UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

Série Q

Supplément 29
(12/1999)

SÉRIE Q: COMMUTATION ET SIGNALISATION

Modélisation des services: évolution vers l'utilisation de techniques orientées objet

Recommandations UIT-T de la série Q – Supplément 29

(Antérieurement Recommandations du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE Q

COMMUTATION ET SIGNALISATION

SIGNALISATION DANS LE SERVICE MANUEL INTERNATIONAL	Q.1–Q.3
EXPLOITATION INTERNATIONALE AUTOMATIQUE ET SEMI-AUTOMATIQUE	Q.4–Q.59
FONCTIONS ET FLUX D'INFORMATION DES SERVICES DU RNIS	Q.60–Q.99
CLAUSES APPLICABLES AUX SYSTÈMES NORMALISÉS DE L'UIT-T	Q.100–Q.119
SPÉCIFICATIONS DES SYSTÈMES DE SIGNALISATION N° 4 ET N° 5	Q.120–Q.249
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION N° 6	Q.250–Q.309
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION R1	Q.310–Q.399
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION R2	Q.400–Q.499
COMMUTATEURS NUMÉRIQUES	Q.500–Q.599
INTERFONCTIONNEMENT DES SYSTÈMES DE SIGNALISATION	Q.600–Q.699
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION N° 7	Q.700–Q.849
SYSTÈME DE SIGNALISATION D'ABONNÉ NUMÉRIQUE N° 1	Q.850–Q.999
RÉSEAUX MOBILES TERRESTRES PUBLICS	Q.1000–Q.1099
INTERFONCTIONNEMENT AVEC LES SYSTÈMES MOBILES À SATELLITES	Q.1100–Q.1199
RÉSEAU INTELLIGENT	Q.1200–Q.1699
PRESCRIPTIONS ET PROTOCOLES DE SIGNALISATION POUR LES IMT-2000	Q.1700–Q.1799
RNIS À LARGE BANDE	Q.2000–Q.2999

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

Supplément 29 aux Recommandations UIT-T de la série Q

Modélisation des services: évolution vers l'utilisation de techniques orientées objet

Résumé

Dans le présent supplément sont comparés différents types de méthodologies de modélisation des services pour déterminer leur applicabilité à l'élaboration de protocoles pour le réseau intelligent (RI) dans le cadre de l'ensemble CS-4 du RI. L'évolution des techniques SIB est étudiée et différentes technologies sont envisagées telles les interfaces API.

Ce supplément vient compléter les informations contenues dans la Recommandation Q.65.

Justification

Le présent supplément rassemble des éléments de discussion sur les aspects modélisation des services relatifs à l'ensemble CS-4 du RI.

Source

Le Supplément 29 aux Recommandations UIT-T de la série Q, élaboré par la Commission d'études 11 (1997-2000) de l'UIT-T, a été approuvé le 3 décembre 1999 selon la procédure définie dans la Résolution 5 de la CMNT.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

La Conférence mondiale de normalisation des télécommunications (CMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de la CMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente publication, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente publication puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des publications.

A la date d'approbation de la présente publication, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente publication. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Supplément 29 aux Recommandations UIT-T de la série Q

Modélisation des services: évolution vers l'utilisation de techniques orientées objet

1 Domaine d'application de la modélisation des services pour l'ensemble CS-4 du RI

Le présent supplément répond aux objectifs suivants:

- identifier et comparer différentes méthodologies de modélisation des services;
- déterminer l'applicabilité des méthodologies de modélisation des services ainsi identifiées, en particulier:
  - la gestion des services: modélisation des données;
  - l'objet de la logique de service: modèle dynamique de service (comportement de la logique de service);
  - la description de la relation entre le modèle de données et le modèle dynamique de service;
- déterminer l'applicabilité des méthodologies d'élaboration de protocoles ainsi identifiées, en particulier, en ce qui concerne:
  - leur compatibilité avec un affinage progressif depuis des capacités de service et de réseau jusqu'au niveau protocole à la fois pour le modèle de données et le modèle dynamique de service;
  - leur capacité à intégrer les protocoles existants dans le modèle défini avec la méthodologie utilisée;
- étudier les aspects évolution de la modélisation des services, en particulier en ce qui concerne:
  - le passage de la méthodologie de modélisation des services par modules SIB aux méthodologies utilisées pour l'ensemble CS-4 du RI;
- déterminer le champ d'application de la modélisation des services pour l'ensemble CS-4 du RI;
- choisir la méthode à appliquer à la modélisation des services dans l'ensemble CS-4 du RI.

1.1 Références

Les Recommandations UIT-T et autres références suivantes contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour le présent supplément. Au moment de la publication, les éditions indiquées étaient en vigueur. Toute Recommandation ou autre référence est sujette à révision; tous les utilisateurs du présent supplément sont donc invités à rechercher la possibilité d'appliquer les éditions les plus récentes des Recommandations et autres références indiquées ci-après. Une liste des Recommandations et des suppléments UIT-T en vigueur est publiée régulièrement.

[1] Recommandations UIT-T de la série Q.122x, *Recommandations relatives à l'ensemble CS-2 du RI*.

[2] Recommandation UIT-T Q.1223 (1997), *Plan fonctionnel global de l'ensemble de capacités 2 du réseau intelligent*.

[3] Recommandation UIT-T Z.100 (1999), *SDL: langage de description et de spécification*.

2 Définitions et abréviations

2.1 Abréviations

Le présent supplément utilise les abréviations suivantes:

API interface de programmation d'application (*application programming interface*)

GFP plan fonctionnel global (*global functional plane*)

INCM modèle conceptuel du réseau intelligent (*IN conceptual model*)

ODP traitement réparti ouvert (*open distributed processing*)

OMT technique de modélisation d'objet (*object modelling technique*)

SDL langage de description et de spécification (*specification description language*)

SQL langage d'interrogation structuré (*structured query language*)

STD diagrammes de transition d'Etat (*state transition diagrams*)

TINA architecture de réseaux d'information sur les télécommunications (*telecommunication information networking architecture*)

2.2 Définitions

Le présent supplément définit les termes suivants:

2.2.1 **API**: une interface API est essentiellement un ensemble d'opérations (*ou de méthodes*) qui peuvent être invoquées sur une composante, chacune de ces opérations obligeant la composante à montrer ses fonctionnalités comportementales. Chaque opération est spécifiée syntaxiquement sous forme d'un identificateur identifiant l'opération invoquée et les paramètres qui affectent d'une certaine manière le comportement de la composante.

2.2.2 **appel API**: un appel API est équivalent au terme "opération" avec lequel il est parfois utilisé de manière interchangeable dans la description d'une interface API (voir ci-dessus). En réalité, une interface API est composée d'un ensemble d'appels API individuels.

3 Prescriptions applicables à la modélisation des services CS-4 du RI

Le présent paragraphe porte sur les prescriptions qui ont été estimées importantes pour les techniques de modélisation et les méthodologies de modélisation des services du RI, à savoir:

- un **affinage progressif** du service (caractéristiques) jusqu'au niveau du protocole doit pouvoir être réalisé;
- la prise en charge de différents mécanismes de **transparence**, à savoir:
  - la transparence *technologique*: la modélisation des services du RI ne doit pas être dépendante de la technologie utilisée, c'est-à-dire du type de réseau, du système d'exploitation ou du langage de programmation. Un type important de transparence dans le contexte de la modélisation des services CS-4 du RI est la transparence relativement à la technologie du réseau. L'objectif de l'ensemble CS-4 du RI est l'intégration d'un certain nombre de technologies de réseau, y compris les réseaux orientés connexion (par exemple, le RTPC, les IMT-2000) et les réseaux de type sans connexion (par exemple, les réseaux IP, les réseaux de données). Certains des services ciblés dans l'ensemble CS-4 du RI seront probablement associés à une technologie spécifique de réseau, d'autres nécessiteront un interfonctionnement entre différentes technologies de réseau, ou leur comportement dépendra de la technologie mise en œuvre à l'endroit où le service est invoqué. On peut penser à un service avec interaction d'utilisateur, lequel peut être invoqué par des utilisateurs du RNIS-LB ou des utilisateurs des réseaux mobiles, ou à

un service de conférence multimédia fonctionnant avec plusieurs technologies de réseau. Le RI peut présenter une architecture intégrée pour les services sur plusieurs technologies de réseau. Il convient d'étudier jusqu'à quel niveau il est possible d'avoir une transparence technologique réalisée.

- Transparence d'*accès*: masquage des différences de représentation des données et des différences de mécanisme d'invocation (c'est-à-dire assurant différents mappages du contenu informationnel des échanges de protocole RI avec les interfaces API programmables).
- Transparence aux *défaillances*: masquage, depuis un objet, des défaillances affectant des objets (y compris lui-même) et leur éventuel rétablissement.
- Transparence à la *migration*: masquage, depuis un objet, de la capacité d'un système à modifier l'emplacement des interfaces avec cet objet.
- Transparence à la *relocalisation*: masquage de la relocalisation d'une interface d'objet par rapport aux autres interfaces qui lui sont liées.
- Transparence à la *duplication*: masquage de l'utilisation d'un groupe d'objets mutuellement compatibles du point de vue du comportement afin de prendre en charge une interface.

