UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T Q.65

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

(06/2000)

SÉRIE Q: COMMUTATION ET SIGNALISATION

Fonctions et flux d'information des services du RNIS – Méthodologie

Méthode fonctionnelle unifiée de caractérisation des services et des capacités des réseaux et utilisation des techniques alternatives orientées objet

Recommandation UIT-T Q.65

(Antérieurement Recommandation 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
Méthodologie	**Q.60–Q.67**
Services de base	Q.68–Q.79
Services complémentaires	Q.80–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.799
INTERFACE Q3	Q.800–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.

Recommandation UIT-T Q.65

Méthode fonctionnelle unifiée de caractérisation des services et des capacités des réseaux et utilisation des techniques alternatives orientées objet

Résumé

La présente Recommandation UIT-T remplace la Recommandation UIT-T Q.65 (06/97) existante. Elle contient la méthodologie fonctionnelle unifiée (UFM, *unified functional methodology*) qui décrit une architecture fonctionnelle commune pour la fourniture de services et traite des besoins de signalisation pour la mise en place de services. La méthode générale permettant d'élaborer des Recommandations portant sur les services de commutation et de signalisation du RNIS est décrite dans l'UIT-T I.130 et se constitue de trois phases. La méthode a été étendue au-delà du domaine du RNIS afin d'inclure des services fournis au sein de réseaux de types divers ou entre de tels réseaux. La méthodologie fonctionnelle unifiée combine la démarche traditionnelle décrite dans la version 1988 de la présente Recommandation avec certaines des démarches utilisées traditionnellement dans la méthode de description du réseau intelligent (RI). La méthode détaillée d'élaboration de la partie de la phase 2 de ces Recommandations est décrite dans la présente Recommandation. Le texte principal de la présente Recommandation fournit une information générale au sujet de la méthodologie fonctionnelle unifiée, des étapes de la méthode, des conventions utilisées par les outils de description ainsi que des directives pour son utilisation.

Source

La Recommandation Q.65 de l'UIT-T, révisée par la Commission d'études 11 (1997-2000) de l'UIT-T, a été approuvée le 15 juin 2000 selon la procédure définie dans la Résolution 1 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 Recommandation, 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 Recommandation 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 Recommandations.

A la date d'approbation de la présente Recommandation, 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 Recommandation. 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

Recommandation UIT-T Q.65

Méthode fonctionnelle unifiée de caractérisation des services et des capacités des réseaux et utilisation des techniques alternatives orientées objet

1 Introduction

La présente nouvelle Recommandation contient la méthodologie fonctionnelle unifiée (UFM, *unified functional methodology*), qui décrit une architecture fonctionnelle commune pour la fourniture de services et traite des besoins de signalisation pour la mise en place de services. La méthode générale permettant d'élaborer des Recommandations portant sur les services de commutation et de signalisation du RNIS est décrite dans UIT-T I.130 et se constitue de trois phases. La méthode a été étendue au-delà du domaine du RNIS afin d'inclure des services fournis au sein de réseaux de types divers ou entre de tels réseaux. Cette nouvelle méthodologie fonctionnelle unifiée combine la démarche traditionnelle décrite dans la version 1988 de la présente Recommandation avec certaines des démarches utilisées traditionnellement dans la méthode de description du réseau intelligent (RI) et celles qui sont utilisées dans la méthodologie fonctionnelle unifiée. Elle donne à l'utilisateur de la Recommandation le choix d'utiliser soit les méthodes qui sont décrites dans la version de 1997 de UIT-T Q.65 soit les techniques qui sont employées dans le secteur informatique. Les méthodes détaillées d'élaboration de la partie de la phase 2 de ces Recommandations sont décrites dans la présente Recommandation.

Le texte principal de la présente Recommandation fournit une information générale au sujet de la méthodologie fonctionnelle unifiée, des étapes de la méthode, des conventions utilisées par les outils de description ainsi que des directives pour son utilisation. L'Appendice I présente un résumé pour l'établissement d'un avant-projet de description de service de la phase 2 à l'aide de la méthodologie faisant appel au concept de module indépendant du service (SIB, *service independent building block*). L'Appendice II contient le modèle fonctionnel unifié, notamment les communications à large bande et la séparation des supports. L'Appendice III donne des "Exemples de classes d'interfaces formées à partir des spécifications relatives aux services ou aux applications".

1.1 Méthodologie fonctionnelle unifiée: résumé

La méthodologie fonctionnelle unifiée (UFM) permet la description fonctionnelle de services au moyen soit de diagrammes de flux d'information, d'actions d'entités fonctionnelles (FEA, *functional entity actions*) et du langage de description et de spécification (SDL, *specification description language*), soit de descriptions de classes d'interfaces, du langage générique de description d'interface (IDL, *interface description language*) et du langage de description de spécification (SDL), en partant d'une architecture fonctionnelle unifiée unique. Dans le premier cas, le concept de module indépendant du service (SIB, *service independent building block*) a été adopté pour traiter des besoins de création de service ainsi que pour introduire des blocs de diagrammes de flux, de diagrammes SDL et d'actions FEA pouvant être réutilisés et catalogués. L'utilisateur a également la possibilité d'employer le langage de modélisation unifié (UML, *unified modelling language*) en même temps que les descriptions détaillées des classes d'interfaces pour appeler des méthodes pouvant s'appliquer aux interfaces de programmation d'application (API, *application programming interface*) et au langage IDL. Ce nouveau concept rend l'élaboration du langage SDL plus aisée. L'utilisateur peut employer les deux méthodes simultanément en choisissant celle qui facilite la description d'une interface particulière. Le modèle unifié permet de décrire de façon semblable toute architecture de réseau (RNIS, RNIS-LB, RI, IMT-2000 et RGT).

La méthodologie doit répondre aux besoins existants, mais elle doit également pouvoir évoluer afin de prendre en compte des extensions et de nouvelles technologies. La méthode contient un grand nombre d'éléments de la version 1988 de la présente Recommandation, de sorte qu'elle peut être utilisée immédiatement, même en cours d'évolution. L'utilisation du concept de module SIB ou de techniques orientées objet (O-O) pour décrire la relation entre objets permet d'étendre le processus de définition du service de la phase 2. De nouveaux modules SIB ou classes d'interfaces seront créés en fonction des besoins. Un catalogue de modules SIB ou de classes d'interfaces sera maintenu afin de servir de référence pour la création et la définition de services.

Les principes définissant le domaine d'application de la méthodologie UFM sont les suivants:

1) la méthodologie fonctionnelle unifiée permet de créer des descriptions fonctionnelles, lorsqu'existent des descriptions de la phase 1 ou des spécifications de capacités de réseau pour la création d'un service;

2) la méthodologie se fonde principalement sur l'expérience de la méthodologie de la phase 2 du RNIS [UIT-T Q.65 (1988)], de la méthodologie du RI [UIT-T Q.1210 (1995)] et des techniques de calcul réparties déduites de celles de Rumbaugh, Booch *et al.* et affinées à l'aide de la méthodologie UML.
   - L'UIT-T Q.65 fournit une quantité importante d'informations générales concernant la description de services au moyen d'un modèle fonctionnel, d'entités fonctionnelles et d'actions d'entités fonctionnelles;
   - la méthodologie du RI répond aux besoins de flexibilité, d'indépendance par rapport au service et de possibilité de réutilisation;
   - la méthodologie UML est un outil permettant une description détaillée des classes d'interfaces et de leurs relations. Cette caractéristique ajoutée à plusieurs types différents d'outils usuels facilite l'élaboration du langage IDL qui est à la base de l'interface de programmation d'application (API, *application programming interface*).

3) La méthode est fondée sur un modèle fonctionnel unifié.
   - L'architecture fonctionnelle unifiée est étendue afin d'y inclure de nouveaux besoins et peut l'être davantage afin de s'adapter aux besoins des utilisateurs;
   - un ensemble adéquat d'entités fonctionnelles est choisi pour chaque description de service afin de composer le modèle;
   - ceci conduit à un ensemble cohérent de diagrammes de flux et de communications API et de diagrammes SDL.

4) La gestion d'une bibliothèque de modules SIB (contenant des flux d'information et des diagrammes SDL correspondants) permet l'utilisation de ces blocs pour l'élaboration de l'architecture fonctionnelle dans le plan fonctionnel réparti (voir Figure 1).
   - Des ensembles de modules SIB sont en cours de création afin de prendre en charge les ensembles de capacités définies dans les Recommandations du RI;
   - les modules SIB sont mappés avec l'architecture fonctionnelle complète au moyen d'actions d'entités fonctionnelles prédéfinies, de diagrammes SDL prédéfinis et de flux d'information prédéfinis;
   - les modules SIB peuvent constituer des outils efficaces pour l'élaboration de la phase 2; d'autres caractéristiques (par exemple les diagrammes SDL ou les diagrammes de flux d'information) sont susceptibles d'être automatisées.

5) La gestion d'une bibliothèque de lots, de classes d'interfaces et de langage IDL permet de définir l'interface API dans une interface particulière. Ils permettent également de déterminer l'architecture fonctionnelle dans le plan fonctionnel réparti (voir Figure 1).

6) La méthodologie répond aux besoins actuels, mais devra évoluer pour accueillir des améliorations ou de nouvelles technologies.

- Possibilité d'utilisation de la méthode, même en cours d'évolution;
- démarche par phases utilisant l'architecture unifiée pour l'ensemble du domaine de compétence de l'UIT-T;
- de nouveaux modules SIB seront nécessaires (services existants et nouveaux services cibles);
- de nouvelles communications avec les interfaces API seront élaborées;
- de nouvelles relations devront être définies;
- de nouveaux services cibles devront être pris en considération (par exemple des appels multiparticipants);
- le processus n'est pas automatisé actuellement, mais la réutilisation joue un rôle majeur;
- un nouveau savoir-faire doit être acquis pour la description de services à partir de modules SIB;
- une vérification supplémentaire portant sur des services existants sera utile (pour la précision de la méthode et l'acquisition d'expérience).

7) La méthodologie unifiée pour l'élaboration d'une description de la phase 2 devra inclure les points suivants:

- identification de la capacité de service ou de réseau;
- modèle fonctionnel et définition d'entités fonctionnelles (méthode fonctionnelle unifiée);
- identification de modules SIB et/ou de paquetages (facultative);
- définition d'actions d'entités fonctionnelles et/ou de langage IDL;
- flux d'information et/ou communications API;
- diagrammes SDL (description dynamique) (facultatifs);
- scénarios (allocation physique d'entités fonctionnelles).

8) Définition de scénarios physiques à un moment adéquat.

- Des décisions doivent être prises en ce qui concerne les implémentations recommandées pour le réseau;
- le catalogue d'entités physiques croît compte tenu des extensions faites au RI, au RNIS-LB, au réseau IMT-2000 et à d'autres réseaux;
- ceci augmente considérablement le nombre de scénarios possibles.

9) Elaboration de Recommandations traitant du perfectionnement des outils et de l'automatisation du processus de description.

- les UIT-T Z.100 (SDL) et Z.120 (Diagrammes de succession de messages – Flux d'informations) peuvent être utilisées de manière plus efficace;
- les outils peuvent être utilisés pour la validation des diagrammes SDL et des diagrammes de flux;
- il convient de diffuser les règles et les conventions pour la définition et l'assemblage des modules SIB; une certaine automatisation de ce processus est possible.

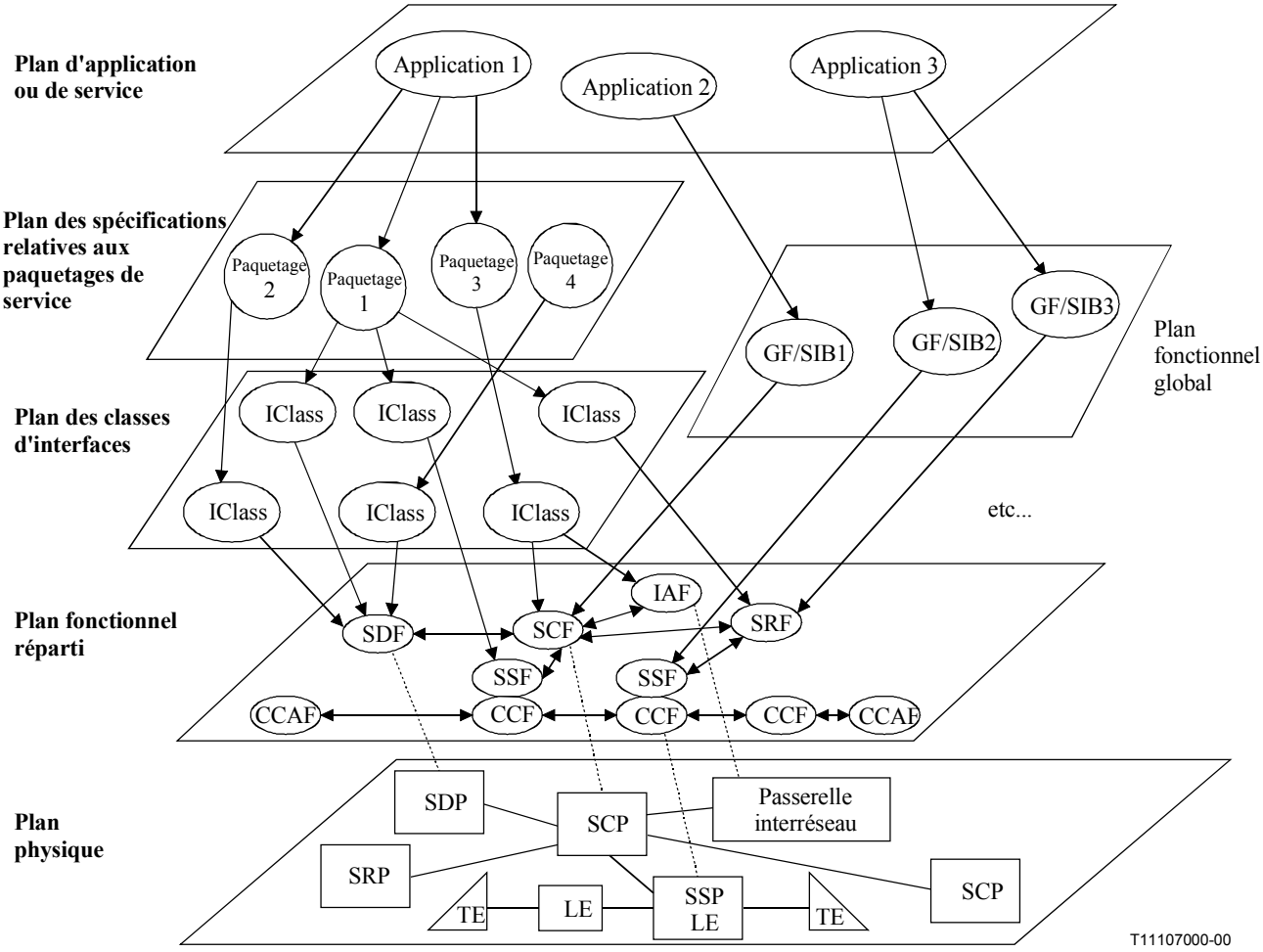


Figure 1/Q.65 – Modèle conceptuel de la méthodologie indiquant les relations entre services, les fonctions globales/modules indépendants du service, les paquetages de service, les classes d'interfaces, les entités fonctionnelles et les entités physiques

La figure ci-dessus représente le modèle fonctionnel de réseau RI modifié. Le modèle fonctionnel est conçu de manière à permettre à l'utilisateur de choisir soit l'approche comportant les modules SIB, soit celle qui emploie les méthodes O-O. Les résultats sont à peu près les mêmes. Si l'on utilise les modules SIB, le résultat consiste en des flux d'informations qui sont mappés sur des entités fonctionnelles, tandis qu'à l'aide des méthodes O-O, le résultat consiste en un ensemble de communications avec les interfaces API qui sont aussi mappées sur des entités fonctionnelles. Cette question fera l'objet d'une explication au paragraphe 3.

Toutes les fonctions élémentaires FE qui sont décrites dans le plan DFP de la Figure 1 peuvent être situées dans un seul domaine de réseau, à l'exception de la fonction d'accès à l'intelligence (IAF, *intelligence access function*). Cette fonction peut être employée par un autre réseau non RI pour reproduire la fonctionnalité qui est assurée par la fonction SCF dans le réseau RI. Il est nécessaire, pour cette raison, d'assurer un genre de fonctionnalité passerelle afin de protéger la séparation entre les réseaux. Cette fonctionnalité passerelle peut assurer une protection de sécurité/coupe-feu et également des capacités de mappage de protocole. En conséquence, dans la Figure 1, l'équivalent du plan physique est représenté comme étant une passerelle interréseau. Ce point d'entrée ou de sortie du réseau RI fournit aussi un accès entre les réseaux RI et IP.

1.2 Définition de la phase 2 (résumé)

– Spécification d'un modèle fonctionnel, pour la description d'un service spécifique ou d'une capacité réseau spécifique, au moyen d'entités fonctionnelles (FE, *functional entities*) appartenant au modèle fonctionnel unifié;

– spécification des actions d'entités fonctionnelles (FEA) nécessaires;

– spécification des flux d'information ou des communications API entre entités fonctionnelles;

– spécification de la description SDL de chaque entité fonctionnelle;

– recommandation d'un nombre limité de scénarios réalistes pour l'allocation des entités fonctionnelles aux entités physiques.

Pour les capacités de services et de réseau devant être normalisées, l'information entrante de la phase 2 est constituée des descriptions de la phase 1 concernant les capacités de réseau et les capacités de services de base et supplémentaires. En qui concerne les capacités de services et de réseau pour lesquelles la phase 2 ne fournit pas de description détaillée, la méthode de la phase 2 peut utiliser comme information entrante les descriptions SIB ou les spécifications relatives aux lots de service des fonctions de service. La description de la phase 1 considère le réseau comme une entité globale fournissant ces services à l'utilisateur (le réseau peut, dans ce contexte, englober certaines capacités de l'équipement utilisateur). La description de la phase 2 définit les fonctions nécessaires et leur répartition au sein du réseau. La phase 2 utilise et interprète les interactions utilisateur/réseau de la phase 1, comme indiqué par la Figure 2.

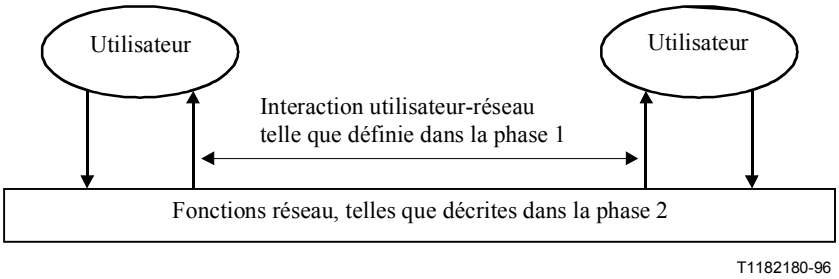


Figure 2/Q.65 – Relations entre la phase 1 et la phase 2

La phase 2 de la méthode utilise des techniques fournissant les caractéristiques souhaitables suivantes:

– spécification fonctionnelle unique pouvant être mise en œuvre par un certain nombre de réalisations physiques en vue de la fourniture du service;

– définition précise des capacités fonctionnelles et de leur distribution possible dans les équipements réseau (et, dans certains cas, les équipements utilisateur) afin de prendre en charge les capacités réseau ainsi que les services de base et complémentaires;

– description détaillée des fonctions qui seront à la disposition des flux d'information et des communications API, mais sans en préciser l'implémentation;

– prescriptions au sujet des capacités de protocole et de commutation, constituant l'information entrante de la phase 3 de la méthode.

L'information issue de la phase 2 est utilisée:

a) en phase 3 par les concepteurs de protocole, dans le but de spécifier les protocoles utilisés entre entités physiques distinctes;

b) par les concepteurs des commutateurs (et d'autres nœuds), dans le but de spécifier les prescriptions fonctionnelles pour ces nœuds;

c) par les planificateurs de réseau.

La présente Recommandation décrit en détail les six étapes de la phase 2. La succession de ces étapes représente un déroulement théorique idéal de l'application de la méthode; la mise en œuvre pratique peut toutefois conduire à un certain nombre d'itérations avant la définition complète des informations produites par la phase 2. L'Appendice I contient un résumé de description de la phase 2 faite en utilisant la méthodologie fonctionnelle unifiée.

Les étapes suivantes varient un peu selon que l'on emploie la démarche utilisant les modules SIB ou la méthodologie UML qui est orientée objet. La plupart des aspects de l'étape 1 sont les mêmes, indépendants de la démarche. Le paragraphe 3 décrit en détail les méthodes UML et donne des exemples où il convient. Un exemple pratique de l'utilisation de la méthodologie UML est donnée à l'Appendice III.

2 Etapes de la méthode

2.1 Etape 1 – Elaboration du modèle fonctionnel

Un modèle fonctionnel est élaboré pour chaque capacité de service de base, de service complémentaire et de service réseau. Le modèle fonctionnel est mis en correspondance, dans chacun des cas, avec les besoins et les caractéristiques du service concerné. Le modèle fonctionnel utilisé dans la description d'un service en phase 2 identifie des entités fonctionnelles et leurs relations mutuelles. L'affinement du modèle fonctionnel initial est fait par l'élaboration et/ou l'itération des étapes 2 à 6 décrites ci-dessous. Le modèle fonctionnel définitif représente le résultat final de la phase 2.

Puisque la présente Recommandation permet à l'utilisateur de choisir soit la démarche SIB soit la démarche orientée objet, il est prévu une alternative pour les étapes 2 à 4. La Figure 2.1 indique les étapes qui doivent être suivies au cours des deux démarches.

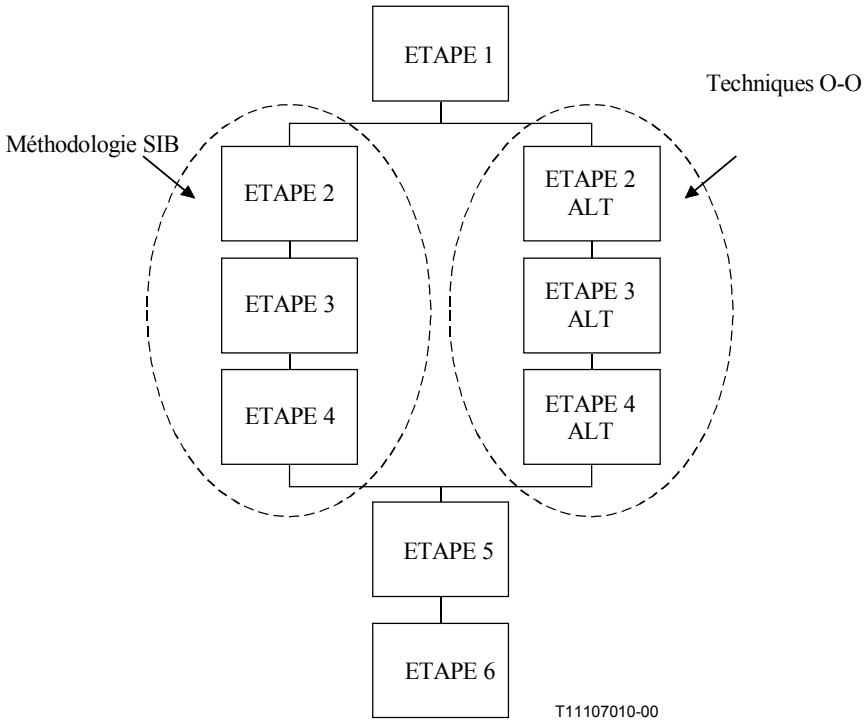


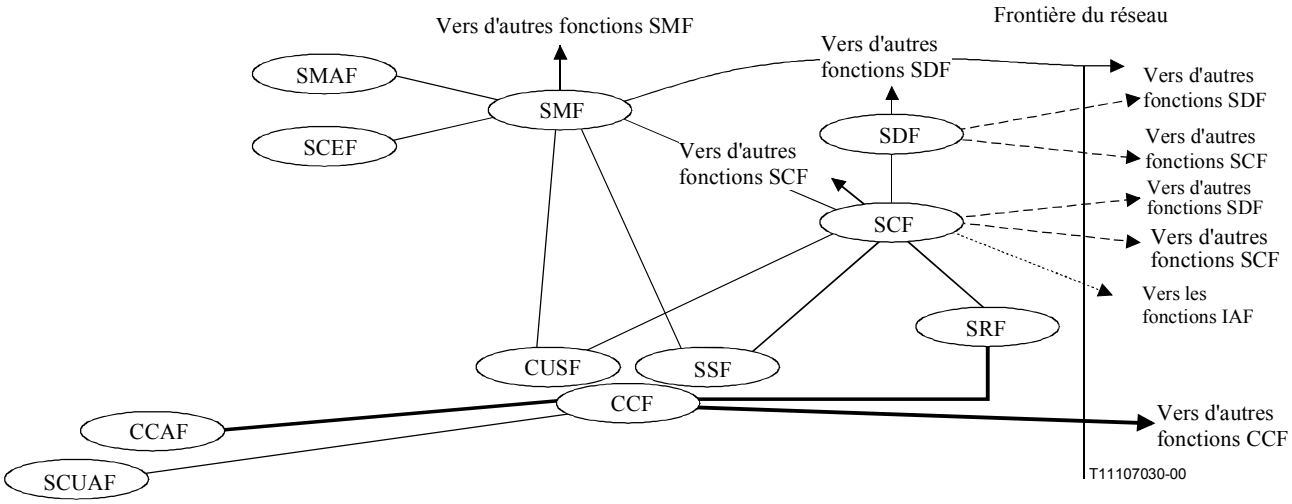
Figure 2.1/Q.65 – Emploi des étapes 1 à 6

2.1.1 Modèle fonctionnel unifié

Le modèle fonctionnel unifié de la méthodologie de la phase 2 tient compte des entités fonctionnelles communes, des relations communes et des flux d'information communs pour les réseaux RNIS, RI, RNIS-LB, IMT-2000 et RGT.

