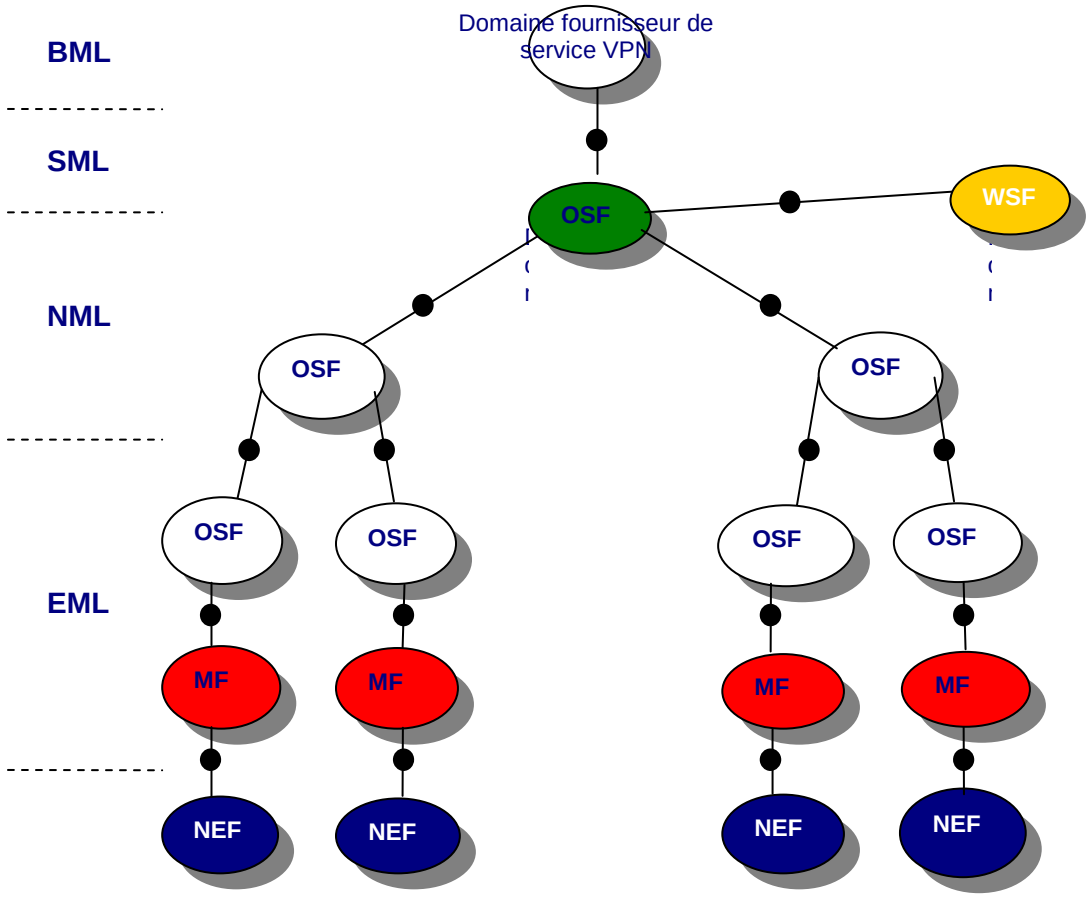


Fig. 5.6: Exemple d'architecture

La couche NML se compose d'un bloc OSF qui prend en charge un domaine d'opérateur de réseau. Le bloc OSF donne une vision unifiée et gère les capacités du réseau afin de supporter le service VPN offert au client. L'interface entre les blocs fonctionnels OSF de la couche NML et les blocs fonctionnels de la couche EML se fait en un point de référence q_3.

A la couche SML, le bloc OSF fournit le service de gestion de configuration de VPN à ses clients. Le dialogue entre le client et l'OSF se fait à travers le bloc WSF. L'OSF de la couche SML est subdivisé en deux parties dont l'une fait partie de la couche NML. Cette partie gère des connexions de sous-réseau, alors que l'autre dans la couche SML gère des liens logiques entre les différents sites distribués du client. L'interface entre le bloc fonctionnel de la couche SML et ceux de la couche NML se fait en un point de référence x, car dans la configuration choisie, le service est offert par un fournisseur de services à valeur ajouté ou VASP (*Value Added Service Provider*).

Dans la couche BML, l'OSF interagit avec l'OSF de la couche SML via un point de référence q_3. Il se charge par exemple des tâches de fixation d'objectifs.

Echanges et protocoles de gestion

La spécification ITU-T des points de référence q et x se base sur les normes de gestion OSI relatives au modèle gestionnaire-agent [X701], les services CMISE [X710] et le protocole CMIP [X711], pour plus de détails, se référer au chapitre 4 où tous ces éléments sont présentés.

Architecture physique du TMN

L'architecture physique correspond à la réalisation physique de l'architecture fonctionnelle. Chaque bloc fonctionnel devient un ou plusieurs blocs physiques remplissant certaines fonctions. Le TMN est alors vu comme un ensemble de blocs physiques connectés, chacun exécutant un ensemble de fonctions du TMN. Lorsque les blocs de fonctions connectés sont réalisés dans des équipements différents, un point de référence devient une interface. Dans la nomenclature TMN, les noms des points de référence apparaissent en lettres minuscules alors que les interfaces ont leur nom en lettres capitales. Le TMN propose une interface pour chaque point de référence.

Les blocs physiques

Chaque bloc de fonction est réalisé par un ou plusieurs blocs physiques comme suit:

Le système d'exploitation ou de gestion ou OS (*Operation System*) réalise les fonctions de l'OSF.

- Le réseau de communication de données ou DCN (*Data Communication Network*) réalise les fonctions du bloc fonctionnel DCF. Le DCN est un réseau de communication à l'intérieur d'un TMN qui fournit le support de communication pour les informations de gestion. Il représente une forme de mise en œuvre des couches 1 à 3 du modèle de référence OSI. Ce réseau ne fournit aucune fonctionnalité des couches 4 à 7.
- L'élément réseau ou NE (*Network Element*) réalise les fonctions NEF. Il est l'unique équipement résidant dans le réseau de télécommunication géré. Son rôle principal est de prendre en charge le trafic et non la gestion. Mais il est l'élément sur lequel sont effectués la supervision, le contrôle et la gestion.
- Le poste de travail ou WS (*Work Station*) est le système qui exécute les fonctions WSF. Il s'agit d'un terminal connecté à l'OS ou à la fonction de médiation à travers le DCN.
- L'adaptateur Q ou QA (*Q Adaptor*) est un dispositif utilisé pour connecter au TMN un équipement ne présentant pas l'interface de type Q. l'adaptateur Q ne réalise que des fonctions d'adaptation d'interface Q.
- L'équipement de médiation ou MD exécute les fonctions MF. Ce dispositif peut éventuellement fournir les fonctions OSF, QAF et WSF. L'équipement MD peut être mis en œuvre sous forme de hiérarchie de dispositifs en cascade.

Le tableau 5.1 résume la correspondance entre nœuds du TMN et blocs de fonctions:

Nœuds TMN	Blocs de fonction
Système d'exploitation (OS)	OSF
Dispositif de médiation (MD)	MF
Adaptateur Q (QA)	QAF
Elément de réseau (NE)	NEF
Poste de travail (WS)	WSF

Tableau 5.1: Equivalences entre nœuds TMN et blocs de fonctions

Les interfaces

Les interconnexions des nœuds OEs, OSs, WS, QAs, et MDs à travers le réseau de transmission de données (DCN) sont réalisées par des interfaces normalisées. Ces interfaces garantissent l'interopérabilité entre systèmes interconnectés afin d'accomplir une fonction de gestion de TMN donnée. Il est donc nécessaire d'employer des protocoles de communication compatibles avec des définitions de messages génériques indépendantes des types d'équipements. L'interface Qx sert pour la connexion entre NEs et MDs, alors que l'interface Q3 est utilisée pour connecter les NEs aux OSs.

Les protocoles utilisés aux interfaces Q sont décris dans la recommandation G.733 de l'ITU-T. Cette dernière recommande deux suites de protocoles pour relier des équipements de transmission à un TMN: les suites de protocoles à piles réduites A1 et A2 essentiellement utilisées comme suite de protocoles Qx et les suites de protocoles à piles complètes de 7 couches pouvant être utilisées pour Qx et Q3.

Dans le cas des protocoles de type A, les couches 4, 5 et 6 du modèle de référence OSI sont utilisées de façon transparente. Ces protocoles sont adaptés pour des équipements de complexité moyenne. Dans le cas des protocoles à piles complètes, les couches 1, 2 et 3, sont définies dans la recommandation Q.811 [Q.811] alors que les couches 4, 5, 6 et 7 sont définies dans la recommandation Q.812 [Q.812]. Un réseau de transport X.25 peut être mis en place pour fournir les couches basses 1, 2 et 3. Les potentialités du canal D du RNIS peuvent être mises à profit. Enfin, les facilités de transport offertes par le canal sémaphore CCITT n°7 peuvent être considérés. A la couche 7, il est fait référence à CMISE/CMIP pour les services de type transaction, et FTAM (File Transfer, Access and Management) pour les services de type de transfert de fichier.

L'interface X est appliquée au point de référence x. Elle est utilisée pour l'interconnexion entre deux TMNs ou entre un TMN et un réseau possédant une interface de type TMN. L'interface X est basée sur CMISE/CMIP avec des éléments de sécurité.

L'interface F permet le raccordement de divers terminaux au TMN. Le tableau 5.2 donne les équivalences entre interfaces et points de référence:

Interfaces TMN	Points de référence TMN
Q3	q3
Qx	qx
F	f
X	x

Tableau 5.2: Equivalences entre interfaces et points de référence

Exemple d'architecture physique de gestion d'un réseau privé virtuel large bande

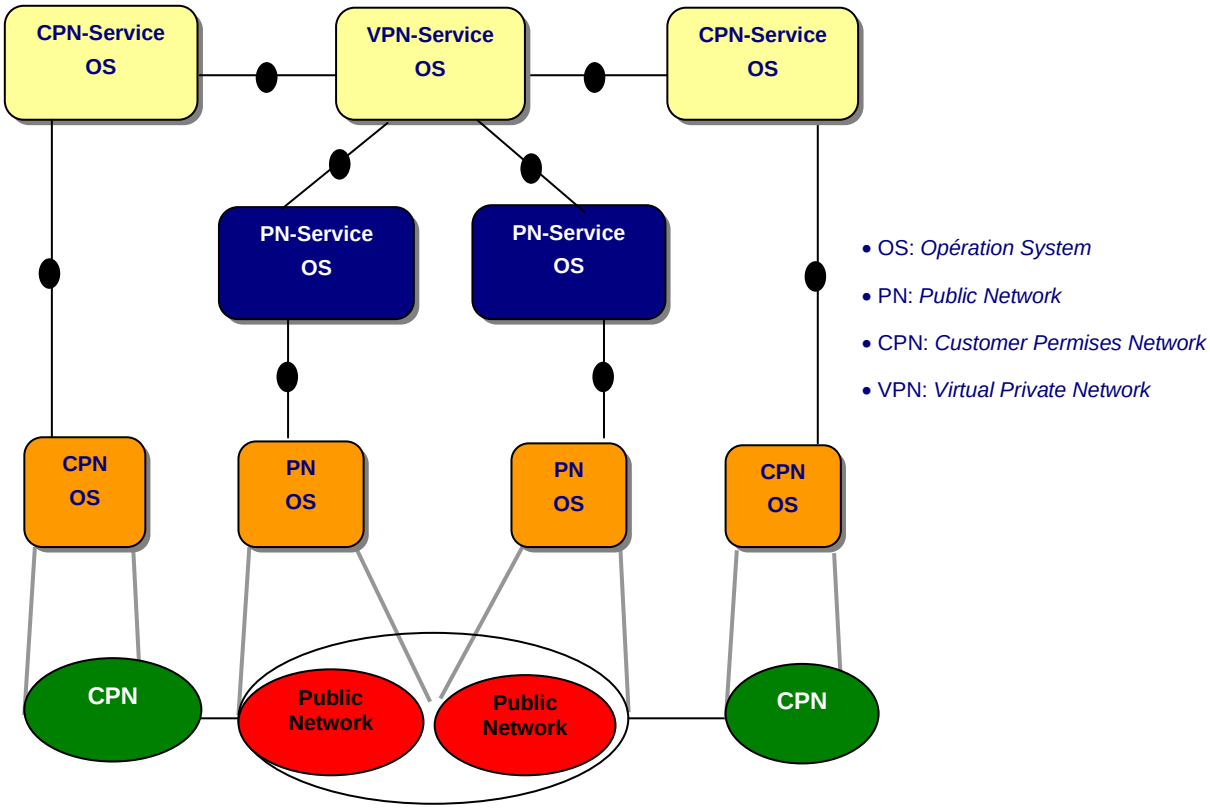


Fig. 5.7: Architecture physique de gestion du service de réseau privé virtuel

Cet exemple présente l'architecture physique de gestion d'un réseau privé virtuel large bande ou BVPN. Cette architecture comporte différents OS, à savoir l'OS du réseau privé local d'un site de l'entreprise ou CPN (*Customer Permises Network OS*) qui gère les ressources du CPN, l'OS réseau public (PN OS) qui gère les ressources du réseau public, l'OS PN-Service qui est responsable de la gestion des services offerts sur le réseau public, l'OS CPN-Service dont le rôle est d'administrer les services fournis sur le CPN et enfin le VPN-Service OS pour la gestion du réseau privé virtuel.

En terme d'interfaces, l'interface X permet la coopération entre le client du service de réseau privé virtuel, le fournisseur de service et l'opérateur de réseau. Quant à l'interface Q3, elle permet l'interopérabilité entre les OSs d'un même domaine de gestion.

Conclusion

La gestion de réseau et de service de télécommunication reste un défi majeur des environnements qui sont aujourd'hui multi-fournisseurs et intégrant plusieurs technologies. Le réseau de gestion des télécommunications TMN qui a été présenté dans ce chapitre tente de répondre à cette problématique à travers la définition d'interfaces uniformes standardisées dans les recommandations de l'ITU-T. Le processus de déploiement du TMN a été lent de par la complexité de l'architecture, l'inertie des systèmes légataires et le manque de volonté industrielle. Ce dernier s'explique par l'intérêt des fournisseurs à lier les équipements et l'offre de gestion. Cependant, le futur devrait présager de

l'avènement du TMN car les grands acteurs des télécommunications prennent conscience de l'importance de l'interopérabilité entre systèmes de gestion dans un contexte de dérégulation.

Le TMN qui est basé actuellement sur l'architecture de gestion système OSI devrait dans le futur proche s'appuyer sur les concepts et principes du traitement réparti ouvert. L'introduction de ce dernier dans le TMN devrait accroître la flexibilité et l'ouverture des composants existants en facilitant leur intégration avec les composants tiers. Cette avancée du TMN constitue une avancée primordiale pour son intégration dans l'architecture TINA qui promeut aussi la convergence entre les architectures IN et TMN pour la création et la gestion de service, ainsi que pour la gestion des réseaux supports des services.

Chapitre 6 - TINA, l'architecture de réseau d'information de télécommunications

Introduction

A l'heure actuelle l'offre en termes de services de télécommunication avancés comme les services multi média et réseaux privés virtuels large bande est encore insuffisante. Les coûts et les délais pour introduire, maintenir et étendre un service sont encore importants. Les technologies utilisées pour leur mise en œuvre sont parfois inefficaces. Or, la demande ne cesse de croître. Cette génération de services requiert entre autre une introduction plus flexible et une gestion plus intégrée. Pour satisfaire cette demande de services par les usagers, les fournisseurs de réseaux et les fournisseurs de services ont besoin d'une infrastructure dans laquelle les services, la gestion de ces derniers ainsi que celle des réseaux supports, puissent être introduits aussi facilement et rapidement que possible. Le logiciel jouera donc un rôle majeur dans cette nouvelle infrastructure. Par conséquent, les coûts de télécommunication seront en majeure partie dus à ce logiciel qu'il sera nécessaire de développer pour faire fonctionner et gérer le réseau et les applications qu'il supporte.

Pour répondre à la problématique posée, il sera fait appel au traitement réparti ouvert et à l'approche orientée objet. En effet, un service de télécommunication est par définition une application distribuée s'exécutant sur différents nœuds d'un réseau de télécommunication, d'où la nécessité du traitement réparti pour réduire la complexité de ce service. Par ailleurs, l'approche orientée objet est quant à elle la technologie de base utilisée pour la création et la gestion de services car elle satisfait aux critères de réutilisabilité, extensibilité et maintenance requis par les logiciels de télécommunication. Les efforts de normalisation en TMN (*Telecommunication Management Network*), IN (*Intelligent Network*), SDH (*Synchronous Digital Hierarchy*) et ATM (*Asynchronous Transfer Mode*) représentent aussi des solutions qui répondent à différents aspects du problème. De plus les deux architectures IN [Q.12xx] et TMN [M.3010] se recouvrent. Par exemple, une application TMN telle que la facturation et une application IN telle que le réseau privé virtuel, sont en relation car la taxation du VPN doit être mise en œuvre de manière consistante par la taxation vue d'un point de vue TMN. Cela montre que tant que les deux architectures IN et TMN ne sont pas intégrées, l'interconnexion entre applications TMN et IN restera complexe à mettre en œuvre. Il est donc nécessaire de disposer d'une architecture intégrant IN et TMN.

La normalisation actuelle ne traite pas des concepts de réutilisation, extensibilité et maintenance. C'est pour cette raison que le consortium TINA (*Telecommunication Intelligent Network Architecture*) a été créé. Il a pour but de définir et de valider par l'exemple, une architecture pour l'introduction de services de télécommunication qui supporte la mise en œuvre de ces services, leur gestion, et la gestion des réseaux supports de ces services.

Ce chapitre présente l'architecture de TINA. On traitera dans le paragraphe 6.1 les différents travaux sur lesquels s'appuie TINA-C pour la définition de son architecture. Dans le paragraphe 6.2, on présentera une vue d'ensemble de l'architecture TINA avec sa subdivision en quatre sous-architectures. Puis, on décrira ces dernières dans les paragraphes suivants. Le paragraphe 6.3 concernera la description de l'architecture de traitement dérivée du modèle de référence ODP. La présentation de l'architecture de service fera l'objet du paragraphe 6.4; l'architecture de gestion et ses concepts sous-jacents seront quant à eux décrits au paragraphe 6.5; l'architecture de réseau et ses modèles d'information sont détaillés au paragraphe 6.6.

Présentation des travaux TINA

Le projet TINA réutilise des travaux existants autant que possible. La réutilisation s'opère sur quatre axes comme l'indique la figure 4.1. Les axes considérés sont le traitement réparti, les architectures de réseau d'information, la création de service et la gestion des ressources. Le terme ressource est considéré dans sa dimension générique; en effet, une ressource peut représenter une ressource de réseau, de système, de service ou d'application.

La norme ODP (*Open Distributed Processing*) de l'ISO [ODP 96] est réutilisée dans l'axe traitement réparti, elle fournit un cadre pour la mise en œuvre de systèmes distribués. Afin d'introduire des services de télécommunication basés sur les concepts et les principes définis par TINA-C, TINA tient compte du standard CORBA qu'on traitera au chapitre 7 (*Common Object Request Broker*) défini par l'OMG.