- **Extensibilité des services**: il doit être possible de prendre en charge des adjonctions aux fonctionnalités de service RI en procédant à une extension des modèles et des composantes de services existants.
- **Echelonnabilité du service**: il doit être possible de prendre en charge et de permettre un échelonnement des capacités du service en termes de nombre d'utilisateurs, de nombre de nœuds et de nombre de domaines administratifs, etc.
- **Traitement de la version de service**: les techniques de modélisation des services doivent faciliter l'utilisation en parallèle de plusieurs versions de modèles et de composantes de services.
- **Possibilités de réutilisation des services**: il doit être possible de réutiliser des modèles d'un service RI (capacité) pour la spécification d'un autre service RI, et non pas de commencer la modélisation à zéro dans le cas d'une fonctionnalité chevauchante. Cela s'applique aux composantes logicielles qui mettent en œuvre la spécification.
- **Exploitation, administration et maintenance**: les méthodologies et les techniques de modélisation doivent présenter une certaine souplesse lorsqu'il s'agit de la modification de fonctionnalités et de modèles de données pendant les phases d'exploitation, d'administration et de maintenance. Par exemple, des modifications de fonctionnalités qui se produisent pendant la maintenance doivent être facilement reflétées dans le modèle de service RI.
- **Réduction des conflits de service**: les méthodologies et les techniques de modélisation doivent réduire les risques d'interaction entre services.
- **Modification/personnalisation souple des services**: les fournisseurs de réseaux et les fournisseurs de services doivent pouvoir différencier et personnaliser les services afin de répondre à des besoins spécifiques du marché.

  Les utilisateurs doivent pouvoir personnaliser les services dans une certaine mesure afin de satisfaire leurs besoins personnels. Dans le court terme, cette souplesse sera limitée à une personnalisation des données et à une sélection à partir d'une liste de caractéristiques prédéfinies. A long terme, l'empaquetage des services pourrait être également assuré.
- **Prise en charge d'un environnement à plusieurs intervenants**: la modélisation des capacités de service CS-4 du RI doit être en mesure de tenir compte des rôles respectifs des différents intervenants dans la fourniture des services (détaillant, opérateur, fournisseur de contenu, etc.).

3.1 Modélisation des services

L'un des objectifs de la modélisation est de faire en sorte que les développeurs et concepteurs des services disposent du même modèle de logique de service. Pour cela, la technique de modélisation doit cacher tous les détails de la plate-forme pour l'utilisateur mais, dans le même temps, permettre aux développeurs des services de traduire le modèle comportemental sous une forme exécutable adaptée. Afin de satisfaire à cette exigence, il est proposé de représenter la technique de modélisation comportementale avec une notation indépendante du langage de programmation, du système d'exploitation et de la base de données. Il est donc proposé d'utiliser le langage SDL sans données et des diagrammes de transition d'état pour représenter le pur comportement d'une classe abstraite.

3.2 Logique de service couvrant une seule classe

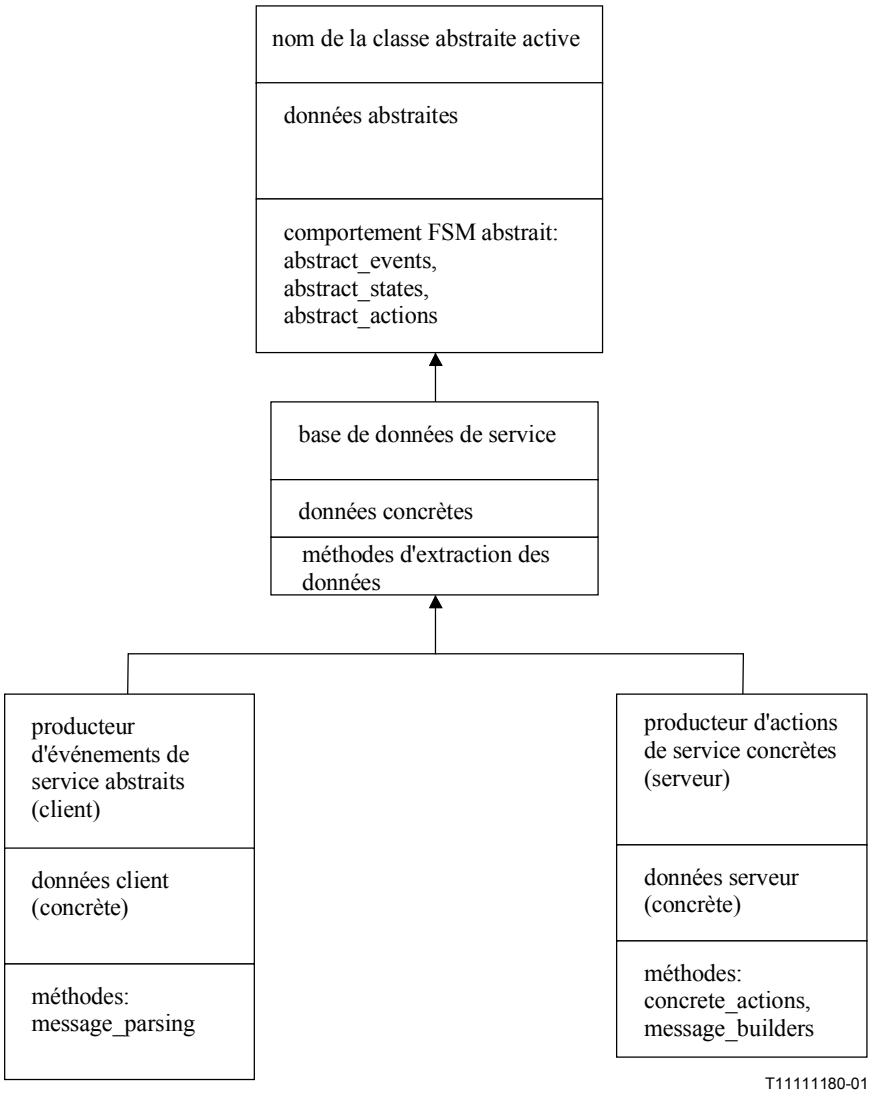
Les techniques de modélisation OMT et autres introduisent le concept de classe active. Une classe active est définie comme étant une classe dont le comportement est le mieux modélisé en utilisant un modèle de machine à états finis (FSM, *finite state machine model*). Afin de conférer une indépendance par rapport au langage au niveau de la logique de service, il est proposé que le modèle de machine à états finis utilisée ne contienne aucune donnée. Ainsi, une classe abstraite représentant un simple service contient un modèle de données abstraites et un modèle comportemental abstrait strictement séparés.

Il est important de souligner ici le fait que la séparation entre modèle de données abstraites et modèle comportemental abstrait n'est proposée qu'au niveau de la modélisation. Aucune recommandation n'est formulée ici ou proposée selon laquelle la séparation entre modèle de données abstraites et modèle comportemental abstrait doit être étendue au code généré automatiquement ou manuellement.

Un exemple de séparation entre modèle de données abstraites et modèle de comportement abstrait pourrait être celui d'un service VPN avec un modèle de service complexe. Ainsi, le paramètre de service VPN Participant_ID est un exemple de donnée abstraite au niveau modélisation. Un modèle de classe abstraite dans sa section données abstraites ne doit pas refléter l'endroit dans le modèle comportemental où cette donnée est utilisée. Par ailleurs, le modèle comportemental abstrait d'une classe abstraite peut nécessiter l'apparition d'un événement New_Participant_Arrived, sans qu'il soit nécessaire de mentionner le lieu où une donnée concernant un participant particulier est stockée, dans la section données abstraites de la classe abstraite.

Cette séparation à l'intérieur de la classe supérieure doit permettre l'interchangeabilité des objets modèle de service et la création de besoins de service standard (voir la Figure 1). Les flèches dans ce modèle représentent l'héritage de façon compatible avec la notation OMG. Dans le cas d'une mise en œuvre en C++, les actions abstract_actions et les événements abstract_events sont souvent mis en œuvre comme des fonctions purement virtuelles.

La Figure 2 montre un exemple de classe abstraite active appelée objet de service VPN, qui peut être utilisée pour la mise en œuvre du service de réseau virtuel privé. Cette classe contient des exemples d'événements abstract_events, d'actions abstract_actions et d'états abstract_states.



NOTE – Il est possible de mettre en œuvre le message_parser et le message_builder sous forme de classes séparées pour des services plus complexes.