La Figure 3 présente une instance de modèle fonctionnel unifié élaboré à partir de l'ensemble des entités fonctionnelles spécifiées par la méthodologie fonctionnelle unifiée. Le modèle fonctionnel identifie et nomme les entités fonctionnelles individuelles ainsi que leurs types. Il identifie également les relations et types de relations entre entités fonctionnelles communicantes. Les entités fonctionnelles sont représentées par des cercles et la relation entre deux entités fonctionnelles communicantes est identifiée par une ligne les connectant.

NOTE – Le modèle de la Figure 3 est un modèle composite basé sur l'appel de base, l'appel CS-2 (Q.71) du RI, le réseau IMT-2000 et le RNIS-LB. Ce modèle est présenté comme une base à partir de laquelle peut être édifié un modèle fonctionnel de la phase 2 spécifique, dont le but est la prise en charge d'une capacité donnée de service ou de réseau. L'Appendice II présente une vision actuelle de l'intégration entre le RI et le RNIS-LB, avec une séparation totale des commandes d'appel et de connexion. Ce type de modèle constituera la base de la prochaine version de la présente Recommandation.



CCAF	fonction d'agent de commande d'appel	CCF	fonction de commande d'appel	SSF	fonction de commutation de services
SCF	fonction de commande de service	IAF	fonction d'accès intelligent	SDF	fonction de données de service
SRF	fonction de ressource spécialisée	SCUAF	fonction d'accès utilisateur de commande de services		

Figure 3/Q.65 – Le modèle fonctionnel unifié

2.1.2 Entités fonctionnelles

Les entités fonctionnelles sont issues, au départ, d'une réflexion globale concernant les fonctions de réseau nécessaires pour prendre en charge le service. Les entités fonctionnelles sont définies comme suit:

– une entité fonctionnelle est un regroupement de fonctions, localisées en un point unique et constituant un sous-ensemble de l'ensemble total des fonctions fournissant le service;

– une entité fonctionnelle est décrite sous la forme de la commande d'une instance de service (par exemple d'un appel ou d'une connexion);

– une entité fonctionnelle est visible pour les autres entités fonctionnelles qui doivent communiquer avec elle en vue de fournir un service (c'est-à-dire que les entités fonctionnelles sont des entités adressables par le réseau);

– un modèle fonctionnel peut contenir des entités fonctionnelles de types différents. Le type d'une entité fonctionnelle est caractérisé par le regroupement particulier des fonctions qui la composent. Il s'ensuit que deux entités fonctionnelles ou plus sont du même type si elles sont constituées des mêmes regroupements de fonctions;

– un type d'entité fonctionnelle distinct est en général défini pour tout regroupement différent de fonctions pouvant être réparties sur des équipements physiques différents. Toutefois, lorsqu'il existe un grand nombre de fonctions communes à plusieurs regroupements obligatoires, il peut être commode de définir ces derniers comme sous-ensembles d'un type unique plutôt que comme des types distincts;

– les entités fonctionnelles sont élaborées pour chaque service de base et pour chaque service complémentaire. Le même type d'entité fonctionnelle peut apparaître plus d'une fois dans un modèle fonctionnel et peut également apparaître dans le modèle de plus d'un service.

Les définitions suivantes concernent les entités fonctionnelles utilisées dans le modèle fonctionnel unifié. Un grand nombre de ces entités fonctionnelles proviennent d'études faites dans le cas du RI pour les services support au sein du réseau.

2.1.2.1 fonction CCA (CCAF, *CCA function*)

La fonction CCAF est la fonction d'agent de commande d'appel (CCA, *call control agent*) qui fournit l'accès pour des utilisateurs. Elle constitue l'interface entre les fonctions utilisateur et les fonctions de commande d'appel réseau. La fonction CCAF remplit les fonctions suivantes:

a) fourniture de l'accès utilisateur par interaction avec l'utilisateur dont l'instance de service ou l'appel doit être établi, maintenu et libéré;

b) accès aux capacités de la fonction de commande d'appel (CCF, *call control function*) qui fournissent le service, en utilisant les demandes de service (par exemple d'appel, de transfert, de maintien, etc.) pour l'établissement, le traitement et la libération d'un appel ou d'une instance de service;

c) réception des indications issues de la fonction de commande d'appel (CCF) en relation avec l'appel ou le service et en les relayant, si nécessaire, vers l'utilisateur;

d) gestion d'une information d'état de l'appel/du service telle qu'elle est perçue par l'entité fonctionnelle;

e) interface en cas de besoin avec la fonction d'accès utilisateur à la commande de service (SCUAF, *service control user access function*) pour un service non lié à un appel.

2.1.2.2 fonction CC (CCF, *CC function*)

La fonction CCF est la fonction de commande d'appel (CC, *call control*) au sein du réseau qui fournit la commande et le traitement d'appel/de service. Elle remplit les fonctions suivantes:

a) établir, traiter et libérer les appels ou les connexions comme "demandé" par la fonction SCUAF;

b) fournir la capacité d'associer et de mettre en relation des entités fonctionnelles de la fonction SCUAF qui sont impliquées dans une instance particulière d'appel, de connexion ou des deux (pouvant résulter de demandes de la fonction CUSF);

c) gérer les relations entre les entités fonctionnelles de la fonction SCUAF impliquées dans un appel (par exemple en supervisant les caractéristiques globales de l'instance de l'appel, de la connexion ou des deux);

d) fournir des mécanismes d'activation de l'accès à des fonctions du RI (par exemple en retransmettant des événements vers les fonctions SSF/CUSF).

e) gérer les données de ressource des appels de base (par exemple, les références d'appel).

2.1.2.3 fonction SS (SSF, *SS function*)

La fonction SSF est la fonction de commutation de services qui fournit, en coopération avec la fonction CCF, un ensemble de fonctions nécessaires à l'interaction entre la fonction CCF et une fonction de commande de services (SCF, *service control function*), et est associée à la fonction NCSF, si nécessaire, pour le traitement de services non liés à un appel. Elle remplit les fonctions suivantes:

a) étendre la logique de traitement de la fonction CCF afin d'y inclure la reconnaissance de déclencheurs de commande de service et d'interagir avec la fonction SCF;

b) gérer la signalisation entre la fonction CCF et la fonction SCF;

c) modifier les fonctions de traitement d'appel/de connexion (au sein de la fonction CCF) comme nécessaire pour traiter, sous le contrôle de la fonction SCF, des demandes de services fournis par le RI;

d) assurer l'interface avec la fonction CUSF pour le traitement d'interactions non liées à un appel;

e) prendre en charge le cas de relais, dans lequel elle assure le transfert d'informations entre les fonctions SCF et SRF en utilisant éventuellement les capacités d'interaction avec l'utilisateur liée à l'appel hors canal (OCCRUI, *out-channel call related user interaction*).

2.1.2.4 fonction SC (SCF, *SC function*)

La fonction SCF est une fonction qui gère les fonctions de commande d'appel dans le traitement des demandes de service fourni par le R1 ou des demandes de service client. La fonction SCF peut interagir avec d'autres entités fonctionnelles pour accéder à des logiques de traitement supplémentaires ou pour obtenir des informations (de service ou d'utilisateur) nécessaires au traitement d'une instance logique d'appel/de service. Elle remplit les fonctions suivantes:

a) assurer l'interface et l'interaction avec les entités fonctionnelles telles que la fonction de commutation de services/commande de service, la fonction de ressource spécialisée (SRF, *specialized resource function*), la fonction données de service (SDF, *service data function*), d'autres fonctions de commande de service (SCF, *service control function*), les fonctions d'accès intelligent (IAF, *intelligent access function*) et la fonction de services non liés à un appel (CUSF, *call unrelated service function*);

b) contenir la logique et les capacités de traitement nécessaires à la prise en charge des tentatives pour un service fourni par le RI, pouvant ou non être liées à un appel;

c) assurer l'interface et l'interaction d'une manière sécurisée avec d'autres fonctions SCF pour la commande répartie de service et les notifications de service non sollicitées. La commande répartie de service a pour effet de transférer l'exécution de la logique de service entre deux fonctions SCF;

d) assurer l'interface et l'interaction avec des fonctions SDF pour l'acquisition sécurisée de données et leur manipulation;

e) mise à la disposition d'un point d'interconnexion avec le réseau à des fins d'interfonctionnement de réseau en masquant d'une manière efficace la structure spécifique du réseau;

f) assurer l'interface et l'interaction avec la fonction SRF pour des interactions non liées à un appel, en lui indiquant le scénario d'interaction utilisateur à exécuter, en lui fournissant les informations supplémentaires dont elle a besoin au cours de l'exécution de ce scénario et en attendant la fin de son exécution;

g) assurer l'interface et l'interaction avec la fonction SRF pour les interactions non liées à un appel, en supervisant la disponibilité des ressources de la fonction SRF pour la commande de certaines de ces ressources en dehors du contexte d'un appel;

h) fournir, à des fins d'interfonctionnement de réseaux, des mécanismes de sécurité permettant le transfert sécurisé d'informations à travers les frontières entre les réseaux.

2.1.2.5 fonction SD (SDF, *SD function*)

La fonction SDF contient des informations de client et de réseau accessibles en temps réel par la fonction SCF pendant l'exécution d'un service fourni par le RI. Elle remplit les fonctions suivantes:

a) assurer l'interface et l'interaction avec des fonctions SCF pour la manipulation et l'acquisition sécurisée de données au moyen d'interrogations simples de base de données ou de scénarios de gestion de données;

b) assurer s'il y a lieu l'interface et l'interaction avec d'autres fonctions SDF, en permettant de masquer l'emplacement des données dans le réseau. Ceci peut être utilisé pour assurer la transparence de la répartition des données (par exemple, vis-à-vis de la fonction SCF);

c) fournir, à des fins d'interfonctionnement de réseaux, des mécanismes de sécurité permettant le transfert sécurisé d'informations à travers des frontières entre réseaux;

d) assurer l'interface et l'interaction avec d'autres entités SDF afin de permettre la copie des données et leur regroupement avec les droits d'y accéder;

e) fournir des fonctionnalités d'authentification et de contrôle d'accès afin d'assurer un accès sécurisé aux données de service;

f) faciliter la coopération dans la gestion du trafic, afin d'éviter ou de résoudre des situations d'encombrement lors de l'acquisition de données;

g) fournir la prise en charge de données pour les services de sécurité. Cette prise en charge peut être utilisée par la fonction SDF elle-même pour la gestion sécurisée des données;

h) faciliter l'intervention d'un mécanisme robuste de récupération pour la copie de données (par exemple dans le cas d'indisponibilité de la fonction SDF);

i) fournir des scénarios d'accès aux données (méthodes) pouvant être appelés par la fonction SCF, afin de simplifier le transfert d'informations à travers l'interface SCF-SDF. Ce type de scénarios d'accès aux données permet effectivement une manipulation simplifiée des données d'une entrée. La fonction SCF continue à assurer les fonctions logiques de traitement propre au service et les fonctions de pilotage de la commande d'appel pour la fonction SSF.

NOTE – La fonction SDF contient des données liées à la fourniture ou à l'exploitation de services fournis par le RI. Elle n'englobe donc pas nécessairement les données fournies par des tiers, telles que des informations de carte de crédit, mais peut fournir l'accès à de telles données.

2.1.2.6 fonction SR (SRF, *SR function*)

La fonction SRF fournit les ressources spécialisées nécessaires à l'exécution de services fournis par le RI (par exemple les récepteurs numériques, les annonces vocales, les circuits de conférence, etc.). Elle remplit les fonctions suivantes:

a) assurer l'interface et l'interaction avec les fonctions SCF et SSF (ainsi qu'avec la fonction CCF);

b) contenir éventuellement la logique et les capacités de traitement pour la réception/l'émission et la conversion d'informations en provenance des utilisateurs;

c) contenir éventuellement des fonctionnalités comparables à la fonction CCF, afin de gérer les connexions support des ressources spécialisées.

2.1.2.7 fonction SRF de reconnaissance automatique de la parole (ASR, *automatic speech recognition*)

La ressource ASR permet à l'utilisateur de services RI d'introduire des commandes et des données à l'aide de sa voix. La reconnaissance ASR peut, ou non, dépendre du locuteur. Dans le cas de reconnaissance automatique de parole dépendante du locuteur, il est nécessaire de fournir un mécanisme permettant à l'utilisateur de gérer directement ses modèles de signal vocal nécessaires à la reconnaissance des commandes et des données; un tel mécanisme doit permettre à l'utilisateur de revoir, mettre à jour, supprimer et insérer:

– les modèles de signaux vocaux;

– la correspondance entre les modèles et le format interne utilisé par l'entité SRF pour les signaux vocaux reconnus (par exemple, entre l'entrée vocale d'un nom et la chaîne correspondante de caractères ASCII).

Ce mécanisme peut soit être piloté par la fonction SCF, soit être réalisé directement par l'entité SRF sans intervention de la fonction SCF. Dans ce dernier cas, l'entité SRF fournit à la fonction SCF, sur demande de cette dernière, une information au sujet du résultat de l'opération. La ressource ASR de base doit fournir la possibilité de reconnaissance de mots isolés (c'est-à-dire les dix chiffres et un certain nombre de commandes de base telles que "oui" ou "non", prononcés dans la langue du fournisseur du réseau local au moins) d'une manière indépendante du locuteur à travers le RTPC.

Considérant qu'une reconnaissance ASR multilingue peut également être utile, on a admis que la fonction SRF doit pouvoir traiter l'indication de la langue utilisée pour l'entrée vocale, de la même manière que pour la génération d'annonces décrites ci-dessus.

2.1.2.8 fonction SRF de texte vers la parole (TTS, *text to speech*)

La fonction SRF peut fournir une fonction de texte vers la parole. Cette fonction est constituée de deux fonctions logiques. La première convertit le texte d'entrée vers une représentation phonétique prosodique. La deuxième fonction produit le signal vocal synthétisé ainsi que le traitement et l'assemblage des éléments vocaux.

2.1.2.9 fonction IA (IAF, *IA function*)

La fonction d'accès intelligent (IAF, *intelligent access function*) fournit un accès entre la fonction SCF d'un réseau à structure de RI et une entité qui n'est pas un réseau à structure de RI. Cette dernière entité peut être un autre réseau ou un abonné (réseau privé, base de données simple utilisée par exemple dans le service de contrôle de continuité, terminal ou autocommutateur privé). Elle remplit les fonctions suivantes:

a) fournir un accès, dans les deux directions, pour l'entité SCF du réseau à structure de RI;

b) effectuer le mappage des informations entre la représentation interne et la représentation externe;

c) être située dans l'entité qui n'est pas le réseau à structure de RI;

d) assurer des capacités de passerelle englobant les fonctionnalités de sécurité et de coupe-feu.

2.1.2.10 fonction CUS (CUSF, *CUS function*)

La fonction CUSF est le service indépendant de l'appel (CUS, *call-unrelated service*) qui, en association avec les fonctions CCF et SSF, assure un ensemble de fonctions de service non liées à un appel pour une interaction hors canal avec une fonction SCUAF. Elle assure également l'ensemble de fonctions nécessaires pour les interactions entre les fonctions SCUAF et SCF. Elle remplit les fonctions suivantes:

a) étendre la logique de la fonction CCF de manière à inclure la reconnaissance du déclenchement de commandes de service et à interagir avec la fonction SCF;

b) gérer la signalisation entre les fonctions CCF et SCF;

c) modifier les fonctions de traitement d'association ou de connexion (dans la fonction CCF), comme nécessaire pour le traitement de demandes d'utilisation de service fourni par le RI sous la supervision de la fonction SCF;

d) modifier les fonctions de traitement d'interaction non liée à un appel (dans la fonction CUSF), comme nécessaire pour le traitement de demandes d'utilisation de service fourni par le RI sous la supervision de l'entité SCF;

e) prendre en charge une interaction non liée à un appel, pouvant être initialisée par l'utilisateur ou par l'entité SCF;

f) assurer l'interface avec la fonction SSF pour le traitement des interactions liées à un appel.

2.1.2.11 fonction SCUA (SCUAF, *SCUA function*)

La fonction SCUAF est la fonction d'agent utilisateur de commande de service (SCUA, *service control user agent*) qui fournit l'accès pour les utilisateurs. Elle constitue l'interface entre un utilisateur et la fonction de service non lié à un appel (CUSF). Elle remplit les fonctions suivantes:

a) fournir l'accès utilisateur en interagissant avec celui-ci pour établir, maintenir et libérer, comme nécessaire, une instance de service non lié à un appel;

b) accéder aux capacités de fourniture de service de la fonction de commande d'appel (CCF) en utilisant les demandes de service (par exemple, l'établissement, l'enregistrement d'un lieu) pour l'établissement, pour le traitement et pour la libération d'une association ou d'une instance de service;

c) recevoir les indications relatives au service non lié à un appel en provenance de la fonction CCF et les transmettre, comme nécessaire, à l'utilisateur;

d) gérer l'information d'état du service, telle qu'elle est perçue par cette entité fonctionnelle.

NOTE – Les ensembles CS-2 RI ne définissent pas si la fonction SCUAF constitue une abstraction pour une nouvelle entité fonctionnelle de services complémentaires associés à un appel. De même, les ensembles CS-2 RI ne définissent pas quelle est la relation à utiliser pour modéliser une interaction utilisateur liée à un appel (la relation existante entre les fonctions CCAF et CCF ou une relation explicite entre certaines entités fonctionnelles).

2.1.2.12 fonction SM (SMF, *SM function*)

La fonction SMF est la fonction de gestion de service. Le présent paragraphe décrit un certain nombre de fonctions SMF du RI. Ces fonctions peuvent être regroupées en cinq catégories:

1) fonctions de mise en place du service;

2) fonctions de fourniture du service;

3) fonctions de commande de l'exploitation du service;

4) fonctions de facturation;

5) fonctions de supervision du service.

- Les fonctions de mise en place du service sont les suivantes:
  - allocation de scénarios de service:

    cette fonction transmet les scénarios de service, détermine quelle est la partie du réseau concernée par ces scénarios et gère les éléments réseau correspondants;
  - allocation de données génériques de service:

    cette fonction transmet les données génériques de service, détermine quelle est la partie du réseau concernée par ces données et gère les éléments réseau correspondants;
  - introduction et allocation de données de routage de signalisation:

    cette fonction transmet les données de routage de signalisation, détermine quelle est la partie du réseau concernée par ces données et gère les éléments réseau correspondants.

Elle décharge les données de routage de signalisation vers le réseau de signalisation SS7 et détermine quels sont les éléments réseau SS7 concernés par l'allocation de ces données;

– introduction et allocation de données de déclenchement:

cette fonction transmet les données de déclenchement, détermine quelle est la partie du réseau concernée par ces données et gère les éléments réseau correspondants. Elle décharge les données de déclenchement vers le RTPC;

– introduction et allocation de données de ressources spécialisées:

cette fonction transmet les données de ressources spécialisées, détermine quelle est la partie du réseau concernée par ces données et gère les éléments réseau correspondants;

– essais du service:

cette fonction recueille le logiciel de service, en provenance de la fonction d'environnement de création de service, qui doit être chargé dans un réseau RI autonome à des fins de test d'une réalisation d'un nouveau service. Cette fonction charge les données spécifiques du service et des abonnés au service. Elle gère l'exploitation des essais concernés.

• Les fonctions de fourniture du service sont les suivantes:

– introduction et allocation des données spécifiques d'abonné:

cette fonction recueille les données spécifiques de l'abonné du service et les gère dans les bases de données d'abonné et de contrat. La fonction effectue la traduction des données de service et d'abonné en données spécifiques du réseau. Cette fonction détermine quelle est la partie du réseau concernée par les données et gère les éléments réseau correspondants.

• Les fonctions de commande de l'exploitation du service sont les suivantes:

– maintenance du service:

les fonctions de maintenance du service sont les suivantes:

– maintenance logicielle:

la maintenance logicielle consiste à modifier la logique de service (la modification de la logique de service fait partie de la fonction SCEF). L'introduction de scénarios modifiés dans un réseau à structure de RI est faite dans la mise en place du service;

– mise à jour des données génériques du service:

cette fonction transmet les données génériques du service, détermine quelle est la partie du réseau concernée par ces données et gère les éléments réseau correspondants;

– mise à jour des données spécifiques de l'abonné:

cette fonction fournit les fonctions de commande pour les données spécifiques de l'abonné du service et les fonctions d'administration des bases de données d'abonné et de contrat. Cette fonction détermine quelle est la partie du réseau concernée par les données et gère les éléments réseau correspondants;

– mise à jour des données d'acheminement de signalisation:

cette fonction fournit les fonctions de commande pour les données de routage de signalisation, détermine quelle est la partie du réseau concernée par ces données et gère les éléments réseau correspondants. Elle décharge les données de routage de signalisation vers le réseau de signalisation SS7 et détermine quels sont les éléments réseau SS7 concernés par l'allocation de ces données;

– mise à jour des données de déclenchement:

  cette fonction fournit les fonctions de commande pour les données de déclenchement, détermine quelle est la partie du réseau concernée par ces données et gère les éléments réseau correspondants. Elle décharge les données de déclenchement vers le RTPC;

– mise à jour des données de ressources spécialisées:

  cette fonction fournit les fonctions de commande pour les données de ressources spécialisées, détermine quelle est la partie du réseau concernée par ces données et gère les éléments réseau correspondants;

– adaptation de la fonction SMAF:

  la fonction SMAF fournit l'interface entre l'exploitant de réseau ou l'abonné du service et l'entité SMF. Cette interface doit pouvoir faire l'objet d'adaptations concernant les données abonné ou réseau. Un abonné du service peut, par exemple, changer de type de périphérique (données spécifiques d'abonné passant d'un téléphone à touches multifréquences vers un terminal vidéotex). Cette modification de périphérique peut également entraîner une modification dans des options de menu;

– reconfiguration du service:

  cette activité consiste à modifier l'allocation de scénarios de service, de données génériques de service et de données spécifiques de l'abonné. Le motif d'une telle modification peut, par exemple, être une modification de configuration réseau ou une amélioration de performance des services;

– activation ou désactivation du service:

  cette activité fournit à l'exploitant de réseau la possibilité d'activer ou de désactiver d'une manière temporaire tout ou partie d'un service. Un service de vote à distance peut, par exemple, n'être utilisé à des fins de maintenance que dans des intervalles de temps définis;

– suppression du service:

  un service sera retiré de l'exploitation;

– sécurité:

  deux types de sécurité existent pour la fonction SMF: le contrôle d'accès et le contrôle des données. Le contrôle d'accès inclut l'identification, l'authentification et l'autorisation (supervision de la commande) de l'abonné du service et de l'exploitant de réseau. Le contrôle des données inclut la vérification des données entrées par l'abonné du service et par l'exploitant de réseau.

• Les fonctions de facturation sont les suivantes:

– génération et stockage des enregistrements de taxation:

  cette fonction supervise l'utilisation du service et journalise les enregistrements d'appel;

– collecte des enregistrements de taxation:

  cette fonction collecte les enregistrements d'appel et les enregistrements détaillés de gestion, puis procède à leur uniformisation et à leur corrélation. Cette fonction journalise les enregistrements d'appel;

– modification des tarifs:

  cette fonction détermine la structure et le contenu du tarif pour un service nouvellement créé ou les modifie pour un service existant.