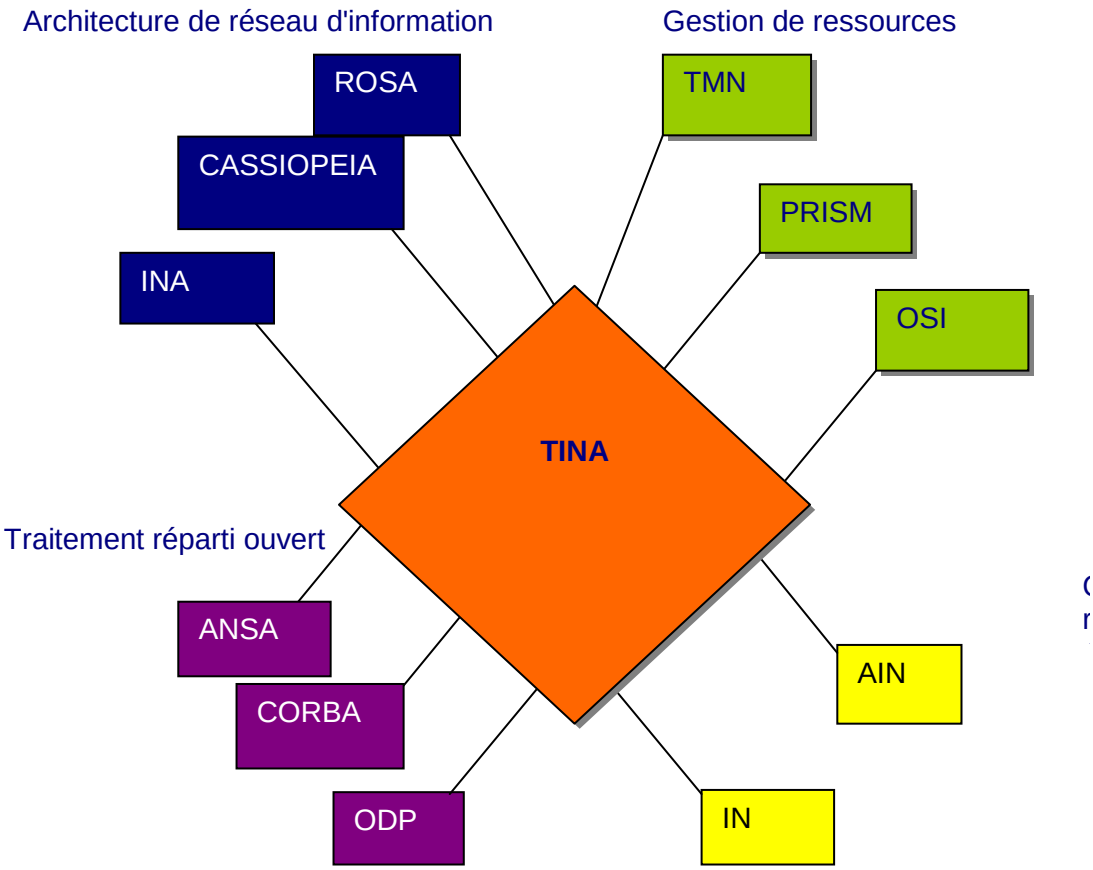


Fig. 6.1: axes de TINA

L'axe de création de service s'appuie sur les recommandations relatives à la normalisation de l'IN par l'ITU-T, et dans une certaine mesure sur celles de l'AIN (*Advanced Intelligent Network*) produites par BellCore.

L'axe gestion des ressources s'appuie sur la normalisation OSI qui utilise l'approche objet pour la gestion de réseau (s'appuyant sur le modèle de référence OSI), et fournit des notations textuelles semi-formelles à savoir GDMO (*Guidelines for the Definition of Managed Objects*) et *GRM (General Relationship Model*) pour spécifier l'information de gestion (objets et relations entre objets). De plus, TINA réutilise les travaux de l'ITU-T concernant l'architecture TMN, notamment son architecture informationnelle et fonctionnelle et les étend pour prendre en compte les dimensions service et système qui ne sont pour l'instant pas traitées par l'ITU-T. Pour l'aspect gestion de service, TINA s'inspire des résultats du projet européen PRISM (*Pan-european Reference-configuration for IBC Services*) qui définit une configuration de référence de gestion de service SMRC (*Service Management Reference Configuration*) ainsi qu'une méthode pour la mise en œuvre de services de gestion. Deux applications sont étudiées dans PRISM [PRI 93] pour valider les concepts proposés: le réseau privé virtuel ou VPN (*Virtual Private Network*) et la télécommunication personnelle universelle ou UPT (*Universal Personal Telecommunication*). Le service UPT offre la possibilité à ses usagers d'accèder aux services de télécommunication quelle que soit leur localisation et quel que soit leur terminal, aussi bien pour établir des appels que pour les recevoir.

L'axe architecture de réseau d'information s'appuie sur l'architecture INA (*Information Networking Architecture*) [RUB 94] proposée par BellCore, elle est définie comme un système de télécommunication permettant

l'accès et la gestion d'information à tout instant et à tout lieu, et sous toute forme. On constatera que l'architecture TINA ressemble plus à une spécification de l'architecture TMN qu'à une intégration entre TMN et IN.

Architecture TINA

Un service réalisé dans TINA, comme tout autre service de télécommunication ou d'information, consiste en un ensemble de composants de services interagissant entre eux. Chaque composant de service peut être lui-même décomposé en un ensemble d'objets de traitement. La communication ou l'interaction entre ces objets est supportée dans TINA par des mécanismes de distribution offerts par un environnement de traitement réparti.

Architecture d'ensemble TINA

Afin d'obtenir une bonne structuration, modularité et réutilisabilité dans l'architecture TINA, trois catégories de composants sont définies, à savoir, composants de service, composants de ressources et d'éléments (voir figure 6.2).

La première catégorie comporte des composants permettant la création de services de télécommunication et services de gestion. TINA-C a entre autre prédéfini certains composants appartenant à cette catégorie et devant prendre place dans tout déploiement de service de télécommunication. A titre d'exemple, le gestionnaire de session de communication dont le rôle est d'allouer les ressources de communication associées à une session de service. Les composants de ressources fournissent une représentation abstraite des ressources physiques disponibles. Parmi les composants de ressources sont présents les composants de ressources de transmission et des ressources de commutation. La troisième catégorie fait référence aux ressources physiques telles que les ordinateurs, les commutateurs, les équipements spéciaux (multimédia).

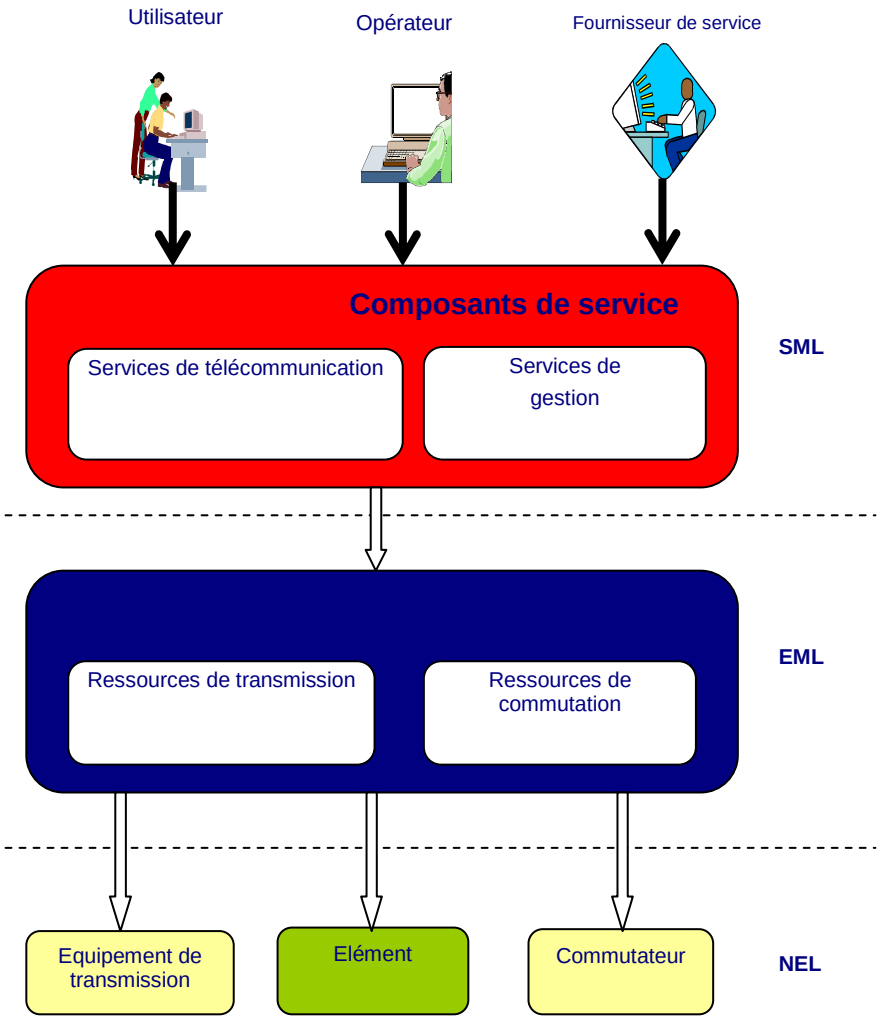


Fig 6.2: Vue d'ensemble de l'architecture TINA

L'architecture TINA est subdivisée en quatre architectures appliquant directement la méthodologie ODP (Open Distributed Processing) (voir figure 6.3), à savoir:

- l'architecture de traitement (computing architecture),
- l'architecture de service (service architecture),
- l'architecture de gestion (management architecture),
- l'architecture de réseau (network architecture),

L'architecture de traitement s'appuie fortement sur ODP afin d'offrir les bases pour la réutilisation et l'interopérabilité du logiciel de télécommunication. Des concepts de modélisation ainsi qu'une terminologie de base sont fournis pour les points de vue information, traitement et ingénierie définis dans la norme ODP. Les concepts de modélisation sont à appliquer pour la conception et le déploiement des services de télécommunication et services de gestion.

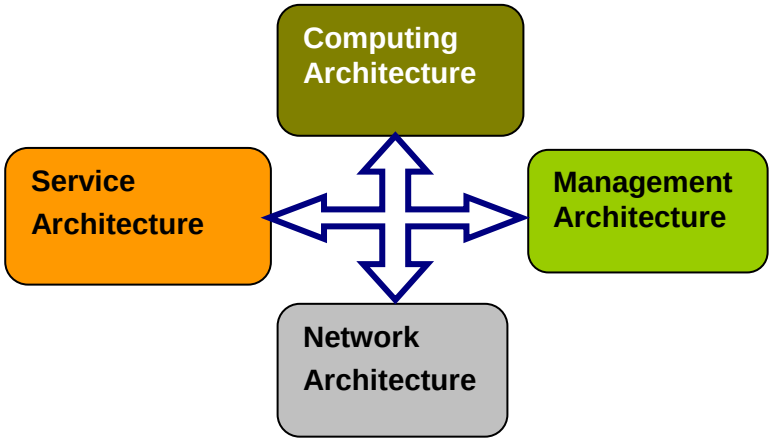


Fig. 6.3: Subdivision de l'architecture TINA

L'architecture de service (*service architecture*) offre des concepts et des principes pour réaliser, déployer et exploiter des services de télécommunication. Comme on vient de voir, un service dans le système TINA se compose de plusieurs objets interagissant entre eux et déployés sur un environnement de traitement réparti. L'architecture de service définit les objets requis pour réaliser un service, leur composition et leurs interactions. De plus un composant de service universel (USCM) a été proposé pour promouvoir la réutilisation dans la mise en œuvre de services. Trois concepts importants sont à retenir dans cette architecture:

- le concept de session qui concerne les activités de services,
- le concept d'accès relatif aux associations de l'usager et du terminal avec les services et le réseau,
- le concept de gestion pour la gestion de services.

L'architecture de gestion (*management architecture*) permet de mettre en œuvre des applications de gestion de service, de système distribué et de ressource au sens réseau grâce aux principes et aux concepts génériques qu'elle définit. Dans ces derniers, on retrouve les aires fonctionnelles de la gestion, à savoir: gestion de la configuration, fautes, sécurité et comptabilité. TINA-C définit deux nouvelles aires initialement considérées comme des fonctionnalités de la gestion de la configuration, il s'agit de la gestion de la souscription dans l'architecture de service TINA-C, et de la gestion des connexions dans l'architecture de réseau TINA-C. Un autre concept important est celui d'objet de traitement qui sert à modéliser des gestionnaires et des agents.

L'architecture de réseau (network architecture) définit un ensemble de concepts et de principes pour la spécification, la conception, l'implantation et la gestion de réseaux de transport. Elle définit également un modèle informationnel générique de ressources de réseau NRIM (*Network Resource Information Model*), des graphes de connexion fournissant une vue orientée service de la connectivité, et la gestion des connexions qui établit, contrôle et gère les connexions dans le réseau.

L'architecture de traitement

L'architecture de traitement définit les concepts et les principes de modélisation permettant de mettre en œuvre des services de télécommunication ou de gestion dans TINA, et donc fournit les bases pour les trois autres architectures. Elle offre également un environnement de traitement réparti (DPE, *Distributed Processing Environment*) qui

permet aux objets d'interagir entre eux de manière transparente. La définition de cet environnement se base sur les concepts d'ODP (traitement réparti ouvert ou *Open distributed processing*). Ce dernier est une norme pour la construction de tout système distribué, et n'est pas spécifique du monde des télécommunications.

L'architecture de traitement TINA-C cherche à raffiner et adapter la norme ODP pour qu'elle soit applicable aux services de télécommunication qui sont par définition des applications distribuées. A ce propos, la norme ODP comporte cinq points de vue qui fournissent des éléments de base pour spécifier des systèmes ODP. Ce sont:

- point de vue entreprise,
- point de vue information,
- point de vue traitement,
- point de vue ingénierie,
- point de vue technologie.

L'architecture TINA-C se concentre en particulier sur les points de vue information, traitement et ingénierie. Les prochains paragraphes présentent les concepts de modélisation de ces trois points de vue.

Modélisation d'information dans TINA

Elle a pour but de décrire la structure modélisant l'information dans le système TINA. Elle spécifie d'une part les entités et de l'autre part les relations entre ces entités. La modélisation d'information dans TINA prend en compte les contraintes et les règles qui gouvernent le comportement des entités et des relations, y compris les règles de création et de suppression.

Chaque entité est modélisée sous forme d'un objet d'information. Le concept de type d'objet est dans TINA-C considéré comme étant un prédicat caractérisant une collection d'objets. Les objets sont décrits par des gabarits spécifiant leurs caractéristiques communes. Ceci avec un certain niveau de détails suffisants pour réaliser leur instanciation. Un gabarit caractérise donc l'ensemble des objets qu'il instancie. Les actions de chaque objet sont décrites dans son gabarit par leurs noms et des paramètres en entrée et en sortie. Ces actions spécifient des changements d'états possibles. L'état d'un objet caractérise l'information qu'il détient à un instant donné. A ces actions sont aussi associées des pré-conditions et des post-conditions.

Les relations entre objets matérialisent les dépendances entre objets et facilitent ainsi la compréhension du modèle. Chaque relation est définie par un paramètre *arité* qui représente le nombre d'objets participant à une relation. La majorité des relations ont généralement une *arité* égale à 2, et sont appelées relations binaires. Toute action peut avoir un état et des attributs qui changent de valeurs par le biais des actions. Comme pour les objets, un type de relation est un prédicat caractérisant une collection de relations. Une position dans une relation est appelée un rôle. Un type de relation associe chaque rôle à un type d'objet. Par exemple, *études* est un type de relation binaire avec les deux rôles *étudiant* et *école*. Le rôle *étudiant* est associé au type *personne* alors que le rôle *école* est associé au type *établissement_scolaire*. Ainsi toute *personne* qui étudie dans un *établissement_scolaire* est une instance de ce type de relation, (Ahmed, INPT). Il est possible de définir des contraintes sur les types de relation, par exemple la suppression d'un objet doit impliquer la suppression

d'autres objets dans la relation. Par ailleurs la cardinalité de rôle est définie comme un ensemble d'entiers positifs qui contraint le nombre de relations de ce type qui ont le même objet dans l'autre rôle.

Les types d'objets et de relation peuvent aussi être représentés graphiquement en utilisant la notation OMT (*Object Modeling Technique*) [RUM 91]. Les types d'objets sont représentés par des rectangles où leurs noms sont en gras dans la première partie du rectangle (voir figure 6.4). Les attributs figurent dans la deuxième partie du rectangle, chaque nom d'attribut peut être suivi par des détails optionnels comme le type et la valeur par défaut. Il n'est pas nécessaire de spécifier des attributs dans une classe. Dans le dernier tiers du rectangle sont placées les opérations. Comme pour les attributs chaque nom d'opération peut être suivi par des détails optionnels, à savoir la liste d'arguments et le type du résultat. La spécification d'opérations est aussi optionnelle. Elle dépendend également du niveau de détail souhaité. Le sous-typage est représenté par un triangle.

Les types de relation sont représentés par des arcs entre les types d'objet. Le nom de la relation apparaît en italique; il est optionnel. Les types de relation d'arité n sont représentés par un losange connectant leurs arcs aux types d'objet impliqués. La cardinalité s'exprime par un nombre ou un ensemble d'intervalles. Un petit rond noir correspond à "many" signifiant zéro ou plus.

Il est également possible de représenter textuellement les types d'objet et de relation. Pour ce, TINA-C se base sur les notations proposées par la gestion OSI. Il s'agit du GDMO (voir chapitre 4) qui permet de spécifier semi-formellement les types d'objet et GRM [X.725] défini afin de spécifier des types de relation. Ces notations textuelles sont adaptées par TINA-C afin de les rendre conformes aux concepts de modélisation d'information. Elles sont appelées Quasi-GDMO et Quasi-GRM.