Figure 1 – Classe abstraite représentant: a) une logique de service simple; b) une partie de logique de service complexe; c) un gestionnaire de logique de service

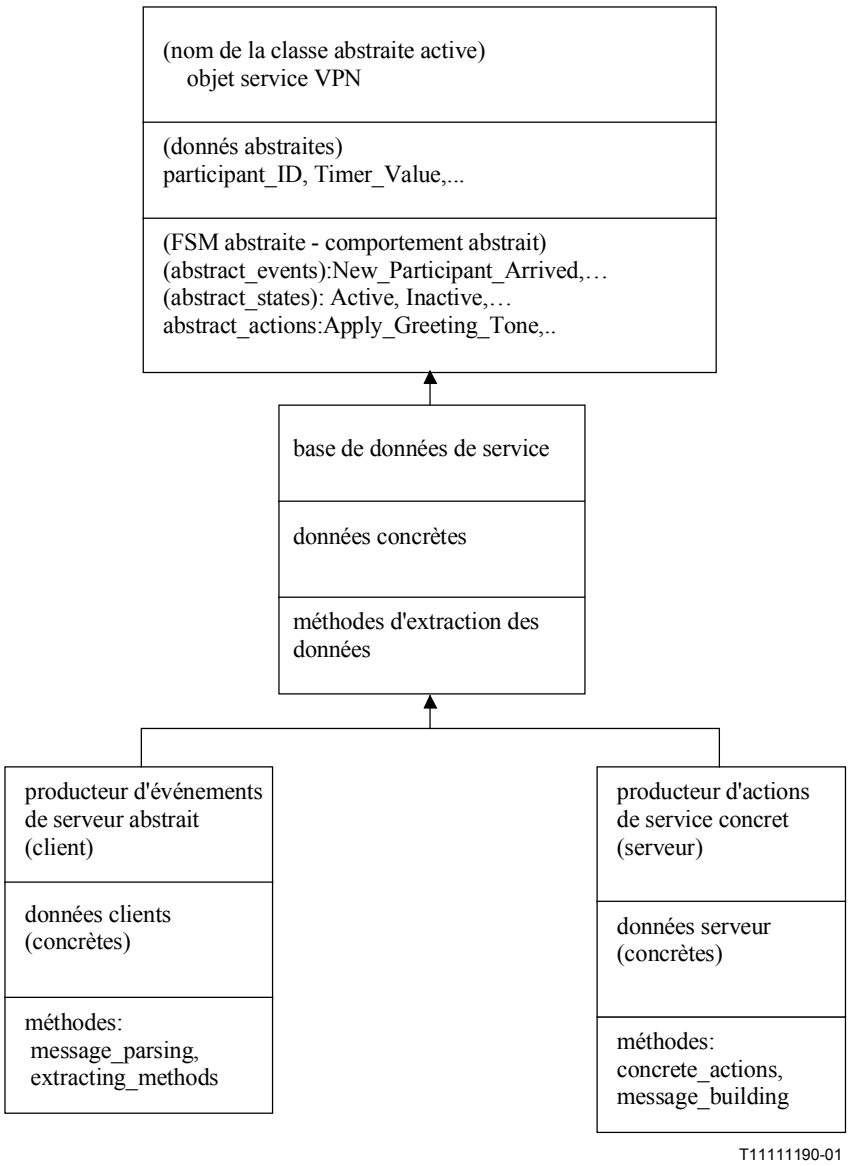


Figure 2 – Exemple de classe abstraite représentant une logique de service VPN

3.2.1 Logique de service s'étendant sur plusieurs classes

Pour des services plus complexes, la logique de service peut être modélisée par une série de classes actives coopérantes. Un objet est défini ici comme une classe instanciée. Il peut être utile d'étudier si le modèle de classe coopérante convient mieux à la modélisation des services RI que le modèle de classe confinée.

D'après certaines indications, le modèle de classe coopérante assure mieux l'encapsulation des classes en cachant les détails d'implémentation. C'est sur ce modèle de classe coopérante que se fonde le langage de programmation objet Smalltalk. Dans ce cas, chacune des classes actives doit être modélisée de la même façon que la logique de service couvrant une seule classe. Les classes coopérantes envoient des demandes entre elles et reçoivent des réponses. Cela représente un flux d'informations de commande entre les objets.

Il convient de noter que le terme message tel qu'il est utilisé dans le présent supplément implique un flux physique d'informations et non pas un flux logique d'informations. Un message peut contenir une donnée ainsi qu'une information de contrôle. Aussi, est-il recommandé de représenter le flux de l'information de commande entre objets séparément du flux de données entre les objets. Un avantage

connu d'une telle séparation est la simplification des outils de validation automatique du service. L'outil de validation de l'environnement de création de service (SCE, *service creation environment*) doit générer une séquence de diagrammes de flux de communication entre les objets à partir des modèles de comportement de la machine FSM associée aux objets coopérants. D'autres contributions sur les prescriptions concernant les SCE et le SMS pourraient être fournies ultérieurement. Le modèle présenté s'applique à la communication entre objets de type homologue ainsi qu'au cas où une fonction de gestionnaire de la logique de service est assignée à l'une des classes. Cela correspond au choix du concepteur du service qui a utilisé une forme hiérarchique de modélisation de service et non pas une forme décentralisée.

4 Méthodologies et techniques de modélisation associées

4.1 Traitement réparti ouvert (ODP)

Le modèle de référence du traitement ODP est une méthodologie générique répartie orientée objet et adaptée à la fois aux applications traditionnelles de télécommunication (tel le RI) et aux applications de traitement de l'information. Il se fonde sur deux puissantes tendances en matière de technologie logicielle:

- les techniques de spécification orientées objet utilisées à différents niveaux d'abstraction permettant un degré élevé d'affinage progressif et de contrôle d'homogénéité;
- des environnements de traitement décentralisés de type objet (plate-forme CORBA par exemple) permettant la fourniture de services répartis, et ce qui est le plus important, permettant une transparence et un interfonctionnement de répartition dans des systèmes répartis.

Le modèle de référence du traitement ODP prescrit que plusieurs descriptions, avec différents niveaux d'abstraction, doivent être fournies pour tout service au stade de l'étude ou de développement. Cinq niveaux d'abstraction – appelés points de vue dans la terminologie ODP – ont été définis dans l'ODP et sont considérés comme englobant différents domaines de préoccupation qu'il faut couvrir dans le processus de développement des services.

Les spécifications doivent être fournies pour chacun de ces points de vue en utilisant un modèle de point de vue correspondant ou un langage de point de vue adéquat. Les cinq points de vue ODP sont les points de vue *entreprise, information, traitement, ingénierie et technologie*.

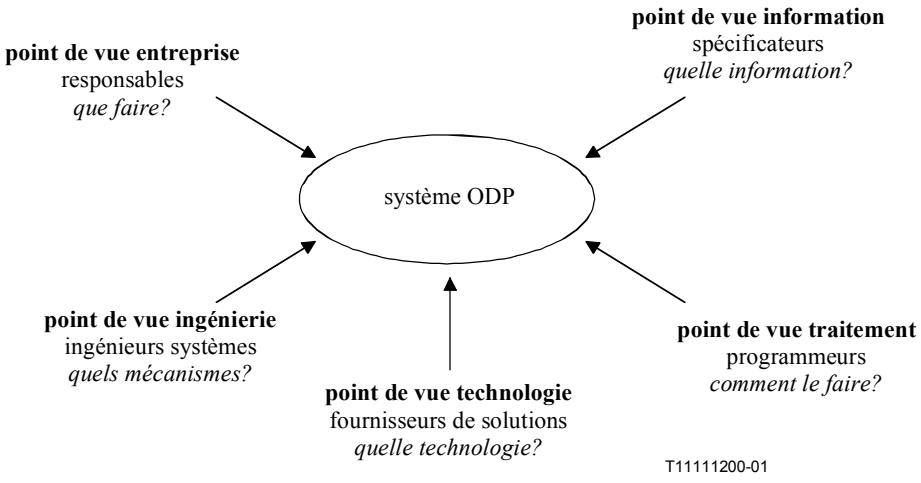


Figure 3 – Différents niveaux d'abstraction (points de vue) dans l'ODP

4.1.1 Point de vue entreprise

Les spécifications entreprise d'un service RI décrivent le service du point de vue des organisations et des personnes qui utiliseront ou exploiteront ce service. Ce concept utilisé pour les spécifications entreprise fait intervenir:

- les capacités de service recherchées sont définies par les spécifications au niveau stratégique (justification);
- les intervenants (et autres institutions participantes);
- les rôles (qui sera impliqué et quel rôle aura-t-il);
- l'environnement réglementaire (obligations des intervenants);
- l'ensemble de règles et de règlements qui régissent l'exploitation du service (règles à respecter pour l'exploitation du service).

En général, une spécification entreprise est écrite avec un format textuel ou en utilisant une notation graphique de haut niveau.

4.1.2 Point de vue information

Une spécification d'information définit un modèle très abstrait d'entités du monde réel et de leurs relations ainsi que les contraintes qui régissent leur comportement. Une spécification d'information englobe en général:

- la spécification de la structure informationnelle du service (objets);
- la spécification des interrelations entre objets informationnels (associations);
- la spécification des opérations et des contraintes qui régissent l'évolution dynamique possible du service considéré comme une collection d'objets.

Le point de vue information peut être développé au moyen, par exemple, du langage UML, en particulier en utilisant les capacités pour la modélisation des données (modèle statique).

Directives applicables au mappage du point de vue entreprise avec le point de vue information:

- les règles de mappage peuvent être établies pour traduire la spécification entreprise en termes d'information. Il convient de noter qu'il ne s'agit là que de directives.

4.1.3 Point de vue traitement

Une spécification de traitement représente une mise en œuvre abstraite du service considéré en termes d'objets en interaction, dans laquelle les lieux, les accès, la distribution et les défaillances sont transparents. Plus simplement, à ce niveau d'abstraction, un service est représenté comme une configuration dynamique d'objets d'interaction.