- Les fonctions de supervision de service sont les suivantes:
  - démarrage des mesures et collecte des données de mesure:

    cette fonction supervise l'utilisation et les performances du service, ainsi que les performances du réseau. Elle utilise à cet effet des résultats de mesures concernant les sous-systèmes sous-jacents, à savoir, la fonction de gestion du système SS7 et la fonction de gestion réseau;
  - analyse et compte rendu des données de mesure:

    cette fonction analyse l'utilisation et les performances du service. Elle analyse également les résultats de la fonction de démarrage et de collecte de mesures;
  - réception des informations de données de supervision de dérangements:

    cette fonction agit sur la réception des données de supervision de dérangements en provenance des éléments réseau. Les implications et les impacts sur la performance du service seront calculés et une action adéquate sera effectuée.

2.1.3 Relations entre entités fonctionnelles

Les services sont pris en charge par la coopération des actions d'un ensemble d'entités fonctionnelles. Cette coopération nécessite l'établissement de communications:

- tout couple d'entités fonctionnelles communicantes d'un modèle fonctionnel spécifique de service est considéré comme étant en relation;
- toute interaction entre deux entités fonctionnelles communicantes est appelée flux d'information ou communication API (le second cas se réfère à l'utilisation du modèle UML pour décrire le point de référence). La relation entre tout couple d'entités fonctionnelles est constituée de l'ensemble complet des flux d'information ou de communications API entre ces entités. Dans le dernier cas, on nomme cet ensemble l'ensemble API pour ce point de référence;
- si deux entités fonctionnelles communicantes sont localisées sur des équipements physiques distincts, les flux d'information ou les communications API entre elles définissent les besoins de transfert d'information d'un protocole de signalisation entre les équipements;
- des couples d'entités fonctionnelles distincts peuvent posséder des relations de types différents. Le type d'une relation est caractérisé par les ensembles de flux d'information ou communications API entre deux entités fonctionnelles. Par exemple, la relation entre les entités fonctionnelles FE1 et FE2 et la relation entre les entités fonctionnelles FE3 et FE4 sont dites du même type si elles sont constituées du même ensemble de flux d'information ou de communications API;
- des identificateurs de type (par exemple r1, r2, r3, etc.) différents peuvent être attribués aux relations de manière à identifier sans ambiguïté des ensembles de flux d'information au sein du modèle fonctionnel d'un service. Le même type de relation peut figurer plus d'une fois dans un modèle fonctionnel.

2.1.4 Elaboration du modèle fonctionnel

Sur la base des définitions données ci-dessus, le modèle fonctionnel d'un service donné est élaboré en tenant compte des directives et critères suivants:

- des entités fonctionnelles adéquates sont choisies, compte tenu d'une connaissance anticipée des différents types de réalisation de réseau. Toutes les répartitions raisonnables de fonctions doivent être prises en compte, ce qui ouvre à une Administration la possibilité de choisir la manière d'offrir effectivement le service;

– les types de relation sont attribués au départ sur la base d'estimations concernant la nature probable des interactions de chaque couple d'entités fonctionnelles. Il peut être nécessaire de réviser le modèle initial compte tenu de définitions plus détaillées des actions d'entités fonctionnelles, des flux d'information ou des communications API et du domaine d'allocation physique des entités fonctionnelles;

– le modèle de certains services peut nécessiter un certain nombre de duplications d'une entité fonctionnelle (par exemple dans le cas de fonctions tandem). Le modèle fonctionnel doit se limiter à la description des duplications jusqu'au point où aucune nouvelle combinaison entre relations externes et entités fonctionnelles n'est mise en évidence par une nouvelle répétition. Il s'ensuit qu'une entité fonctionnelle unique peut représenter de multiples entités physiques tandem fournissant la même fonction.

Le modèle fonctionnel unifié est prévu pour prendre en considération des caractéristiques du plus grand nombre possible d'architectures réseau. Dans le cas du RNIS, les fonctionnalités peuvent être réparties entre les fonctions spécifiées, bien qu'un grand nombre des entités fonctionnelles du modèle sont localisées au même endroit dans l'implémentation physique. Dans le cas du RI et du réseau IMT-2000, le modèle fonctionnel et les entités fonctionnelles sont représentatives de l'ensemble de capacités du RI et des architectures du réseau IMT-2000 (voir UIT-T Q.1711). Dans le cas du RNIS-LB, le modèle fonctionnel unifié identifie des relations de bord à bord entre les fonctions de commande d'appel serveuses, et des relations de lien à lien utilisant les fonctions de commande d'appel de transit. Les configurations de point à multipoint sont prises en considération, en fonction des besoins, par l'identification d'instances supplémentaires d'entités fonctionnelles similaires et de relations. Pour les réseaux IP, le modèle fonctionnel fournit un lien avec les objets gérés qui peuvent être considérés comme existant au sein des entités fonctionnelles et auxquels, en tant que tels, on peut s'adresser et accéder par ces entités fonctionnelles. Le modèle fonctionnel permet aussi d'accéder aux fonctions qui assurent le mappage de protocole pouvant être nécessaire entre les réseaux RI et IP sous la forme de fonctions passerelles.

2.1.5 Relations entre les modèles de service de base et de service complémentaire

Le modèle fonctionnel d'un service complémentaire correspondra à un modèle de service de base. Les directives suivantes doivent être utilisées lorsqu'est fait le choix entre, d'une part, l'implantation d'entités fonctionnelles liées à un service complémentaire au même lieu que les entités fonctionnelles du service de base, et d'autre part, la définition de nouvelles entités fonctionnelles délocalisées:

– un regroupement de fonctions au sein d'un modèle de service complémentaire doit être localisé au même lieu qu'une entité fonctionnelle de service de base (voir, par exemple, la Figure 5) s'il modifie un objet (par exemple la connexion d'appel) se trouvant sous la commande du service de base;

– une entité fonctionnelle qui n'est pas localisée au même lieu qu'une entité fonctionnelle de service de base ne doit pas, en règle générale, avoir besoin d'une information détaillée sur l'état de l'appel/de la connexion. Une entité fonctionnelle distincte peut également être caractérisée par le fait d'être engagée dans une relation transactionnelle avec une entité fonctionnelle du service complémentaire localisée au même lieu qu'une entité fonctionnelle du service de base (par exemple pour fournir la traduction du numéro à l'entité fonctionnelle du service de base);

– une relation entre les entités fonctionnelles pour le service complémentaire est définie si, et seulement si, les entités fonctionnelles du service complémentaire sont nécessaires pour la communication et que ces besoins de communication ne sont pas satisfaits par les flux d'information définis au sein du service de base;

– une relation entre le modèle de service complémentaire et le modèle du service de base peut être élaborée à partir de la comparaison des modèles. Les entités fonctionnelles du service complémentaire doivent tenir compte de tous les scénarios définis dans l'étape 6, aussi bien pour le modèle du service de base que pour le modèle du service complémentaire.

La Figure 4 illustre ces relations.

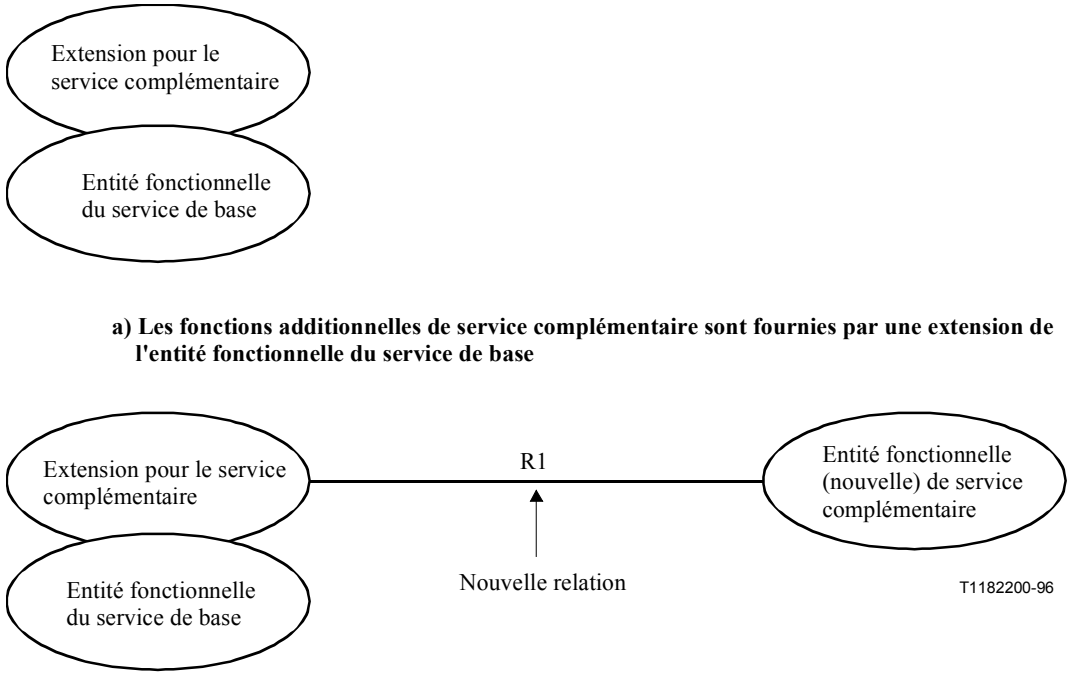


Figure 4/Q.65 – Variantes pour l'addition de fonctions de service complémentaire au modèle fonctionnel du service de base

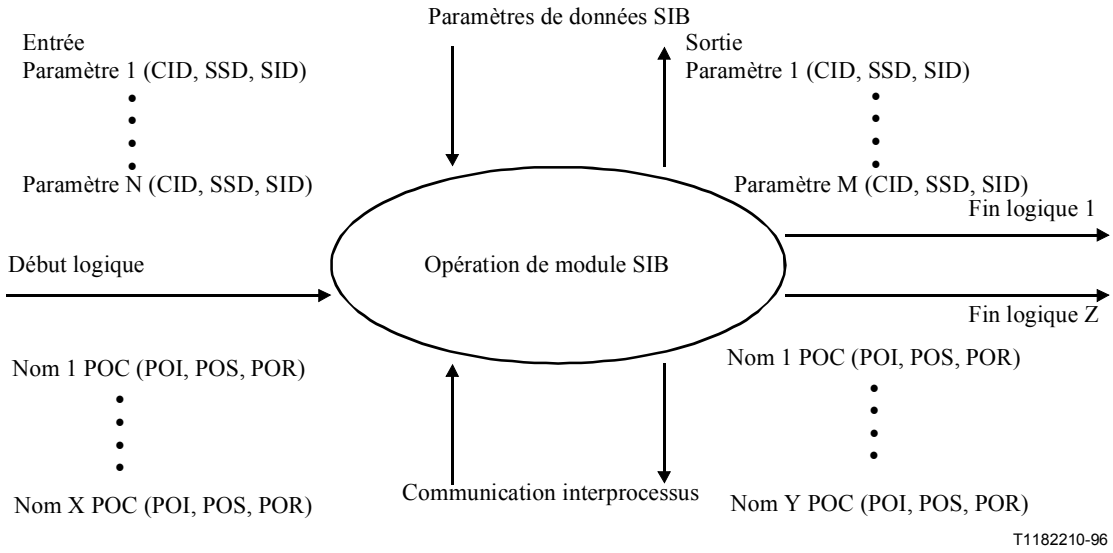


Figure 5/Q.65 – Représentation graphique d'une opération de module SIB

2.2 Etape 2 (facultative) – Description des fonctions d'un service au moyen d'un module SIB (voir 3.1 pour l'alternative de l'étape 2 employant les techniques orientées objet)

Modules indépendants du service

Un module SIB est une capacité réutilisable normalisée pour l'ensemble du réseau, résidant dans le plan fonctionnel global, et utilisée pour créer des fonctions du service (voir la Figure 1). Les modules SIB sont de nature globale. Leur réalisation n'est pas traitée à ce niveau, mais peut être trouvée dans les descriptions du plan fonctionnel réparti (DFP, *distributed functional plane*) et du plan physique. Les modules SIB sont réutilisables et peuvent être combinés pour réaliser les services décrits dans le plan de service. Les capacités fournies au sein du module SIB sont décrites par l'ensemble de fonctions qui peuvent y être appelées. L'ensemble des opérations offertes par un module SIB constitue l'interface de ce bloc. Toute opération définit une fonction qui peut être réalisée en relation avec la capacité du module SIB. Les modules SIB complexes, par exemple ceux qui modélisent des activités persistantes, sont définis à partir d'un certain nombre d'opérations qui permettent le pilotage de l'activité du module SIB. Les modules SIB sont définis de manière à être indépendants d'un service spécifique et de la technologie qui les prendra en charge ou avec laquelle ils seront réalisés.

Caractéristiques des modules SIB

- Les modules SIB sont des blocs de construction monolithiques (les détails de leur implémentation sont cachés) que le concepteur de services utilisera pour élaborer de nouveaux services.
- Toutes les fonctions du service (SF, *service features*) sont décrites par un module SIB ou une combinaison de modules SIB.
- Toute fonction du service peut être définie par un nombre fini de modules SIB.
- Un module SIB définit une activité complète.
- Un module SIB est réalisé dans le plan fonctionnel réparti par des actions d'entité fonctionnelle qui peuvent résider dans une ou plusieurs entités fonctionnelles.
- Une opération de module SIB possède un seul point logique d'entrée et un ou plusieurs points logiques de retour. Les données nécessaires à chaque opération de module SIB sont définies par les paramètres de données de prise en charge d'opération de module SIB et par des paramètres de données d'instance d'appel.
- Les modules SIB sont de nature globale et leurs localisations n'ont pas à être prises en considération, étant donné que le réseau dans sa totalité est considéré comme une entité unique dans le plan fonctionnel global.
- Les modules SIB sont réutilisables. Ils sont utilisés pour d'autres services sans subir de modification.

Le présent sous-paragraphe est conçu pour servir d'introduction pour l'utilisation des modules indépendants du service (SIB) dans la méthodologie fonctionnelle unifiée (UFM) à des fins de description de services. Le concept de module SIB a été adopté aussi bien pour répondre aux besoins de création de services que pour introduire des blocs de diagrammes de flux, de diagrammes SDL et d'actions d'entités fonctionnelles qui sont réutilisables et peuvent être catalogués. L'adjonction du concept de module SIB a été faite afin d'alléger le travail de définition de services de la phase 2. De nouveaux modules SIB seront créés en fonction des besoins et une bibliothèque de modules SIB sera gérée comme référence lors de la création et de la définition de services.

NOTE – L'utilisation des modules SIB pour les descriptions de l'étape 2 ne se limite pas aux services pris en charge par un RI.

La manière de définir et d'utiliser les modules SIB dans le modèle conceptuel du réseau intelligent fournit la base pour une méthode souple de description et de création de services, ainsi qu'une méthode de référence à des diagrammes de flux et à des diagrammes SDL prédéfinis. Le choix du concept de module SIB [tel qu'il est référencé dans les UIT-T Q.1213 et Q.1214 (1995)] constitue un ajout majeur à la méthodologie de la phase 2 décrite dans la présente Recommandation.

Un aspect commun de la méthodologie unifiée est que les modules SIB sont décomposés en actions d'entités fonctionnelles, en diagrammes SDL et en flux d'information. L'attrait de cette méthode réside dans la possibilité de réutilisation (les modules SIB correspondent à des actions d'entités fonctionnelles prédéfinies, décrites par des flux d'information et des diagrammes SDL prédéfinis). Il ne peut être démontré en toute rigueur que des modules SIB peuvent être combinés de toutes les manières possibles pour créer de nouveaux services, mais différentes méthodes sont en cours d'examen (par exemple, l'extension du processus de diagrammes SDL en vue d'y inclure la combinaison de modules SIB). La vérification des modules SIB existants (c'est-à-dire la vérification que chacun d'eux est défini de manière correcte) est possible par l'analyse des diagrammes SDL qui les définissent. La vérification de combinaisons de modules SIB est possible en analysant les diagrammes SDL créés lorsque des modules SIB sont assemblés en vue de décrire des fonctionnalités de services et des capacités de réseau.

Il existe un besoin de nouveaux modules SIB ainsi que d'élaboration de règles de combinaison et de règles d'application. Les modules SIB doivent être créés de haut en bas en partant des services cibles, mais également de bas en haut en partant de l'expérience acquise avec la méthodologie de description de service. La méthodologie peut, de cette manière, répondre aux besoins existants, mais est également capable d'évoluer pour prendre en compte des extensions et de nouvelles technologies. Grâce à un effort continu de définition de modules SIB et de règles d'application, les modules SIB sont susceptibles de devenir l'outil principal de la phase 2 pour la description et la création de services.

2.2.1 Définition de module SIB

Terminologie:

BCP	processus d'appel de base (*basic call process*)
CID	données d'instance d'appel (*call instance data*)
GSL	logique de traitement de service global (*global service logic*)
HLSIB	module SIB de haut niveau (*high level SIB*)
POC	point de commande (*point of control*)
POI	point de lancement (*point of initiation*)
POR	point de retour (*point of return*)
POS	point de synchronisation (*point of synchronization*)
SID	données d'instance de service (*service instance data*)
SSD	données de prise en charge de service (*service support data*)

2.2.1.1 module SIB de processus d'appel de base

Le processus d'appel de base (BCP, *basic call process*) est responsable de fourniture de la continuité de l'appel de base entre des participants du réseau. Le processus d'appel de base peut être considéré comme un processus spécialisé de service fournissant des capacités de traitement d'événements pour l'appel de base, mais également comme un module SIB spécialisé fournissant un ensemble d'opérations SIB telles que:

- la connexion d'appels avec disposition appropriée;

- la déconnexion d'appels avec disposition appropriée;
- la mémorisation de l'identificateur de circuit à des fins de traitement ultérieur de cette instance d'appel.

2.2.1.2 module SIB de haut niveau (HLSIB, *high level SIB*)

Les modules SIB de haut niveau sont, comme des modules SIB normaux, des parties réutilisables d'une fonction de service; ils se composent d'opérations SIB et d'autres modules SIB de haut niveau pouvant être exécutés de manière séquentielle. Les modules SIB de haut niveau possèdent les caractéristiques supplémentaires suivantes:

- les modules SIB de haut niveau ne peuvent contenir que d'autres modules SIB de haut niveau et des opérations SIB;
- un module SIB de haut niveau donné ne peut être utilisé comme composant au sein du même module SIB, c'est-à-dire qu'une utilisation récursive n'est pas possible;
- les modules SIB de haut niveau situés au niveau le plus bas ne contiennent que des opérations SIB, c'est-à-dire qu'aucun autre détail ne sera visible au niveau du plan fonctionnel global;
- un seul des modules SIB (de haut niveau) sera le premier à être exécuté dans un module SIB de haut niveau, de sorte que les modules SIB de haut niveau n'ont qu'un seul point d'entrée (début logique), comme c'est le cas pour des modules SIB normaux. De même, un module SIB de haut niveau peut avoir un ou plusieurs points de retour (fins logiques).

2.2.1.3 logique de traitement de service global

La logique de traitement de service global (GSL, *global service logic*) peut être considérée comme la "colle" qui définit l'ordre dans lequel sont chaînées des opérations SIB, en vue de construire des processus de service permettant de réaliser les fonctions du service. Toute instance de la logique de traitement de service global est, d'une manière potentielle, propre à un appel donné, mais elle utilise les éléments communs spécifiques suivants:

- points d'interaction (points d'entrée, de synchronisation et de retour) des processus de service, y compris le processus de commande d'appel de base;
- opérations SIB;
- connexions logiques entre opérations SIB ainsi qu'entre opérations SIB et points d'interaction de processus de service;
- paramètres de données d'entrée et de sortie, données de prise en charge de service et données d'instance d'appel définies pour chaque module SIB.

La logique de traitement de service global "chaînera" ces éléments l'un à l'autre, compte tenu de leurs fonctionnalités, en vue de fournir un service spécifique.

2.2.2 Paramètres de données de module SIB

Les modules SIB sont, par définition, indépendants du service ou de la fonction du service pour la représentation duquel ils sont utilisés. Ils n'ont aucune connaissance au sujet d'autres modules SIB utilisés pour décrire la fonction du service.

Certains éléments dépendant du service sont nécessaires pour décrire les fonctions du service au moyen de ces modules SIB génériques.

Une dépendance du service peut être décrite au moyen de paramètres de données permettant l'adaptation d'un module SIB afin de réaliser la fonctionnalité souhaitée. Les paramètres de données sont spécifiés indépendamment de tout module SIB et mis à la disposition du module SIB par la logique de traitement de service global.

Les paramètres de données se constituent de paramètres d'entrée et de paramètres de sortie. Toute opération de module SIB nécessite deux types généraux de paramètres: des paramètres dynamiques appelés données d'instance d'appel (CID, *call instance data*) et des paramètres statiques appelés données de prise en charge de service (SSD, *service support data*) et données d'instance de service (SID, *service instance data*).

La distinction entre les paramètres formels de module SIB et les paramètres effectifs permet une plus grande flexibilité dans l'attribution du type de données du paramètre SIB. Les paramètres SIB formels sont ceux qui seront utilisés pour les descriptions de module SIB dans la présente Recommandation. Les paramètres SIB effectifs n'apparaissent que dans une instance de module SIB au sein d'une logique de traitement de service global (GSL) spécifique.

2.2.2.1 données d'instance d'appel (CID, *call instance data*)

Les données d'instance d'appel définissent des paramètres dynamiques dont les valeurs diffèrent pour chaque instance d'appel. Elles sont utilisées pour spécifier des détails propres à l'abonné, tels que l'information de ligne appelante ou l'information de ligne appelée. L'origine de ces données peut être la suivante:

- fourniture par le processus d'appel de base (par exemple information de ligne appelante);
- génération par une opération de module SIB (par exemple un numéro traduit); ou
- entrée par l'abonné (par exemple un numéro composé par l'abonné ou son code PIN).

2.2.2.2 données de prise en charge de service (SSD, *service support data*)

Les données de prise en charge de service définissent des paramètres de données nécessaires à une opération de module SIB et qui sont propres à une fonction du service. Lorsqu'une opération de module SIB figure dans la logique globale de service d'une description de service, la logique globale de service spécifiera des valeurs de données de prise en charge de service pour le module SIB. Les données de prise en charge de service sont des paramètres fixes. Il s'agit de paramètres de données qui ont la même valeur pour toutes les instances d'appel. Par exemple, la donnée de prise en charge de service "indicateur de fichier" utilisée par le module SIB traducteur doit être spécifiée d'une manière unique pour toute occurrence de ce module SIB dans une fonction de service donnée. On dit que la donnée de prise en charge de service "indicateur de fichier" est fixe dans la mesure où sa valeur est déterminée par la description du service ou de la fonction de service et non par l'instance d'appel.

Si un service ou une fonction de service utilise des occurrences multiples d'un même module SIB, les paramètres de données de prise en charge de service sont définis d'une manière unique pour toutes les occurrences.

2.2.2.3 données d'instance de service (SID, *service instance data*)

Les données d'instance de service définissent des données en relation avec un profil d'abonné au service qui existent avant l'invocation du service et qui peuvent être modifiées ou mises à jour en fonction de l'activité de traitement du service. Ce type de données peut être lu pendant l'exécution du service et stocké en vue d'une utilisation lors d'appels ultérieurs du service.

2.2.3 Conventions de modélisation de module SIB

Représentation graphique

La description des opérations exécutées par un module SIB utilise une représentation graphique illustrée par la Figure 6. Toute opération de module SIB est caractérisée par des paramètres d'entrée et de sortie, un flux logique d'entrée et un ou plusieurs flux logiques de sortie. Ces flux logiques sont indiqués par des flèches pleines dans les parties gauche et droite du diagramme. La spécification de chaque flux est indiquée sous la flèche correspondante. Les paramètres d'entrée et de sortie sont identifiés par des flèches en pointillé dans la partie supérieure du diagramme et leurs spécifications

se trouvent sur le côté de la flèche en pointillé. La déclaration du type de données SSD, SID ou CID est indiquée à côté des paramètres d'entrée et de sortie. Les points de commande (POC) sont indiqués de même dans la partie basse du diagramme.

Les modules SIB peuvent être définis avec divers niveaux de détail. L'opération de composition permet de définir des modules SIB à partir de modules SIB de taille plus réduite afin de constituer un module SIB de haut niveau (HLSIB). L'opération de décomposition permet, en sens inverse, de répartir le niveau de détail d'un module SIB de haut niveau en blocs réutilisables de taille plus réduite. La Figure 6 présente plusieurs couches de modules SIB de haut niveau avec différents niveaux de détail. Il peut être intéressant de cataloguer certains modules SIB de haut niveau ou des combinaisons de modules SIB simples afin de faciliter la réutilisation de fonctionnalités plus complexes.