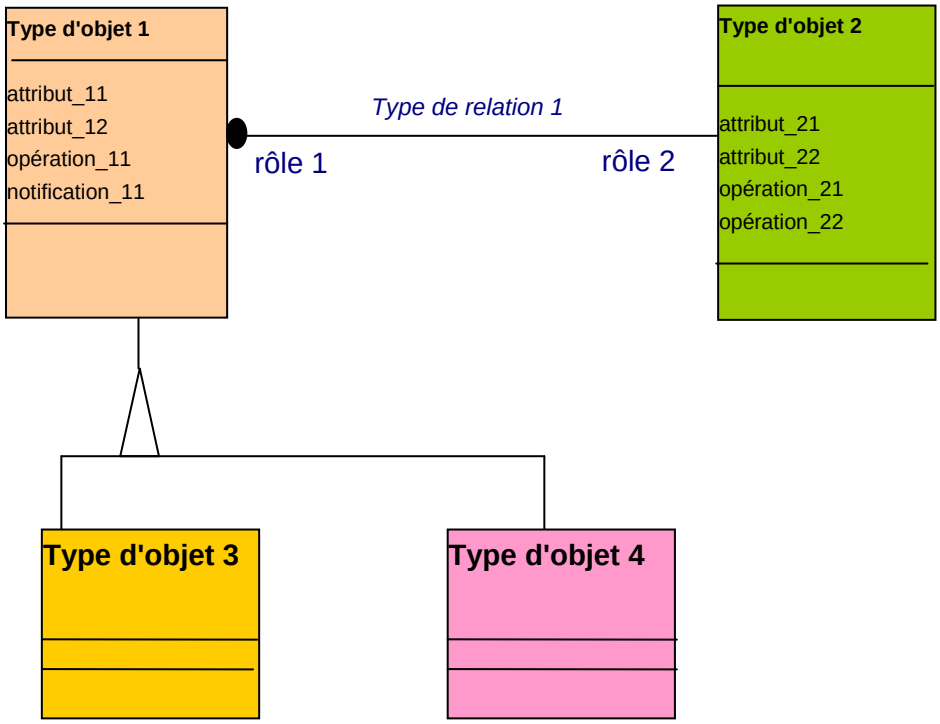


Fig. 6.4: notations graphiques pour les types d'objet et de relation définies par OMT

Modélisation de traitement dans TINA

Du point de vue traitement, une application distribuée consiste en un ensemble d'objets de traitement. Chaque objet de traitement encapsule des données de traitement et fournit un ensemble de fonctions (services) accessibles par d'autres objets. Ces services sont offerts au travers d'interfaces de traitement.

TINA-C a défini le langage ODL (*Object Definition Language*) [KIN 95] pour la spécification semi-formelle des objets de traitement et de leurs interfaces. L'ODL est une extension de l'OMG-IDL (voir chapitre 7 et annexe 1), il est en fait un sur-ensemble d'OMG-IDL. Ce qui permet aux services de télécommunication ou de gestion spécifiés avec TINA-ODL d'être construits au-dessus de plates-formes réparties ORB (*Object Request Broker,* voir chapitre 7) commerciales conformes aux spécifications de l'OMG.

La spécification d'objet est introduite par le mot clé OBJECT suivi du nom attribué à un objet.

OBJECT NetworkConnectionManager
BEHAVIOR "Cet objet crée des connexions de réseau"
REQUIRES ElementConnectionManagerIF;
INITIAL NetworkConnectionManagerConfigurationIF;
CONSTRAINT {Attributs de service: débit de connexion, temps d'établissement…}
SUPPORTS NetworkConnectionManagerIF

Une spécification de comportement (BEHAVIOR) définit en langage naturel le rôle d'un objet qui offre des services via des interfaces; la clause REQUIRES indique les interfaces requises qui sont offertes par d'autres objets et utilisées par cet objet. La clause INITIAL désigne l'interface d'initialisation de l'objet. la clause CONSTRAINT décrit les contraintes applicables à l'objet comme les contraintes de qualité de service, d'usage et de gestion. Enfin la clause SUPPORTS déclare les interfaces offertes ou supportées par l'objet. La spécification d'objet suivante décrit un gestionnaire de connexion de réseau.

Quant à la spécification d'interface, elle est introduite par le mot clé INTERFACE suivi du nom d'une interface; dans certains cas une interface peut hériter d'une ou plusieurs interfaces dont les noms sont indiqués. Une spécification de comportement BEHAVIOR décrit le service offert et précise l'usage du service stipulant des contraintes de séquencement sur les opérations définies sur cette interface. Ces contraintes doivent être vérifiées lors de l'invocation d'opérations. Une spécification d'attribut de service déclare les types et les identifiants d'attribut de qualité de service. Une clause signature d'interface permet de spécifier une signature d'interface ou de flot. La spécification d'interface suivante décrit l'interface

`NetworkConnectionManagerConfigurationIF`

qui hérite de l'interface ServiceManagementIF.

INTERFACE NetworkConnectionManagerConfigurationIF:
ServiceManagementIF {
BEHAVIOR "Cette interface permet de configurer l'objet de traitement NetworkConnectionManager. L'opération readState retourne la valeur de l'état de cet objet. L'opération WriteState permet de changer la valeur de l'état de l'objet "
USAGE "l'opération Init doit être invoquée avant toute autre opération définie. Des invocations concurrentes de l'opération ReadState sont permises mais toutes les demandes d'opérations WriteState sont bloquées tant qu'une opération WriteState est en cours de traitement"
Void Init;

Modélisation d'ingénierie dans TINA

La modélisation d'ingénierie TINA-C décrit l'organisation d'une infrastructure abstraite appelée TINA DPE (TINA Distributed Processing Environment, voir figure 6.5) qui permet d'exécuter les objets de traitement TINA-C [GRA 95]. Les applications TINA consistent en un ensemble d'objets de traitements s'appuyant sur cet environnement de traitement réparti TINA, qui permet leur déploiement, leur exécution et leurs interactions.

Dans le point de vue ingénierie, un objet de traitement devient un objet de traitement d'ingénierie eCO (Engineering Computational Object). Il existe une différence d'abstraction entre ces deux objets. En effet, l'objet de traitement a la vue uniquement des interactions avec d'autres objets de traitement, alors qu'un eCO peut interagir avec d'autres objets d'ingénierie différents des eCOs. Ceci revient au fait que les eCOs interagissent avec des eCOs qui établissent leurs communications avec d'autres eCOs ou qui offrent des transparences à la distribution.

L'infrastructure TINA DPE offre des services aux objets des applications qu'elle exécute. Ces services offrent une interface de traitement par laquelle ils sont accèssibles à partir des objets de traitement TINA. Les services offerts sont de deux types: des fonctions pour supporter les mécanismes d'interactions entre objets ou des fonctions suffisament génériques pour être appliquées à un grand nombre de domaines.

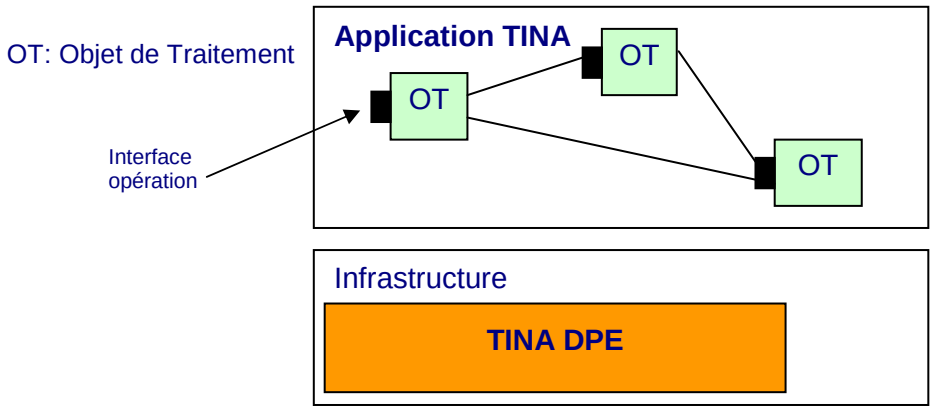


Fig. 6.5: TINA DPE, l'infrastructure abstraite

Ces services sont encapsulés dans le DPE (voir figure 6.6). Les services existants actuellement sont au nombre de six:

• service de courtage: publie et recherche les interfaces qu'utiliseront des objets pour offrir des services à d'autres objets.

• service de transaction: établit une communication transactionnelle entre les objets possèdant les propriètés ACID (Atomicité - Concurrence - Intégrité - Durabilité).

• service conteneur: permet de stocker des spécifications de types d'objet et d'interface. Le service de conteneur d'implémentation permet de stocker l'implémentation de l'objet sous la fome d'un code exécutable et d'informations liées à son implémentation.

• service de notification: offre la possibilité aux objets d'émettre ou de recevoir des notifications sans tenir compte des objets avec lesquels ils communiquent.

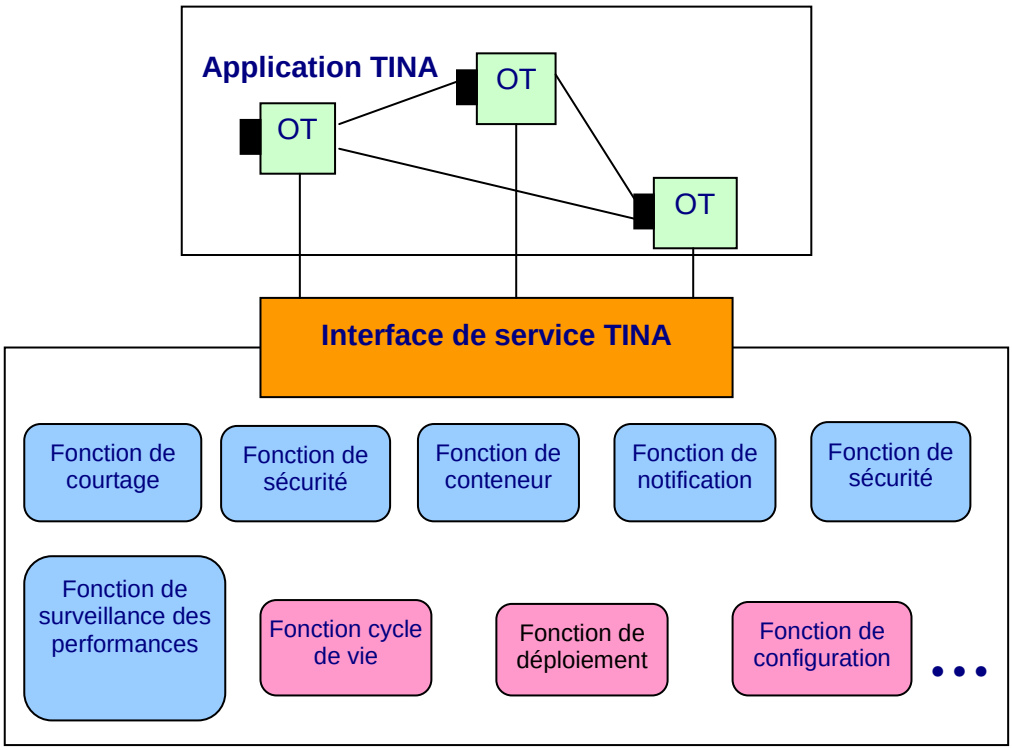


Fig. 6.6: Services et fonctions du DPE TINA

- service de sécurité: permet aux applications TINA de faire l'authentification, l'autorisation, le contrôle d'accès, etc.
- service de surveillance des performances: fournit les mesures et les rapports de performance de l'activité des ressources de réseau.

TINA-C a aussi proposé trois autres services relatifs à la gestion du cycle de vie des objets:

- service de cycle de vie: permet la création, la suppression, l'activation, la désactivation et la migration d'eCOs.
- service installation: permet le déploiement des eCOs.

- service de configuration: donne des informations sur la configuration des eCOs.

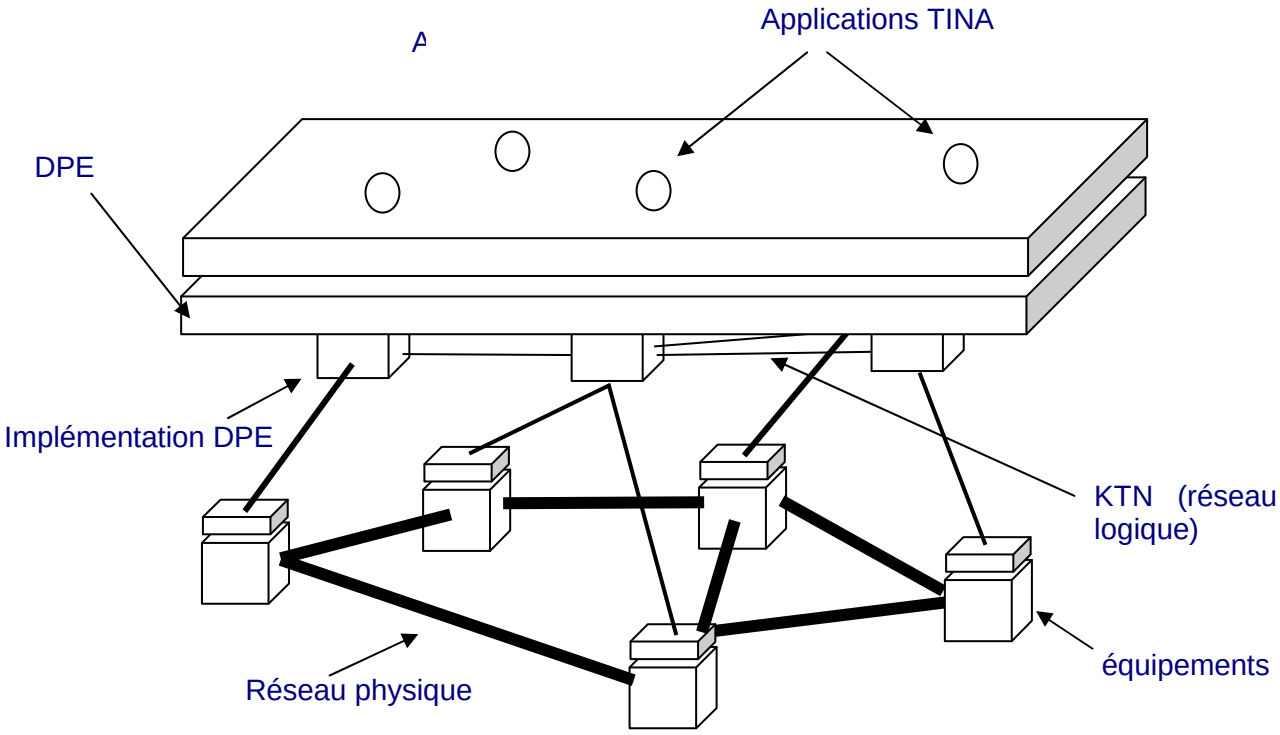


Fig. 6.7: le DPE vu par TINA-C

La figure 6.7 montre les différents niveaux du DPE TINA. Le niveau des applications TINA communique avec un niveau DPE s'appuyant sur un réseau abstrait mettant en œuvre un réseau de gestion, le KTN ou Kernel Transport Network. Les nœuds du réseau KTN communiquent avec des nœuds du réseau physique.

Il est à noter que le KTN est un équivalent du DCN (Data Communication Network) du TMN, mais avec la différence notable qu'il se situe à un niveau de réseau logique et non physique.

Un des aspects puissants du DPE est justement le découplage entre un niveau de réseau logique sur lequel est établit le réseau de gestion KTN, et un niveau de réseau physique mettant en œuvre les équipement eux-mêmes. Ce découplage permet au DPE d'être totalement affranchi des technologies de réseaux, des topologies etc.

On peut considérer le DPE comme une sorte de super-CORBA (cf. chapitre suivant). La richesse des concepts proposés ici est également à comparer à une architecture émergente: les GRIDs. En effet, un nœud DPE semble finalement assez proche de ce que propose un nœud GRID en termes de vision objet, d'autonomie, d'intelligence locale active coopérante. La vision DPE dénote ici une grande avance sur son temps.

L'architecture de service

L'architecture de service consiste en un ensemble de concepts, de principes et de règles qui permettent de concevoir et construire des systèmes au sens large du terme. L'architecture de service de TINA-C

définit donc les concepts et les principes de base nécessaires à la mise en œuvre de services TINA. Ces derniers sont regroupés en trois catégories:

- services de télécommunications: établissent la connexion et transportent l'information entre les différents terminaux connectés sur le réseau de télécommunication.
- services de gestion: assurent la gestion de la configuration, des fautes, des performances, de la comptabilité et de la sécurité des ressources TINA.
- services d'information: permettent le stockage et la visualisation des ressources d'information tels que le son, la vidéo, le texte, etc.

La notion de session

Afin de pouvoir déployer les services multimédia, TINA-C a introduit la notion de session généralisant ainsi le concept d'appel. ce dernier étant uniquement dédié à la négociation et à l'allocation des ressources de communication ne supporte pas les fonctionnalités spécifiques au service. Par contre, la session de service peut utiliser différents modèles et procédures d'appel pendant sa durée de vie en fonction des besoins des applications.

Une session est définie par la relation temporaire entre un ensemble de ressources ayant une tâche à accomplir pendant une période bien définie. Ces ressources peuvent être des utilisateurs par exemple. Il existe quatre types de session (voir figure 6.8):

- session de service (*Service Session*): elle concerne la simple activation d'un service. Elle présente des informations de gestion et de contrôle du service comme le nombre d'utilisateurs, profils de service, etc. Mais aussi des informations caractérisant une utilisation du service comme le rôle des utilisateurs impliqués, capacité de communication allouée, etc. Elle se décompose en deux sessions:
  - session de service usager (*User Service Session*): permet à l'usager de personnaliser sa vue de la session de service. Elle facilite l'utilisation du service en cachant sa complexité à l'usager. Ce dernier est sûr que ses préférences et son environnement seront supportés par le service. Une session usager est créée lorsqu'un utilisateur se joint à une session de service; elle est supprimée lorsque celui-ci quitte la session.
  - session de service fournisseur (*Provider Service Session*): fournit à l'ensemble des clients du fournisseur un environnement permettant l'exécution d'un service. Elle définit l'ensemble des activités d'un service et fournit le contrôle requis sur ce service.

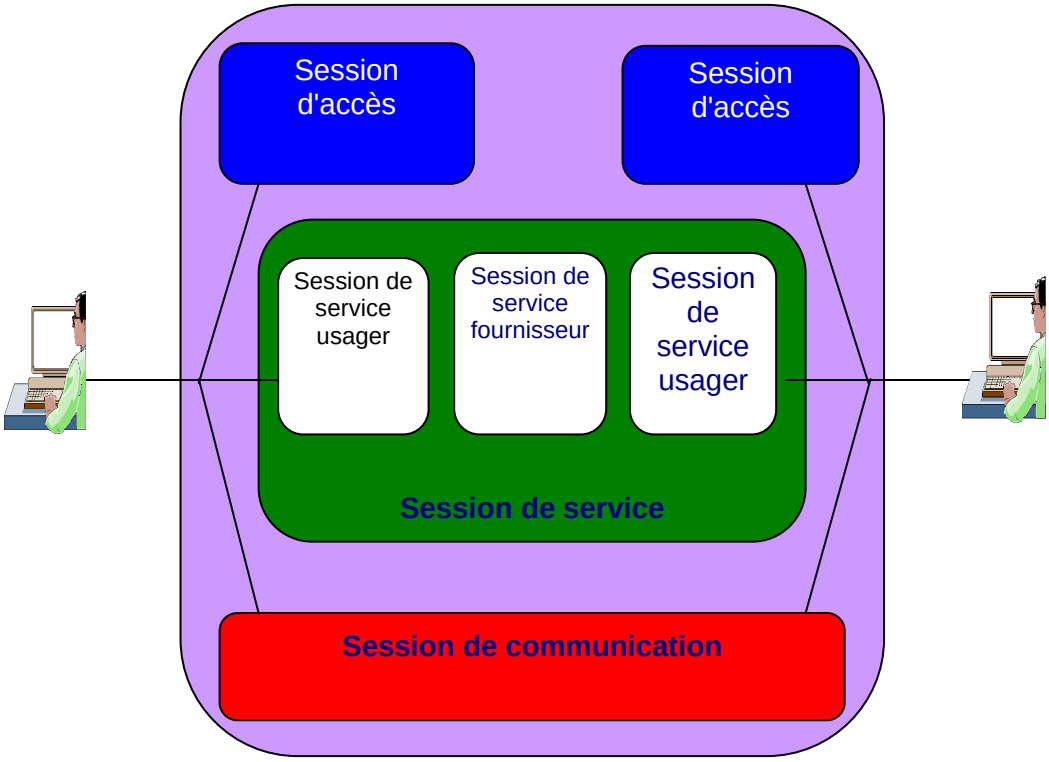


Fig. 6.8: Concept de session dans TINA

- session d'accès (*Access Session*): gère l'implication de l'utilisateur dans plusieurs services simultanément. En effet, un utilisateur peut se connecter à un système et lancer des sessions de services et en même temps se joindre à plusieurs sessions de service simultanées. La session d'accès gère également la connexion de l'utilisateur au système.
- session de communication (*Communication Session*): représente les connexions qu'utilise le service dans le réseau de transport: chemin de communication, point de terminaison, qualité de service, etc.