Une caractéristique importante de cette spécification de traitement est sa capacité à en saisir les aspects temps réel et les aspects probabilistiques. Les objets liants, par exemple, décrivent le comportement de communication, qui respecte certaines contraintes de qualité de service. Ces exigences non fonctionnelles sont particulièrement pertinentes lorsqu'il s'agit de la prise en charge des interactions multimédias dans un environnement réparti.

La spécification du point de vue traitement contiendra en général des entités fonctionnelles bien connues (SCF, SFF, SRF, etc.) et des objets de traitement, décomposés ensuite selon les besoins de décentralisation. Les interactions entre les objets de traitement s'exprimèrent en termes "d'opérations" (opérations INAP) et de "flux" (voix, vidéo et données).

Le point de vue traitement pourrait être développé en utilisant par exemple le langage UML, pour la spécification des objets calcul et, par exemple, le langage IDL pour la spécification des interfaces entre objets.

Bien que l'ODP ne recommande pas de solution de mappage entre les types d'objets informationnels et traitement, la simple propriété suivante est un point de départ: il n'y a pas de correspondance biunivoque. Une spécification de type d'objet traitement est obtenue en prenant une spécification de type d'objet information et en ajoutant la description des opérations pour lesquelles il a un rôle de serveur. Toutefois, certaines considérations concernant les possibilités de répartition modifieront ce mappage.

4.1.4 Point de vue ingénierie

Une spécification d'ingénierie détermine la manière dont les descriptions de traitement sans distribution peuvent être réalisées en termes de composantes génériques de système et de protocoles de communication génériques (tel le SS7). Elle met l'accent sur la façon dont les interactions entre les objets sont obtenues et sur les ressources qui sont nécessaires pour cela. Du point de vue ingénierie, un système ODP est considéré comme un ensemble de systèmes de traitement. Les détails concernant les réseaux de communication sous-jacents, les systèmes d'exploitation et le logiciel sont masqués par un environnement réparti uniforme et de base.

En terminologie ODP, les machines SSF-FSM et les modèles BCSM pourraient être considérés comme des *talons* ou des *objets de protocole*. Les signatures de ces opérations correspondraient aux définitions ASN.1 des opérations INAP.

La spécification d'ingénierie serait très semblable à la spécification de protocole.

Elle pourrait être élaborée en utilisant le SDL et l'ASN.1. Le SDL est utilisé pour décrire les objets et leur comportement et l'ASN.1 les signatures des opérations visibles aux interfaces externes (correspondant ainsi aux messages de protocole). La raison justifiant l'utilisation de l'ASN.1 et non l'IDL est qu'un grand nombre de protocoles avec lequel le RI doit interfonctionner sont déjà spécifiés en ASN.1.

La cohérence entre les spécifications traitement et les spécifications ingénierie peut être maintenue au moyen d'un outil. Par exemple, les objets traitement, spécifiés en UML/OMT, peuvent être "collés" dans la spécification SDL comme des objets SDL (types). L'outil conservera ses liaisons entre les objets UML/OMT et SDL, et les vérifications d'homogénéité peuvent être effectuées entre les deux modèles. De même, les diagrammes de processus SDL peuvent être automatiquement générés à partir des descriptions des graphiques d'état UML/OMT.

4.1.5 Point de vue technologie

Le point de vue technologie décrit l'implémentation d'un système en termes de composants matériels et logiciels. Il peut être nécessaire de prendre en considération les contraintes de coût et de disponibilité. Les choix influencent la mise en œuvre, le point de vue technologique n'entre pas dans le cadre de la normalisation.

4.2 Evaluation de l'ODP

Les avantages apportés par l'ODP en matière de modélisation des services du RI sont les suivants:

- *affinage progressif des spécifications*: montre plusieurs scénarios possibles d'affinage des spécifications dans l'ODP (voir Figure 4). Il faut noter que l'affinage des spécifications et les transformations peuvent être effectués de manière itérative.

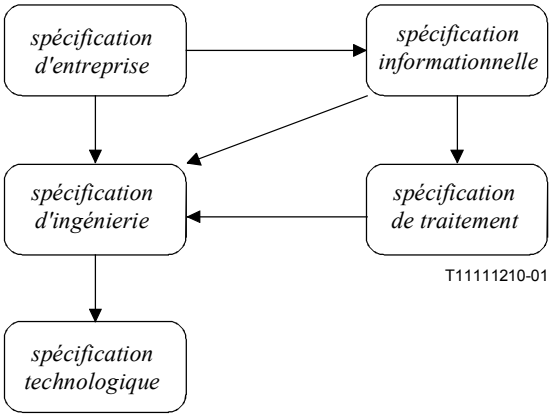


Figure 4 – Scénarios d'affinage possibles des spécifications en ODP

- *Mécanismes de transparence*: l'ODP dispose de plusieurs mécanismes de transparence à différents niveaux d'abstraction. Ces mécanismes sont les suivants: la transparence d'accès, la transparence de lieu, la transparence de défaillance, la migration, la transparence et la transparence de transaction.
- *Portabilité du service et réutilisation des composantes de service*: ces deux exigences peuvent être facilement prises en charge dans un environnement de traitement décentralisé orienté objet, tel qu'une plate-forme CORBA.
- *Interaction de caractéristique*: l'ODP prescrit nettement l'utilisation des techniques de description formelle (FDT, *formal description techniques*) pour les différents points de vue. L'utilisation des techniques FDT améliore la description du comportement du service et réduit fortement les risques d'interaction entre services.
- *Fonctions de nomage et de courtage*: elles font partie des fonctions définies par l'ODP. La fonction de courtier fournit le moyen de rendre public les offres de service et le moyen de découvrir des offres de service via des demandes de service.

L'ODP ne définit pas de méthode de développement de chacun des points de vue ou n'indique pas le langage qui doit être utilisé pour le décrire. Toutefois, l'ODP doit être appliqué en association avec les techniques de modélisation appropriées pour chacun des points de vue.

4.3 Langage unifié de modélisation (UML)

Présentation générale de l'UML

Le langage unifié de modélisation (UML, *unified modelling language*) est un langage qui reflète le consensus qui se dégage dans le monde de "l'orienté objet" sur les concepts essentiels de modélisation. Il permet l'expression d'écarts en termes de mécanismes d'extension. Les développeurs de l'UML avaient les objectifs suivants lors de l'élaboration de ce langage:

- définir une sémantique et une notation suffisantes pour couvrir de façon directe et économique une grande variété de questions récentes relatives à la modélisation;
- définir une sémantique suffisante pour traiter certaines questions relatives à la modélisation future, en particulier associée à la technologie des composants, à l'informatique décentralisée, aux cadres et à l'exécutabilité;
- définir des mécanismes d'extensibilité de sorte que des projets individuels peuvent étendre le métamodèle pour leurs applications à faible coût. Les utilisateurs ne doivent pas être obligés d'ajuster le métamodèle UML en lui-même;
- définir des mécanismes d'extensibilité de sorte que les futures approches de modélisation peuvent se développer au sommet de l'UML;

- définir une sémantique suffisante pour favoriser l'échange de modèles parmi différents types d'outils;
- définir une sémantique suffisante pour spécifier l'interface avec les répertoires pour l'échange et la mémorisation des défauts du modèle.

4.3.1 Evaluation du langage UML

L'UML est particulièrement utilisé dans les points de vue information et traitement de l'ODP. Bien que l'UML puisse également être appliqué à d'autres points de vue ODP, ses applications sont actuellement limitées à ces deux points de vue. Etant donné qu'un grand nombre de protocoles avec lesquels RI doit interfonctionner sont déjà spécifiés en ASN.1 et en SDL, on peut envisager pour assurer une rétrocompatibilité d'utiliser l'ASN.1 et le SDL de préférence à l'UML dans le point de vue ingénierie.

5 Avantages de l'utilisation de l'orientation objet pour la modélisation des services

La modélisation orientée objet est fondée sur le principe d'"objet". Les objets sont des composantes autonomes, c'est-à-dire que leurs propriétés peuvent être décrites de manière indépendante du monde extérieur. La définition d'un objet comprend les attributs et les méthodes. Les attributs décrivent les données dans l'objet, les méthodes, les opérations qui peuvent être imposées à ces données. Les objets peuvent invoquer des méthodes dans d'autres objets. Parmi les concepts importants de la modélisation orientée objet il y a l'héritage et l'encapsulation.