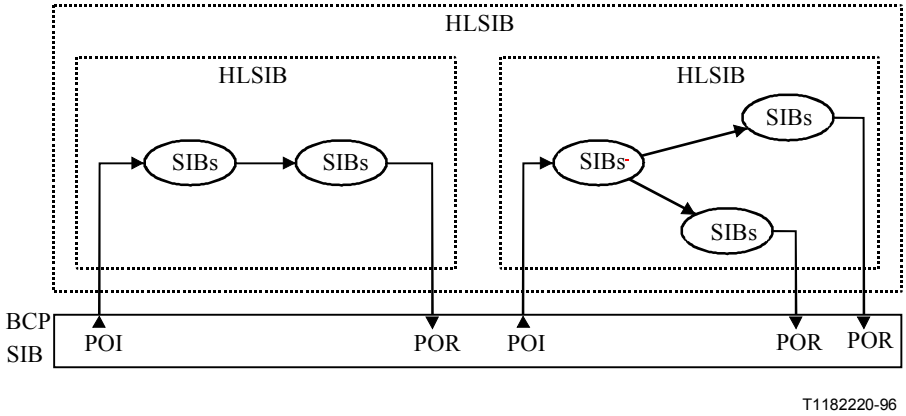


Figure 6/Q.65 – Combinaisons de modules SIB et HLSIB (exemple générique)

La Figure 7 donne un exemple de diagramme de logique globale de service. Le diagramme de logique globale de service contenant un module SIB, ou une combinaison de modules SIB, qui réalise une fonction de service donnée, constitue une "passerelle" idéale d'une description de la phase 1 vers une description de la phase 2. Le diagramme décrit la capacité de service ou de réseau avec une notation abrégée qui véhicule une quantité importante d'information en utilisant des moyens graphiques relativement simples. Les définitions apparaissant dans le diagramme contiennent toutefois l'ensemble des diagrammes SDL, flux d'information et actions d'entité fonctionnelle nécessaires pour une description complète de la phase 2.

La Figure 7 donne une description possible du service final de filtrage. En partant du point d'initialisation correspondant à l'arrivée de l'appel, le module SIB "Filtre" est utilisé afin de déterminer si l'utilisateur appelant figure sur la liste des utilisateurs autorisés à réaliser un appel pour la destination. L'appel est autorisé si l'appelant figure sur la liste et le processus d'appel de base poursuit dans ce cas le traitement d'appel en utilisant les données existantes. Dans le cas contraire, le module SIB d'interaction utilisateur (UI, *user interaction SIB*) est utilisé pour envoyer à l'appelant un message de déconnexion adéquat, après quoi le processus d'appel de base libère l'appel.

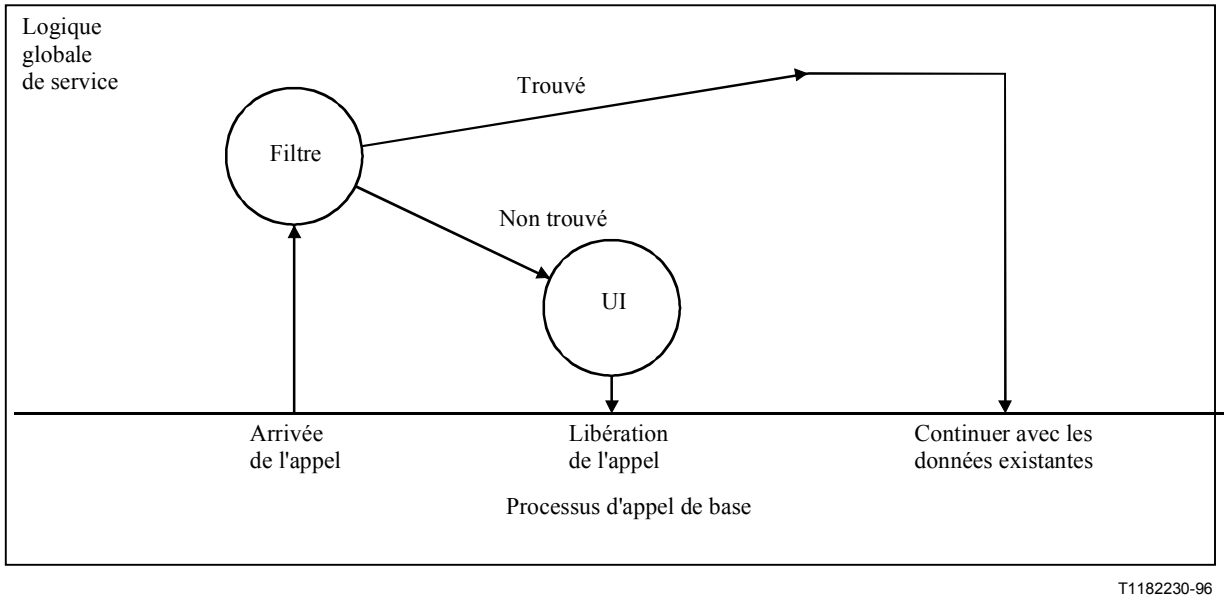


Figure 7/Q.65 – Processus fonctionnel global du service de filtrage d'arrivée

2.2.4 Modélisation de fonctions de service par des modules SIB

L'utilisation de modules SIB et de modules SIB de haut niveau est recommandée pour la description des fonctions de services et des capacités de réseau, car cette pratique encourage la réutilisation de diagrammes SDL, de flux d'information et d'actions d'entité fonctionnelle existants. Le diagramme de logique globale de service, présentant un module SIB ou une combinaison de modules SIB réalisant une fonction de service particulière, constitue un moyen économique et efficace pour transmettre à la description de la phase 2 l'information fonctionnelle contenue dans les modules SIB.

Les directives suivantes concernent l'utilisation des modules SIB dans la méthode de la phase 2:

- fournir des diagrammes de logique globale de service contenant le point de lancement (POT, *point of initiation*) et les points de retour du module SIB du processus d'appel de base;
- fournir un mappage des modules SIB utilisés vers les descriptions d'entités fonctionnelles contenues dans le modèle fonctionnel unifié. Fournir également les références pour les définitions des modules SIB utilisés [par exemple le paragraphe 5/Q.1213 (1995)]. Ceci peut être fait facilement sous forme de tableau (voir 2.2.6);
- fournir les paramètres de données SIB nécessaires pour le service et identifier les valeurs de données, compte tenu du contexte du service.

Dans le cas où toutes les fonctions du service peuvent être décrites à l'aide de modules SIB, des références spécifiques doivent être fournies, dans les paragraphes adéquats de la description de la phase 2 pour la prise en charge des diagrammes SDL, des flux d'information et des actions d'entité fonctionnelle. Il n'est pas nécessaire de faire un avant-projet explicite pour ces paragraphes (dans l'hypothèse où l'exactitude des descriptions des modules SIB a été démontrée). La "preuve" de la correction des diverses combinaisons de modules SIB et de modules SIB de haut niveau sera faite par l'expérience; les services de la bibliothèque de modules SIB conserveront cette information). S'il n'existe pas de modules SIB disponibles pour décrire une fonction donnée, la méthodologie fonctionnelle unifiée peut être utilisée comme par le passé pour élaborer d'une manière explicite les diagrammes SDL, flux d'information et actions d'entité fonctionnelle. Si on identifie certains modules SIB permettant de décrire partiellement un service, il est suggéré de fournir les diagrammes SDL et les flux d'information d'une manière explicite en indiquant quelles parties proviennent des définitions de module SIB.

2.2.5 Liste des modules SIB disponibles

Modules indépendants du service de RI CS-1R:

1) algorithme *algorithm;*
2) authentification *authenticate;*
3) taxation *charge;*
4) comparaison *compare;*
5) distribution *distribution;*
6) limite *limit;*
7) journalisation d'information d'appel *log call information;*
8) file d'attente *queue;*
9) filtre *screen;*
10) gestion de données de service *service data management;*
11) notification de statut *status notification;*
12) traduction *translate;*
13) interaction utilisateur *user interaction.*

D'autres modules SIB sont définis dans les Recommandations de la série Q.12x3.

2.2.6 Mappages des modules SIB vers des entités fonctionnelles

Le Tableau 1 montre le mappage des modules SIB vers des entités fonctionnelles.

Tableau 1/Q.65 – Mappage entre modules SIB et entités fonctionnelles

Modules SIB	Entités fonctionnelles			
	SSF/CCF	SCF	SRF	SDF
Algorithme		X		
Taxation	X	X		
Comparaison		X		
Distribution		X		
Limite	X	X		
Journalisation d'information d'appel	X	X		X
File d'attente	X	X	X	
Filtre		X		X
Gestion de données de service		X		
Notification de statut	X	X		
Traduction		X		X
Interaction utilisateur	X	X	X	
Vérification		X		
Processus d'appel de base	X	X		
Authentification		X		X

2.3 Etape 3 – Diagrammes de flux d'information (voir 3.2 pour l'alternative de l'étape 3 employant les techniques orientées objet)

2.3.1 Identification des flux d'information

La répartition des fonctions nécessaires à la fourniture d'un service, tel qu'il est défini par le modèle fonctionnel, nécessite des interactions entre entités fonctionnelles. De telles interactions sont décrites sous la forme de "flux d'information" et porteront un nom décrivant l'objectif du flux d'information.

Les diagrammes de flux d'information sont créés en vue de recueillir tous les flux d'information nécessaires pour les cas généraux d'exécution réussie du service. Il peut être nécessaire de créer des diagrammes de flux d'information pour d'autre cas, en fonction des besoins. La Figure 8 présente la forme générale d'un diagramme de flux d'information pour un service de base ou un service complémentaire.

Les diagrammes de flux d'information de services complémentaires ne doivent pas nécessairement dupliquer les descriptions de flux d'information faisant partie d'un service de base. Il est toutefois possible qu'une description de service complémentaire identifie de nouveaux besoins de flux d'information entre les entités fonctionnelles représentant le service de base, auquel cas une description correspondante est nécessaire.

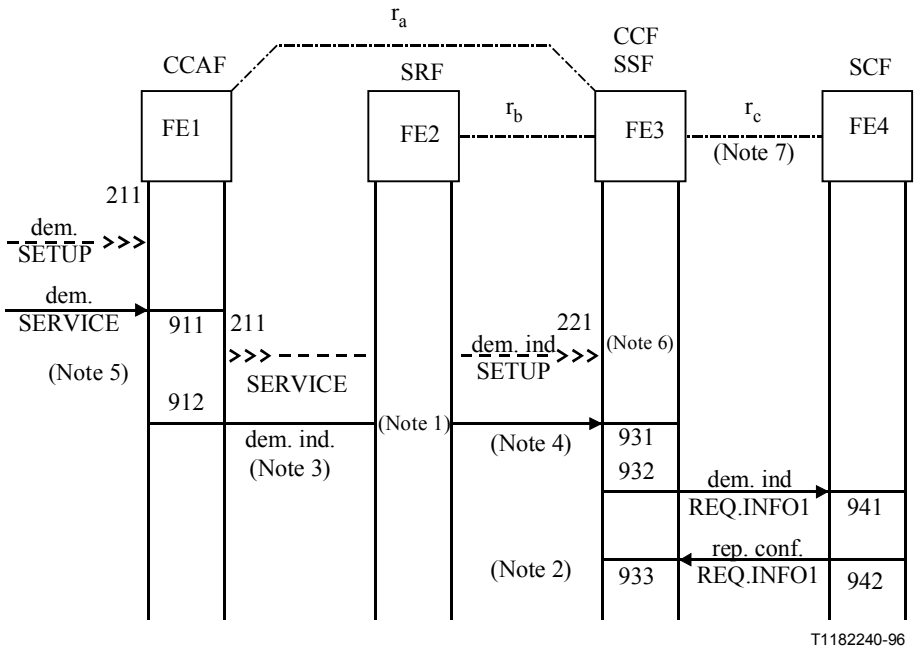


Figure 8/Q.65 – Exemple de diagramme de flux d'information

Notes concernant la Figure 8:

NOTE 1 – La réception et l'émission d'entrées/sorties utilisateur et les flux d'information sont représentés par des lignes horizontales entre les colonnes des entités fonctionnelles concernées. L'absence d'une telle ligne indique a contrario l'absence de réception et d'émission.

NOTE 2 – Un numéro de référence est attribué à tout point de la séquence globale au niveau duquel sont indiquées des actions d'entité fonctionnelle.

NOTE 3 – Les flux d'information sont représentés par des flèches avec le nom du flux d'information indiqué au dessus et en dessous de la flèche. Le nom descriptif figure en majuscules au-dessous de la flèche et l'étiquette (par exemple "dem.ind.") est indiqué en minuscules en dessus de la ligne. Pour des flux d'information non confirmés et la partie "demande" des flux d'information confirmés l'étiquette "dem.ind." figure en minuscule sous la flèche du flux d'information. Pour la partie "confirmation" des flux d'information confirmés, la notation "rep. conf." est utilisée.

NOTE 4 – Dans la colonne d'une entité fonctionnelle donnée:

– les actions indiquées en dessous d'une ligne représentant la réception d'une entrée utilisateur ou d'un flux d'information sont conditionnées par la réception de cette information (c'est-à-dire qu'elles ne peuvent être effectuées avant). L'action 931, par exemple, ne peut être effectuée avant la réception de l'information SERVICE;

– de même, les actions indiquées en dessous d'une ligne représentant l'émission d'une sortie utilisateur ou d'un flux d'information doivent être terminées avant l'émission du flux. Il s'ensuit que l'information REQ.INFO1 ne peut être émise avant que les deux actions 931 et 932 aient été effectuées. Aucune implication n'existe au sujet de l'ordre d'exécution envisagé pour les actions 931 et 932;

– les actions indiquées en dessous d'une ligne représentant l'émission d'une sortie utilisateur ou d'un flux d'information n'ont pas besoin d'être effectuées avant l'émission (quoique cela puisse être le cas dans beaucoup d'implémentations). Aucune contrainte n'est envisagée pour ce qui est de l'ordre relatif de l'émission et de l'action immédiatement suivante. Il s'ensuit que l'action 942 peut être effectuée avant, après ou en même temps que l'émission de la partie "demande" du flux d'information REQ.INFO1.

NOTE 5 – Les interactions de service en phase 1 constituent des entrées et des sorties pour le diagramme de flux d'information de la phase 2. Les interactions de service en phase 1 issues de l'utilisateur sont de la forme demande XXXXX ou réponse XXXXX. Les interactions de service en phase 1 à destination de l'utilisateur sont de la forme indication XXXXX ou confirmation XXXXX.

Les directives suivantes sont à prendre en considération lors de l'établissement d'un avant-projet pour ces flux d'information:

• les colonnes représentent chacune des entités fonctionnelles identifiées par le modèle fonctionnel du service. Les flux d'information sont présentés dans leur ordre d'apparition lors du traitement d'un appel. L'ordre des actions d'entité fonctionnelle apparaissant entre les flux d'information n'est pas significatif;

• un flux d'information sera caractérisé, dans les graphes utilisant des flèches, par les termes demande/indication ou réponse/confirmation. Ceci se répercute sur la primitive communiquée au système de signalisation sous-jacent, comme l'illustre la Figure 8. Le nom de la primitive est en général déduit directement du nom du flux d'information. Les termes sont indiqués en liaison avec le flux d'information, de manière à indiquer la relation entre les diagrammes SDL de la phase 2 et les diagrammes SDL du système de signalisation sous-jacent.

Un numéro de référence identifie sans ambiguïté un point donné dans la séquence de flux d'information de la phase 2 et apparaît en conséquence dans le flux d'information. Ce numéro sert également de pointeur vers une description des actions requises à ce point de la séquence (voir 2.4 ci-dessous). Une courte description des actions d'entité fonctionnelle peut également apparaître sur la partie concernée des diagrammes de flux d'information. Le système de numérotation des références à utiliser est décrit ci-dessous.

Un numéro de la forme XYZ est attribué par le concepteur de la description de la phase 2. Ce numéro identifie un point donné de la description procédurale de la phase 2 (diagrammes de flèches et diagrammes SDL), au niveau duquel sont décrites des actions d'entité fonctionnelle. X doit être égal à "9" pour des actions de service complémentaire. Y est le numéro de l'entité fonctionnelle où l'action est exécutée. Z énumère les actions d'une entité fonctionnelle donnée (Z=1, ..., 9, A, ..., Z, a, ..., z). Ce numéro est non ambigu dans le domaine de la description de la phase 2 d'un service donné (avec toutes ses variantes).

NOTE 6 – Les flux d'appel de base sont indiqués par des lignes en pointillé et des chevrons. Les flux de service complémentaire sont indiqués par des lignes pleines et des flèches. Les informations transférées "dans la bande" sont indiquées par des lignes tiretées doubles.

NOTE 7 – Les relations entre entités fonctionnelles peuvent également figurer sur le diagramme.

2.3.2 Définition des flux d'information individuels

La sémantique et le contenu informationnel de chaque flux d'information sont déterminés. Un flux d'information donné peut être identifié comme devant être confirmé et entraîner en conséquence un flux d'information en retour portant le même nom.

Les flux d'information confirmés se présentent sous la forme d'une demande pour une action (dans une direction) et d'une confirmation de l'exécution de l'action (dans la direction de retour). Les flux d'information confirmés sont en général nécessaires à des fins de synchronisation. Les deux cas principaux sont la demande d'allocation ou de libération d'une ressource partagée.

Les flux d'information sont véhiculés par des protocoles de système de signalisation lorsque les entités fonctionnelles qui interagissent sont implémentées au niveau d'emplacements physiques distincts. Lorsque l'implémentation est faite en un seul lieu, les flux d'information sont internes et ne nécessitent pas de protocole de système de signalisation.

Des tableaux doivent être produits pour indiquer tous les éléments de chaque flux d'information en mentionnant quels sont les éléments obligatoires ou non et quelle relation est utilisée pour transférer l'information (voir Tableau 2).

Tableau 2/Q.65 – Exemple de définition des flux d'information individuels

REQ.INFO1

Relation	Elément	dem. ind	rép. conf
r_a, r_b, r_c	Demande de numéro taxé	Obligatoire	–
r_a, r_b, r_c	Numéro taxé	–	Obligatoire
r_a, r_b, r_c	Demande de type de service	Facultatif	–
r_a, r_b, r_c	Type de service	–	Facultatif

2.4 Etape 4 – Actions d'entité fonctionnelle (voir 3.3 pour l'alternative de l'étape 4 employant les techniques orientées objet)

Les actions de la phase 2 effectuées au sein d'une entité fonctionnelle sont identifiées et énumérées, en partant de la réception de chaque flux d'information jusqu'à la transmission du flux d'information qui en résulte. Toutes les actions visibles de l'extérieur (notifiées de façon explicite ou implicite à d'autres entités fonctionnelles) en font partie. Ces actions identifiées sont alors représentées sur les diagrammes de flux d'information et les diagrammes SDL par de brefs commentaires, ou d'une manière séparée en utilisant des numéros de référence.

NOTE – L'implémentation des principes de taxation des Recommandations de la série D est une affaire nationale. En conséquence, les actions d'entité fonctionnelle qui ne concernent que la taxation ne doivent être présentes que s'il existe un besoin direct de prise en charge de signalisation, ou lorsque le service est susceptible d'être fourni par des réseaux publics à travers une frontière internationale. Lorsque les principes de taxation de la série D doivent être représentés dans les descriptions de service en phase 2, la note de bas de page suivante doit être mentionnée pour toute action d'entité fonctionnelle contenant une fonction de taxation.

"Cette action d'entité fonctionnelle décrit des actions qui peuvent être adoptées par des Administrations en vue de prendre en charge les principes de taxation des Recommandations de la série D."

Le sous-paragraphe concernant les actions d'entité fonctionnelle contient des descriptions d'actions requises pour chaque entité fonctionnelle et chaque action d'entité fonctionnelle est identifiée par un numéro de référence.

La Figure 9 illustre la forme de représentation des actions d'entité fonctionnelle.

action d'entité fonctionnelle de FE2

921:

- interaction avec l'utilisateur en vue d'accumuler l'information
- sélectionner les ressources d'accès réseau
- réserver les équipements nécessaires, dans les deux directions

922:

- interaction avec l'utilisateur en vue d'obtenir l'adresse d'appel
- déterminer et indiquer la fin de numérotation

Figure 9/Q.65 – Exemple de descriptions d'actions d'entité fonctionnelle

2.5 Etape 5 (facultative) – Diagrammes SDL pour des entités fonctionnelles

Le langage de description et de spécification de systèmes (SDL, voir UIT-T Z.100) est utilisé dans sa représentation graphique pour décrire d'une manière formelle un service donné en terme d'actions effectuées dans les entités fonctionnelles impliquées dans la fourniture du service et sous forme des flux d'information résultant de ces actions ou déclenchant ces actions. Les diagrammes SDL sont basés sur (ou sont cohérents avec) les diagrammes de flux d'information créés lors de l'étape 2 de la méthode de description du service. Ils fournissent le détail du fonctionnement du service dans les cas normaux, les cas d'échec et les cas anormaux.

Les flux d'information, qui sont complètement définis sous forme de leurs contenus et de leurs points de départ et d'arrivée dans une partie du diagramme SDL, sont pris en charge par des procédures de signalisation spécifiées dans les descriptions de service en phase 3.

L'UIT-T Z.100 contient la description du langage SDL. Le paragraphe ci-dessous en résume les éléments.

NOTE – On estime que l'étape 5 (SDL) n'est facultative pour l'étape 2 que si des diagrammes SDL formels sont spécifiés dans la description de l'étape 3.

2.5.1 Caractéristiques générales du langage SDL

Les éléments de description d'un service en langage SDL sont les systèmes, blocs, processus et procédures (voir 2.4/Z.100 et Annexe B/Z.100). Les systèmes, blocs et processus peuvent être définis comme une instanciation des types système, bloc et processus. Les procédures ou les types système, bloc et processus sont décrits dans un ou plusieurs diagrammes. Les diagrammes de procédure sont présents si les diagrammes de processus contiennent des appels de procédure.

L'information est véhiculée entre blocs ou processus dans des signaux qui peuvent avoir des paramètres contenant des données. Les données ont une définition formelle fournie par des types de données.

2.5.1.1 Diagramme système

Le diagramme de type de système décrit le système, dans le contexte de description de la phase 2, sous la forme de blocs, de canaux et de signaux. Les blocs représentent l'ensemble du réseau et des fonctions d'accès réseau devant être exécutées par le système afin de fournir un service. Les signaux véhiculés par les canaux (voir 2.5/Z.100) décrivent les informations échangées entre les blocs, ainsi qu'entre les blocs et l'environnement du système. Les canaux sont connectés aux blocs et au cadre délimitant un diagramme par l'intermédiaire de portes nommées, si le bloc est défini comme une instance du type bloc. La liste des signaux véhiculés par les canaux du système fait partie du diagramme du système.

Les Figures 10a et 10b donnent un exemple de diagramme système pour un système nommé BasicService (supposé être le service support par circuit commuté). Le diagramme contient, dans l'exemple, deux cadres dénommés "définition" (Figure 10a) et "structure" (Figure 10b).

NOTE – Le nombre de cadres utilisés pour tracer un diagramme est arbitraire et choisi normalement en fonction de considérations de lisibilité et d'espace disponible pour le graphisme.

Le cadre "définitions" a le contenu suivant:

a) un symbole de type de bloc contenant le nom du type de bloc (BS_St2_BT) précédé du mot-clé "virtual". Ceci signifie que le type de bloc (BS_St2_BT) peut être redéfini, par exemple lorsqu'il devient nécessaire d'incorporer au service de base une nouvelle logique de service prenant en charge un service complémentaire;

b) un symbole de texte contenant la liste des signaux pouvant être transférés entre le système et l'environnement, précédée du mot clé SIGNAL;

c) des symboles de texte contenant des descriptions de listes de signaux (mot clé SIGNALLIST) nommées respectivement Afw, Abw, Bfw et Bbw;

d) un symbole de texte contenant les définitions (mots clés NEWTYPE, ENDNEWTYPE) d'un type de données RejectCause. Les variables de ce type peuvent prendre les valeurs (mot clé LITERALS) RequestNotValid, NoRoute, NoBresources, NoResponse et Busy. Il n'est pas nécessaire de définir de nouveau les types de données normalisés [par exemple Boolean, Integer (voir l'Annexe D/Z.100)];

e) un symbole de texte contenant une liste de synonymes qui ne sont pas spécifiés dans la description du service. Le mot-clé SYNONYM indique que ces noms sont donnés à des valeurs définies d'une manière externe (mot clé EXTERNAL), par exemple NoAnswerTime possédant le type de données normalisé Duration représente une valeur définie d'une manière externe.