Ainsi TINA sépare complètement les aspects relatifs aux services (session d'accès, session de service, session de service fournisseur et usager) et les aspects relatifs aux capacités de communication (session de communication).

Les composants de service TINA

TINA propose plusieurs composants de services réutilisables pour la création d'un grand nombre de services. Ces composants relatifs aux sessions introduites précédemment sont de trois types :

- composants relatifs à la session d'accès,
- composants relatifs à la session de service.
- composants relatifs à la session de communication

La première catégorie comporte les composants suivants:

- l'agent usager ou UA (*User Agent*): représente l'utilisateur dans le système TINA. Il permet à son usager d'initialiser une nouvelle session de service ou de se joindre à des sessions existantes. Il est indispensable pour l'accès aux services par l'utilisateur, il est indépendant des services.
- L'agent fournisseur ou PA (*Provider Agent*): également indépendant des services permet à son utilisateur d'accéder à son UA. En terme de

session d'accès, il transporte des données d'un utilisateur vers son UA pour créer de nouvelles sessions de service, pour se joindre à une session existante, etc.

- L'agent initial ou IA (*Initial Agent*): représente le point d'accès initial à un fournisseur de service. Il est aussi indépendant des services.

La deuxième catégorie se compose des composants de service suivants:

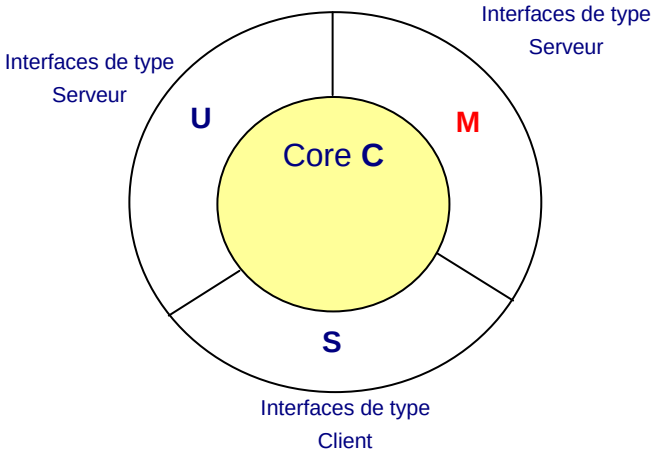
- l'application usager ou UAP (*User Application*): représente des applications dans le domaine usager. Il offre l'interface utilisateur d'une application en jouant le rôle de point terminal dans une session de service.
- le Service Factory ou SF est un composant de service spécifique à un type de service donné, dont le rôle est de créer les composants de la session de service pour ce type de service. Pour des besoins de gestion, il peut fournir la liste des sessions de service actives et peut libérer certaines d'entre elles à la demande.
- Le gestionnaire de session de service ou SSM (*Service Session Manager*): supervise et contrôle les ressources utilisées dans la session de service. Il supporte l'ajout / le retrait / l'invitation / d'utilisateurs à/de la session en interagissant avec les UAs correspondants. Enfin, il offre des fonctionnalités de gestion pour la session comme la facturation. Un composant SSM est créé par un SF lorsqu'un service est invoqué à travers une session d'accès.
- le gestionnaire de session utilisateur ou USM (*User Session Manager*): supervise et contrôle la session usager, il représente l'utilisateur avec ses caractéristiques particulières. Un composant USM est créé par un SF lorsqu'un usager se joint à une session de service, elle est supprimée lorsque l'utilisateur quitte la session de service.

La troisième catégorie de composant se compose du gestionnaire de session de communication ou CSM (*Communication Session Manager*): il offre un service de connectivité au SSM. Il est indépendant des services. Il transforme une demande applicative (établir/maintenir/libérer une association) en connexions de réseau de bout en bout. Il représente donc le point de contrôle pour l'allocation des ressources de communication.

En résumé, les composants relatifs à la session d'accès offrent un cadre pour la fourniture d'accès sécurisés et personnalisés aux services. Les composants relatifs à la session de service fournissent un cadre pour la définition de services accessibles et gérables. Les SSMs et les USMs sont instanciés par des SFs à partir de demandes des UAs. Quant au composant UAP, il permet les interactions entre l'utilisateur et la session de service, tandis que le composant relatif à la session de communication alloue les ressources de communication.

Le composant de service universel

TINA définit un modèle de composant de service universel appelé USCM (*Universal Service Component Model*) qui permet de décrire chaque composant de service de télécommunication. En effet, pour décrire un service de télécommunication il faut séparer les aspects relatifs à l'accès au service, à sa logique, à sa gestion et à l'utilisation des ressources ou



- **M:** *Management Sector*.
- **U**: *Usage Sector*.
- **S**: *Substance Sector*.
- **C**: *Core Sector*.

Fig. 6.9: structure du modèle USCM

autres services. Ce modèle se compose de quatre parties (voir figure 6.9):

- Core: correspond à la logique du service, c'est à dire ses fonctionnalités spécifiques qu'attendent les utilisateurs,
- Usage: correspond aux interfaces utilisateur du service pour y accéder et l'utiliser,
- Substance: correspond aux dépendances du service envers des ressources externes ou d'autres services.
- Management: aspects liés à la gestion du service.

Les interactions permises entre les composants de service TINA sont au nombre de quatre (voir figure 6.10):

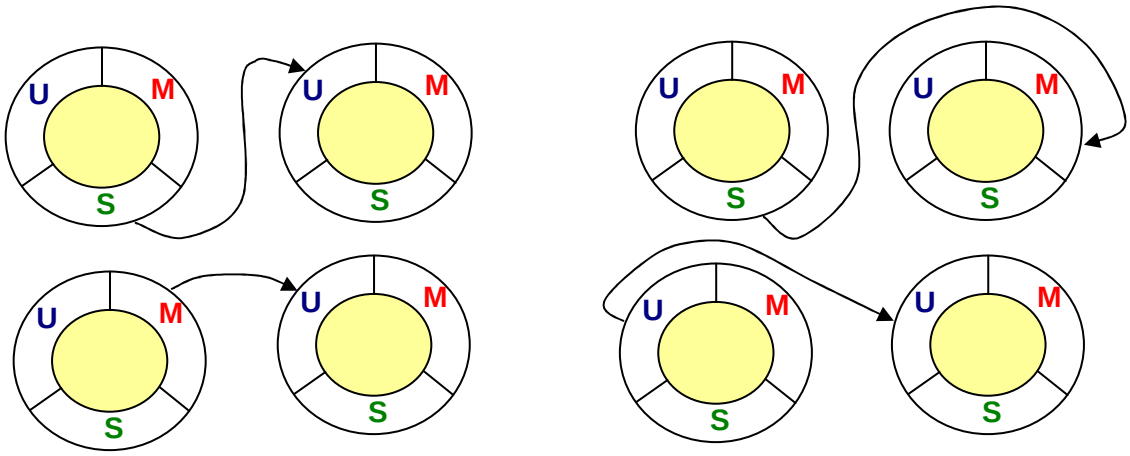


Fig. 6.10: interactions permises entre composants de service TINA

- Interaction S-U: a lieu lorsqu'un composant nécessite un service extérieur;
- Interaction S-M: se produit quand un composant de gestion (gestionnaire) veut superviser un composant de service. Le secteur *Substance* du gestionnaire invoque le secteur *Management* du

composant service à l'aide d'opérations comme Get, Set, reroute, réinitialise, etc;

- Interaction M-U: a lieu lorsqu'un composant de service est en faute, il émet une notification depuis son secteur *Management* où la faute est détectée vers du secteur Usage d'un gestionnaire qui analysera et corrigera la faute;
- Interaction U-U: se produit quand un composant de service ne peut offrir le service demandé par son client, il délègue alors la demande à un autre composant en mesure de le fournir.

L'architecture de gestion

La gestion dans TINA se répartit en trois types: gestion de service, gestion de ressource, et gestion du DPE (*Distributed Processing Environnment*) (voir figure 6.11):

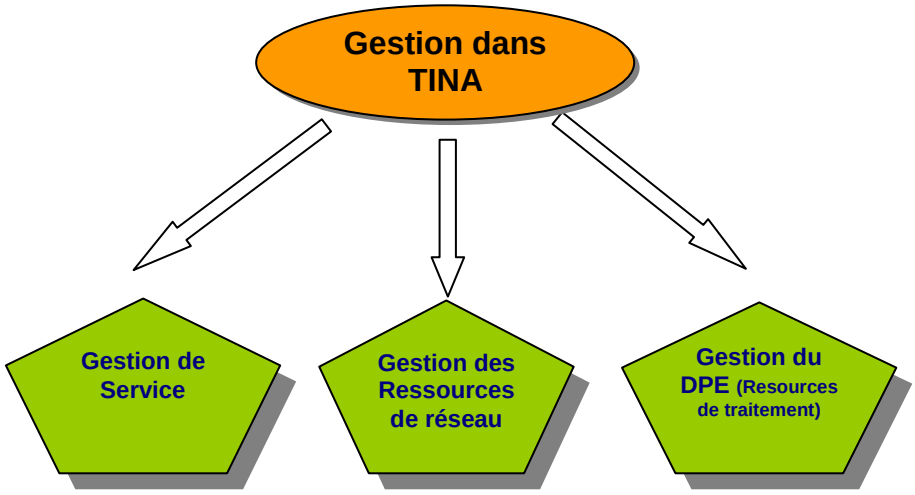


Fig. 6.11: Gestion dans TINA

La gestion de service concerne tous les services définis dans l'architecture TINA. La gestion des ressources s'applique à toutes les ressources de réseau utilisées pour la fourniture de ces services. Quant à la gestion du DPE, elle gère l'infrastructure de l'environnement de traitement réparti. Ce dernier fournit les transparences de distribution aux applications de télécommunications et de gestion de TINA-C. La gestion du DPE est encore en cours de spécification par TINA-C.

La gestion de service prend place dans l'architecture de service car un service de télécommunication doit intégrer des fonctionnalités de gestion pour qu'il soit complètement défini et prêt à être déployé. La gestion des ressources est intégrée dans les architectures de gestion et de réseau. La gestion du DPE est mise en œuvre dans l'architecture de traitement.

Les aires fonctionnelles de gestion

TINA-C reprend les cinq aires fonctionnelles du TMN (gestion de la configuration, des fautes, des performances, de la comptabilité et de la sécurité). TINA-C décompose la gestion de la configuration en:

- gestion des connexions: se charge de l'établissement, du contrôle et de la libération des connexions;

- gestion de configuration des ressources: permet l'installation, la fourniture, le contrôle et la gestion des ressources de réseau.

Les principes de gestion dans TINA-C

L'architecture de gestion TINA-C se base sur les concepts d'architecture que TINA-C a défini mais aussi sur les recommandations de gestion ITU-T et ISO. L'architecture logique répartie en couches définie dans la recommandation ITU-T M.3010 est applicable dans l'architecture de gestion TINA-C. Les couches considérées sont les couches gestion d'élément de réseau, gestion de réseau et gestion de service. TINA-C regroupe les deux couches basses en une couche appelée couche gestion de ressource. Les objets de gestion OSI ou ITU-T sont spécifiés dans TINA-C sous forme d'objets d'information selon les concepts de modélisation d'information. Les blocs de fonctions peuvent être représentés par un ou plusieurs objets de traitement. Par exemple, un bloc OSF qui réalise une application de facturation ou de configuration peut être mis en œuvre par plusieurs objets de traitement. Du fait que les objets de traitement sont spécifiés en utilisant l'approche orientée objet, ils sont donc plus facilement réutilisables que les blocs de fonctions TMN (voir figure 6.12)

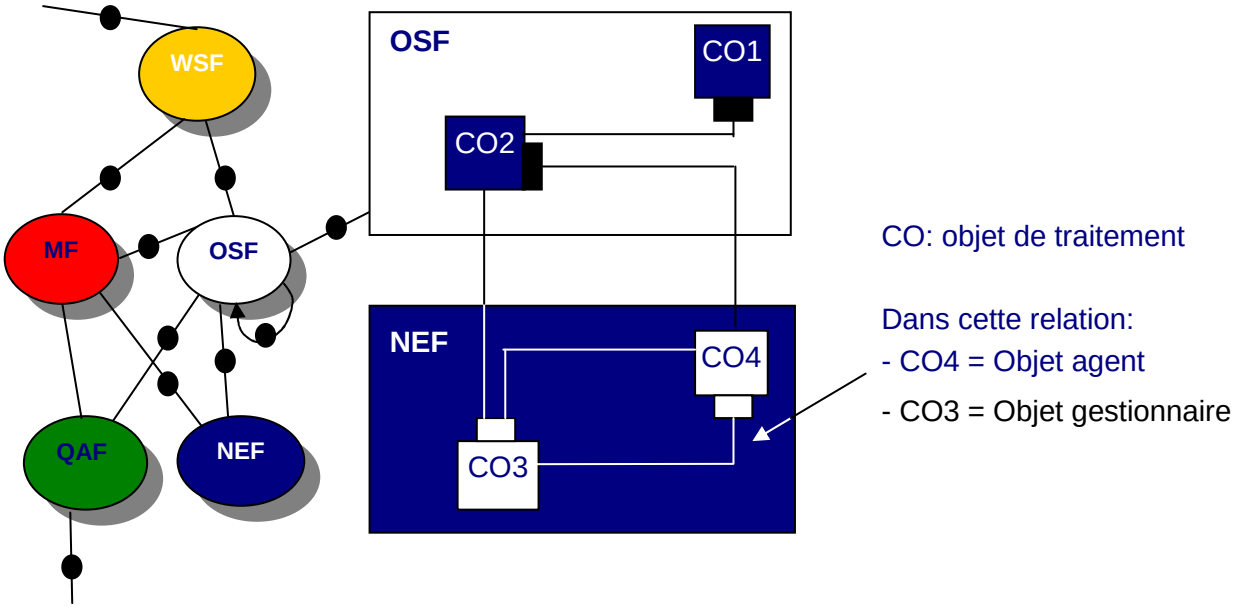


Fig. 6.12: Blocs de fonction, objets de traitement, points de référence et interfaces

Les points de référence TMN peuvent quant à eux aussi être supportés par un ensemble d'interfaces opération. Ces dernières sont spécifiées grâce au langage ODL; elles sont typées et décrivent un service bien défini à la différence des points de référence qui fournissent une description générale d'une fonctionnalité. Des interactions en cascade entre les objets de traitement existent, dans le sens où un objet de traitement peut jouer en même temps le rôle de gestionnaire dans une interaction et le rôle d'agent dans l'autre.

L'architecture de réseau

L'architecture de réseau TINA-C définit le modèle informationnel générique de ressource de réseau NRIM (Network Resource Information Model) ainsi qu'une architecture de gestion des connexions permettant l'établissement, le contrôle et la gestion des connexions dans un réseau.

Présentation du modèle NRIM

Le modèle informationnel NRIM décrit les classes d'objets qui modélisent les ressources de réseau tout en restant indépendant de toute technologie de commutation. Ces classes sont réutilisables afin de faciliter la mise en œuvre d'applications de gestion de service et de réseau [FUE 95]. Le modèle NRIM a été influencé par deux recommandations de l'ITU-T, à savoir M.3010 qui décrit un modèle informationnel générique de réseau et G.803 qui définit une architecture de réseau de transport basée sur la hiérarchie digitale synchrone, SDH (*Synchronous Digital Hierachy*). L'apport de NRIM par rapport aux deux recommandations consiste en la prise en compte par NRIM du niveau gestion de ressources (gestion de réseau) que M.3010 ne spécifie pas. Le modèle d'information TINA-C est applicable aux différents modèles existants qui décrivent le niveau élément de réseau, à savoir l'ATM, SONET, SDH, etc. Ceci revient à la généricité du modèle NRIM. Ce dernier sépare les aires fonctionnelles du modèle informationnel de la gestion OSI et propose des classes d'objets pour les aires gestion des fautes et gestion de la configuration.

NRIM est défini par un ensemble de fragments tout comme le modèle M.3010 de l'ITU-T (voir figure 6.13). Chaque fragment traite un aspect bien spécifique.

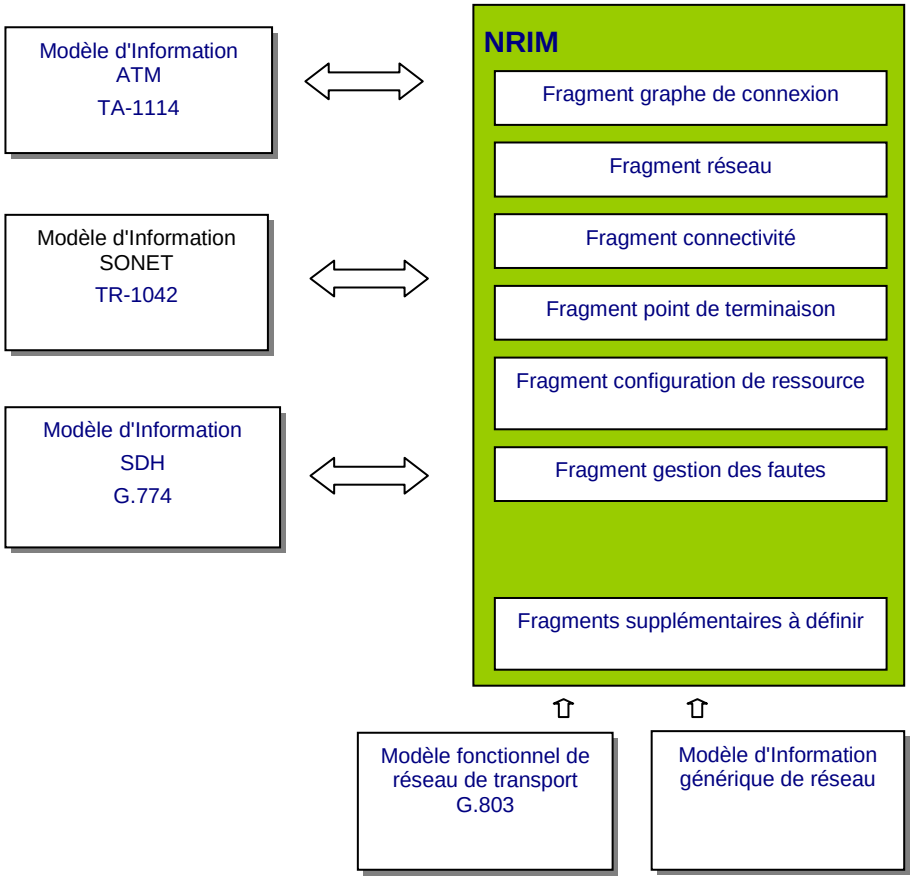


Fig. 6.13: le modèle d'information de ressources de réseau défini par TINA-C et ses fragments

• Fragment graphe de connexion: CG (*Connection Graph*) permet d'exprimer la connectivité requise par les services de télécommunication durant leur exécution indépendamment de toute technologie. Le modèle comporte trois objets (voir figure 6.14):

- le port (Port) modélise le point d'accès à un lien;
- le lien (Line) représente un flot d'information entre des ports dans le graphe de connexion.
- le vertex (Vertex) modélise l'ensemble des ressources impliquées dans le transport des flux d'information. il peut représenter par exemple des équipements tels que la caméra, la moniteur, le microphone, etc.