Compte tenu du domaine d'application et de la complexité des services envisagés pour l'ensemble CS-4 du RI (résultant de l'intégration des services mobiles, des services à large bande et des services IP) l'utilisation du paradigme orienté objet dans le domaine de la normalisation de l'ensemble CS-4 du RI présente trois avantages:

- les méthodes orientées objet semblent adaptées à la spécification des services du RI pendant toute la phase de conception. Toutefois, il convient de vérifier si la modélisation du service RI est réellement liée à un domaine d'application dans lequel les méthodes orientées objet peuvent être aussi appliquées avec succès;
- les méthodes orientées objet sont déjà largement utilisées en génie logiciel;
- un environnement de traitement réparti (DPE, *distributed processing environment*) orienté objet peut être utilisé comme modèle de traitement uniforme permettant une répartition transparente de l'intelligence du réseau (logique de service, création et gestion des services). Un scénario de migration pourrait être celui illustré à la Figure 5 par l'interfonctionnement d'un DPE CORBA (l'intelligence de réseau) et d'un équipement du RI existant (SSP) via une interface passerelle (SS7-IDL):

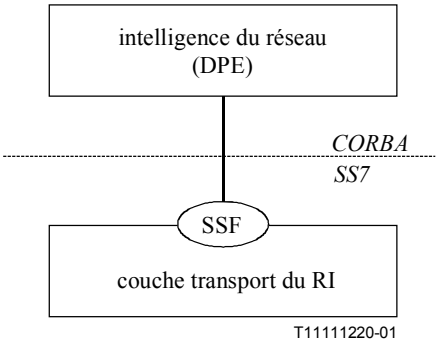


Figure 5 – Scénario d'évolution possible d'un RI

Les avantages liés à l'utilisation d'un environnement DPE orienté objet (par exemple une plate-forme CORBA) pour l'intelligence du réseau sont les suivants: amélioration des possibilités de réutilisation des services qui deviennent aussi plus généraux (composition ou décomposition), facilité de développement, d'installation et d'intégration, création de liens dynamiques et possibilité de reconfiguration des composantes du service.

L'architecture CORBA (*common object request broker architecture*: architecture commune des gestionnaires de demande d'objet) permet l'utilisation de méthodes "objet" dans un environnement décentralisé. Cela signifie que l'architecture CORBA permet de masquer l'emplacement de l'objet adressé. Les interfaces entre les objets spécifiés dans le langage CORBA IDL assurent une transparence technologique, car l'interface masquera la façon dont les objets ont été implémentés. Il faut cependant noter que, actuellement, l'un des principaux inconvénients liés à l'utilisation de ces environnements dans le contexte du RI concerne les exigences en temps réel.

La définition des interactions entre un objet à travers une interface logique est généralement désignée comme étant une interface API. La Figure 6 illustre l'utilisation des interfaces API dans le contexte du RI.

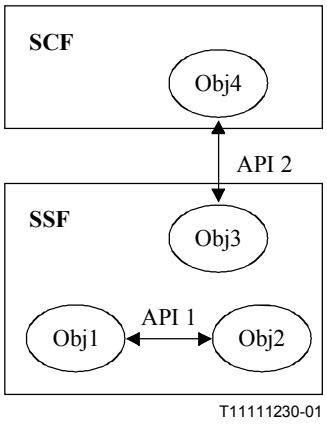


Figure 6 – Exemple d'utilisation d'interface API dans le contexte du RI

Dans la Figure 6, l'objet 1 et l'objet 2 se trouvent dans la fonction SSF. L'objet 1 peut être, par exemple, un objet modèle d'appel et l'objet 2 une table d'état de déclenchement. Lorsque la SCF arme un point de détection d'événement particulier (EDP, *event detection point*), l'objet 1 invoque une méthode (appel API) applicable à l'objet 2 modifiant l'état d'un déclencheur particulier. L'appel API pourrait par conséquent être "arm edp2".

Lorsque l'objet 3 situé dans la fonction SSF invoque une méthode concernant l'objet 4 (appel API 2) qui se trouve résidant dans la SCF, la méthode franchit l'interface externe et le résultat est qu'elle **peut** être mappée en une opération de protocole. Les propriétés de l'appel API, c'est-à-dire son contenu, devront peupler l'opération protocolaire physique. Par conséquent, une correspondance biunivoque existe entre le contenu de l'appel API et les attributs de, par exemple, l'opération INAP. Ainsi, imaginons que l'objet 3 représente un objet "accès au service" et l'objet 4 une logique de service du RI. A la réception de l'information provenant de l'utilisateur, l'objet 3 peut invoquer une méthode dans l'objet 4 comme "receipt of user dialled digits" (réception des chiffres composés par l'utilisateur). Cette méthode peut être réalisée comme étant l'opération INAP "AnalysedInformation" (information analysée).

5.1 Etude de l'utilisation des interfaces API dans l'ensemble CS4 du RI

Dans le secteur des télécommunications en pleine évolution, il est nécessaire plus que jamais de se tenir informé des nouvelles technologies et des nouveaux concepts réseau, en particulier dans le domaine des *technologies de l'information*.

Il existe une nécessité commune à toutes ces architectures, à savoir la réutilisation des scripts de service et la capacité à envoyer et recevoir des informations empaquetées sous une forme reconnaissable. Les protocoles normalisés et les protocoles spécifiques fonctionnent parallèlement de sorte qu'il est nécessaire d'effectuer un mappage d'un protocole à l'autre au niveau des passerelles ou des supports d'interfonctionnement.

Il est maintenant de plus en plus nécessaire de décrire les services en utilisant un format commun. Les services dans le domaine des technologies de l'information utilisent des techniques associées à des interfaces de programmation d'application (API). Les normes internationales, en particulier celles qui sont associées au réseau intelligent, peuvent bénéficier de ces travaux.

5.1.1 Rappel

Lorsqu'il s'agit de proposer un nouveau service, deux questions doivent être posées, à savoir:

1) comment le service sera-t-il pris en charge avec d'autres technologies de transport (*interfonctionnement service-réseau*)?

2) comment ce service interfonctionnera avec d'autres services (*interfonctionnement service-service*)?

En général, il n'est pas facile d'apporter des réponses à ces questions, mais le fait de les poser aboutira à des définitions de service bien plus génériques, souples et évolutives. Ainsi, la définition d'un service de courrier vocal pour le RTPC doit être applicable aux réseaux IP et aux réseaux GSM, et doit pouvoir interfonctionner avec des services tels la mobilité personnelle, le mail et la visiophonie.

Une approche de type composant utilisant des interfaces API bien définies devrait être étudiée pour spécifier et élaborer les services de manière à obtenir améliorer l'interfonctionnement service-réseau et service-service.

Les sous-paragraphes 5.1.2 à 5.1.4 présentent un cadre possible applicable à l'utilisation d'une approche de type composante pour l'élaboration de services de télécommunications lors de la spécification d'un réseau intelligent et de protocoles.

5.1.2 Cadre d'utilisation des interfaces API

Ce cadre est fondé sur le modèle de traitement client/serveur. Le client représente l'entité demandant l'exécution d'une fonction, le serveur représente l'entité qui exécute cette fonction et (facultativement) renvoie le résultat. Le client et le serveur peuvent être représentés comme étant des composantes (ensemble d'objets logiciels ou de modules SIB). L'interface entre le client et le serveur est définie par l'interface API présentée au client par le serveur, et est activée par l'interlogiciel sous-jacent (protocoles de transport par exemple).

Cette approche est complémentaire des méthodologies déjà adoptées pour les normes du réseau intelligent, et les améliore en adoptant un point de vue orienté logiciel – en particulier celle du plan fonctionnel réparti (DFP, *distributed functional plane*). Les entités fonctionnelles identifiées dans le plan DFP sont des composantes initiales valables possibles avec lesquelles les interfaces API sont définies. Par exemple, les flux d'informations en direction et depuis la SSF/CCF constituent une bonne base de définition d'une interface API.

Au fur et à mesure qu'un plus grand nombre d'entités fonctionnelles sera définie (par exemple, la commande de connexion support pour le RNIS-LB, la messagerie vocale, le courtage de service), il faudra normaliser l'interface avec beaucoup de ces capacités. L'approche par composantes permet de

définir ces capacités par leurs interfaces API de manière à pouvoir les intégrer facilement du point de vue logiciel. Les interfaces API peuvent alors être utilisées comme base d'élaboration des protocoles.

Pour les composantes complexes, celles-ci peuvent être subdivisées en objets, le protocole étant définis à partir de ceux-ci. Pour des composantes simples, on peut déduire le protocole directement. En définissant l'interface API dans le détail, le mappage API-protocole est simplifié et plus à l'épreuve du temps. Par exemple, il sera plus facile de faire évoluer les normes actuelles sur le RI et de les aligner avec les technologies de traitement réparties.

La Figure 7 représente les relations entre les services et les composants. Chaque composant est défini par son interface API – l'interface API reflète la fonctionnalité complète de la composante (y compris l'utilisation, la gestion, etc.).