Le cadre dénommé "structure" décrit les blocs du système BasicService. Le système contient, dans cet exemple, une instance BS_St2 du type de bloc BS_St2_BT englobant toutes les actions de réseau relatives au service de base. Les utilisateurs du service sont situés en dehors du système, c'est-à-dire dans l'environnement. Seules leurs interactions avec le système sont décrites par la phase 2, et non leurs actions. Les interactions sont représentées par les signaux définis dans les listes de signaux et sont transférés, en provenance et à destination de l'environnement, au moyen de canaux nommés. Dans l'exemple, les signaux appartenant aux listes de signaux AFw et ABw sont transférés depuis et vers l'environnement sur le canal nommé UserAInterface connecté à l'instance de bloc BS_St2 au niveau de la porte To_A. De même, les signaux appartenant aux listes de signaux BFw et BBw sont transférés depuis et vers l'environnement sur le canal nommé UserBInterface connecté à l'instance de bloc BS_St2 au niveau de la porte To_B.

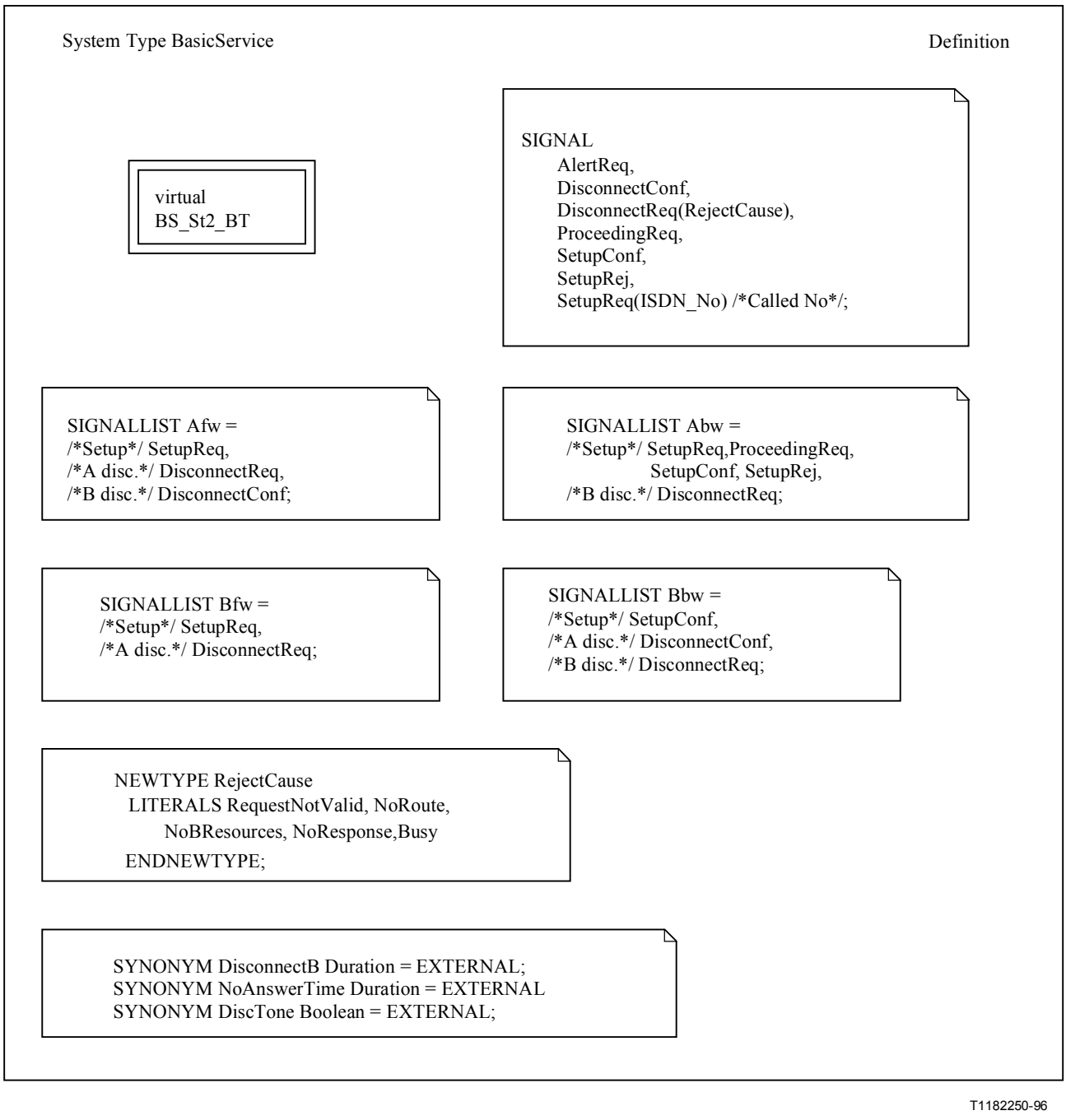


Figure 10a/Q.65 – Exemple de définition de diagramme de type de système

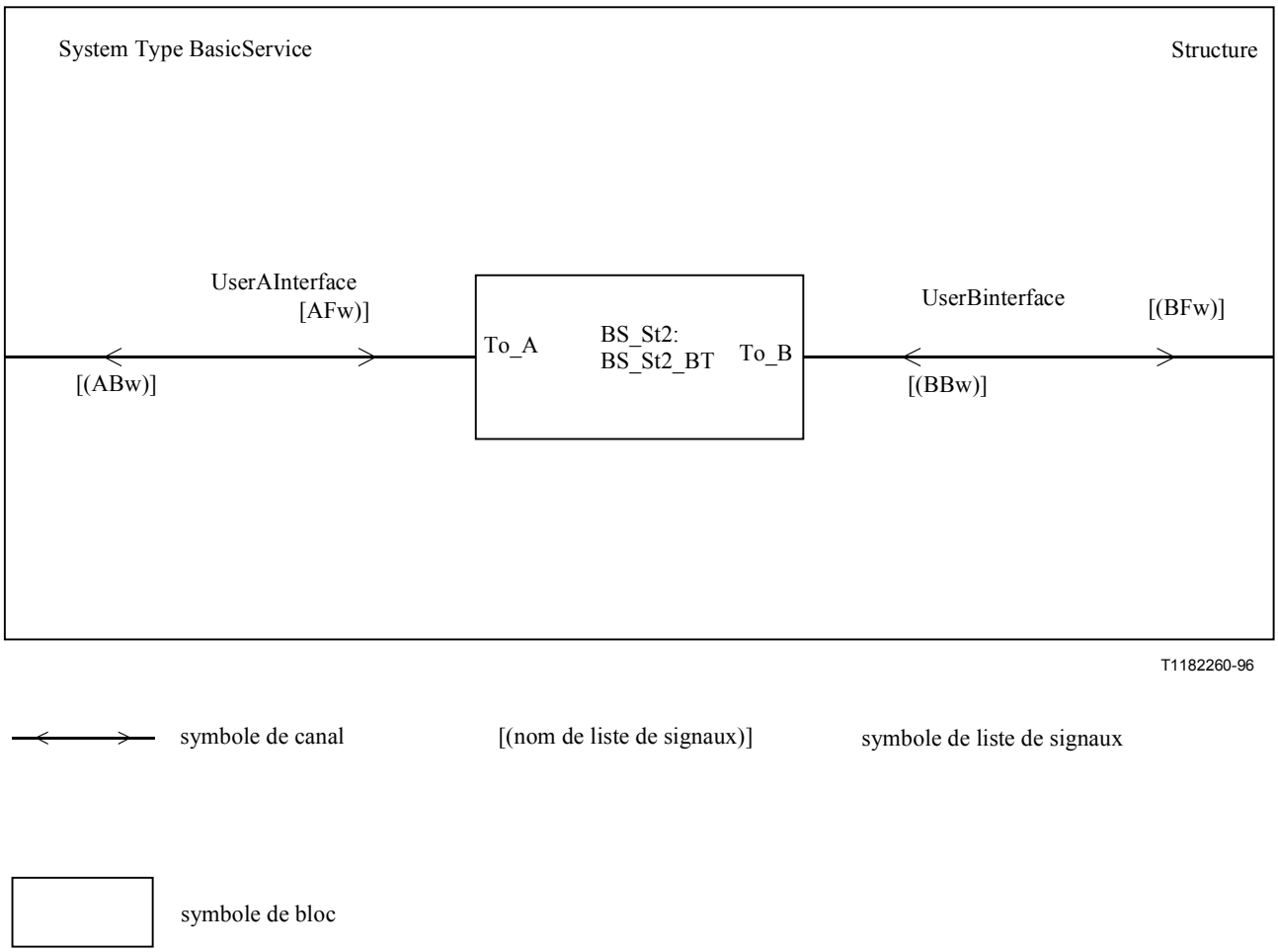


Figure 10b/Q.65 – Exemple de structure de diagramme de type de système

2.5.1.2 Diagramme de blocs

Le diagramme de type de bloc définit, dans le contexte de la description de la phase 2, les types de processus utilisés dans le système ainsi que la manière dont ces types sont structurés et communiquent en vue de la prise en charge d'un service. Les cadres des Figures 11a "Définition" et 11b "Structure" donnent le diagramme du type de bloc BS_St2. Comme déjà indiqué pour le diagramme de type de système, ce type de bloc a été défini comme type virtuel afin de permettre sa redéfinition éventuelle. Le bloc contient des instances de 5 types de processus se trouvant dans les cinq entités fonctionnelles (à savoir, les fonctions d'agent de commande d'origine et de terminaison d'appel et les fonctions de commande d'origine, de transit et de terminaison d'appel) qui sont impliquées dans l'initiation et la terminaison d'une instance du service de base ou, en d'autres termes, dans l'établissement et la libération d'un appel de base.

Les cinq types de processus de l'exemple de service de base ont été définis comme types virtuels afin de permettre leur redéfinition ultérieure (par exemple, en vue d'insérer une logique de service complémentaire).

Le transfert d'information est représenté dans le diagramme de processus par des routes de signal qui connectent, au moyen de portes nommées, une instance de processus à une autre instance de processus ou à un canal. Dans l'exemple, la route R1 interconnecte les instances de processus CCAFO et CCFO au moyen de la porte OG du processus CCAFO et de la porte IC du processus CCFO. De même, la route UIA connecte l'instance de processus CCAFO au processus UserAInterface au moyen de la porte To_A. Les signaux transférés sur chaque route sont décrits dans des listes de signaux.

Les symboles de processus contiennent les noms des instances et types de processus, séparés par une virgule, ainsi que deux nombres, par exemple (1,1), qui suivent le nom de l'instance de processus. Ces nombres représentent respectivement le nombre d'instances du processus existant au moment de sa création et le nombre maximal d'instances du processus.

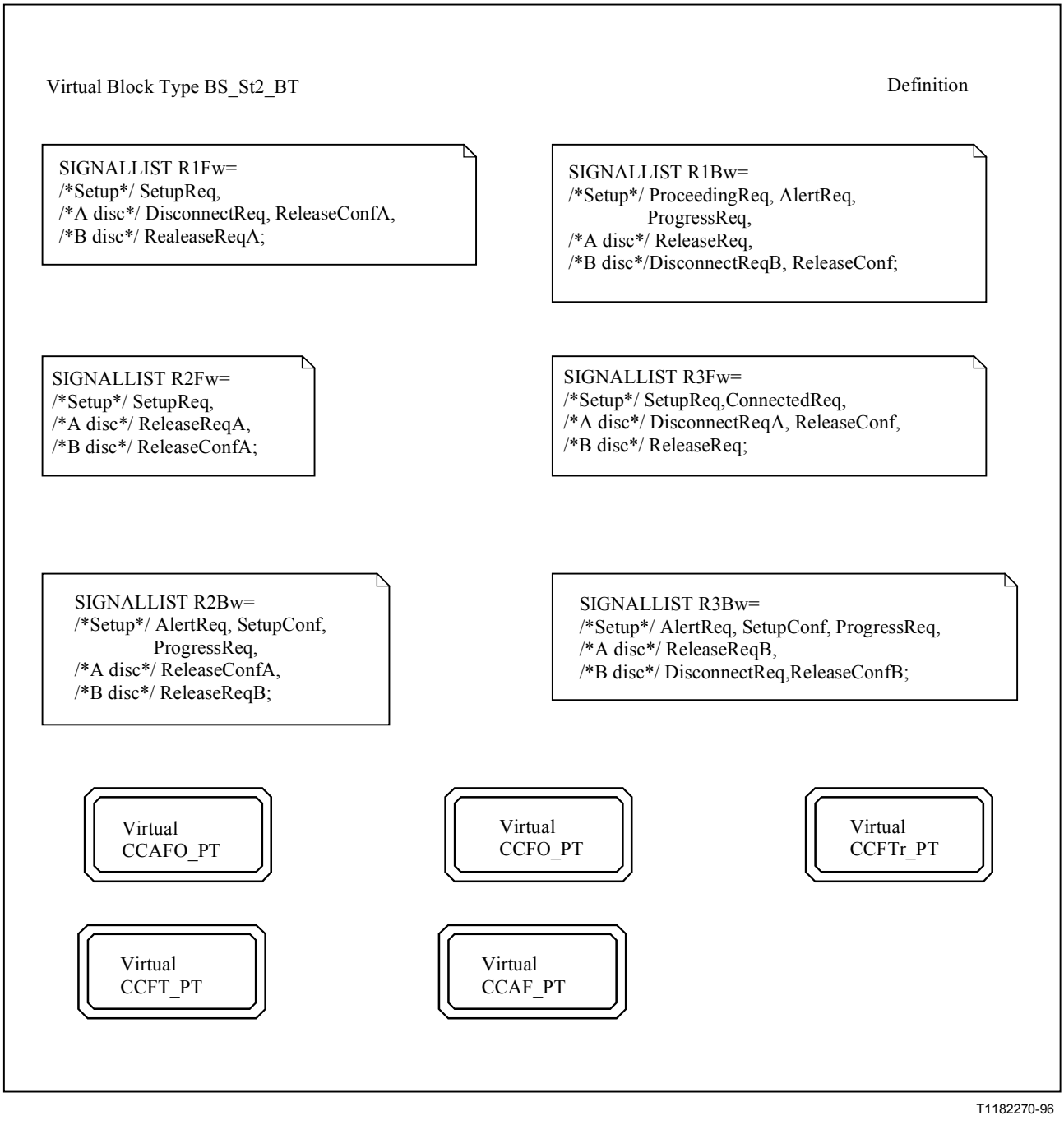


Figure 11a/Q.65 – Exemple de diagramme de définition de type de module

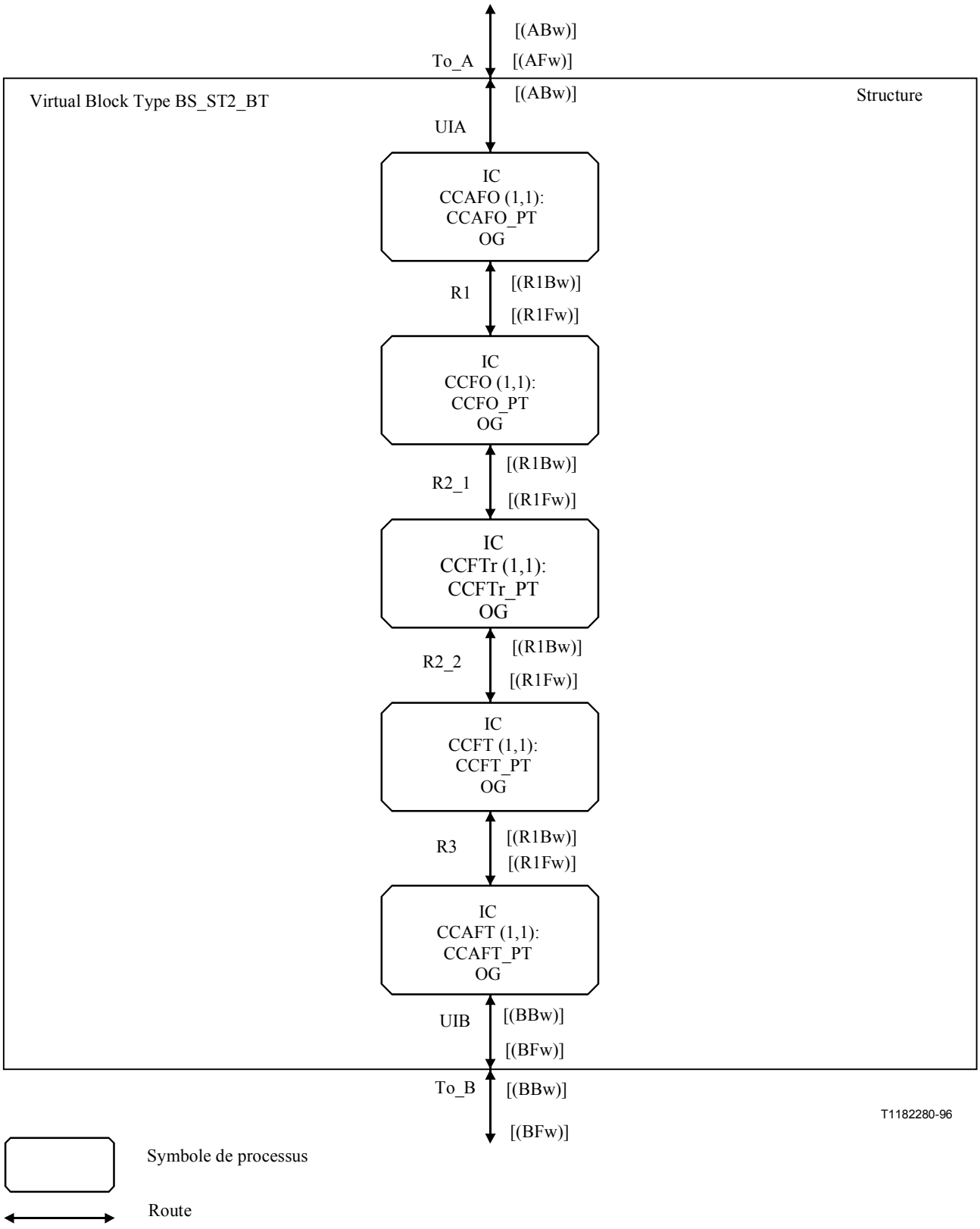
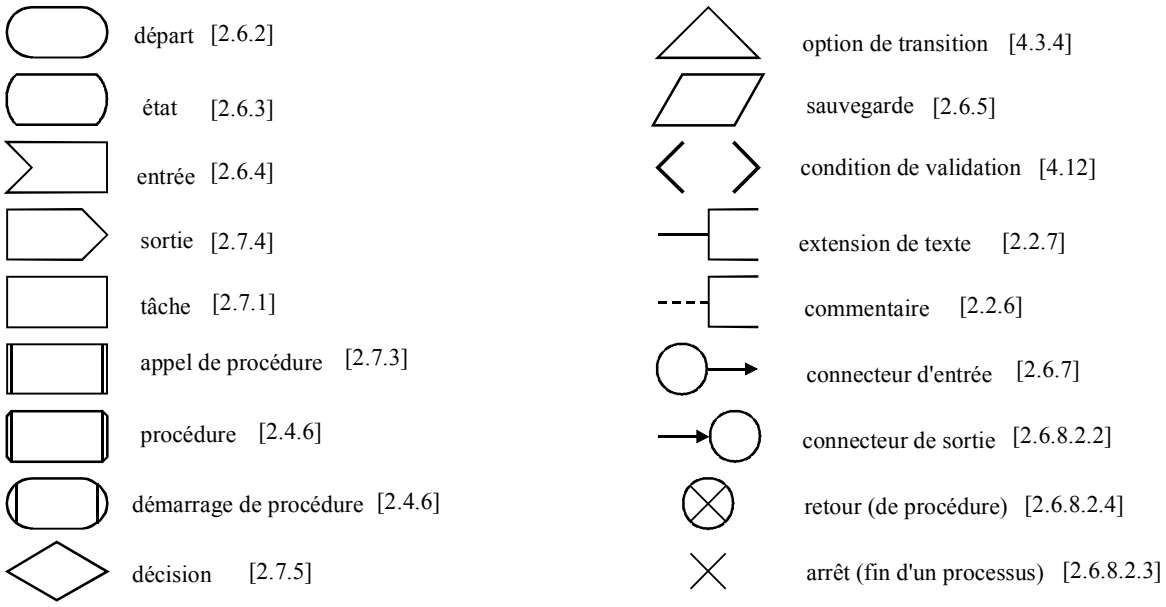


Figure 11b/Q.65 – Exemple de diagramme de structure de type de module

2.5.1.3 Diagramme de processus

Un diagramme de processus décrit la séquence d'actions (transactions) effectuées par une instance de processus lorsqu'un signal est reçu. Le Tableau 3 présente les symboles graphiques utilisés le plus fréquemment dans des diagrammes de processus décrivant des services. Les nombres entre crochets indiquent les sous-paragraphes de l'UIT-T Z.100 dans laquelle ces symboles sont décrits plus en détail.

Tableau 3/Q.65 – Symboles SDL



T1182290-96

Deux diagrammes de processus sont donnés en exemple. La Figure 12a contient un fragment du type de processus virtuel CCAFO_PT appartenant au type de module BS_St2_BT et la Figure 12b contient un fragment du type de processus virtuel CCFT_PT (voir Figures 11a et 11b).

Les notes explicatives suivantes s'appliquent à la Figure 12:

a) le symbole de texte contient la déclaration (mot clé DCL) des variables utilisées par le processus. Un nom et un type de données sont définis pour chaque variable, par exemple les variables dénommées "calling" et "called" sont du type prédéfini PId (identificateur d'instance de processus) et identifient donc des processus. Les variables Connected et Resourcesoccupied sont également de type prédéfini (par exemple Boolean), alors que les variables CalledNr et Reason sont respectivement des nouveaux types ISDN_No et RejectCause, qui sont propres à cette description de système;

b) les symboles d'entrée et de sortie contiennent le nom du symbole SDL qu'ils représentent. Les expressions entre parenthèses se trouvant après le nom de signal, par exemple SetupReq (CalledNr) indiquent une information (un paramètre) véhiculée par le signal;

c) l'émetteur est déclaré par un mot clé, par exemple dans le symbole de tâche "Calling:=Sender", utilisé pour identifier le processus qui a émis le signal d'activation de la transition en cours. Dans l'exemple des processus de la Figure 13, les deux processus "Calling" et "Called" sont identifiés de cette façon. Ces processus se trouvent, dans ce cas particulier, dans l'environnement et ne sont pas pris en compte dans la description du système donnée en exemple.