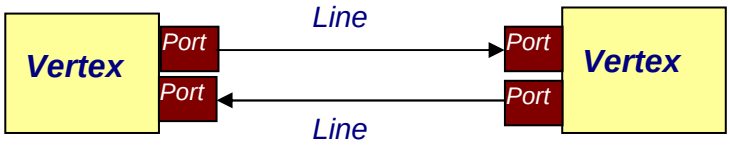


Fig. 6.14: le graphe de connexion

• Fragment réseau: modélise la structure et la subdivision d'un réseau en couches. Ce dernier contient des chemins, des connexions, des liens topologiques et des sous-réseaux. Un chemin transporte des informations entre points de terminaison du réseau en couches. Une connexion décrit la connectivité entre deux sous-réseaux. Un ensemble de connexions peut être groupé pour former un lien topologique.

• Fragment connectivité: la connexion et la connexion de sous-réseaux sont deux concepts qui permettent de modéliser la connectivité à

l'intérieur d'un réseau en couches. Le premier concept décrit la connectivité entre deux sous-réseaux (connexion de liens), elle est pré-configurée. Le deuxième concept qui celui de la connexion de sous-réseaux ou SNC (*Subnetwork connection*) décrit la connectivité à l'intérieur d'un sous-réseau.

- Fragment point de terminaison: les points de terminaison sont des points d'accès à un réseau en couches ou à un sous-réseau. Un point de terminaison de connexion de réseau NWCTP (*Network Connection Termination Point*) termine une connexion ou une connexion de sous-réseau. Un point de terminaison de chemin NWTTP (*Network Trail Termaination Point*) termine un chemin. Il représente le point d'accès à un réseau en couches. Le point de terminaison de lien LTP (*Link Termination Point*) modélise une terminaison d'un lien topologique.
- Fragment configuration de ressource: trois types de ressources sont présents dans l'architecture TINA, à savoir les ressources de réseau, les ressources du DPE et les ressources de service. Une classe d'objet appelée *configurable* est une super classe de toute classe d'objet à configurer. Les attributs de cette classe sont *état administratif* et *état opérationnel*. Des actions et notifications de base relatives à la configuration sont aussi définies pour la classe d'objet *configurable* tels que lock, unlock et notification de changement d'état.
- Fragment gestion des fautes: ce fragment définit des objets supports pour la gestion des fautes. La généricité de ces objets leur permet d'être non seulement applicables aux ressources de réseau mais aussi aux ressources du DPE ou celles de service. L'aire fonctionnelle de la gestion des fautes TINA-C recouvre les activités de surveillance d'alarme, de localisation de fautes, de correction de fautes et de tests/diagnostics.

Architecture de gestion des connexions TINA-C

La gestion des connections dans TINA a pour but d'établir les connections que demandent les services de télécommunication. Le terme session a un sens différent selon le niveau auquel on se place. Au niveau le plus haut de l'architecture, il représente une liaison entre interfaces flot d'objets de traitement. Cette liaison ou association est décomposée de manière récursive pour devenir une connexion de transport de bout en bout, puis une connexion de sous-réseau et enfin une connexion entre ports d'un commutateur. L'architecture des connections doit être vue comme une application distribuée qui s'exécute sur le DPE. La fonctionnalité de gestion des connections est définie par un ensemble d'objets de traitement selon les concepts de modélisation de traitement; ces objets sont:

- le gestionnaire de session de communication CSM (*Communication Session Manager*);
- le coordinateur de réseau en couche LNC (*Layer Network Coordinator*);
- les gestionnaires de connection CPs (*Connection Performers*).

Un service de télécommunication est du point de vue traitement réparti un ensemble d'objet de traitement s'exécutant sur un DPE. Ces derniers interagissent au travers d'interfaces flot. Les interfaces opération sont liées dynamiquement et implicitement par les services du DPE. Alors que les interfaces flot sont liées uniquement de façon explicite. L'objet CSM, défini au niveau le plus haut de l'architecture de gestion des connexions offre les services nécessaires pour lier des interfaces flot. Le client de

l'objet CSM, généralement un service de télécommunication, indique les références des interfaces à interconnecter, ainsi que les caractéristiques de la connection logique telle que la qualité de service. Cette spécification de connection est indépendante de la technologie, de la topologie du réseau support et de la distribution.

L'objet CSM doit prendre en compte le fait que la connection entre interfaces d'objet de traitement doit être mise en œuvre dans un réseau de télécommunication. Elle peut devenir par exemple une connection ATM de bout en bout. Cette dernière connecte les points de terminaison des terminaux sur lesquels sont exécutes les objets de traitement dont les interfaces flot sont à lier. L'interconnection de points de terminaison dans un réseau de télécommunication est réalisée par l'objet CC. Le client de l'objet CC, à savoir l'objet CSM ou tout client souhaitant une connection de niveau transport, spécifie les adresses de points de terminaison à interconnecter, ainsi que les caractéristiques de la connection.

Cette spécification de connection est indépendante de la technologie et de la structure du réseau sous-jaçent.

Pour offrir son service, l'objet CC doit sélectionner l'objet LNC afin que celui-ci établisse un chemin dans un réseau en couches. Un réseau en couches est décomposé en domaines, chacun sous la responsabilité d'un opérateur de réseau par exemple. Chaque domaine a son propre objet LNC. L'objet CC demande la mise en œuvre d'un seul chemin à un seul objet LNC; ce dernier prend en charge la fédération entre les domaines nécessaires. Pour le client d'un objet LNC, ce dernier est le seul point d'accès dans tout réseau en couches. Il est à noter que des objets LNC appartenant à des réseaux en couches différents n'interagissent pas entre eux. Un objet CPE LNC est présent dans le domaine de l'utilisateur. il met en œuvre la fédération entre domaine public (réseau d'opérateur) et domaine privé (réseau privé local).

Un objet NML CP met à la disposition de l'objet LNC des services lui permettant d'établir un chemin dans le réseau en couche. Le NML CP contrôle un sous-réseau dans un réseau en couches, il y fournit l'interconnection de points de terminaison. Il permet donc l'établissement, le contrôle et la libération de connections de sous-réseau point-à-point unidirectionnelles et bidirectionnelles ou point-à-multipoint unidirectionnelles. Pour réaliser la connection physique de sous-réseau, l'objet NML CP utilise les services des objets EML

L'objet EML CP accède aux agents des équipements du réseau. Il permet donc d'établir, de contrôler et de libérer les connections dans l'équipement dont il est responsable.

La figure 6.15 est un exemple possible d'architecture de gestion des connections résumant les enchaînements décrits précédemment. Il adopte l'architecture logique répartie en couches définie par le TMN. Les services de télécommunication se situent dans la couche SML. La couche NML contient les objets CSM, CC, LNC, et NML CP. Les objets EML CP sont présents à la couche EML. La couche NEL est naturellement formée des éléments physiques de télécommunication.

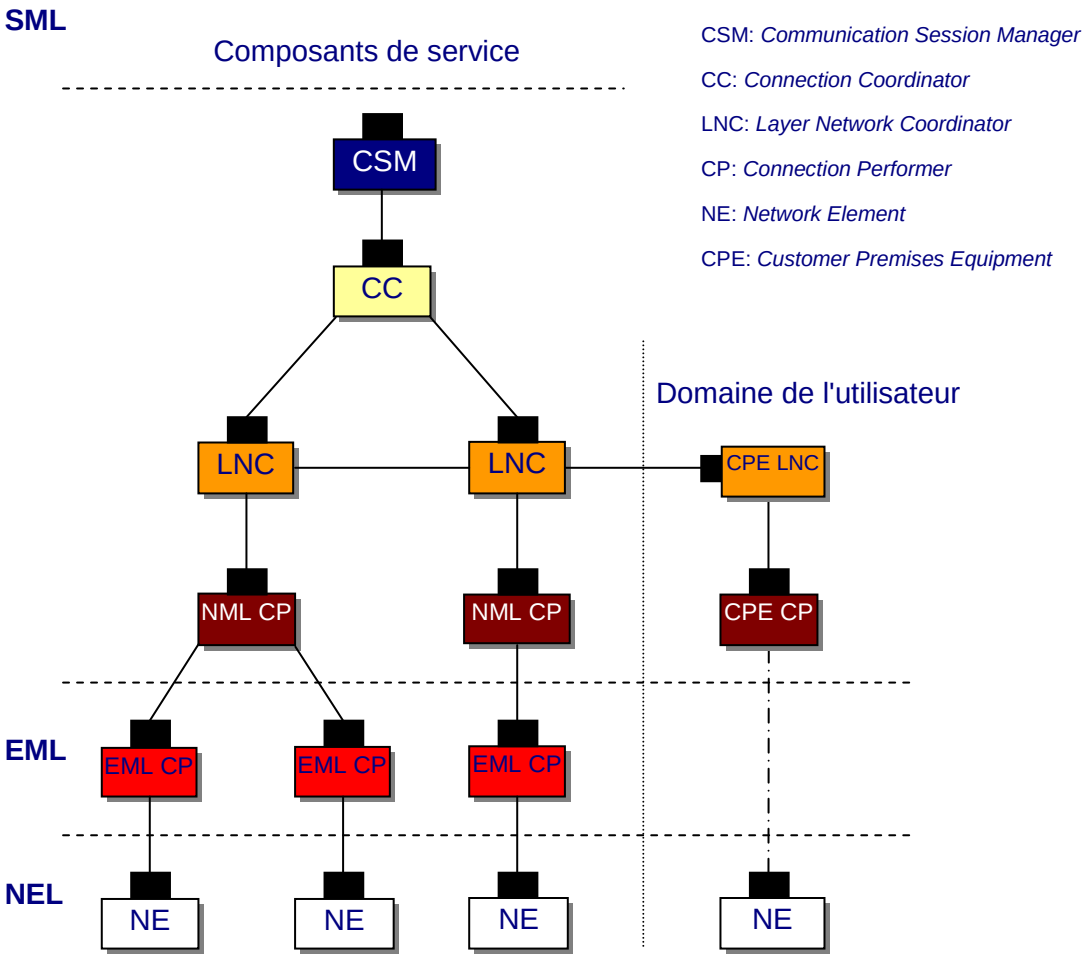


Fig. 6.15: Exemple d'architecture de gestion des ressources

Conclusion

Le consortium TINA a spécifié une architecture logicielle adéquate. Cette dernière intègre le traitement réparti et l'approche orientée objet aux systèmes de télécommunication. TINA facilite ainsi la réutilisation des spécifications, la répartition flexible du logiciel et l'homogénéité dans la mise en œuvre des services. Plusieurs opérateurs faisant partie du consortium TINA-C envisagent de migrer à long terme de l'IN (*Intelligent Network*) vers TINA. Certains d'entre eux ont déjà mis en œuvre des prototypes TINA pour éprouver les concepts, principes et règles définies.

Un résultat majeur de TINA est d'avoir influencé l'ITU-T suffisamment pour accepter CORBA comme alternative à CMIS/P pour le protocole.

TINA, malgré un manque certain de réalisations industrielles, reste un réservoir à idées d'importance majeure. Il est probable que TINA a pêché par son avance sur son temps, en restant incompris de la majorité de la communauté TMN traditionnelle. Mais cette situation n'est peut-être pas définitive.

On verra au chapitre 8 que le NGOSS du TeleManagement Forum réutilise avec intelligence un grand nombre de concepts de TINA.

Chapitre 7 - CORBA et la gestion des réseaux

Introduction

La révolution que connaît actuellement la programmation distribuée revient aux spécifications des standards pour les systèmes distribués et à l'utilisation des langages orientés objet. CORBA est notamment une spécification normative qui résulte d'un consensus entre les membres de l'OMG, un consortium qui regroupe aujourd'hui plus de 850 acteurs de l'industrie informatique. Dans ce chapitre on présentera la norme CORBA, son architecture, ses composants et ses objectifs.

Survol de CORBA

CORBA est l'acronyme de *Common Object Request Broker Architecture,* qui est la solution de l'OMG (*Object Management Group*) pour l'interopérabilité des machines et des logiciels. Le nombre et la diversité de ces derniers ne cessant pas de croître met en évidence l'idée d'une standardisation des outils de communication. CORBA vient justement pour permettre aux applications de communiquer entre elles sans se préoccuper de la machine où elles se trouvent non plus de ceux qui les ont conçues. La première version CORBA 1.1 a vu le jour en 1991, elle a défini l'IDL (*Interface Definition Language*) et l'API qui permet l'interaction d'objets client/serveur à travers un bus de communication appelé ORB (*Object Request Broker*). La deuxième version CORBA 2.0 apparue en 1994, définit une vraie interopérabilité puisqu'elle spécifie comment des ORBs de différents constructeurs peuvent communiquer [OMG].

OMA, L'architecture globale de CORBA

L'OMG définit aussi une vision globale de la construction d'applications réparties: l'Object Management Architecture Guide (OMAG). Cette architecture globale, appelée aussi l'OMA, vise à classifier les différents objets qui interviennent dans une application en fonction de leurs rôles. En effet la norme CORBA traite les éléments constitutifs des systèmes d'objets distribués, à savoir: un langage de description d'objets (IDL), une infrastructure de distribution des objets appelée Bus d'Objets Répartis (ORB), une description de services communs nécessaires aux objets applicatifs (Services Objet Communs ou "CorbaServices"), une description de services communs aux applications (Utilitaires Communs ou "CorbaFacilities"), une collection de descriptions de services par branche industrielle (Interfaces de Domaine ou "Domain Interfaces") et une spécification normative d'interopérabilité entre ORB (Corba2) [OMG] (*voir figure 7.1)*.

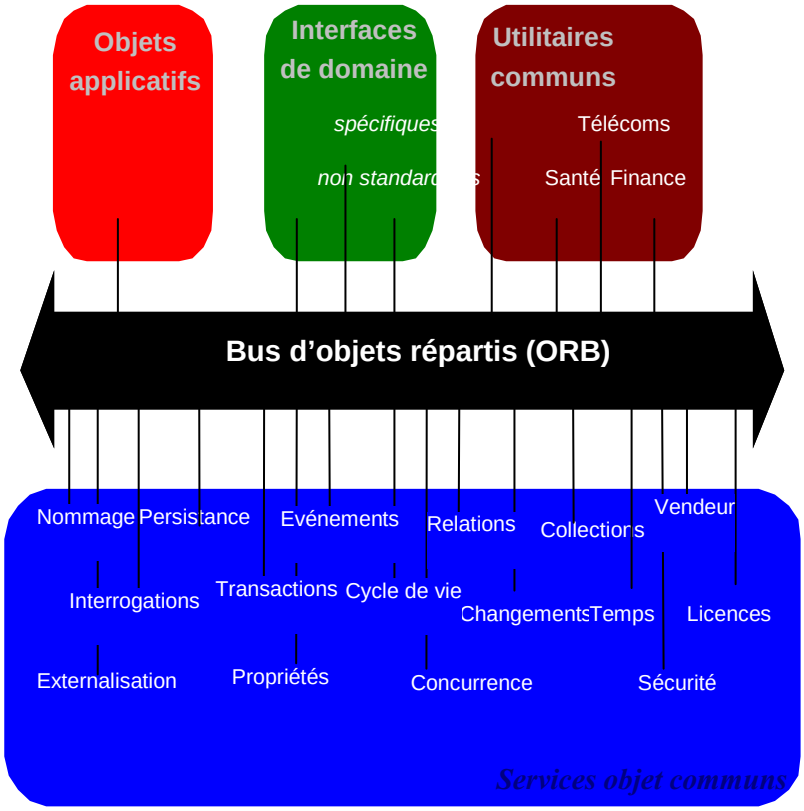


Fig. 7.1: OMA, l'architecture globale de CORBA

- Le **bus d'objets répartis** est la clé de voûte de l'architecture globale de l'OMG. Il assure le transport des requêtes entre tous les objets CORBA. Il offre un environnement d'exécution aux objets masquant l'hétérogénéité liée aux langages de programmation, aux systèmes d'exploitation, aux processeurs et aux réseaux. Ce bus est plus amplement détaillé dans la suite.
- Les **services objet communs** (*CorbaServices*) fournissent sous forme d'objets CORBA, spécifiés grâce au langage OMG-IDL, les fonctions systèmes nécessaires à la plupart des applications réparties. Actuellement, l'OMG a défini des services pour les annuaires (Nommage et Vendeur), le cycle de vie des objets, les relations entre objets, les événements, les transactions, la sécurité, la persistance, etc. Au fur et à mesure des besoins, l'OMG ajoute de nouveaux services communs. Ces services seront détaillés dans le paragraphe suivant.
- Les **utilitaires communs** (*CorbaFacilities*) sont des canevas d'objets qui répondent plus particulièrement aux besoins des utilisateurs. Ils standardisent l'interface utilisateur, la gestion de l'information, l'administration, etc.
- Les **interfaces de domaine** (*Domain Interfaces*) définissent des objets de métiers spécifiques à des secteurs d'activités comme la finance (monnaie électronique), la santé (dossier médical) et les télécoms (transport multimédia). Leur objectif est de pouvoir assurer l'interopérabilité sémantique entre les systèmes d'informations d'entreprises d'un même métier: *les «Business Object Frameworks »* (BOFs).

- Les **objets applicatifs** (*Application Objects*) sont ceux qui sont spécifiques à une application répartie et ne sont donc pas standardisés. Toutefois, dès que le rôle de ces objets apparaît dans plus d'une application, ils peuvent alors rentrer dans une des catégories précédentes et donc être standardisés par l'OMG [OMG].

Les services objet communs

Les services objets communs couvrent les services de nommage, de persistance, d'annuaire, de cycle de vie des objets, etc. Ces services tous décrits à l'aide de l'IDL, sont nécessaires dans les systèmes distribués. Ils permettent d'isoler clients et serveurs des détails d'implémentation dus à leur localisation ou à leur état d'activation. Leur objectif est donc de standardiser les interfaces des fonctions système indispensables à la construction et à l'exécution de la plupart des applications réparties.