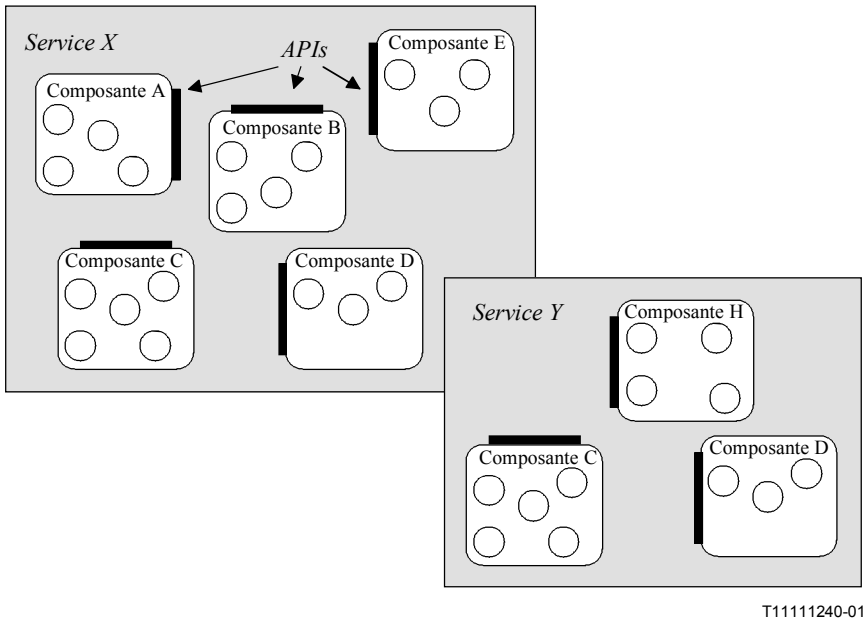


Figure 7 – Services/Composantes/API

5.1.3 Vue d'ensemble des interfaces API

L'interface API est l'élément clé de la définition des composantes client/serveur. L'API constitue la définition de la fonctionnalité d'une composante. Il s'agit essentiellement d'un ensemble d'opérations (ou de *méthodes*) qui peuvent être invoquées sur la composante, chacune d'entre elles forcera la composante à révéler sa fonctionnalité comportementale. Chaque opération est spécifiée syntaxiquement sous la forme d'un identificateur pour l'opération invoquée et les paramètres qui agissent d'une certaine manière sur le comportement de la composante considérée.

Ainsi, l'opération API suivante (*add_transaction*) sur une composante de facturation provoquera l'adjonction de la transaction identifiée par le paramètre *transaction_id*, à la facture du client identifiée par le paramètre *customer_id*:

```
add_transaction ( transaction_id, customer_id )
```

L'interface API est abstraite par rapport à la technologie utilisée pour réaliser la composante et est spécifiée au moyen d'un langage de définition d'interface abstraite. En plus de la syntaxe, l'interface API spécifie la sémantique concernant la façon dont la fonctionnalité est invoquée et le comportement de la composante, en termes de manière dont les attributs publics et privés sont utilisés et modifiés, et les résultats et les erreurs renvoyés. L'interface API définit l'ensemble complet des fonctionnalités exécutables par une composante.

Une présentation particulière d'une API, représente la façon dont la composante est réalisée et la syntaxe et la sémantique concrètes qui permettent d'invoquer cette capacité. La présentation peut restreindre la fonctionnalité qui est accessible à une composante d'invocation. Une composante peut avoir plusieurs présentations mais une seule interface API. Par exemple, la présentation d'utilisation d'une interface API SSF/CCF ne comporterait que les opérations de traitement d'appel, et pourrait être réalisée en utilisant les protocoles INAP ou ISUP; la présentation de gestion prendra en charge les capacités tel l'espacement des appels, et pourrait être réalisée en utilisant les protocoles INAP ou CMIP.

5.1.4 Exemples d'API pour le traitement d'appel

La composante traitement d'appel (SSF/CCF) a un certain nombre de clients potentiels – utilisateurs finaux (appelés ici correspondants finaux), les fournisseurs de services (appelés ici tiers), la gestion, etc. Dans cet exemple, seuls sont considérés les correspondants finaux et les tiers. La différence entre un correspondant final et un tiers tient à ce que les correspondants finaux demandent l'établissement des appels et des connexions supports, leur maintien et leur libération. Le tiers peut agir au nom d'un participant final, ou modifier la façon dont le traitement d'appel a lieu, avec ou sans connaissance des correspondants finaux.

Chaque "appel API" (opération) décrit une opération effectuée par un correspondant final ou par un tiers. L'appel API sera homogène entre les points A et B, bien que le protocole sous-jacent et les mécanismes de transport de réseau peuvent changer entre les deux points précités. L'exemple suivant (Figure 8) permet de préciser ce qui vient d'être dit.

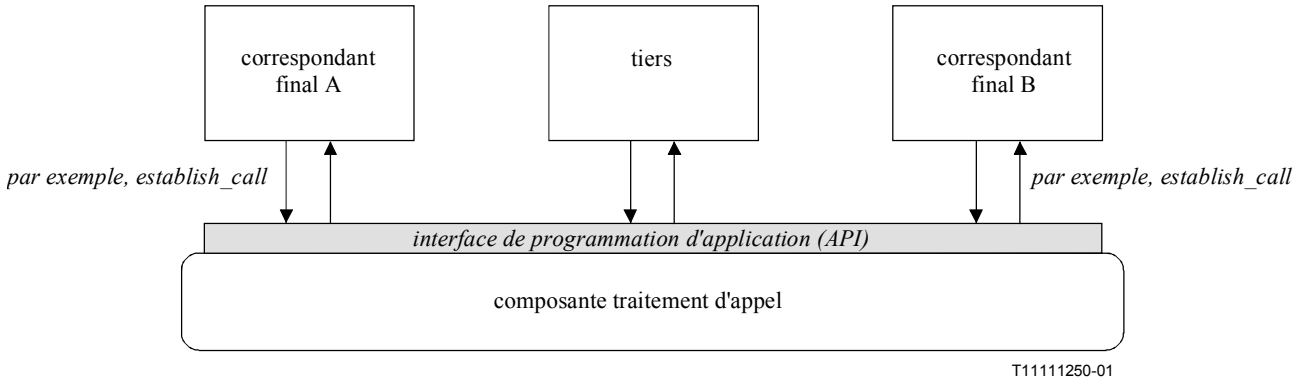


Figure 8 – Exemple d'interface API de traitement d'appel

Supposons qu'un événement (tel décrochage d'un combiné téléphonique) est détecté par un utilisateur final d'un service du RNIS (exécuté dans le terminal). Il est peu probable que le terminal du RNIS ait une relation avec le réseau jusqu'à ce que l'utilisateur compose un numéro. Lorsque le service du correspondant final A a collecté suffisamment de chiffres composés par l'utilisateur, il invoque l'opération API "establish_call" (établissement d'appel) sur la composante traitement d'appel.

Cela se traduit par l'envoi de la signalisation nécessaire à travers le réseau entre les différents objets composante traitement d'appel, par exemple au moyen des protocoles DSS1 et ISUP. Lorsque l'appel atteint l'extrémité distante, la composante traitement d'appel invoque un appel API sur une composante de service dans le service du correspondant final B, et pour des raisons de symétrie on peut appeler cette opération "establish call".

Dans un scénario plus complexe, le tiers pouvait s'inscrire auprès du traitement d'appel pour recevoir une notification d'événement lorsque des critères spécifiques sont vérifiés. Si tel est le cas, le traitement d'appel invoquera une opération sur la composante de service tiers qui a demandé la notification.

Il est assez facile de voir comment l'interface API traitement d'appel avec un tiers peut effectuer le mappage avec normes du RI (par exemple, protocole INAP). Il est moins facile de mapper l'interface API correspondant final, puisque cette question n'a pas été étudiée dans les normes du RI.

5.2 Approche SIB

Le présent sous-paragraphe vise à souligner les inconvénients associés à l'utilisation de l'approche SIB du RI.

La subdivision des services en parties plus petites et réutilisables a été, dans le passé, assez utile pour le RI et est encore utilisée par de nombreux constructeurs dans leurs environnements d'exécution de la logique de service (SLEE, *service logic execution environments*).

La grande idée sous-jacente à cette méthodologie est de définir le plus grand nombre de blocs de données réutilisables et pouvant être exécutés dans de nombreuses applications des services. Le concept de réutilisation de service est important et efficace et, en réalité, peut être mis en œuvre indépendamment de la façon dont on modélise un service. Toutefois, lorsqu'on compare la méthodologie SIB avec les méthodologies orientées objet, les inconvénients des SIB sont très apparents. Voici la liste de ces inconvénients:

- tout d'abord, il est très difficile d'affiner (les SIB) du (GFP) jusqu'au niveau Plan fonctionnel réparti (DFP);
- l'affinage depuis le plan service vers les SIB du plan GFP utilise une seule étape intermédiaire, à savoir celui des modules SIB de haut niveau. De ce fait, l'affinage du plan de service vers le plan GFP n'est pas très précis. Afin d'obtenir un affinage plus progressif, le concept de module SIB de haut niveau n'est pas suffisant, il est nécessaire d'avoir un affinage plus itératif;
- la modélisation du plan GFP au moyen de modules SIB ne permet pas d'obtenir un modèle de service complet, étant donné que les données de service sont subordonnées aux fonctions exécutées par un service RI. Les données nécessitent une attention plus grande, par exemple pour permettre une meilleure gestion des données de service;
- l'approche SIB appliquée à la modélisation du plan GFP des ensembles CS1 et CS2 diffèrent des techniques de modélisation appliquées en technologie de l'information. Toutefois, il est de plus en plus nécessaire d'aligner les produits provenant du secteur des technologies de l'information (par exemple, service de facturation et service clientèle);
- les spécifications techniques utilisées dans le plan fonctionnel global (GFP) et dans le plan fonctionnel réparti (DFP) ne permettent pas de disposer de mécanismes de contrôle d'homogénéité entre les spécifications et ainsi limitent les compositions de service et leur réutilisation. Plus généralement, il n'existe pas de méthode globale de spécification préconisée pour le RI.