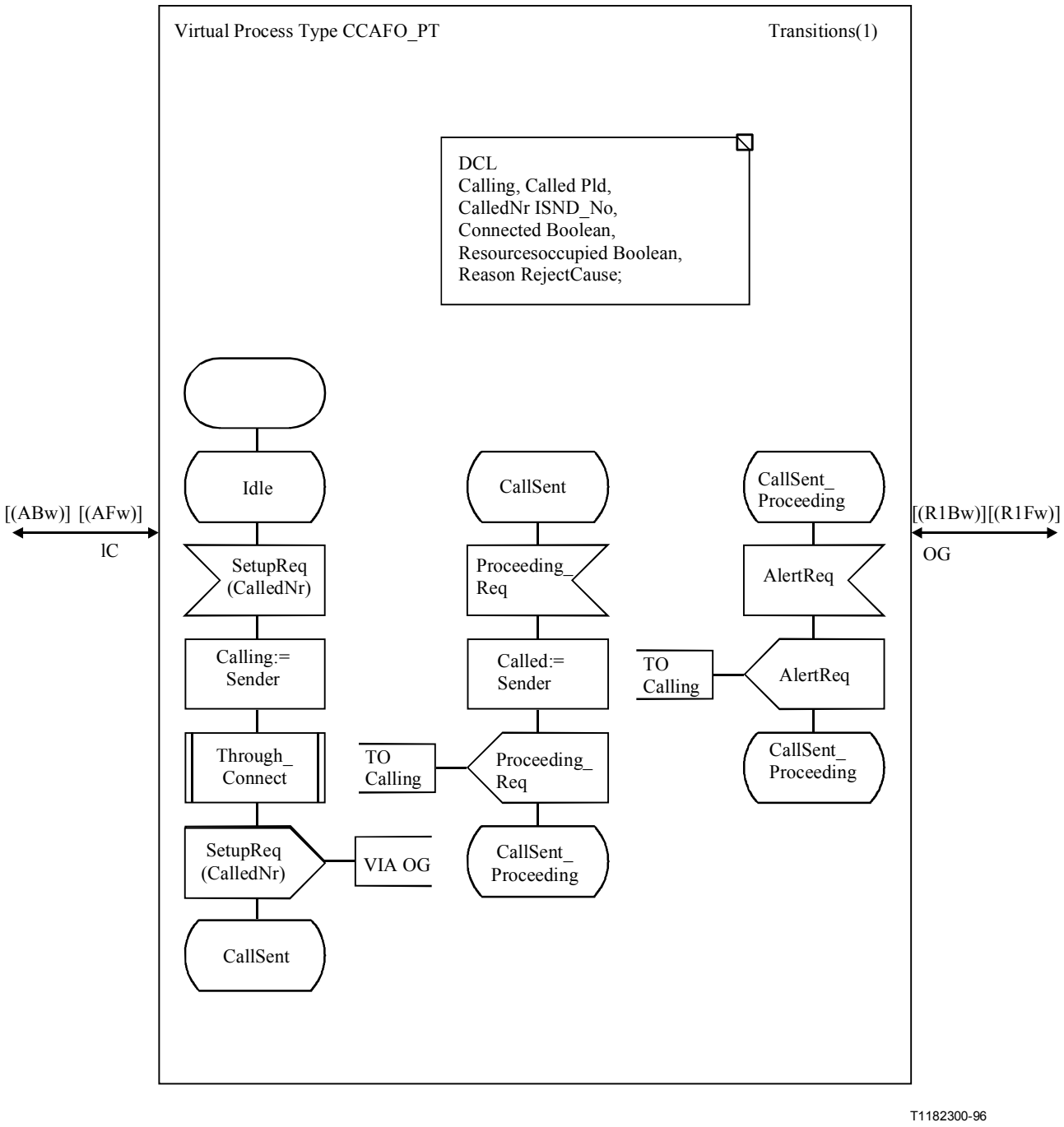


Figure 12a/Q.65 – Exemple 1 de diagramme de processus – Fragment du processus CCAFO

La Figure 12b donne une illustration de la méthode de création, dans une transition existante, d'une ouverture vers des extensions futures. L'ouverture est fournie par l'insertion d'un appel de procédure, dans l'exemple vers la procédure SupSvc1, faite en un point de la transition où on anticipe une extension ultérieure de la logique de traitement. Cette procédure est "vide" au départ. Elle est redéfinie en vue de fournir une logique supplémentaire lorsque ceci devient nécessaire (voir 2.5.2) et c'est pour cette raison qu'elle est définie comme étant virtuelle.

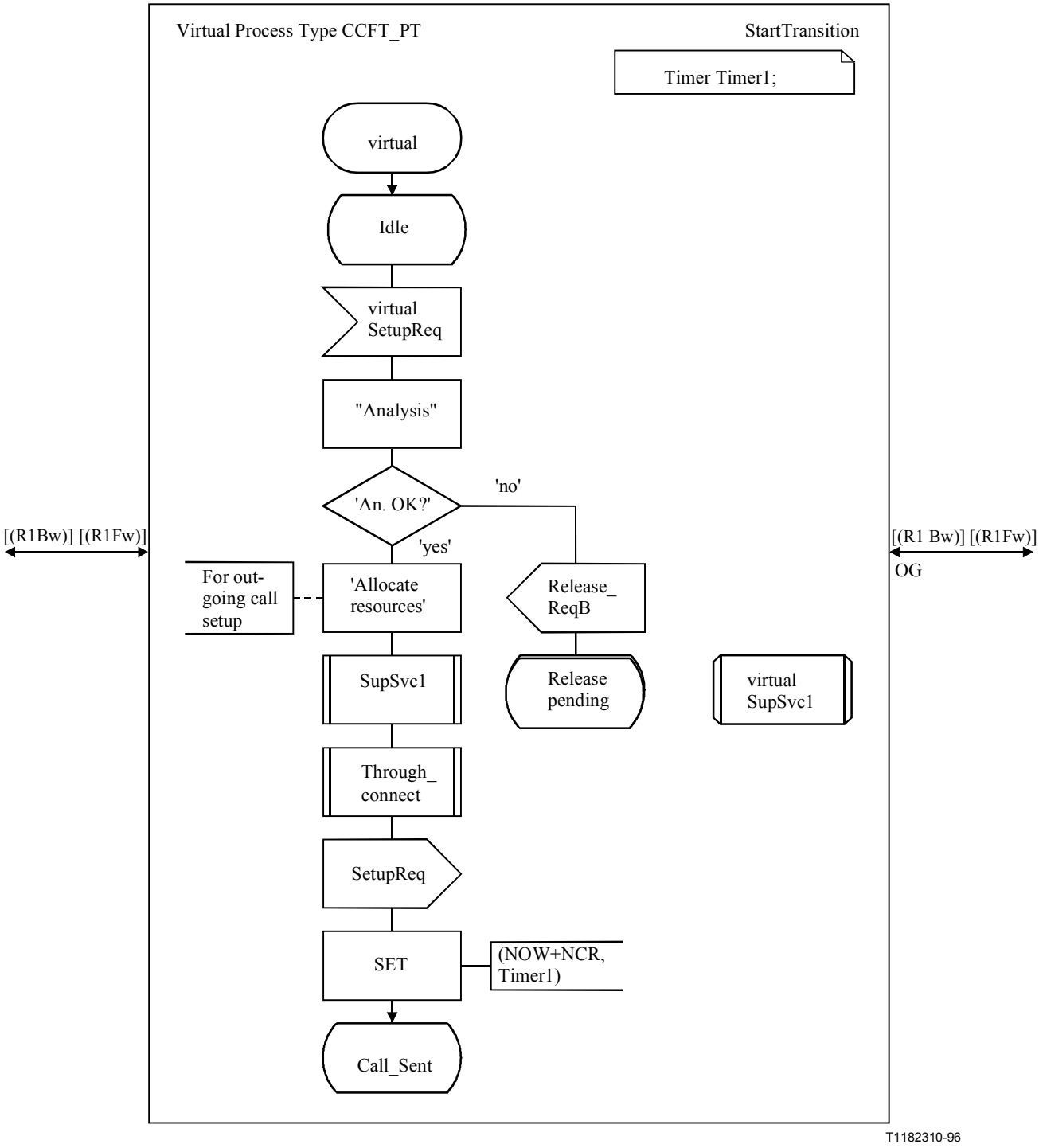


Figure 12b/Q.65 – Exemple 2 de diagramme de processus – Fragment du type de processus CCFT

2.5.1.4 Diagramme de procédure

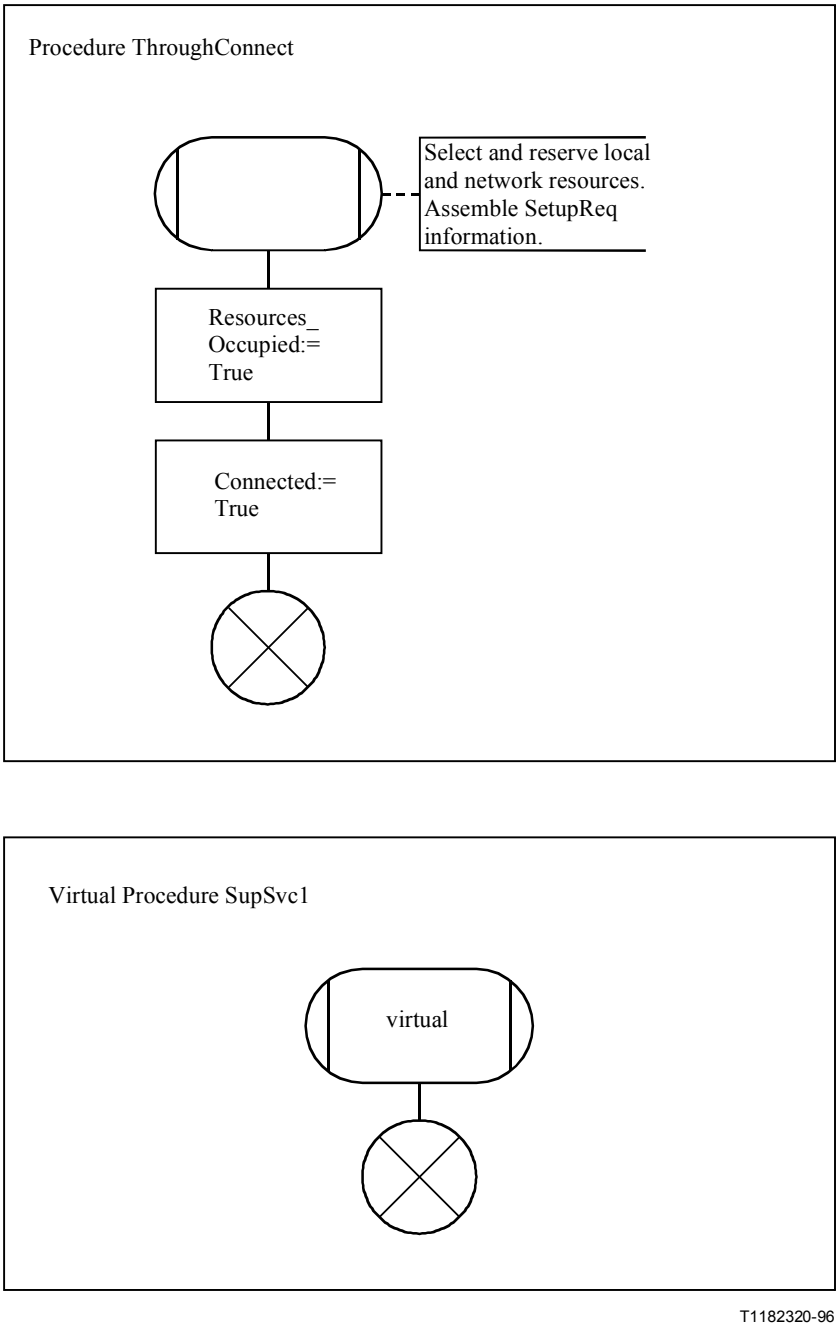


Figure 13/Q.65 – Exemples de diagramme de procédure

Les procédures éventuellement appelées par les types de processus du système sont décrites dans des diagrammes de procédure. Les exemples de la Figure 13 présentent la procédure ThroughConnect d'établissement de connexion appelée par des instances de processus CCAFO et CCFT et une procédure SupSvc1 appelée par une instance de processus CCFT. Comme il est prévu que ce dernier subira des modifications, la procédure SupSvc1 a été définie comme étant virtuelle. Comme noté précédemment, celle-ci est vide et se comporte pour l'instant comme une réservation de place pour l'addition ultérieure d'une logique de service.

Le Tableau 3 donne l'explication des symboles utilisés.

2.5.2 Ajout de fonctionnalités à des descriptions de service de la phase 2 existantes

2.5.2.1 Spécialisation

Le besoin d'ajouter de nouvelles fonctionnalités à une description de service existante se manifeste à certaines occasions. L'ensemble de modifications, devant être faites pour le service support de base par circuit commuté (service de base) lorsqu'un nouveau service complémentaire est normalisé, constitue un cas typique.

Il est possible d'utiliser le concept de spécialisation du langage SDL lorsque la description du nouveau service (par exemple, du service complémentaire) nécessite la modification de la description d'un service existant (par exemple, du service de base). Une spécialisation permet de décrire des procédures logiques associées à un nouveau service sans affecter la description du service existant en fournissant, par exemple, les possibilités suivantes:

– ajouter et redéfinir des types de bloc dans des diagrammes de système;
– ajouter des canaux dans des diagrammes de type de système afin de véhiculer des signaux existants ou des signaux nouveaux;
– ajouter et redéfinir des types de processus dans des diagrammes de bloc;
– ajouter des routes dans des diagrammes de type de bloc afin de véhiculer des signaux existants et des signaux nouveaux;
– ajouter de nouveaux états et de nouvelles transitions entre états dans des diagrammes de type de processus;
– ajouter de nouvelles transitions vers des états existants, déclenchées par de nouveaux signaux;
– redéfinir des transitions entre états existants;
– redéfinir des procédures.

Les redéfinitions du langage SDL sont rendues possibles en définissant comme virtuels ceux des types de blocs, de types de processus, de procédures et de transitions qui ont une certaine probabilité de subir des modifications lors de l'évolution du système. Les Figures 10a, 11a, 12b et 13 donnent des exemples de définitions virtuelles.

2.5.2.2 Description de service complémentaire

Un service complémentaire est défini par la modification ou la redéfinition de la logique de service de base. Dans cette perspective, les types de bloc, types de processus et transitions qui ont une certaine probabilité de subir des modifications en cas d'ajout d'un service complémentaire ont été marqués comme étant "virtuels" (voir 2.5.2.1).

Les modifications apportées au service de base par l'introduction d'un service complémentaire sont ensuite décrites en énonçant que le système "service complémentaire" hérite du système "service de base" et en ajoutant des blocs, processus, transitions, signaux, etc., nouveaux ou redéfinis comme nécessaire à la prise en charge du service complémentaire. "Inherits" est un mot clé conformément à l'UIT-T Z.100.

La méthode de description est illustrée par les Figures 14 à 17 qui présentent un exemple de service simplifié d'identification des appels malveillants.

Le diagramme de système de la Figure 14 contient les indications suivantes:

a) le type de système MCID est basé sur (hérite du) système BasicService;
b) le module BS_St2 du système BasicService a été redéfini;
c) le canal MCIDUserB a été ajouté. Il est utilisé par le module BS_Stage2 à des fins d'échange des signaux MC_req, MC_conf, et MC_rej avec l'environnement (utilisateur B dans ce cas);

d) un nouveau type de données MC_rejcause a été introduit, pouvant prendre les valeurs (littérales) indiquées.

NOTE – Etant donné qu'il est nécessaire d'indiquer où est attaché le canal MCIDUserB, l'instance de bloc BS_St2 possède un pourtour pointillé indiquant qu'il fait partie du système hérité, c'est-à-dire qu'il ne s'agit pas d'un bloc qui a été ajouté au système BasicService pour les besoins du système MCID.

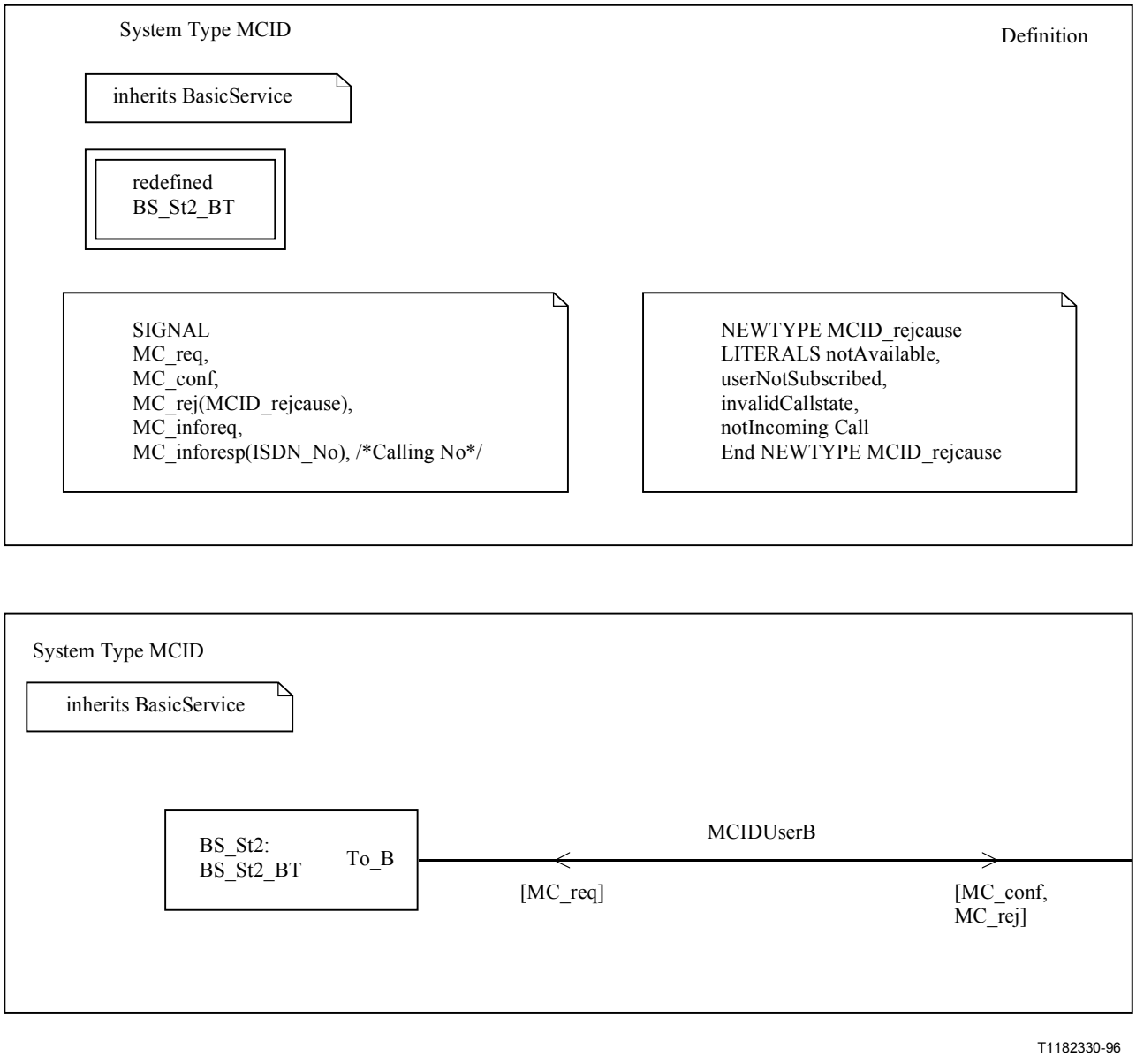


Figure 14/Q.65 – Exemple d'un nouveau type de système (MCID) basé sur un type de système existant (BasicService)

Les Figures 15a et 15b montrent la redéfinition du type de module BS_Stage2_BT résultant de l'introduction du service complémentaire MCID. Cette introduction nécessite l'ajout d'un type de processus MCID_PT et des routes RA, RC1, RC2, et MC_UIB, ainsi que l'ajout de nouveaux signaux dans la liste de signaux de la Figure 15.

Les types de processus qui possèdent des pourtours pointillés ne font pas partie de la redéfinition, mais apparaissent afin qu'il soit possible d'indiquer l'endroit où sont attachées les routes.

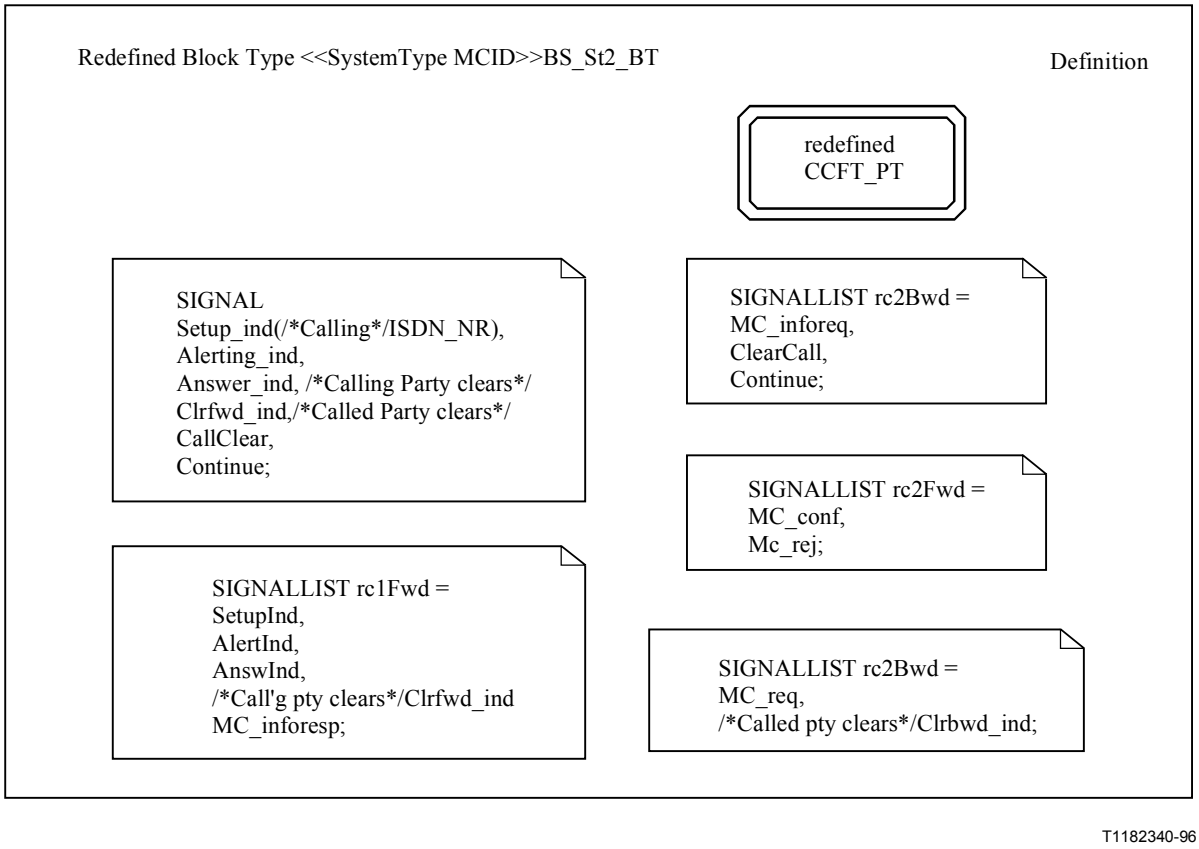


Figure 15a/Q.65 – Redéfinition du type de module BS_St2 (Définition) pour le système MCID

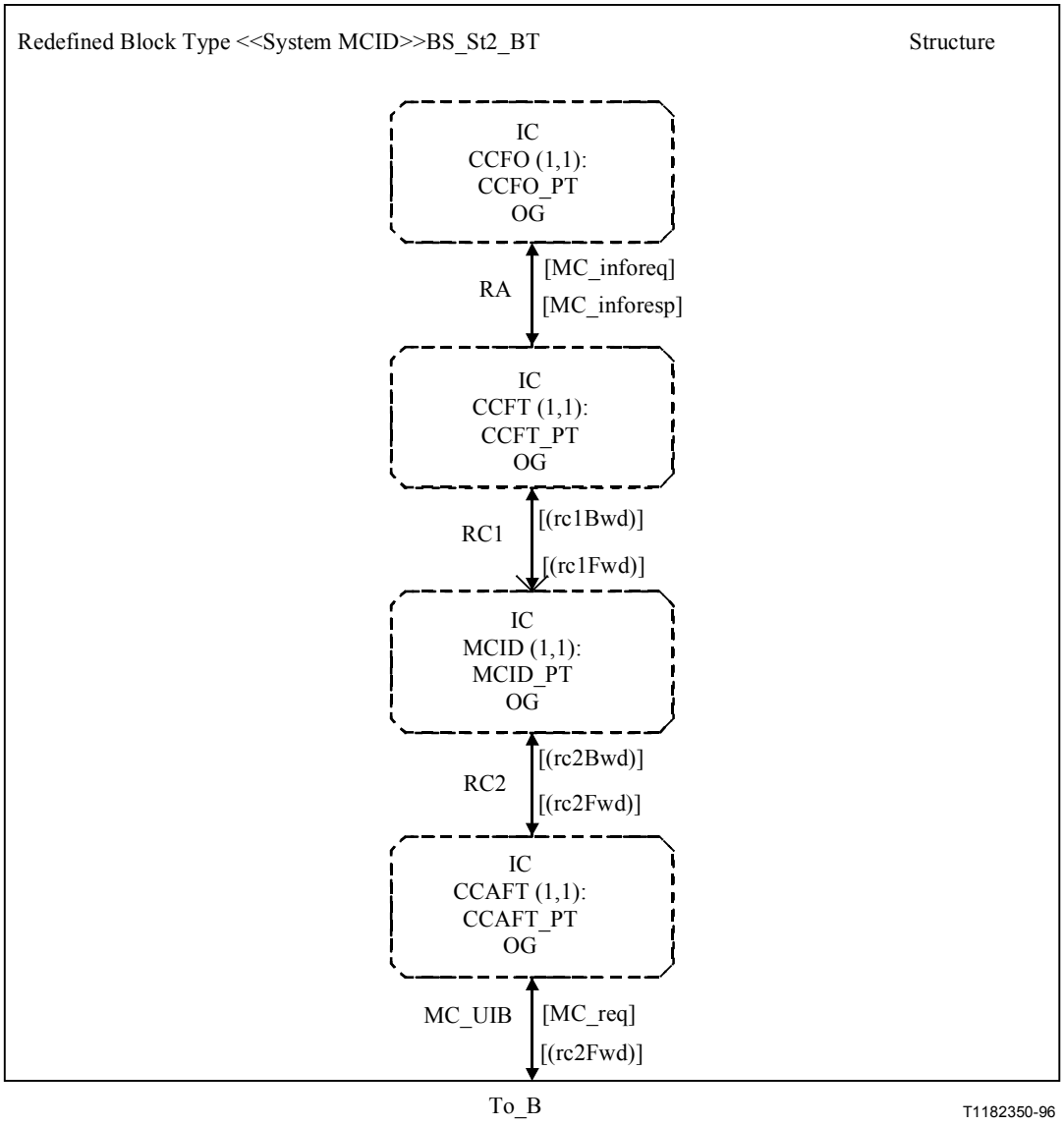


Figure 15b/Q.65 – Redéfinition du type de bloc Type BS_St2 (Structure) pour le système MCID

La Figure 16 présente la redéfinition du type de processus CCFT_PT. La seule modification qu'il est nécessaire de montrer dans ce cas est l'indication de la redéfinition de la procédure SupSvc1, définie comme virtuelle dans le type de système BasicService. La Figure 17 présente la procédure de redéfinition.