La recherche d'objets

Cette catégorie de services offre les mécanismes pour rechercher/retrouver dynamiquement sur le bus les objets nécessaires aux applications. Ce sont les équivalents des annuaires téléphoniques:

- Le service **Nommage** (*Naming Service*) est l'équivalent des «pages blanches»: les objets sont désignés par des noms symboliques. Cet annuaire est matérialisé par un graphe de répertoires de désignation.
- Le service **Courtage** (*Trader Service*) est l'équivalent des «pages jaunes»: les objets peuvent être recherchés en fonction de leurs caractéristiques.

La vie des objets

Cette catégorie regroupe les services prenant en charge les différentes étapes de la vie des objets CORBA.

- Le service **Cycle de Vie** (*Life Cycle Service*) décrit des interfaces pour la création, la copie, le déplacement et la destruction des objets sur le bus. Il définit pour cela la notion de fabriques d'objets «*Object Factory*».
- Le service **Propriétés** (*Property Service*) permet aux utilisateurs d'associer dynamiquement des valeurs nommées à des objets. Ces propriétés ne modifient pas l'interface IDL, mais représentent des besoins spécifiques du client comme par exemple des annotations.
- Le service **Relations** (*Relationship Service*) sert à gérer des associations dynamiques (appartenance, inclusion, référence, auteur, emploi,...) reliant des objets sur le bus. Il permet aussi de manipuler des graphes d'objets.
- Le service **Externalisation** (*Externalization Service*) apporte un mécanisme standard pour fixer ou extraire des objets du bus. La migration, le passage par valeur, et la sauvegarde des objets doivent reposer sur ce service.
- Le service **Persistance** (*Persistent Object Service*) offre des interfaces communes à un mécanisme permettant de stocker des objets sur un support persistant. Quel que soit le support utilisé, ce service s'utilise de la même manière via un «*Persistent Object Manager*». Un objet persistant doit hériter de l'interface «Persistent Object» et d'un mécanisme d'externalisation.

- Le service **Interrogations** (*Query Service*) permet d'interroger les attributs des objets. Il repose sur les langages standards d'interrogation comme SQL3 ou OQL. L'interface *Query* permet de manipuler les requêtes comme des objets CORBA. Les objets résultats sont mis dans une collection. Le service peut fédérer des espaces d'objets hétérogènes.
- Le service **Collections** (*Collection Service*) permet de manipuler d'une manière uniforme des objets sous la forme de collections et d'itérateurs. Les structures de données classiques (listes, piles, tas,..) sont construites par sous-classement. Ce service est aussi conçu pour être utilisé avec le service d'interrogations pour stocker les résultats de requêtes.
- Le service **Changements** (*Versionning Service*) permet de gérer et de suivre l'évolution des différentes versions des objets. Ce service maintient des informations sur les évolutions des interfaces et des implantations. Cependant, il n'est pas encore spécifié officiellement.

La sûreté de fonctionnement

Cette catégorie de services fournit les fonctions système assurant la sûreté de fonctionnement nécessaire à des applications réparties.

- Le service **Sécurité** (*Security Service*) permet d'identifier et d'authentifier les clients, de chiffrer et de certifier les communications et de contrôler les autorisations d'accès. Ce service utilise les notions de serveurs d'authentification, de clients/rôles/droits (et délégation de droits), d'IIOP sécurisé.
- Le service **Transactions** (*Object Transaction Service*) assure l'exécution de traitements transactionnels impliquant des objets distribués et des bases de données en fournissant les propriétés ACID (Atomicité - Concurrence - Intégrité - Durabilité).
- Le service **Concurrence** (*Concurrency Service*) fournit les mécanismes pour contrôler et ordonnancer les invocations concurrentes sur les objets. Le mécanisme proposé est le verrou. Ce service est conçu pour être utilisé conjointement avec le service Transactions.

Les communications asynchrones

Par défaut, la coopération des objets CORBA est réalisée selon un mode de communication client/serveur synchrone. Toutefois, il existe un ensemble de services assurant des communications asynchrones.

- Le service **Evénements** (*Event Service*) permet aux objets de produire des événements asynchrones à destination d'objets consommateurs à travers des canaux d'événements. Les canaux fournissent deux modes de fonctionnement. Dans le mode « push », le producteur a l'initiative de la production, le consommateur est notifié des événements. Dans le mode « pull », le consommateur demande explicitement les événements, le producteur est alors sollicité.
- Le service **Notification** (*Notification Service*) est une extension du service précédent. Les consommateurs sont uniquement notifiés des événements les plus intéressants. Pour cela, ils posent des filtres sur le canal réduisant ainsi le trafic réseau engendré par la propagation des événements inintéressants.
- Le service **Messagerie** (*CORBA Messaging*) définit un nouveau modèle de communication asynchrone permettant de gérer des

requêtes persistantes lorsque l'objet appelant et l'objet appelé ne sont pas présents simultanément sur le bus.

Autres fonctions

- Le service **Temps** (*Time Service*) fournit des interfaces permettant d'obtenir une horloge globale sur le bus (*Universal Time Object*), de mesurer le temps et de synchroniser les objets.
- Le service **Licences** (*Licensing Service*) permet de mesurer et de contrôler l'utilisation des objets, et cela en vue de facturer les clients et de rémunérer les fournisseurs.

Les interfaces de domaine

L'OMG vise à travers ces spécifications d'offrir une description standard des objets et des services communs à une industrie donnée ou à un secteur d'activité. Ces efforts sont effectués au sein de groupes de travail nommés selon chaque secteur, parmi ces groupes:

- **Business Objects DTF** standardise les méthodologies et les technologies pour la conception d'applications métier au dessus de CORBA.
- **Workflow Workgroup** travaille sur les outils CORBA nécessaires au fonctionnement des applications de gestion de flux de travail indispensables aux activités coopératives au sein des entreprises.
- **C4I DSIG** fait la promotion de CORBA au sein des organismes et administrations liés à la défense en jouant un rôle pédagogique.
- **End User SIG** travaille en relation avec les utilisateurs finaux à fin de déterminer leurs besoins, de connaître leurs expériences pour mieux orienter les futurs technologies CORBA.
- **CORBAmed** définit les standards OMG pour le domaine de la médecine. Ces standards permettront l'interopérabilité entre les acteurs de la santé en définissant par exemple la notion d'objets patient.
- **Life Sciences Research DTF** est un nouveau groupe dédié à l'utilisation de CORBA par les instituts de recherche dans les sciences de la vie: les universités et les laboratoires pharmaceutiques.
- **Electronic Commerce** s'intéresse à la définition de technologies CORBA pour les applications de commerce électronique.
- **Financial Services DTF** produit les spécifications OMG liées au domaine de la finance/banque comme, par exemple, la définition d'objets monnaie et d'objets convertisseur de monnaie.
- **Telecom DTF** fait la liaison entre le monde CORBA et le monde des télécommunications.
- **Internet Platform SIG** travaille sur les rapprochements des technologies CORBA avec les technologies d'Internet. Il étudie principalement les relations avec le *World Wide Web*.
- **Manufacturing DFT** tente de combler le fossé entre les technologies OMG et les besoins des industries.
- **Realtime SIG** réfléchit sur la prise en compte des aspects temps réel au sein du bus CORBA.
- **Distributed Simulation SIG** adresse les problèmes liés à la simulation distribuée et son exécution au dessus d'un bus CORBA.

- **Test SIG** doit définir les méthodologies, les outils, les protocoles pour automatiser les tests d'applications réparties CORBA.
- **Object Model Working Group** travaille sur l'unification des modèles orientés objet en vue de créer un standard plus généraliste que celui proposé actuellement par CORBA.
- **UML Revision Task Force** travaille sur le futur du langage UML. Ce langage, standardisé par l'OMG, est une notation graphique pour définir des modèles orientés objet [OMG].

Le langage de description d'objets IDL

Les services CORBA sont décrits par l'IDL (*Interface Definition Language*). C'est un langage de spécification indépendant de tout langage de programmation utilisé pour implémenter les comportements des objets. La communication entre les programmes client et serveur se fait via l'ORB, qui représente en fait l'infrastructure de communication. La robustesse de la norme CORBA revient notamment à cette séparation entre le client et le serveur, qui est à la fois en terme de langage de programmation mis en œuvre, de protocoles réseau et de mécanismes de transport de données. Cet isolement permet aussi de concrétiser les bénéfices attendus de l'adoption des systèmes d'objet distribués.

L'IDL est donc un outil utilisé pour définir les interfaces d'objets d'une façon universelle. C'est un langage de description et non d'implémentation. Il ne comprend pas donc de constructeurs car il n'implémente rien, il ne fait que décrire. Il ajoute un certain nombre de mots clé pour spécifier les systèmes distribués. L'IDL est utilisé pour rendre les applications complètement indépendantes des langages.

L'IDL est le résultat d'un travail de l'OMA (*Object Management Architecture*) qui consistait à donner naissance à un langage indépendant. Le langage OMG-IDL (*Interface Definition Language*) permet d'exprimer, sous la forme de contrats IDL, la coopération entre les serveurs et les clients de services, en séparant l'interface de l'implémentation des objets et en masquant les divers problèmes liés à l'interopérabilité, l'hétérogénéité et la localisation de ceux-ci. Un contrat IDL spécifie les types manipulés par un ensemble d'applications réparties, c'est-à-dire les types d'objets (ou interfaces IDL) et les types de données échangés entre les objets. Le contrat IDL isole ainsi les clients et serveurs de l'infrastructure logicielle et matérielle les mettant en relation à travers le bus CORBA.

De l'IDL aux langages de programmation

Une fois le langage d'implémentation choisi, un mapping (ou projection) entre le source IDL et le langage désiré doit être fait grâce à un compilateur IDL (voir figure 7.2). Le compilateur IDL produit du code source pour le client et pour le serveur dans le langage de programmation cible [CAJB]. Ces compilateurs génèrent automatiquement la souche et la squelette dans la langage de programmation choisi. La souche implante la liaison entre l'application cliente et l'ORB, tandis que la squelette implante la liaison entre l'ORB et le serveur pour l'objet considéré. Il reste

en suite au programmeur que d'écrire, du côté serveur les methodes de l'objet, et du côté client la logique d'envoie des messages.

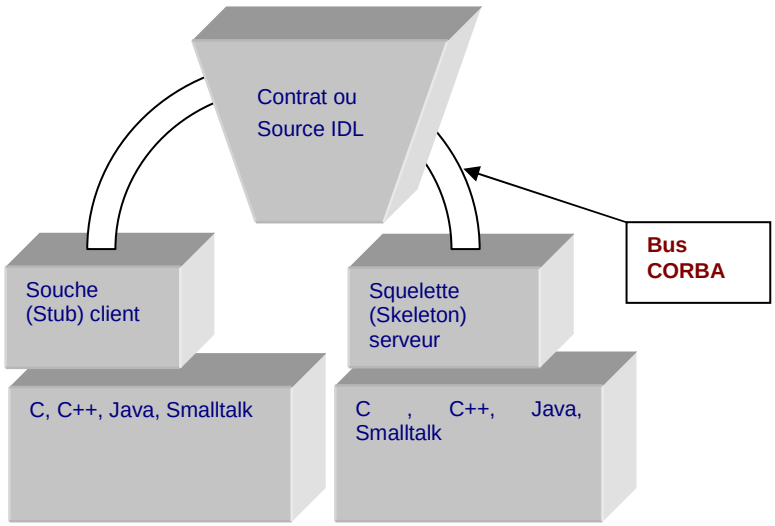


Fig. 7.2: Compilateur IDL

La traduction de l'IDL vers un langage d'implantation s'appelle la projection. Les contrats IDL sont projetés en souches IDL (ou interface d'invocations statiques SII) dans l'environnement de programmation du client et en squelettes IDL (ou interface de squelettes statiques SSI) dans l'environnement de programmation du fournisseur. Le client invoque localement les souches pour accéder aux objets. Les souches IDL construisent des requêtes, qui vont être transportées par le bus, puis délivrées par celui-ci aux squelettes IDL qui les délégueront aux objets. Ainsi le langage OMG-IDL est la clé de voûte du bus d'objets répartis CORBA.

Afin de garantir la portabilité des applications d'un bus vers un autre, les règles de projection sont normalisées. Elles traduisent précisément chaque construction IDL en une ou plusieures constructions du langage cible. Elles définissent également les règles d'utilisation correctes de ces traductions. Les règles existantes actuellement concernent les langages suivants: C, C++, SmallTalk, Ada, Java et Cobol orienté objet. Le tableau 7.1 donne des exemples de ces règles de projection:

Construction OMG-DL	Projection en C++	Projection en Java
`module M { … };`	Au choix du produit CORBA: `namespace M { … }` ou `class M { … }` ou préfixe `M_`	Package M
`Interface J:I {…};`	`Class J:public virtual I {…};`	`Interface J extends I {…}`
`String`	`char *`	`java.lang.String`

Tableau 7.1: Exemples de règles de projection

Ainsi, chaque produit CORBA fournit un pré-compilateur IDL pour chacun des langages supportés, mais aussi de bus CORBA cible. Le code des applications est alors portable d'un bus à un autre car les

souches/squelettes générés s'utilisent toujours de la même manière quel que soit le produit CORBA. Par contre, le code des souches et des squelettes IDL n'est pas forcément portable car il dépend de l'implantation du bus pour lequel ils ont été généré.

Le mode statique et le mode dynamique

Le précompilateur OMG-IDL génère automatiquement les souches de communication avec les objets. Ces souches utilisent le bus pour réaliser la coopération des objets des applications. Cette approche statique est bien adaptée pour la conception et l'exécution d'applications dont les spécifications OMG-IDL sont stables. Néanmoins, dans la plupart des applications complexes, certaines spécifications évoluent au cours du temps par l'ajout de nouvelles opérations et/ou de nouveaux types d'objets ou par la modification des spécifications existantes. Dans ces contextes, l'approche statique, par prégénération automatique de souches, ne convient plus. Elle crée un lien statique (à la compilation) entre les applications clientes et les interfaces IDL des objets utilisés. Ainsi, lorsque les interfaces évoluent, il faut modifier et recompiler toutes les applications clientes et serveurs.

D'un autre côté, CORBA offre un ensemble de mécanismes pour exploiter et implanter dynamiquement des objets répartis: le référentiel des interfaces (IFR), l'interface d'invocations dynamiques (DII) et l'interface de squelettes dynamiques (DSI). Ces mécanismes dynamiques (qu'on détaillera dans la partie 7.3) permettent de construire des applications qui s'adaptent automatiquement aux changements et évolutions des spécifications IDL.

La syntaxe IDL

Ce paragraphe montre à l'aide d'un exemple simple la syntaxe d'IDL. Il s'agit dans cet exemple d'un guichet bancaire qui distribue automatiquement des billets, en utilisant une carte de crédit. La présentation d'un tel objet et ses interactions avec le clavier, le lecteur de cartes et l'imprimante est comme suit:

Dans un premier temps on s'aperçoit que la syntaxe de l'IDL présente d'indiscutables ressemblances avec celle de C++. Cependant, de

```
// Exemple DL pour un distributeur automatique de billets
module DAB{  typedef string BandeMagnetique;
        interface Entrée { typedef string CommandeOpérateur;
                        void SaisieCarte( in BandeMagnetique item);
                        void SaisieClavier( in CommandeOpérateur cmd); };
        interface Sortie{  boolean Impression( in string texte); };
        interface TerminalDAB{ void FinDeTransaction();
        void RapportDeTransaction();
         };
};
```

nombreuses différences existent entre les deux: la sémantique n'est pas la même, les paramètres dans les déclarations doivent être nommés et quantifiés, etc. IDL est un langage de description fortement typé. En effet, toute variable et attribut d'objet doit être d'un type formellement défini. L'IDL comporte une longue liste de types de base à partir desquels de nouveaux types peuvent être définis, ainsi qu'un type passe-partout, le

type any qui englobe tous les autres. Le type any est décodé, à l'arrivée, à l'aide d'une spécification de type appelée typecode. Toutes les définitions de constantes, d'exceptions, d'interfaces et de modules ont une étendue et ne sont valides qu'à l'intérieur de la section où elles apparaissent. Pour importer des définitions extérieures, il faut utiliser l'opérateur "::". Dans notre exemple, la variable "Commande Opérateur" n'est valide que dans l'interface Entrée.

Une interface est un mot clé du langage IDL qui regroupe des sous-ensembles cohérents de constantes, de types, d'exceptions, d'opérations et de variables.

Une opération est définie par trois parties obligatoires et trois autres facultatives. Dans cet exemple aucune option facultative n'apparaît. Les trois options requises sont le type de la valeur renvoyée par l'opération, le nom de l'opération, et enfin la liste des paramètres de l'opération. Chaque paramètre est qualifié par l'un des mots clé *in*, *out*, ou *inout*, suivi de son type et de son nom. Dans l'exemple de l'opération impression, le type de la valeur renvoyée est *boolean*, le nom de l'opération est impression et il n'y a qu'un paramètre de type de base *string,* nommé *texte.* Le mot clé *in* indique que ce paramètre est non modifiable.

L'ORB: le bus d'objets répartis CORBA

L'ORB est l'entité chargée de l'acheminement des requêtes et du retour des résultats ou des exceptions vers le client. Le bus CORBA est donc l'intermédiaire/négociateur à travers lequel les objets vont pouvoir dialoguer. Il fournit les caractéristiques suivantes:

- La **liaison avec «tous» les langages de programmation**: cependant, actuellement l'OMG a seulement défini officiellement cette liaison pour les langages C, C++, SmallTalk, Ada, COBOL et Java.
- La **transparence des invocations**: les requêtes aux objets semblent toujours être locales, le bus CORBA se chargeant de les acheminer en utilisant le canal de communication le plus approprié.
- L'**invocation statique et dynamique**: ces deux mécanismes complémentaires permettent de soumettre les requêtes aux objets. En statique, les invocations sont contrôlées à la compilation. En dynamique, les invocations doivent être contrôlées à l'exécution.
- Un **système auto-descriptif**: les interfaces des objets sont connues du bus et sont aussi accessibles par les programmes par l'intermédiaire du référentiel des interfaces.
- L'**activation automatique et transparente des objets**: les objets sont en mémoire uniquement s'ils sont utilisés par des applications clientes.
- L'**interopérabilité entre bus**: à partir de la norme CORBA 2.0, un protocole générique de transport des requêtes (GIOP pour *General Inter-ORB Protocol*) a été défini permettant l'interconnexion de bus CORBA provenant de fournisseurs distincts, une de ses instanciations est l'Internet *Inter-ORB Protocol* (IIOP) fonctionnant au-dessus de TCP/IP.

Les composantes

Le bus CORBA fournit les composantes et services suivants (voir figure 7.3):

- **ORB** (*Object Request Broker*) est le noyau de transport des requêtes aux objets. Il intègre au minimum les protocoles GIOP et IIOP. L'interface au bus fournit les primitives de base comme l'initialisation de l'ORB.
- **SII** (*Static Invocation Interface*) est l'interface d'invocations statiques permettant de soumettre des requêtes contrôlées à la compilation des programmes. Cette interface est générée à partir de définitions OMG-IDL.
- **DII** (*Dynamic Invocation Interface*) est l'interface d'invocations dynamiques permettant de construire dynamiquement des requêtes vers n'importe quel objet CORBA sans générer/utiliser une interface SII.
- **IFR** (*Interface Repository*) ou banque d'interfaces est le référentiel des interfaces contenant une représentation des interfaces OMG-IDL accessible par les applications durant l'exécution.