6 Passage possible des modules SIB aux capacités de service orientées objet

Le fossé entre les services et les caractéristiques de service d'un côté et les flux d'information et les protocoles de l'autre est énorme. De ce fait l'exhaustivité des flux d'information et des opérations/paramètres de protocole est difficile à vérifier par rapport aux services et aux caractéristiques de service qui doivent être pris en charge. En ce qui concerne les capacités CS-1 et CS-2 du RI, le plan GFP a été utilisé pour obtenir une "couche" entre les services et les caractéristiques de service d'un côté et les flux d'informations et les opérations/paramètres de

protocole de l'autre. Il s'agit là d'un moyen permettant de prendre en charge la validation de l'exhaustivité des protocoles. Afin de pouvoir modéliser des services complexes, les techniques de modélisation actuellement utilisées pour le plan GFP doivent être améliorées. On considère que la technique orientée objet est très prometteuse pour la modélisation des services. En conséquence, il est utile d'étudier le passage des techniques de modélisation actuelles vers une approche plus orientée objet.

6.1 Modèle de classe de service

Le modèle de classe de service comporte deux vues complémentaires:

- *la vue exécution du service.* Cette vue comporte les classes d'objetss les plus élémentaires disponibles pour l'élaboration des services;
- *la vue fourniture/personnalisation du service.* Cette vue montre les classes d'objetss visibles pendant la fourniture et la personnalisation des services. Elle masque les classes d'objetss qui ne peuvent être gérées que pendant le phase active du service.

Ces deux vues constituent ce qui est appelé "modèle d'objet" dans plusieurs méthodologies orientées objet. Cela signifie que ces deux vues offrent une vue statistique du service. On peut accéder à certaines classes d'objets dans la vue exécution du service, et à d'autres classes dans la vue fourniture/personnalisation du service et même à certaines classes dans les deux vues (voir Figure 9). Les instances de la classe d'objetss "appel", par exemple contiennent les données qui changent pendant l'exécution du service. Les instances de la classe d'objetss "Announcement" (annonce) contiennent les données qui peuvent être lues pendant l'exécution d'un service (diffusion d'une annonce) le contenu des données d'annonce peut être modifié avec le système de gestion des services. Par conséquent, la classe d'objetss "Announcement" est l'intersection de la vue "exécution du service" et de la vue "fourniture/personnalisation du service".

Dans la Figure 9, on fait la distinction entre les ressources service et les ressources réseau. Un exemple de classe d'objetss ressource réseau est la classe d'objetss terminal. La relation entre les ressources service et les ressources réseau exprime la façon dont les capacités de service sont mappées en ressources de réseau. Ainsi, un utilisateur peut lancer un appel depuis différents terminaux tel son poste téléphonique classique ou son terminal RNIS. Les classes d'objets dans la vue exécution du service dont les instances sont créées pendant l'exécution du service sont *dynamiques* (par exemple, appel) toutes les ressources sont *persistantes* étant donné que leur durée de vie est plus grande qu'une simple exécution de service.

Pour la modélisation des services, seules les ressources services sont pertinentes. Ces classes d'objets sont les capacités de service utilisées pour composer les services.

Il convient de noter que les classes d'objets représentées dans les Figures 9 et 10 ne sont que des exemples.

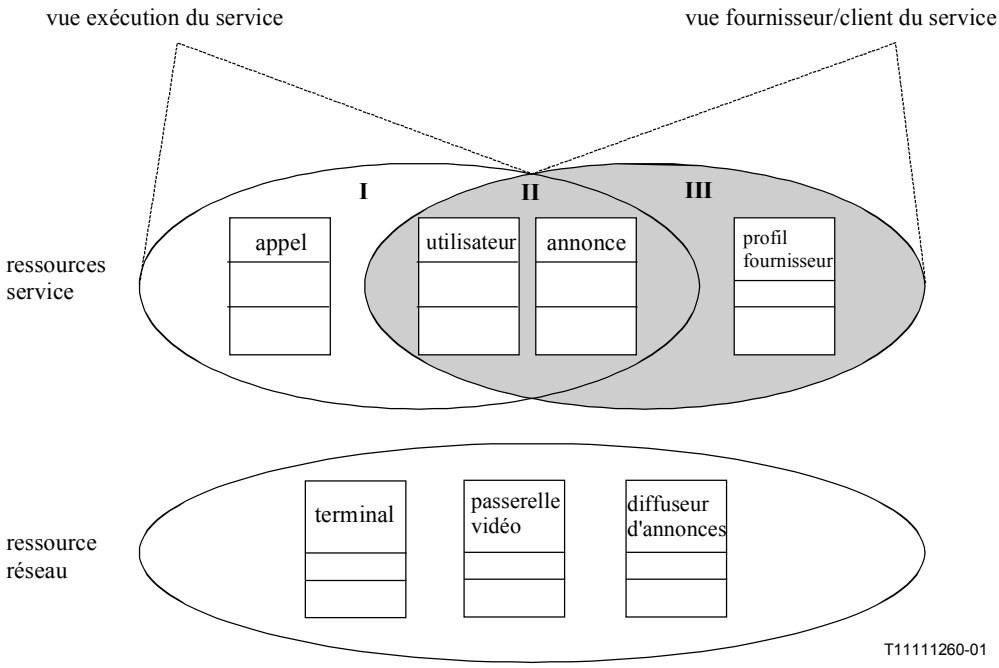


Figure 9 – Différentes vues d'un service

6.2 Exécution du service

La Figure 10 représente le modèle d'objet de la vue exécution du service au niveau le plus élevé. La classe d'objetss *logique de service* représente le service et peut être constituée de zéro ou de plusieurs classes d'objets *logique de service* et de zéro ou de plusieurs classes d'objets *ressource de service*. On peut aussi modéliser un service sous forme d'une composition de caractéristiques de service (plus généralement, cela permet un affinage progressif). Au niveau de décomposition le plus bas, il y a les classes d'objets *ressources de service*. Les classes d'objets *ressource de service* ont un caractère indépendant des services et peuvent être considérées comme l'équivalent orienté objet des modules SIB actuellement utilisés. Une classe d'objetss *ressource de service* peut être réutilisée dans un nombre nul/non nul de classe d'objetss *logique de service* une instance de classe d'objets *Service Resource* peut être réutilisée dans un nombre nul/non nul de classe d'objets *Service Logic*. Une instance de la classe d'objets *ServiceResource* est soit *DynServResource* ou *PersServResource*. Une classe d'objets *DynServResource* représente toutes les classes d'objets dans lesquelles les instances d'objet existent seulement pendant une seule exécution du service (voir I dans la Figure 9). La classe d'objets *PersServResource* représente toutes les classes d'objets pour lesquelles les instances d'objet sont persistantes, c'est-à-dire que leur durée de vie est supérieure à la durée d'exécution d'un seul service, c'est-à-dire la période de temps pendant laquelle un client s'abonne à un service (voir II dans la Figure 9). Le *ServSessionMgr* invoque le service et peut être imaginé comme étant le processus d'appel de base dans le plan GFP du CS-2" du RI. Le *ServSessionMgr* peut être vu lui-même comme une spécialisation de la classe d'objets *DynServResource* mais est décrit séparément en raison de son rôle particulier.

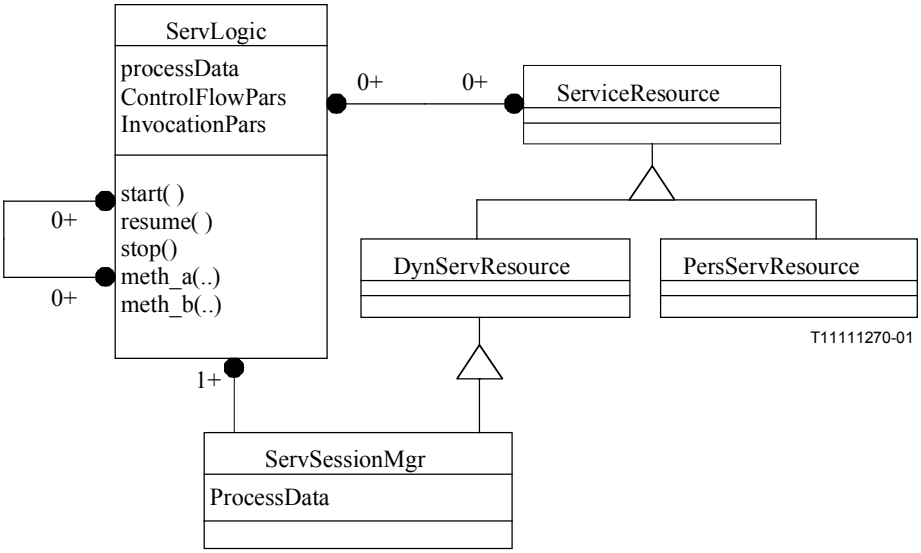


Figure 10 – Modèle d'objet de la vue exécution du service

Les classes d'objets *DynServResource* et *PersServResource* ont chacune un nombre nul ou non nul de spécialisation. Une spécialisation de la classe d'objets *DynServResource* est par exemple la classe d'objets *Call* (voir Figure 11). Une instance de cette classe peut contenir par exemple, des données d'instances d'appel tel le A-number et B-number. Les méthodes associées à cette classe seraient par exemple Add_Party( ), Remove_Party( ) et Connect( ). Des spécialisations possibles de PersServResource seraient par exemple, *Announcement* et *Counter*. Ces classes d'objets sont persistantes dans la mesure où elles existent plus longtemps qu'une seule exécution de service. Elles sont observées dans les deux vues suivantes, exécution du service et personnalisation/fourniture du service, étant donné que les méthodes utilisées dans les instances d'objet de ces classes peuvent être invoquées avec les deux vues. Par exemple, pendant l'exécution de la logique de service, les données d'un objet *Announcement* peuvent être lues; pendant la phase de personnalisation du service, le texte de l'annonce peut être modifié.