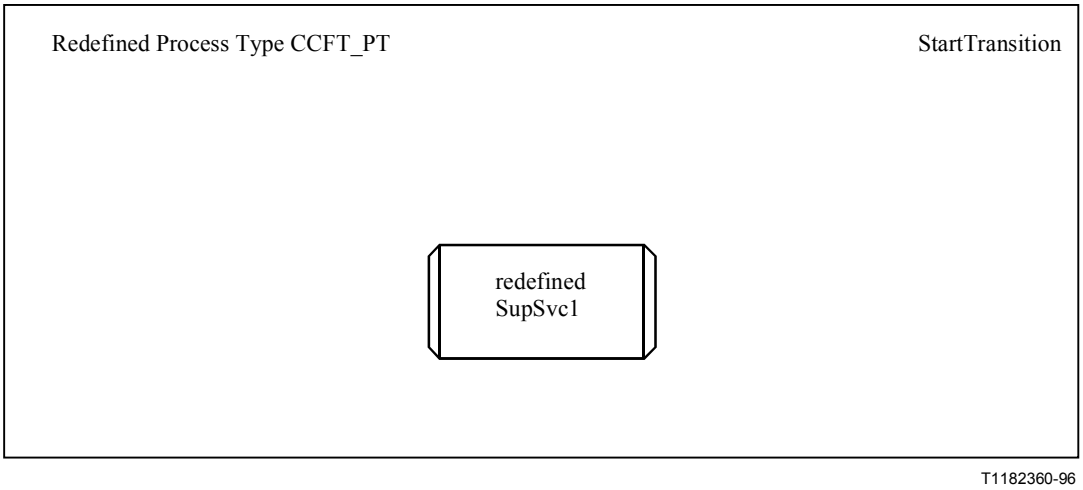


Figure 16/Q.65 – Redéfinition du type de processus CCFT_PT pour le système MCID

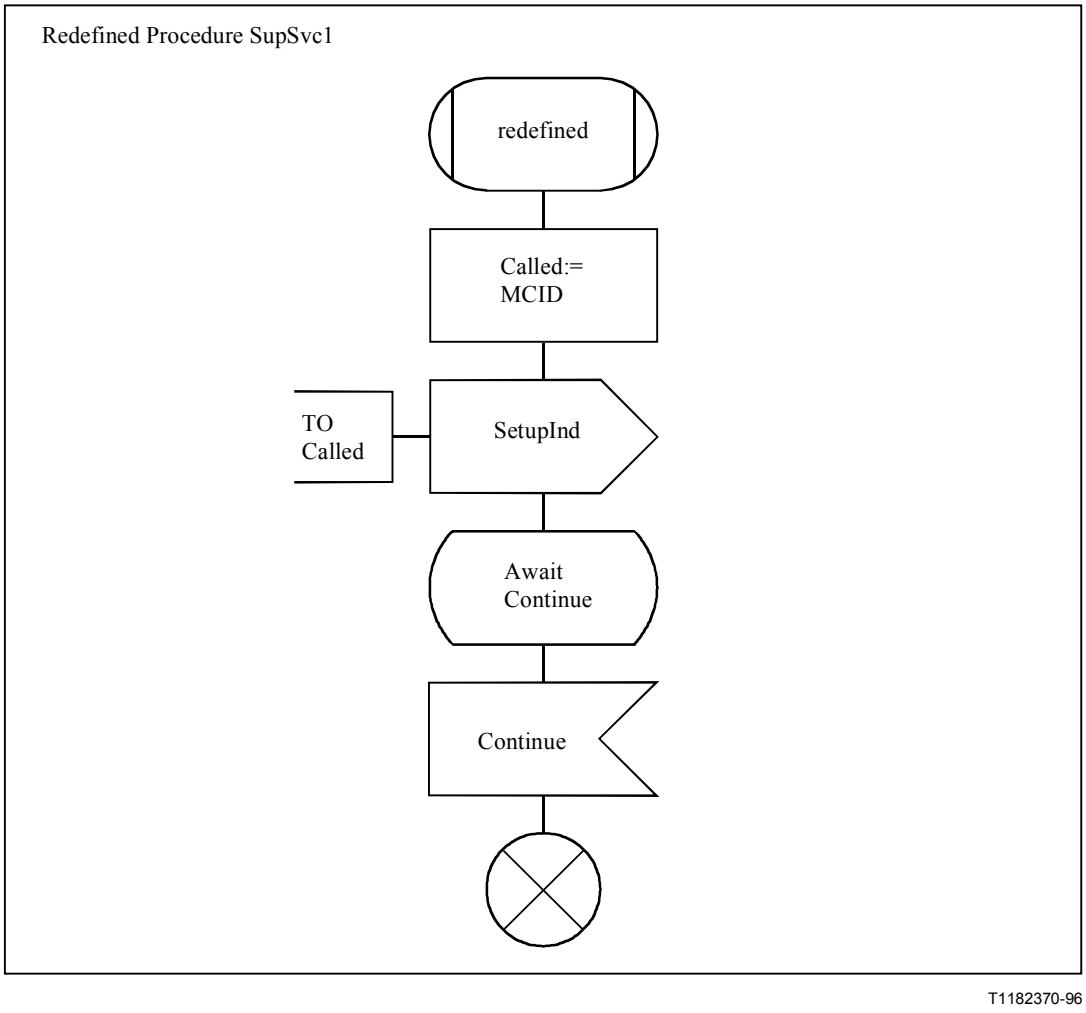


Figure 17/Q.65 – Redéfinition de la procédure SupSvc1 pour le système MCID

2.5.3 Paquetages

Le langage SDL 92 offre la possibilité d'utiliser des types définis dans le système lui-même, ainsi que de types définis dans d'autres systèmes. Ceci est réalisé par le diagramme d'empaquetage servant de conteneur pour des éléments tels que des définitions de type de système, des définitions de type de bloc, des définitions de type de processus, des signaux ou des synonymes.

L'utilisation d'un type défini dans un paquetage est faite en indiquant, dans un symbole de texte situé dans le haut d'un diagramme, le nom du paquetage précédé du mot clé "use". La Figure 18 fournit un exemple de définition de paquetage et d'utilisation d'un type défini dans le paquetage. La figure montre que le type de module B1 du système BCSystem est défini au moyen du type de module Block1_BT dans le paquetage BCBlocks.

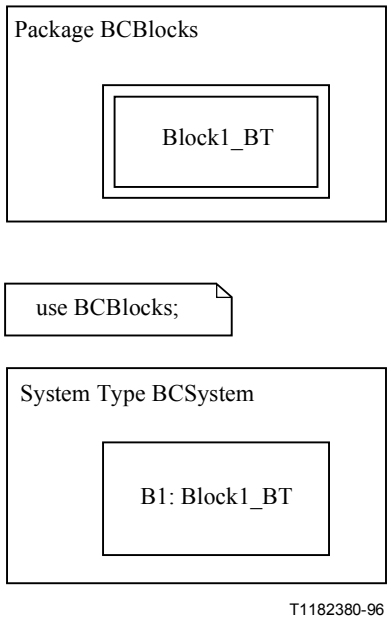


Figure 18/Q.65 – Exemple de définition d'un type de module dans un paquetage et d'utilisation de ce paquetage

2.6 Etape 6 – Allocation des entités fonctionnelles à des emplacements physiques (scénarios)

Dans l'étape 1, un modèle fonctionnel, constitué d'entités fonctionnelles dont chacune possède avec les autres des relations bien définies, est défini pour le service de base et chaque service complémentaire. L'étape 6 se constitue de l'allocation de ces entités fonctionnelles à des emplacements physiques et définit toutes les implémentations physiques intéressantes, appelées scénarios.

Il est possible de décrire plus d'un scénario pour un modèle fonctionnel donné, auquel cas les Administrations auront le choix de l'emplacement à partir duquel sera fourni le service. Une entité fonctionnelle de service complémentaire pourra, par exemple, être localisée dans un commutateur PBX ou dans un centre de commutation.

Il convient de noter les points suivants concernant l'allocation d'entités fonctionnelles:

a) une entité fonctionnelle peut, en principe, être allouée à tout emplacement physique;

b) plusieurs entités fonctionnelles peuvent être allouées à un même emplacement physique;

c) des scénarios réseau prenant en compte la localisation des entités fonctionnelles du service de base doivent être définis pour tout service complémentaire;

d) des emplacements physiques différents pour la localisation d'entités fonctionnelles peuvent impliquer des différences mineures pour les capacités de nœud (par exemple les actions de commutation de l'itinéraire de transmission peuvent dépendre du fait que l'accès se trouve dans un centre de commutation ou un commutateur PBX);

e) les relations entre couples d'entités fonctionnelles du modèle fonctionnel utilisé doivent rester invariantes pour l'ensemble des scénarios recommandés.

Le point e) implique, par exemple, que les flux d'information d'un service complémentaire ne seront pas affectés par la réallocation d'une ou de plusieurs entités fonctionnelles d'un commutateur du réseau public vers un commutateur PBX ou vice versa.

Tous les scénarios identifiés seront pris en considération dans l'étape 3 pour la définition des protocoles de signalisation, des capacités de commutation et des capacités de service.

Les emplacements physiques possibles et les représentations symboliques correspondantes sont les suivantes:

– équipement terminal, type 1 ou adaptateur de terminal: TE;
– terminaison réseau, type 2: NTE (en général dans un PBX);
– commutateur local: CL;
– centre de transit: CT;
– point de commutation du service: SSP;
– point de commande du service: SCP;
– base de données: DB;
– périphérique intelligent: IP.

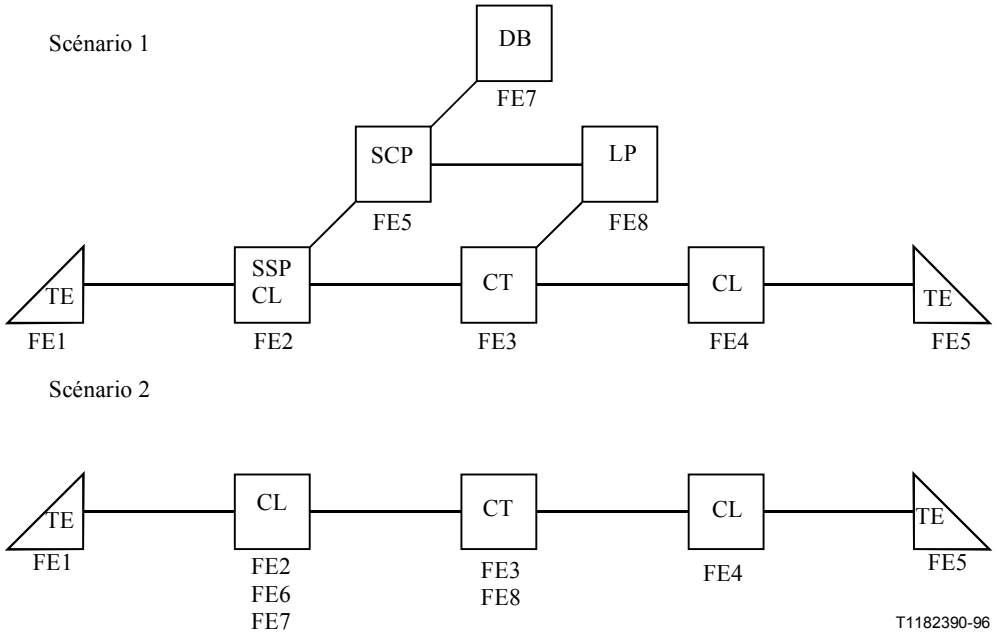


Figure 19/Q.65 – Exemple de deux scénarios allouant des entités fonctionnelles à des emplacements physiques

3 Etapes alternatives employant des techniques orientées objet

Les stades ou les étapes au cours desquels on crée des services ou des applications, depuis leur conception jusqu'à leur implémentation physique (protocole), ont été décrits en détail dans le paragraphe 2. Le présent paragraphe a pour objet de décrire des méthodes alternatives qui emploient des techniques orientées objet. Il est possible que les deux techniques, aussi bien les techniques orientées objet (O-O) que les techniques non O-O, soient utilisées pour décrire les interfaces RI, les méthodes O-O étant employées pour certaines interfaces, tandis que les méthodes non O-O sont employées pour d'autres interfaces. Le résultat consiste en un ensemble de communications API (employant les techniques O-O) et un ensemble de flux d'information (employant les techniques non O-O). Cette situation s'avère confuse même s'il est possible de mapper ces deux résultats sur un protocole d'application de RI par exemple. Si l'on souhaite utiliser les deux méthodes, il est préférable d'employer une méthode pour un ensemble d'interfaces et l'autre méthode pour les interfaces restantes. Les techniques O-O sont utilisées dans le secteur informatique depuis un certain moment déjà et il convient donc que nous donnions des exemples sur la manière dont ces méthodes peuvent être utilisées pour renforcer le développement de réseaux intelligents et faciliter leur intégration dans les environnements informatiques répartis. Différents ensembles de techniques O-O peuvent être utilisés, certains étant meilleurs que d'autres. Le présent paragraphe décrit le langage de modélisation unifié (UML), adapté à la description des alternatives des étapes 2 à 4 de la méthodologie fonctionnelle unifiée. L'étape 1 reste inchangée.

3.1 Alternative de l'étape 2

Applications aux paquetages [facultatives] (voir Figure 1)

En langage UML, le paquetage est un mécanisme général permettant d'organiser les éléments de modélisation par groupe. Dans notre méthodologie, l'application ou le service est construit à l'aide d'un certain nombre de paquetages. Les paquetages sont employés pour regrouper les éléments de modélisation dans des ensembles plus grands qui peuvent être traités comme des groupes. Des paquetages bien conçus regroupent des éléments sémantiquement équivalents qui ont tendance à évoluer comme un groupe.

Chaque paquetage doit avoir un nom qui le différencie des autres paquetages. Un *nom* est une chaîne de caractères. Ce nom seul se nomme *nom simple*; un *nom de chemin* est le nom du paquetage précédé le cas échéant du nom du paquetage dans lequel ce paquetage réside. Un paquetage est généralement représenté repéré par son seul nom, comme dans la Figure 20. Comme pour les classes, les paquetages peuvent être complétés par des valeurs étiquetées ou par des compartiments supplémentaires qui en donnent le détail.

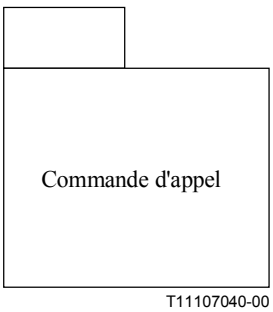


Figure 20/Q.65 – Exemple de paquetage

La division proprement dite en paquetages d'une application ou d'un service peut varier. Certaines applications peuvent comporter de nombreux paquetages, tandis que d'autres n'en comportent qu'un seul. L'application X, dans la Figure 21, peut par exemple contenir des paquetages de commande d'appel, d'authentification et de messagerie, tandis qu'une application de sécurité peut ne contenir qu'un paquetage d'authentification. Les paquetages peuvent être influencés par différentes applications. L'application X peut par exemple nécessiter les sous-éléments d'authentification a, b et c, tandis que l'application Y nécessite les sous-éléments b et d, le paquetage d'authentification résultant comportant en conséquence les sous-éléments a, b, c, et d (voir Figure 21).

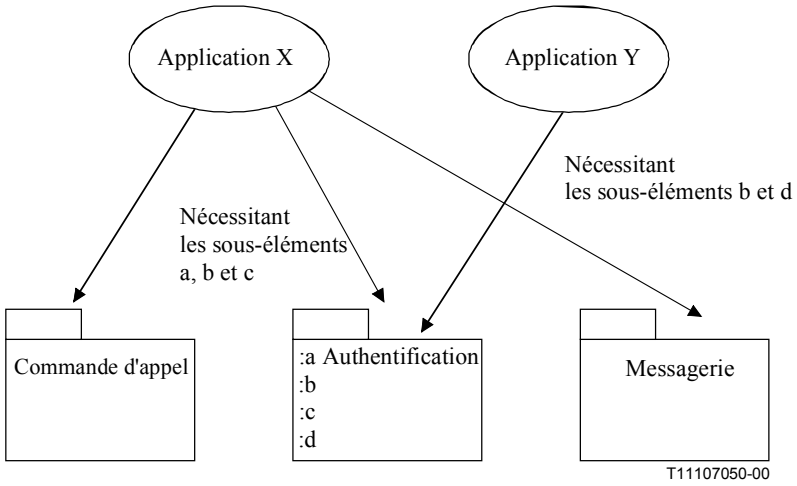


Figure 21/Q.65 – Mappage des paquetages sur des applications

La méthode des paquetages est un moyen qui permet de décrire les secteurs devant être examinés afin de produire les classes d'interfaces nécessaires, et de regrouper un certain nombre de classes qui sont nécessaires pour une interface particulière afin d'assurer le service ou l'application particulière correspondante (voir Appendice III.3.1 pour de plus amples détails sur ce point).

NOTE – Il est possible d'omettre l'étape concernant les paquetages lorsque les classes d'interfaces permettant d'assurer le service ou l'application sont connues.

3.2 Alternative de l'étape 3

Classes d'interfaces et diagrammes de classes

3.2.1 Classes d'interfaces

Une interface concerne un ensemble portant un nom d'opérations qui sont employées pour spécifier un service d'une classe ou d'une composante. A la différence des classes ou des types, les interfaces ne spécifient ni la structure (elles peuvent donc ne pas comporter d'attribut), ni l'implémentation (elles peuvent donc ne pas comporter de méthode qui permette d'effectuer une opération). Comme les classes, les interfaces peuvent comporter un nombre quelconque d'opérations. Ces opérations peuvent encore posséder des propriétés de visibilité, des propriétés de concomitance, des stéréotypes, des marques ou des contraintes.

Les opérations peuvent être repérées par leur nom seulement, ou on peut ajouter leur signature complète et d'autres propriétés, comme indiqué dans la Figure 22.

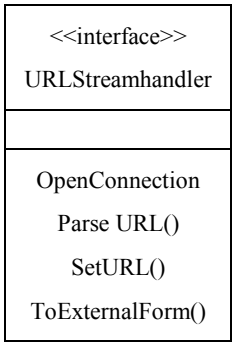


Figure 22/Q.65 – Opérations dans une classe d'interfaces

Une classe d'objets n'est pas décrite, parce que cela fournirait des détails sur l'implémentation relatifs à cette classe, à savoir cela spécifierait comment le système applique la méthode qui a été appelée. Les classes d'interfaces ne décrivent pas les méthodes propres à l'implémentation. Elles peuvent donc être implémentées dans des interfaces physiques de types différents et peuvent être importées d'autres systèmes pour une interface particulière, sans compromettre la procédure qui est employée pour appliquer la méthode invoquée.

Selon la fonctionnalité, les classes d'interfaces peuvent être décrites par paires, la "classe d'interfaces" et la "classe d'interfaces d'application". La "classe d'interfaces de base" réside dans le système d'un côté de l'interface, tandis que l'"application" est de l'autre côté. Généralement, les méthodes appelées par "l'application" nécessitent un résultat qui est implémenté par la "classe d'interfaces de base".

Deux types utiles de classes d'interfaces sont employés pour décrire un système: la classe d'interfaces de "cadre" et la classe d'interfaces de "service générique".

3.2.1.1 Classes d'interfaces de cadre

Les classes d'interfaces de cadre sont des classes qui décrivent un modèle extensible pour les applications dans un domaine. Un cadre est un genre de microarchitecture qui englobe un ensemble de mécanismes travaillant ensemble à la résolution d'un problème commun dans un domaine commun.

Il convient donc de se représenter une "classe d'interfaces de cadre" comme étant une classe qui est presque toujours utilisée de part et d'autre d'une interface, quelle que soit la classe de service qui est par ailleurs appelée. Un exemple en est "Authentication". Il est presque certain que toute communication à travers une interface doit initialement procéder à l'authentification d'une extrémité du système par l'autre extrémité. "Event Notification" et "Integrity Management" constituent d'autres exemples.

Les classes d'interfaces de cadre peuvent presque toujours être décrites par paires, par exemple "Iauthentication" et "IAppAuthentication", comme décrit au 3.2.1, où le "I" qui précède indique qu'il s'agit d'une "interface de classes" et non d'une "classe d'objets".

3.2.1.2 Classes d'interfaces de service générique

Les classes d'interfaces de service générique sont des classes qui satisfont aux spécifications de service de l'interface en question. Une classe de service générique est d'une manière générale "Call Control Manager" par exemple. Les paires résultantes de classes d'interfaces sont "ICallControlManager" et "IappCallControlManager". Il est possible de représenter cette classe de service générique comme un "paquetage".

Les deux différents types de classes d'interfaces doivent être définis de manière détaillée, en d'autres termes il faut les remplir à l'aide des appels de leurs méthodes respectives. Mais avant cette étape finale, il faut décrire les relations entre les classes d'interfaces et cela se fait à l'aide des "diagrammes de classes".

3.2.2 Diagrammes de classes

Les relations entre les classes d'interfaces peuvent être représentées au moyen de "diagrammes de classes" (voir Figure 23).

Un diagramme de classes est un diagramme où est représenté un ensemble de classes, d'interfaces et de collaborations, ainsi que les relations qui existent entre elles. On peut employer des diagrammes de classes pour visualiser de manière statique la conception d'un système. Cette visualisation tient essentiellement compte des spécifications fonctionnelles d'un système, à savoir les services que le système doit fournir aux utilisateurs.

Les diagrammes de classes contiennent habituellement les éléments suivants:

- *classe*: une classe décrit un ensemble d'objets qui ont les mêmes attributs, concernent les mêmes opérations, ont les mêmes relations et la même sémantique. Une classe fait intervenir une ou plusieurs interfaces;
- *interface*: une interface concerne un ensemble d'opérations qui sont utilisées pour spécifier un service d'une classe ou d'une composante;
- *collaboration*: une collaboration est un ensemble de classes, d'interfaces ou d'autres éléments qui travaillent conjointement pour aboutir à un certain comportement coopératif supérieur à la somme des comportements individuels;
- *relation de dépendance, de généralisation ou d'association*: une dépendance est une relation indiquant qu'une modification de la spécification d'un élément peut avoir des effets sur un autre élément qui emploie le premier, mais que le contraire n'est pas nécessaire. Une généralisation est une relation entre un élément général (nommé la superclasse ou le parent) et un type particulier d'un tel élément (nommé sous-classe ou enfant). Par exemple, on peut avoir la classe générale "commande d'appel" et le type particulier "commande d'appel du protocole d'application du RI". Une association est une relation structurelle qui décrit un ensemble de liaisons, une liaison étant une connexion entre objets.

Les diagrammes de classes sont les diagrammes les plus courants dans la modélisation des systèmes orientés objet. Ils jouent un rôle important non seulement dans la visualisation, la spécification et la documentation des modèles structurels, mais aussi dans la construction, à l'aide de l'ingénierie procédant vers l'avant ou vers l'arrière, de systèmes susceptibles d'être exécutés.

Les diagrammes de classes décrivent les types d'objets (implémentations particulières) ou les classes d'interfaces du système et les divers types de relations statiques qui existent entre eux. Les principales relations statiques sont au nombre de deux:

- associations;
- sous-types.