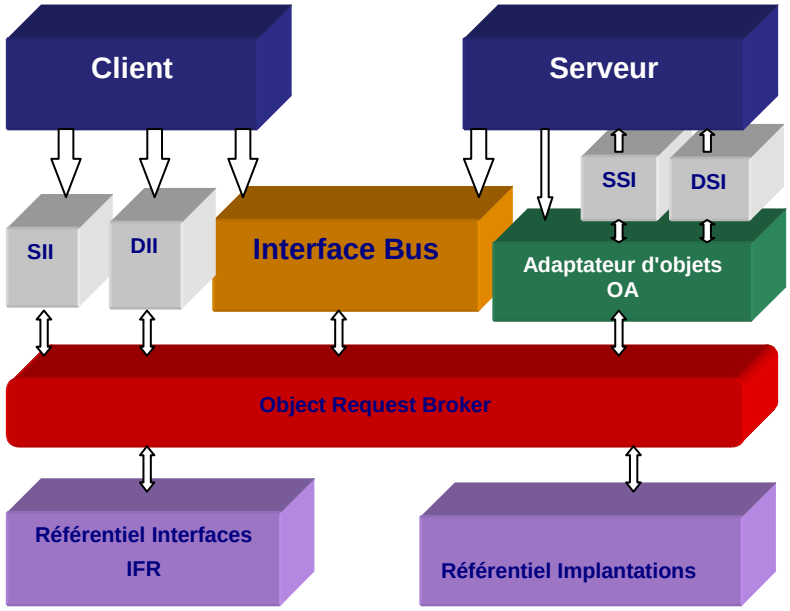


Fig. 7.3: Composantes du bus CORBA

- **SSI** (*Skeleton Static Interface*) est l'interface de squelettes statiques qui permet à l'implémentation des objets de recevoir les requêtes leur étant destinées. Cette interface est générée comme l'interface SII.
- **DSI** (*Dynamic Skeleton Interface*) est l'interface de squelettes dynamiques qui permet d'intercepter dynamiquement toute requête sans générer une interface SSI. C'est le pendant de DII pour un serveur.
- **OA** (*Object Adapter*) est l'adaptateur d'objets qui s'occupe de créer les objets CORBA, de maintenir les associations entre objets CORBA et implémentations et de réaliser l'activation automatique si nécessaire.
- **ImplR** (*Implementation Repository*) est le référentiel des implémentations qui contient l'information nécessaire à l'activation. Ce référentiel est spécifique à chaque produit CORBA.

Ces différentes composantes sont toutes décrites dans le langage OMG-IDL, ce qui les rend accessibles au travers du bus. Cela permet aussi d'homogénéiser les différentes implémentations possibles de la norme car différentes approches peuvent être envisagées pour construire un bus:

- **1 bus = 1 processus**: les objets sont dans le même espace mémoire que les clients. C'est le cas typique des applications embarquées. Ici, le langage OMG-IDL sert à spécifier les objets.
- **1 bus = 1 OS**: les clients et les fournisseurs sont des processus différents sur la même machine. Le bus CORBA peut alors être tout ou partie du système d'exploitation. **1 bus réseau**: les processus sont sur des sites différents et les requêtes sont véhiculées à travers le réseau, c'est le cas sur Internet avec IIOP.
- **Fédération de bus CORBA**: plusieurs bus distincts peuvent être mis ensemble pour former une fédération. Les communications entre ces bus peuvent se faire soit par le protocole commun IIOP soit à travers des passerelles spécifiques ou génériques (DII-DSI).

L'ORB vu du côté client

Du côté du client, l'interface dynamique (DII) permet d'envoyer une requête (voir figure 7.4) vers n'importe quel objet présent sur le réseau à un moment donné, en particulier à des objets pour lesquels le client n'a pas de souche. La spécification DII de la norme CORBA décrit deux modes d'invocation dynamique, un mode synchrone dans lequel le client est bloqué en attendant la réponse du serveur et un mode asynchrone (appelé *deferred synchronous*) où le client n'attend pas la réponse du serveur et peut la demander plus tard. Le serveur quant à lui ne distingue pas les invocations statiques, qui utilisent la souche du côté client et les invocations dynamiques qui ne l'utilisent pas. En effet, dans une invocation statique, le programme client utilise la souche générée à partir du source IDL pour envoyer ses requêtes à l'objet distant, par contre dans une invocation dynamique, le programme client construit dynamiquement une requête sans utiliser la souche. L'ORB ne fait pas de distinction entre les deux mécanismes. L'invocation dynamique est importante dans le sens où elle permet d'isoler les programmes clients de changements dans l'implémentation des objets, changement de programmes ou changement de localisation.

La construction d'une invocation dynamique passe par quatre étapes:

- L'identification de l'objet destinataire de la requête;
- Récupération de l'interface de l'objet destinataire;
- Construction de l'invocation;
- Exécution et réception des résultats et des exceptions éventuelles.

La première étape utilise le service de nommage (Naming service) et le service d'annuaire (*Trader service*) pour identifier les objets. Durant l'exécution, le client étant dynamique découvre tous les services disponibles via l'ORB auquel il est connecté. L'ORB renvoie ensuite grâce à l'opération *get_interface* une référence qui permet d'extraire de la banque d'interfaces l'information nécessaire à la construction de l'invocation dynamique. A partir de cette information, la norme DII spécifie une opération *create_request* pour construire la requête.

La requête est envoyée de manière synchrone avec l'opération *invoke* ou de manière asynchrone avec l'opération *send*, qui rend immédiatement le contrôle au programme client. Une fois fait, le client peut s'enquérir de

l'état de la requête avec l'opération *get_response.* Lorsque celle-ci renvoie un résultat positif, le client peut l'utiliser dans le

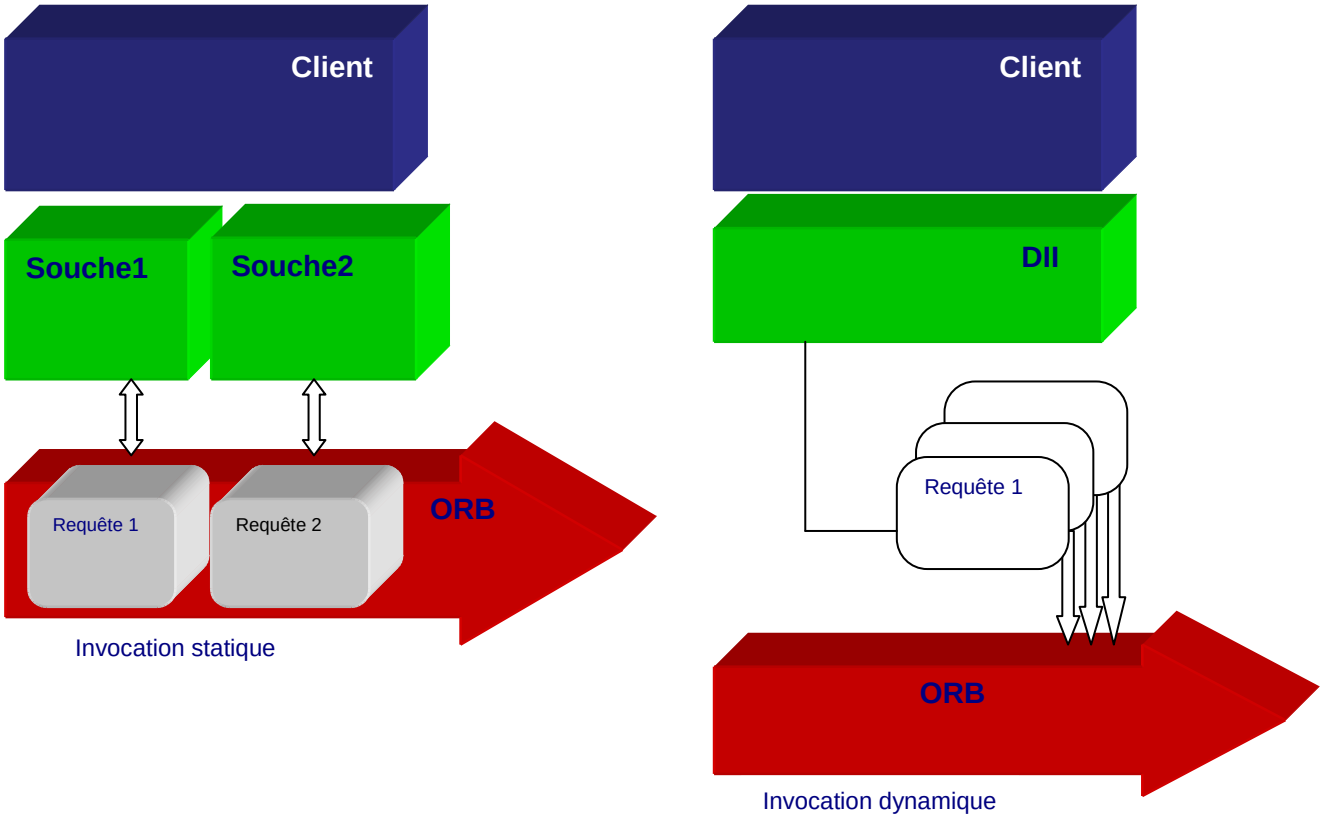


Fig. 7.4: Invocation statique/dynamique côté client

paramètre *result* de l'opération *create_request*, sinon le client est informé d'une exception.

Toute la séquence de ces opérations aboutit donc au même résultat qu'une invocation statique mais permet en plus d'assurer un complète indépendance vis-à-vis des changements d'implantations ou de localisation des serveurs.

Remarque:

Avant de pouvoir fonctionner, l'ORB doit être initialisé à l'aide de l'opération *ORB_init* définie dans le module CORBA de la norme Corba. Cette initialisation peut être faite plusieurs fois et renvoie les mêmes résultats si les mêmes paramètres sont utilisés.

L'ORB vu du côté serveur

Du côté serveur, la situation est plus compliquée qu'elle ne l'est de côté client. L'Adaptateur d'Objet (OA Object Adapter) se charge de donner l'impression au client que l'objet qu'il invoque est toujours actif, et prêt à répondre à ses requêtes. Il définit également l'interface entre le serveur et l'ORB. Cette complexité provient du fait que dans un réseau dynamique, les serveurs peuvent actifs ou inactifs, et chaque serveur peut implémenter un ou plusieurs objets. L'Adaptateur d'Objet permet de préparer les objets à la réception des requêtes et d'informer l'ORB une fois ces requêtes sont exécutées. Il est aussi responsable de la sauvegarde de l'état des objets si ceux-ci sont amenés à être désactivés entre deux requêtes. L'Adaptateur d'Objet assure, plus concrètement, les

fonctions suivantes en s'appuyant sur l'ORB ou sur les Services Objet Communs (*CorbaServices*):

- Génération et interprétation des références aux objets;
- Invocation des méthodes;
- Sécurité des interactions;
- Activation et désactivation des serveurs et des implémentations d'objets;
- Enregistrement des implémentations.

L'Adaptateur d'Objets dépend de la nature du serveur. Plusieurs types d'adaptateurs existent alors, l'un des plus grands et spécifié par la norme Corba est le BOA Adaptateur d'Objets de base (*Basic Object Adaptor*). Il suppose que le serveur est une application séparée de l'application cliente. Le Library Object Adapter, est un Adaptateur d'Objet qui suppose que le serveur est en fait une DLL qui sera chargée dynamiquement dans l'application cliente, et *l'Object-Oriented Database Adaptor*, qui suppose que le serveur est une base de données orientée objet. De même on pourrait imaginer des Adaptateurs d'Objets conçus pour des systèmes d'objets différents, spécifiques à des besoins particuliers.

Du côté du serveur les requêtes passent soit par le squelette associé à un objet donné, soit par le DSI dans le cas dynamique (voir figure 7.5).

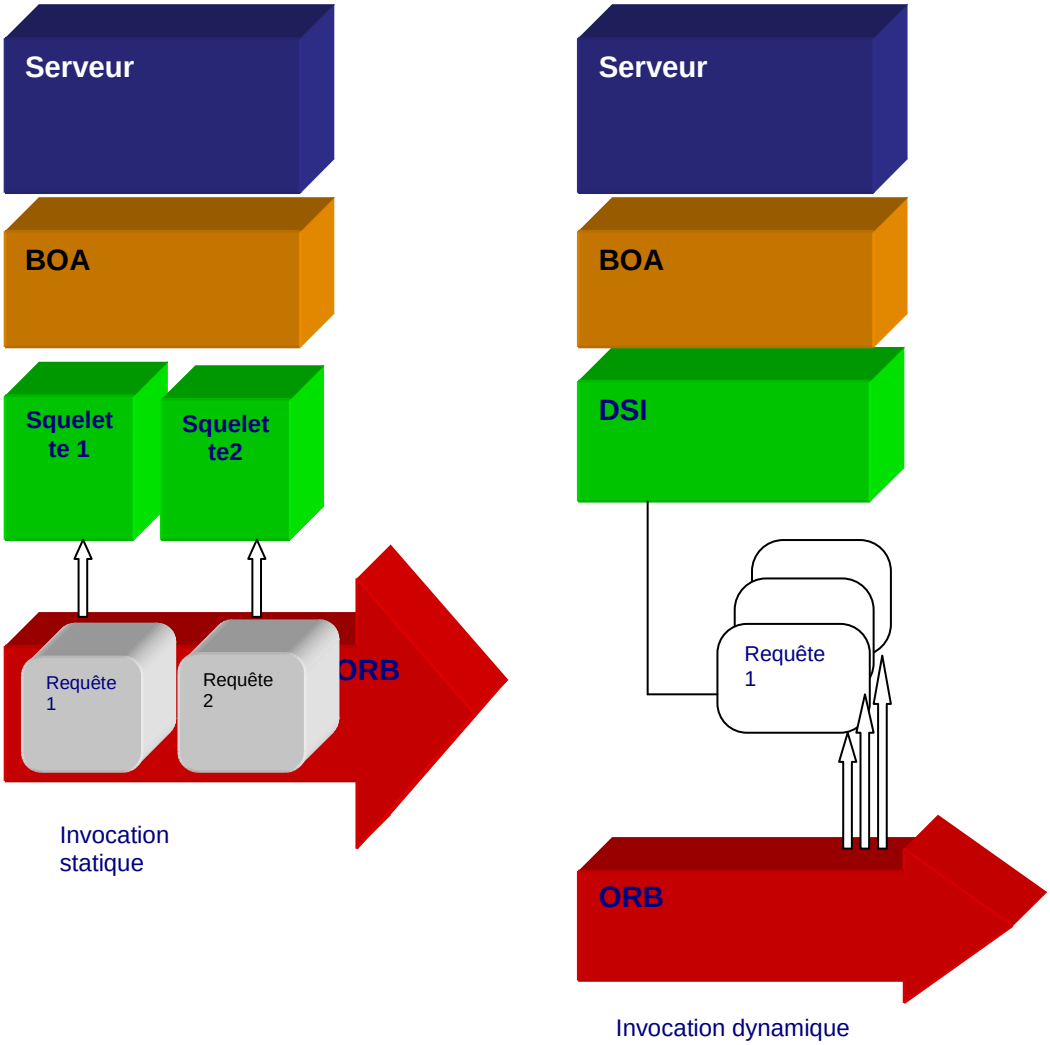


Fig. 7.5: Invocation statique/dynamique côté serveur

Conclusion

Le serveur a la charge de décoder l'objet destinataire. Les objets dans les serveurs sont administrés par le BOA ou par un Adaptateur d'Objets spécifique au serveur correspondant.

Chaque objet du coté serveur est connu par un squelette. De même qu'il existe une interface d'invocation dynamique du côté client, la norme Corba définit également une interface dite de *squelette dynamique* (*skeleton dynamic*) du côté serveur. Le DSI représente en fait l'homologue du DII se trouvant au côté client. Le DSI permet à l'ORB de transmettre des requêtes à des objets pour lesquels le serveur n'a pas de squelette à proprement parler. Ceci est notamment le cas lorsqu'on cherche à intégrer des applications non conformes à CORBA: non orienté objet ou antérieures à la norme, etc. Le DSI est comme le DII, il permet d'offrir une flexibilité et une adaptabilité dans l'évolution et la maintenance des systèmes d'objets distribués. Un des avantages du DSI réside dans son utilisation pour communiquer d'un ORB à un autre. En effet, le DSI permet de construire aisément des passerelles entre ORBs, le serveur du premier ORB est en même temps client du second pour des objets distants en question.

La norme CORBA distingue la couche des services avec le langage de spécification IDL, et la couche transport avec la spécification de l'ORB. Les services sont définis sous forme d'objets en IDL, les compilateurs les transforment en des programmes clients et serveurs dans les langages de programmation choisis par les programmeurs. L'ORB est quant à lui défini de manière très générale, tant du côté client que du côté serveur, dans l'optique d'une intégration facile dans des schémas très variés. Il offre des mécanismes complètement dynamiques (DII et DSI) qui sont particulièrement adaptés à des systèmes d'informations amenés à évoluer rapidement, protégeant ainsi clients et serveurs des changements d'implémentation ou de localisation qui pourraient survenir. Ils permettent également d'interconnecter des ORB différents ou un ORB à une application non orienté objet.

CORBA d'un point de vue gestion de réseaux

Pour conclure ce chapitre, quelques propriétés rendant CORBA très attractif pour la conception de systèmes distribués:

- Courbe d'apprentissage minimale pour un programmeur Java ou C++;
- Véritable orientation objet profonde;
- L'objet est implicitement distant aux yeux du programmeur (sous CMIS/P, l'objet est explicitement distant);
- Adoption très élargie (c'est le bus standard remplaçant le RMI de Java);
- Maturité des produits;
- Adoption par l'ITU-T sous influence de TINA

Remarquons néanmoins que CORBA ne sera jamais un remplaçant 1 pour 1 de CMIS-P, qui offrait de puissants accès (scoping, filtering) et dont la notion de Managed Object ne trouve pas directement d'équivalent sous CORBA.

Chapitre 8 - L'architecture NGOSS du TeleManagement Forum

Le consortium TMF

Le TeleManagement Forum (TM Forum) est un consortium privé à but non lucratif qui fournit une stratégie ainsi que des solutions pratiques pour améliorer la gestion et le fonctionnement des services de communication et d'information. Le TM Forum liste plus de 350 membres comprenant tant des fournisseurs de services, des équipementiers, des sociétés de services ou des intégrateurs, et ce du monde entier.

Parmi les entrants récents, il est à noter une forte participation d'origine chinoise.

L'esprit du TM Forum consiste traditionnellement en l'implication personnelle de consultants privés soutenus par leurs employeurs.

Le cheval de bataille actuel du TM Forum est le *New Generation Operations Systems and Software* (NGOSS). Il s'agit d'un programme intégré innovant, pour le développement et le déploiement de systèmes et de logiciels.

Le programme NGOSS fait partie du programme technique global du TM Forum, qui s'attache à des activités collaboratives facilitant l'implémentation réelle des services de télécommunications.

Le TM Forum contribue au domaine des TIC depuis plus de 15 ans. A citer parmi les résultats les plus connus, le TOM et le TMN++ (vue C++ de l'accès aux objets du TMN).