6.3 Passage des modules SIB du CS-2 aux classes d'objets et aux méthodes associées

Afin de permettre une évolution des techniques de modélisation des services actuellement utilisés vers une approche plus orientée objet, il est utile de mapper les modules SIB en classes d'objets et en méthodes objet. Dans le tableau qui suit la Figure 11, on donne un exemple d'un tel mappage.

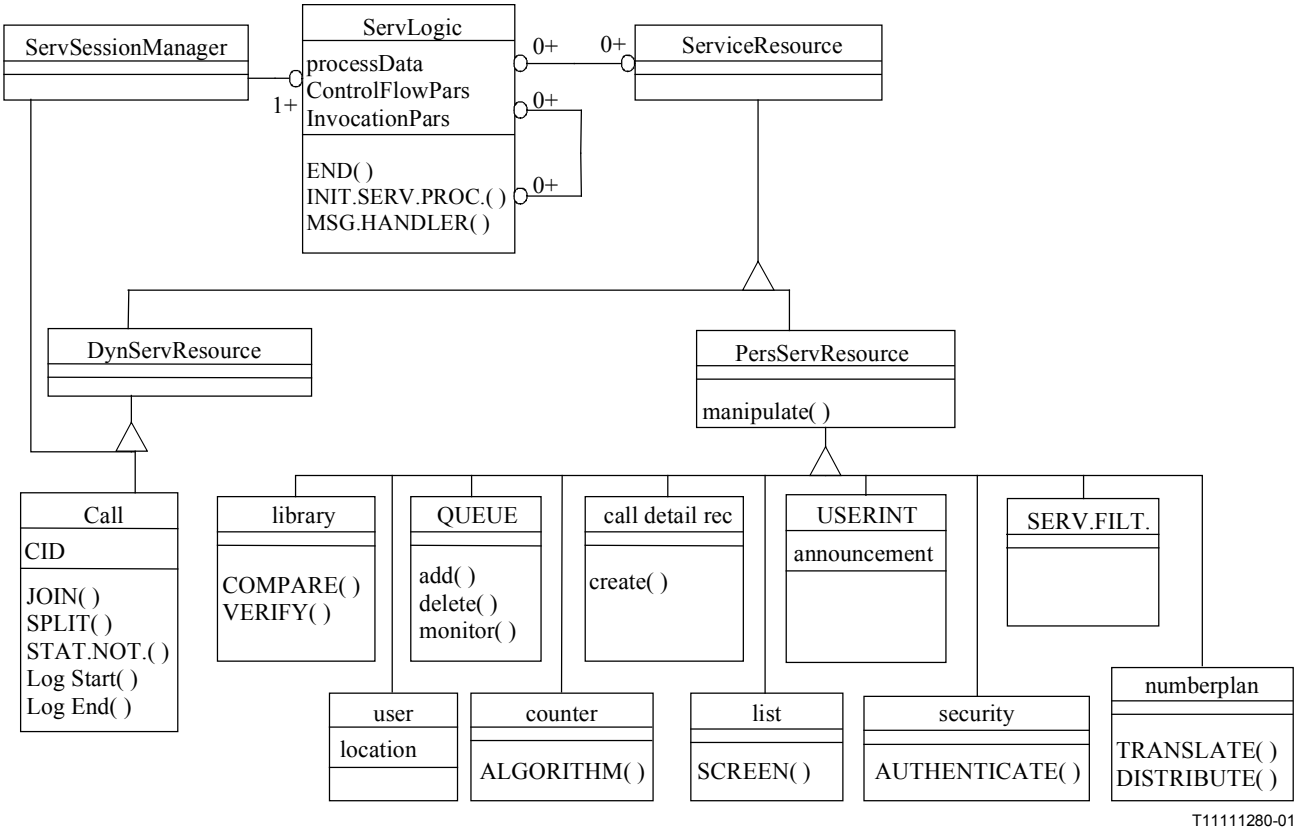


Figure 11 – Passage possible des modules SIB aux classes d'objets et méthodes objet

Le tableau ci-dessous décrit le mappage des modules SIB du CS-2 du RI (Recommandation Q.1223) en classes d'objets utilisées ci-dessus:

SIB	Mappage
Algorithme	Applique un algorithme mathématique à une valeur (par exemple incrémentation ou décrémentation d'un compteur) et peut par conséquent être considéré comme une méthode de la classe d'objets *Counter*.
Authentification	Fonction exécutée aux fins de sécurité et est par conséquence considérée comme une méthode de la classe d'objets *Security*.
Taxation	La taxation peut être considérée comme la création d'un enregistrement des détails relatifs à l'appel et par conséquent représentée par la méthode *Create* dans la classe d'objets *Call Detail Record*.
Comparaison	Fonction utilisée pour vérifier un identificateur par rapport à une valeur de référence spécifiée, telle l'heure de la journée, le lieu d'origine ou une valeur de chiffre. Elle peut être vue comme une opération mathématique générique et est par conséquent représentée par une méthode de la classe d'objets *Library* (qui peut être considérée comme une bibliothèque standard avec des opérations mathématiques et des opérations de chaîne telles qu'on les trouve dans de nombreux langages de programmation).
Distribution	La distribution d'un appel vers une destination conformément à un algorithme particulier peut être considérée comme une méthode de la classe d'objets *Numberplan*.
Fin	Indique la terminaison normale d'un processus de service et peut être considérée comme une méthode de la classe d'objets *Service Logic*.

SIB	Mappage
Initialisation du processus de service	L'invocation d'une logique de service parallèle peut être considérée comme une méthode de la classe d'objets de niveau le plus élevé, c'est-à-dire de la classe d'objets *Service Logic*.
Adjonction	L'adjonction d'un ou de plusieurs correspondants associés à un groupe d'appel courant à un autre groupe d'appel dans le même appel peut être considérée comme une méthode de la classe d'objets *Call*.
Registre d'informations d'appel	La journalisation des données d'instance d'appel identifiées est considérée comme étant les méthodes *Log Start* et *Log End* de la classe d'objets *Call*, dans laquelle les données d'instance d'appel sont des attributs.
Gestionnaire de messages	La communication entre des processus dans un seul service peut être considérée comme une méthode de la classe d'objets la plus élevée, c'est-à-dire la classe d'objets *Service Logic*.
File d'attente	Une instance d'objet *Queue* contient un nombre nul ou non nul de références à des instances d'objet *Call*. Les instances *Call* peuvent être ajoutées, supprimées, etc. de *Queue*.
Filtre	La vérification pour voir si un numéro se trouve sur une liste (par exemple pour le filtrage d'appel) peut être considérée comme une méthode de la classe d'objets *List*.
Gestion des données de service	Exécute des opérations sur des données de service telles l'addition, la suppression, la modification ou l'extraction, etc. Ces opérations sont collectées dans la classe d'objets *PersServResource* comme la méthode *Manipulate*.
Filtre de service	Filtre les appels conformément à un mécanisme spécifié et est considérée comme une classe d'objets distincte avec des méthodes telles *Activate* et *Report*.
Séparation	Sépare un correspondant ou un groupe de correspondants d'un appel en cours et associe un ou des correspondants à un nouvel appel ou à un appel existant. Il pourrait s'agir d'une méthode de la classe d'objets *Call*.
Notification de statut	Capacité d'interrogation sur l'état (modification d'état) des ressources de réseau. Peut être considérée comme une méthode de la classe d'objets *Call,* permettant la vérification de l'état des ressources de réseau qui permettent de réaliser le *Call*.
Traduction	La traduction d'un numéro en un autre numéro pourrait être une méthode de la classe d'objets *Numberplan*.
Interaction utilisateur	Le mécanisme d'invite et de collecte pour fournir à l'utilisateur les informations ou obtenir les informations de la part de l'utilisateur, est considéré comme une classe d'objets distincte.
Vérifier	Peut prendre en charge la vérification de la syntaxe de tous les types de données et est par conséquent représenté comme une méthode faisant partie de la classe d'objets *Library*.

APPENDICE I

Bibliographie

[TINA] TINA-C architecture specifications (e.g. Service Architecture and Information Architecture).

[OMT] Object-Oriented Modelling and Design: *Rumbaugh et autres*.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, de télégraphie, de télécopie, circuits téléphoniques et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	**Commutation et signalisation**
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information et protocole Internet
Série Z	Langages et aspects informatiques généraux des systèmes de télécommunication

Imprimé en Suisse
Genève, 2001