On trouvera sur le site www.TMForum.org de nombreuses informations et documents de haute qualité.

Le NGOSS

Le NGOSS se propose d'offrir un standard pour le développement et le déploiement de composants OSS/BSS facilement intégrables, gérables et flexibles.

Le NGOSS est constitué d'un ensemble de documents jouant le rôle d'outils de spécifications industrielles, et de conseils qui couvrent des domaines stratégiques. Une méthodologie d'utilisation des outils est fournie.

Le NGOSS repose sur une approche orientée cycle de vie en ce qui concerne le développement de systèmes de gestion; cette approche utilise des définitions claires des processus d'affaire (*business processes*), des spécifications pour automatiser ces processus ainsi que des outils de test des systèmes vis-à-vis de la conformance au NGOSS.

En résumé, le NGOSS est un paradigme architectural proposant une approche intégrant quatre axes:

- Une analyse des processus et flux opératoires de l'opérateur (les « workflows » ou « business processes » dans le jargon TMF); cette étape utilise le eTOM (extended Telecom Operations Map) qui est détaillé plus bas;
- L'analyse et la conception du système, qui est basée sur le SID (Shared Information Data);
- L'analyse et la conception de la solution, qui s'intéresse aux problèmes d'interopérabilité et propose une approche orientée contrats pour les interfaces entre systèmes;
- La validation de la solution vis-à-vis des critères de conformité émis par le TMF.

La figure 8.1 illustre l'approche proposée:

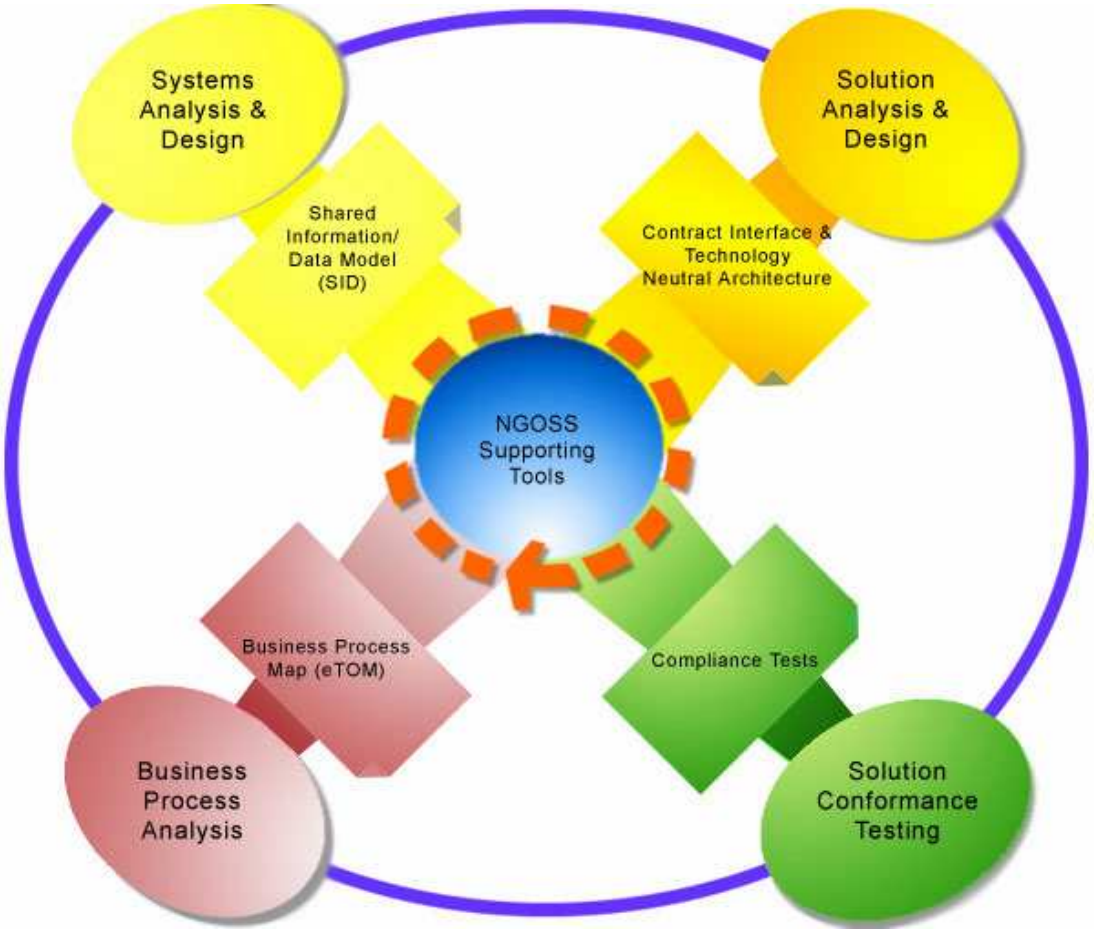


Fig. 8.1: vue d'ensemble du NGOSS

Il est à noter que les solutions basées sur le NGOSS, selon l'esprit des concepteurs, utiliseront des concepts et des technologies conventionnelles disponibles immédiatement sur le marché. Il s'agit donc d'une approche essentiellement pragmatique facilitant l'acceptance. Cet aspect est à comparer avec l'approche de TINA.

Par ailleurs le NGOSS n'est prescriptif que pour les points de référence dits « cardinaux » où l'interopérabilité est obligatoire. Ceci permet à la fois d'accroître la compétitivité (grâce à l'efficacité des composants NGOSS) et de respecter l'utilisation des systèmes légataires.

Les sections suivantes vont détailler les quatre axes du NGOSS: le eTOM, le SID, les contrats et la TNA, la validation et la conformité.

Le eTOM

La figure 8.2 montre les différents composants de la *enhanced Telecom Operations Map* (eTOM).

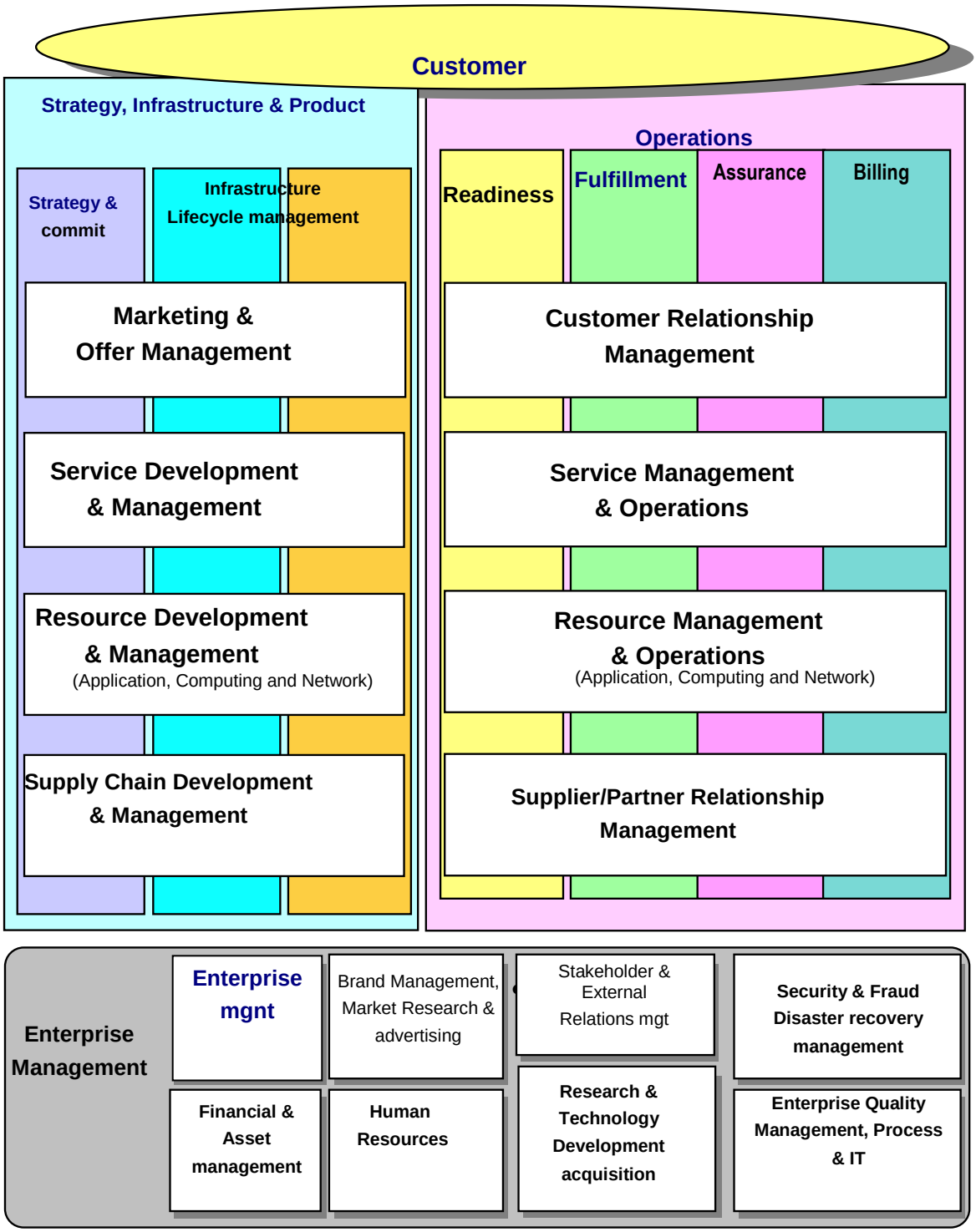


Fig. 8.2: le eTOM

La enhanced Telecom Operations Map est une initiative continue du TM Forum pour fournir un modèle d'affaire et une architecture destinés aux fournisseurs de services et autres acteurs de l'industrie des télécommunications. Il décrit l'ensemble des processus d'entreprise utilisés par un fournisseur de services et les analyse selon différents niveaux de détail selon leur importance et leur priorité. Pour ces entreprises, ces analyses serviront de support au cadre décisionnel par rapport à la gouvernance stratégique; d'autre part, elles fournissent un

point de référence neutre par rapport aux révisions des processus internes, des partenariats, alliances, et autres relations avec les acteurs impliqués. En ce qui concerne les fournisseurs par exemple, le eTOM souligne les frontières potentielles des composants logiciels en fonction des besoins des clients, et dénote les fonctions, les entrées et les sorties nécessaires.

Le eTOM est basé sur le TOM tout en le généralisant pour qu'il devienne un modèle de processus d'affaire complet supportant:

- Un modèle pour l'e-entreprise;
- Un processus systématique de décomposition des processus, des flux et de leurs descriptions.

Ces spécifications de processus, qui comprennent des définitions de haut niveau de l'information requise, représentent un lien fondamental avec le travail systémique du NGOSS.

Le SID (Shared Information Data model)

L'industrie des télécommunications a besoin de technologies avancées pour améliorer le délai de mise sur marché des nouveaux produits et services. Les fournisseurs, de leur côté, savent qu'en proposant des solutions et des composants de gestion intégrés, la future intégration globale en sera simplifiée; avec comme conséquence la diminution du temps nécessaire au déploiement des nouveaux services. Pour obtenir une gestion intégrée de ces différents composants, un vocabulaire commun doit être définit et doit faire l'objet d'un accord global.

Le modèle Shared Information/Data (SID) est conçu pour fournir ce vocabulaire commun. Son but est de fournir un ensemble commun de définitions et de relations concernant l'information et les données utilisés dans les architectures NGOSS. Le SID développe en fait trois points de vue de conceptualisation de l'information:

- Une vue orientée affaire des Entités Affaire (business entities) qui supportent les processus eTOM;
- Une vue système reposant sur la notation UML qui permet de réaliser une solution répondant au besoin d'affaire;
- Une vue orientée implémentation, réalisée en formalisme spécifique de la plateforme de développement.

La version actuelle (Phase III) propose des objets ABE (Aggregate Business Entities) pour les domaines d'affaire Client, Produit, Ressource et ABEs Communs.

La version en cours de développement (Phase IV) introduira de plus une représentation pour la vue système et raffinera les aspects relationnels entre ABEs existants.

Le document GB922 avec ses 14 addendums utilise une approche narrative progressive pour expliquer chacun des Domaines ainsi que les ABEs associés. Les addendums décrivent les Entités d'Affaires à l'aide de l'outil UML ainsi que d'un dictionnaire de données facilitant l'usage des processus d'affaire du eTOM.

Le SID Phase III version 3.5 comprend des définitions d'ABEs et l'UML associé, pour le Contrat (y compris le SLA), l'Interaction d'Affaire, les

Types de Base, le Projet, l'Organisation, l'Equipe, le Calendrier, le Client, le Produit, la Qualité de Service, le Résumé du Service, la Ressource Physique, la Ressource Logique, la Stratégie. De plus, le document GB922 inclus une architecture SID révisée y compris des spécifications d'ABEs jusqu'au niveau 3 et le mapping entre les ABEs et les processus eTOM de niveau 2.

Rôle au sein du NGOSS

Le SID doit être considéré comme un modèle complémentaire du eTOM en ce sens qu'il fournit, d'un point de vue des entités d'affaires, les modèles de référence de l'information et des données ainsi qu'un catalogue commun de vocabulaire supportant les modèles d'affaires du eTOM. En conséquence le SID et le eTOM, utilisés conjointement, offrent un outil efficace pour définir comment les entités du système s'organisent pour répondre à un besoin d'affaire.

Applications

Les éléments du SID peuvent être considérés de plusieurs façons. Un analyste des processus d'affaire pourrait par exemple être intéressé par un point de vue affaire des éléments de données. D'un point de vue système, un concepteur de contrat pourrait s'intéresser aux opérations agissant sur une entité d'affaire.

Conclusion

Les documents du SID consistent en un document principal décrivant le modèle SID, la relation avec d'autres entités du TM Forum, ainsi que des extraits du contenu du modèle. Les nombreux addendums fournissent des définitions d'ABEs et des modèles pour les ABEs communs.

L'interface contractuelle et la TNA

La Technology Neutral Architecture (TNA) du NGOSS est définie par la suite de documents TMF053.

L'une des approches proposées est la création de modèles technologiquement neutres en avance de phase sur le déploiement, et l'uilisation de plateformes spécifiques. Ceci implique le développement de vues d'affaire, de vues système et de vues informationnelles en collaboration avec les activités liées au eTOM, au SID et aux projets Catalyst[3].

D'autre part le TNA spécifie également un certain nombre de services génériques de plateforme (n.b. à comparer avec les services COSS de CORBA). Ceux-ci sont spécifiés à l'aide du SID, résultant en des modèles formels UML.

[3] les projets Catalyst sont des projets industriels concrets basés sur le NGOSS, partenaires du TMForum.

L'interface contractuelle

Le contrat NGOSS est l'unité fondamentale d'interopérabilité au sein d'un système NGOSS. L'interopérabilité est une notion importante concernant les quatre vues définies sur la méthodologie Cycle de Vie du NGOSS. Par exemple, le contrat est utilisé pour définir la spécification d'un service, mais aussi de l'information et le code qui l'implémentent. Le contrat est aussi utilisé pour monitorer le service et ainsi assurer que les obligations externes du contrat (p. ex. concernant un SLA) sont respectées; par ailleurs il définit quelles mesures sont à prendre en cas de violation.

Le contrat définit également la façon d'utiliser le service (c.a.d. son invocation). Ceci représente davantage qu'une spécification d'interface logicielle; on y trouve également la définition de pré- et postconditions, la sémantique associée à l'invocation, les stratégies affectant la configuration et l'utilisation du service, et bien d'autres aspects.

En résumé le contrat est une façon de réifier la spécification d'un service ainsi que d'assurer l'implémentation du service (y compris des obligations envers d'autres entités du système).

La Technological Neutral Architecture

Le document TMF053 décrit les principaux éléments de la TNA:

- Les contrats,
- La transparence de distribution,
- Les services de gestion des processus,
- Les services de gestion de la SID,
- Les services de gestion des stratégies,
- Les services d'adaptation et de médiation.

La validation et la conformité

Le programme de conformité

Le programme de conformité a pour but de fournir à l'intention de l'industrie des télécommunications un ensemble de critères de test ansi que de méthodes de test adaptées. Ce travail cherche à permettre d'identifier des produits ou solutions OSS qui sont partiellement ou pleinement conformes aux principes du NGOSS.

Le programme de test

Le programme de test est une activité continue phasée par les « world events » du TMForum. Le programme de test finalise actuellement la couverture du eTOM.

L'ensemble des tests sont documentés par la suite de documents TMF050.

Prochaines étapes

A long terme le programme de test aboutira à une suite de tests couvrant l'ensemble du NGOSS. Le but est d'accréditer les produits et solutions compatibles par un label NGOSS Powered™.

L'outil de test en ligne

Un service web permet de tester la compatibilité des produits et applications avec les standards du NGOSS. Basé sur le moteur Jrules d'ILOG, cet outil permet d'exécuter les tests de compatibilité NGOSS tels que définis par les documents TMF050 et TMF050a, ainsi que de recevoir des éléments informatifs permettant de procéder à la finition des applications.

En résumé

Le TMForum propose une architecture nouvelle, intégratrice, originale par de nombreux aspects. Il est à souligner l'approche pragmatique désireuse de susciter l'acceptance des concepts par les industriels. D'autre part l'approche proposée est radicalement basée sur la notion de processus d'affaires (les *business processes*) desquels sont déduits l'ensemble des composants des systèmes. Cette approche est très innovante en ce sens qu'elle place, comme on le voit sur la figure 8.2, l'utilisateur du système à la source de toute activité du système de gestion.

Enfin, il y a un rapport subtil entre l'architecture TINA et le NGOSS. Le NGOSS entre en quelque sorte de plain pied par la porte qu'avait entr'ouverte TINA et son architecture de services. On peut considérer que le TMForum reprend de nombreuses idées de TINA (et de CORBA), mais en les adaptant au monde industriel et à ses contraintes, et en les rendant ainsi bien plus séduisantes.

Conclusion

La gestion de réseaux est l'un des domaines les plus complexes auxquels l'on puisse se confronter; elle cumule la distribution, le modèle objet, le temps réel, le transactionnel, la gestion d'équipements complexes. C'est en conséquence une source de coût importante pour l'opérateur, qui se voit contraint d'investir des sommes significatives et des compétences critiques dans une fonction qui semble non immédiatement rentable. Néanmoins, le souhait exprimé ici concerne la compréhension de la nécessité de la fonction Gestion de Réseaux, et son intérêt pour l'entreprise qu'est l'opérateur; ce n'est qu'à travers de la gestion efficace que ce dernier sera en mesure d'offrir les services qui le différencieront de la concurrence, tels que des Réseaux Privés Virtuels, de la connectivité sur demande etc.

Par ailleurs, les auteurs de ce document souhaitent exprimer leur souci de situer cet exposé au fil du temps, les technologies évoluant au fur et à mesure de leur cycle de vie et se succédant les unes les autres. Nous ne sommes pas dans un monde de certitudes dans ce domaine, mais au contraire dans une sphère d'interrogations et de remises en cause permanentes, ce qui de l'avis des auteurs, doit contribuer à une meilleure réflexion et une plus profonde compréhension du fond de la problématique envisagée.