Les diagrammes de classes représentent aussi les attributs et les opérations d'une classe, ainsi que les contraintes qui s'appliquent à la façon dont sont connectés les objets. Dans la présente Recommandation, nous ne considérons pas les attributs parce que ceux-ci dépendent de l'implémentation.

Les associations représentent des relations entre des instances de classes. Des exemples en sont une personne qui travaille pour une société, ou une société qui a un certain nombre de bureaux. Le premier exemple est illustré dans la figure suivante:

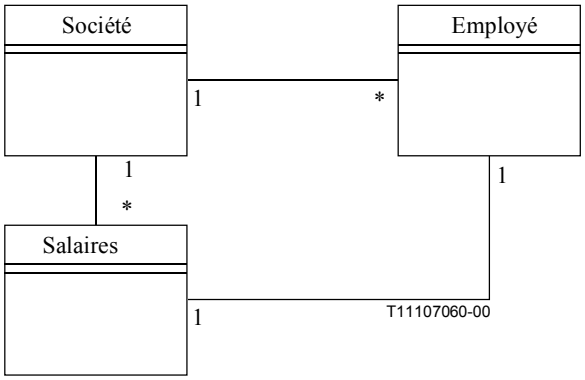


Figure 23/Q.65 – Associations d'instances de classes

La Figure 23 montre la relation entre une société, ses employés et leurs salaires. Dans cette relation, la société a de nombreux employés et un employé reçoit un salaire. En d'autres termes, un salaire est affecté à un employé et un employé n'a qu'un seul employeur.

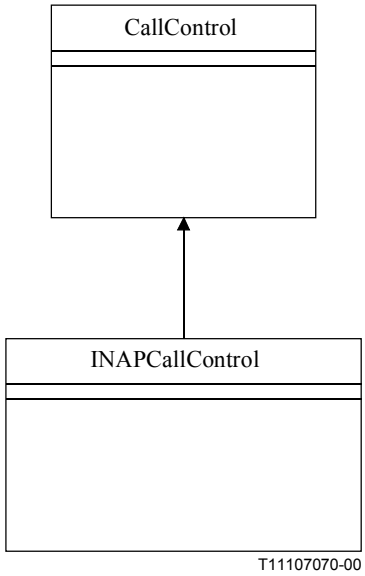


Figure 24/Q.65 – Utilisation des sous-types ou des sous-classes

La Figure 24 explique l'utilisation des sous-types ou des sous-classes. Ici, la commande d'appel est le service générique, tandis que "INAPCallControl" est un type de commande d'appel propre au système associé au protocole d'application du RI (INAP, *intelligent network application protocol*).

3.2.3 Diagrammes de séquences d'interfaces (facultatifs)

L'utilisation de diagrammes de séquences d'interfaces est facultative. La nature de ces diagrammes est semblable à celle des "diagrammes de flux d'information" du 2.3.1. Les diagrammes de séquences d'interfaces peuvent toutefois aussi indiquer les flux de communications API (appels de méthode) (voir 3.3 pour plus d'explication), chaque côté de l'interface étant représenté. Puisque les communications API dans chaque système sont susceptibles d'être propriétaires, on peut restreindre ou omettre l'utilisation de ces diagrammes dans une Recommandation.

Les diagrammes de séquences d'interfaces représentent graphiquement les appels de méthode (communications API) qui traversent une interface particulière, en insistant sur l'ordre dans le temps des messages. Du point de vue des techniques O-O, ils représentent les appels de méthode entre différentes classes d'interfaces. Ce faisant, certains de ces appels de méthode peuvent en fait ne pas traverser une interface mais rester à l'intérieur d'un système particulier. Mais même dans ce cas, pour être complet, on doit indiquer tous les appels de méthode.

Comme le montre la Figure 25, on forme un diagramme de séquences en plaçant les classes d'interfaces (objets) qui participent à l'interaction au sommet du diagramme suivant l'axe X. Généralement, on place à gauche l'objet qui est à l'origine de l'interaction, puis, à sa droite, les objets qui lui sont de plus en plus subordonnés. Ensuite, on place suivant l'axe Y les messages que ces objets envoient et reçoivent, de haut en bas, par ordre croissant dans le temps. Cela permet au lecteur d'avoir une image claire du flux de commandes dans le temps.

Les diagrammes de séquences ont deux caractéristiques qui les distinguent des diagrammes de collaborations. Il y a d'abord l'objet ligne de vie. C'est la ligne verticale qui représente l'existence d'un objet dans le temps. Des objets peuvent être créés au cours de l'interaction, et ils commencent à exister lors de la réception d'un message "nouveau". On peut supprimer des objets de la même manière.

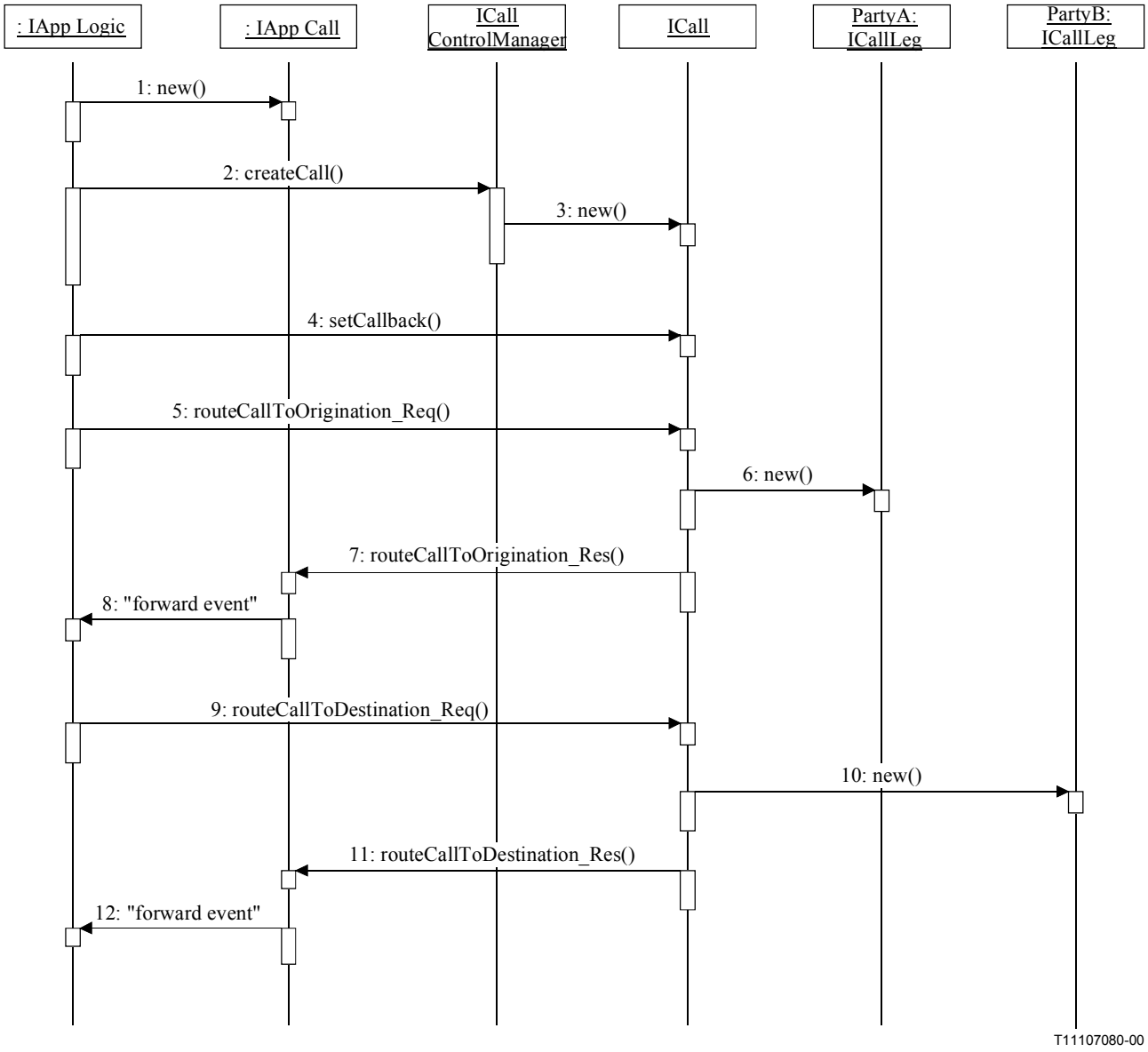


Figure 25/Q.65 – Exemple de diagramme de séquences d'interfaces

Le scénario qui est illustré ci-dessus représente un "appel lancé par un tiers", qui est décrit de manière plus détaillée à l'Appendice III. Dans cet exemple, l'interface proprement dite est située entre les objets IappCall et Icall ControlManager. Les appels de méthode 2, 4, 5, 7, 9 et 11 se manifestent en conséquence réellement en tant que communications API à travers l'interface physique. Les autres appels de méthode sont à l'intérieur des systèmes des deux côtés de l'interface, et donc il n'est pas toujours nécessaire d'indiquer le résultat de l'objet "ciblé" puisque cela peut ne pas avoir d'effet sur l'interface elle-même. Cet exemple nous montre que les communications API qui sont nécessaires pour assurer "l'établissement de la communication lancée par un tiers" pour le service ou l'application sont les communications API 2, 4, 5, 7, 9 et 11.

Plus il y a de services et d'adresses, plus le nombre de classes d'interfaces examiné est grand, et plus la spécification API augmente.

3.3 Alternative de l'étape 4

Descriptions API en langage IDL

Après avoir suivi les deux étapes précédentes, et identifié les classes d'interfaces impliquées dans les services ou les applications qui sont nécessaires dans l'interface dont on dispose, il faut aborder l'étape suivante qui consiste à décrire les appels de méthode en détail. Il faut se souvenir que les appels de méthode sont les messages qui sont envoyés d'une classe d'interfaces à une autre, et que pour cette raison chaque classe d'interfaces ou d'objets doit savoir quelle information il pourrait recevoir. Les appels de méthode ou communications API doivent donc être décrites à l'aide d'un format dont il faut convenir, de façon que les communications API soient normalisées, et que chaque système puisse savoir quelle information il aura à traiter. La présente Recommandation préconise l'utilisation d'un langage de description d'interface générique (IDL, *interface description language*) pour décrire les communications API. Les avantages d'un tel format sont que cette description d'interface peut aisément être convertie en une solution quelconque propre à l'implémentation telle que les architectures CORBA, les protocoles INAP, les applications JAVA.

Le langage IDL détaille chacun des appels de méthode et décrit les paramètres de données qu'ils acheminent. Un exemple est donné ci-après:

```
routeCallToDestination_Req(callSessionID : in TSessionID, responseRequested : in
    TCallResponseRequest, targetAddress : in TAddress, originatingAddress : in TAddress,
    originalDestinationAddress : in TAddress, redirectingAddress : in TAddress, appInfo :
    in TCallAppInfoArr) : TResult
```

Dans cet exemple, le nom de la communication API figure au début (routeCallToDestination_Req). D'autres paramètres sont indiqués en italique de manière à faire apparaître comment la méthode permet les subdivisions. Par exemple, le paramètre *callSessionID : in TsessionID* spécifie l'identificateur de la session de la communication. Cet identificateur qui est spécifié par: *in TsessionID*, est donc défini comme étant un identificateur *sessionID* de norme "type". La première partie du paramètre explique ce qui est acheminé par la méthode, tandis que la seconde partie qui suit le point-virgule indique le format qui est utilisé pour l'acheminement. Les détails précis de ce format ne font pas l'objet de la présente Recommandation.

Afin d'achever le processus de modélisation, il faut revenir à l'étape 5 au 2.5.

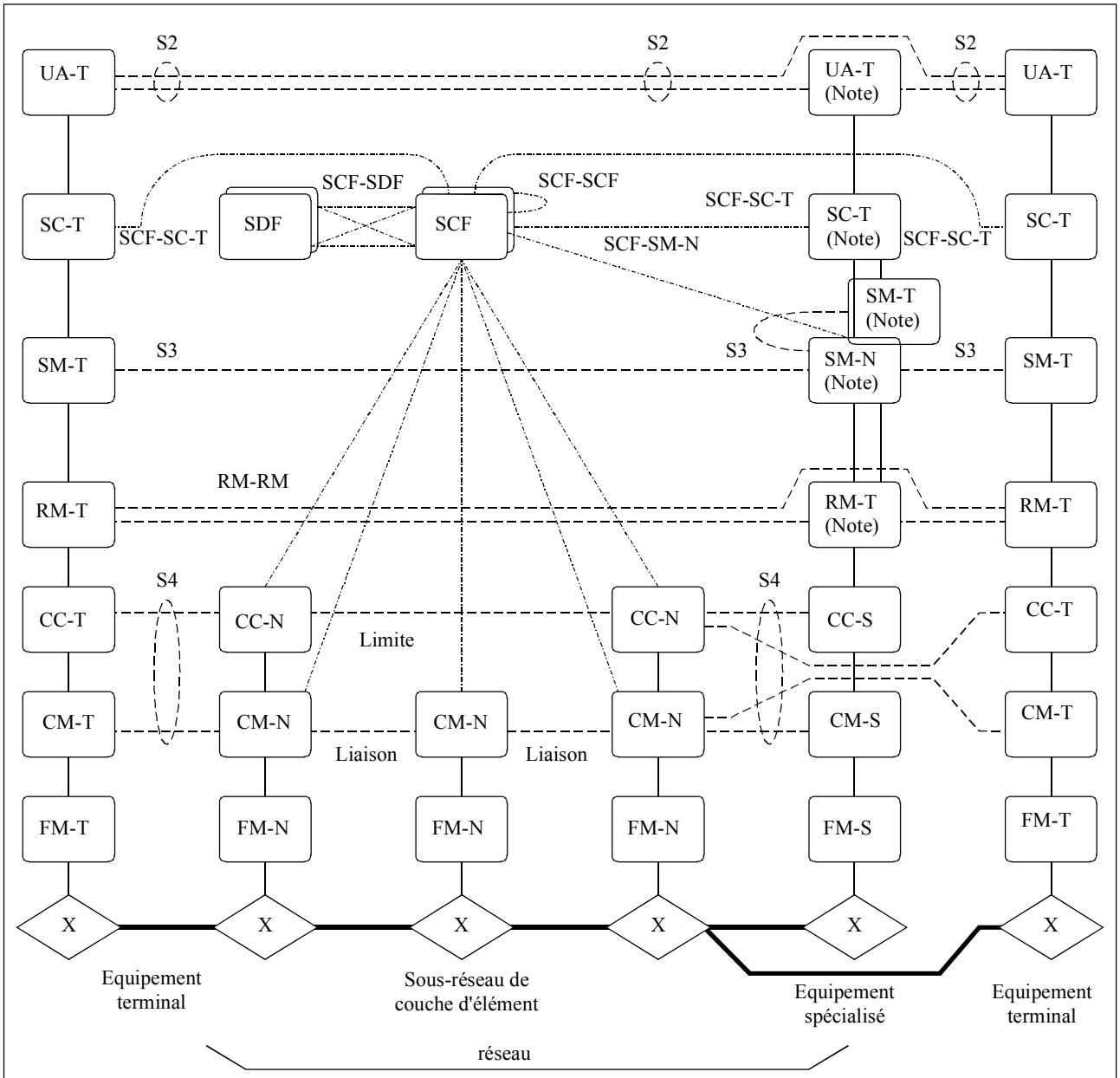
APPENDICE I

Format et table des matières d'une description de la phase 2 utilisant la méthodologie fonctionnelle unifiée

APPENDICE II

Architecture fonctionnelle – Evolution de la Recommandation Q.65

Ce diagramme présente un modèle d'entité fonctionnelle, ne s'inscrivant ni dans le cadre d'une réalisation physique particulière, ni dans le cadre d'une architecture donnée. L'entité "CM-N" indiquée correspond simplement à la composante de commande de connexion en temps réel figurant dans la couche de gestion de réseau de l'architecture fonctionnelle unifiée. L'entité "FM-N" indiquée correspond simplement à la composante de commande des ressources matérielles en temps réel figurant dans la couche de gestion d'élément de l'architecture fonctionnelle unifiée.



UA fonction d'application d'utilisateur, SC fonction de commande de service, SCF fonction de commande de service,
SDF fonction de données de service, SM fonction de gestion de session, RM fonction de gestion de ressources,
CC fonction de commande d'appel, CM fonction de gestion des connexions, FM fonction de gestion des ressources matérielles

NOTE – Ces fonctions peuvent être déclenchées par la fonction SCF. En pareil cas, les relations en direction de ces fonctions aboutiront à la fonction SCF.

La signalisation S2 est écoulée par le circuit SVC de l'utilisateur; la signalisation S3 peut être écoulée par le circuit SVC de signalisation ou le circuit SVC de l'utilisateur; la signalisation S4 est écoulée par le circuit SVC de signalisation. L'équipement spécialisé peut être relié au nœud serveur (SN) ou aux passerelles.

T1183760-97

Figure II.1/Q.65 – Modèle fonctionnel unifié

APPENDICE III

Exemples de classes d'interfaces formées à partir des spécifications relatives au service ou à l'application

III.1 Introduction

Les exemples suivants de classes d'interfaces sont donnés en vue d'aider le lecteur à définir un système. Ils ne doivent être considérés qu'à titre indicatif et non comme étant réels.

La méthodologie qui a fait l'objet d'une explication au paragraphe 3 est employée ici pour montrer comment on déduit la communication API appropriée pour une interface particulière. Nous décrirons un type de service ou d'application et identifierons les classes d'interfaces résultantes. Nous remplirons également les classes d'interfaces avec les flux d'information (appels de méthode) que nous identifions.

III.2 Exemple relatif au service ou à l'application

III.2.1 Appel lancé par un tiers

Supposons, à titre d'exemple, qu'un fournisseur de services extérieur demande l'établissement d'une communication entre deux entités A et B. Il doit donc en faire part à un opérateur de réseau, probablement à la fonction de commande de service (SCF). A ces fins, nous devons identifier les classes d'interfaces qui sont chargées de communiquer ces spécifications. Nous devons donc analyser le service dont nous avons besoin afin de créer deux branches séparées, d'établir une communication entre elles et d'être en mesure de gérer la communication. La première spécification consiste par conséquent à fournir un "**gestionnaire de commande d'appel**". Cette classe d'interfaces est chargée d'établir une communication unique ou des communications multiples. Puisqu'elle peut gérer plusieurs appels, nous devons disposer d'une classe d'interfaces propre à chaque appel individuel et nous devons donc fournir une classe d'interfaces "**commande d'appel**". Comme chaque communication possède des branches qui lui sont associées et que chaque branche est indépendante, nous devons fournir par branche une classe d'interfaces "**branche de communication**". L'examen de notre première analyse du service indique que nous avons jusqu'à présent les classes d'interfaces suivantes:

- gestionnaire de commande d'appel
- commande d'appel
- branche de communication [entité A]
- branche de communication [entité B]

Ces classes d'interfaces, comme décrit auparavant dans le paragraphe 3, sont implémentées par paires, un membre de la paire étant situé dans l'application, tandis que l'autre membre est situé dans le réseau en tant que tel. La terminologie permettant de décrire cela est la suivante:

Classes d'interfaces d'application	Classes d'interfaces côté réseau
IAppCallControlManager	**IcallControlManager**
IAppCAll	**Icall**
Entité A: IAppCallLeg	**Entité A: IcallLeg**
Entité B: IAppCallLeg	**Entité B: IcallLeg**

Il convient de noter que le "I" qui précède le nom indique que nous nous référons ici à la "classe d'interfaces" et non à la classe d'objets. Les classes d'interfaces n'ont aucun attribut qui leur est associé, comme c'est le cas pour les classes d'objets, ce qui rend celles-ci dépendantes de l'implémentation.

A ce point, il est maintenant profitable d'aborder les diagrammes de classes, qui sont importants du point de vue du système.

Le concept de paquetage que nous avons déjà introduit dans la présente Recommandation est examiné de façon plus approfondie ici.

Le "paquetage" importe lorsqu'on divise son "interface" en parties pertinentes. On peut par exemple vouloir regrouper tous les aspects de la commande d'appel dans un paquetage, afin de permettre la réplication de ces classes d'interfaces dans différentes interfaces, sans avoir à rechercher la même spécification à chaque fois. Le paragraphe suivant traite des diagrammes de classes du service de commande d'appel et donne la relation qui existe entre les paquetages et les classes d'interfaces.

III.3 Diagrammes de classes

III.3.1 Diagramme de classes du service générique de commande d'appel

Le paquetage d'application du service générique de commande d'appel représenté dans la Figure III.1 qui est nommé *IAppGCCS* comporte une interface *IAppCallControlManager*, zéro ou plusieurs interfaces *IAppCall* et zéro ou plusieurs interfaces *IAppCallLeg*.

Le paquetage du service générique de commande d'appel représenté dans cette même figure qui est nommé *IGCCS* comporte une interface *ICallControlManager*, zéro ou plusieurs interfaces *ICall* et zéro ou plusieurs interfaces *ICallLeg*.

Figure III.1/Q.65 – Paquetages du service générique de commande d'appel

Le diagramme de classes représenté dans la Figure III.2 montre les interfaces qui forment le paquetage d'application du service générique de commande d'appel et le paquetage du service générique de commande d'appel. La communication entre ces paquetages se fait par l'intermédiaire des voies *+uses the ICallControlManager*, *+uses the ICall* et *+uses the ICallLeg*.

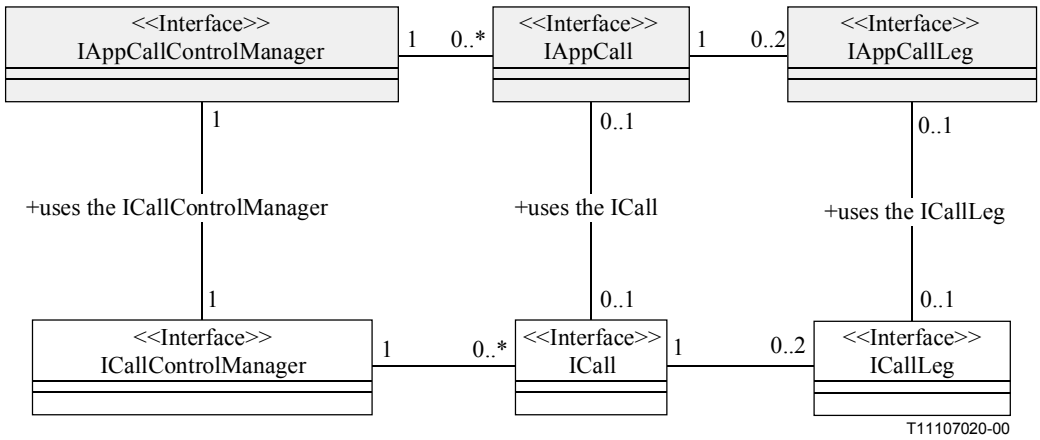


Figure III.2/Q.65 – Diagramme de classes du service générique de commande d'appel

III.3.2 Appels de méthode

Il est maintenant nécessaire de remplir les classes au moyen d'opérations, ou "d'appels de méthode" comme on les nomme. Ce sont des messages qui peuvent être envoyés à la classe pour invoquer un certain résultat. Considérons d'abord les appels ICallControlManager.

III.3.2.1 Appels de méthode ICallControlManager

Le client doit d'abord indiquer au réseau qu'il veut lancer un appel, donc la première méthode est la suivante:

- "createCall".

Ceci est actuellement probablement suffisant. Il convient toutefois de noter que si un nombre plus élevé de services ou d'applications est examiné, de nouvelles méthodes seraient identifiées.

Il faut aborder ensuite la classe d'interfaces Call.

III.3.2.2 Appels de méthode Icall

Les méthodes suivantes sont profitables:

- routeCallToDestination_Req;
- routeCallToOrigination_Req.

Pour le moment, ces méthodes suffisent probablement. Il faut examiner enfin la classe CallLeg.

III.3.2.3 Appels de méthodes IcallLeg

Les méthodes suivantes sont profitables:

- routeCallLegToAddress;
- releaseCallLeg.

Ayant donné ci-dessus une définition provisoire des classes d'interfaces, il faut à présent décrire en détail le langage de description d'interface qui permet d'effectuer ces appels de méthode.

III.3.3 IDL

Seules deux classes d'interfaces seront traitées à titre d'exemple. Ce processus doit être répété pour chacune des classes.

Les classes d'interfaces suivantes sont à nouveau données à titre d'exemple seulement:

III.3.3.1 Langage IDL concernant la classe ICallControlManager

Classe d'interfaces

<<Interface>> ICallControlManager
createCall(appCall : in TIAppCallRef, call : out TICallRefRef, callSessionID : out TSessionIDRef) : Tresult

Figure III.3/Q.65

La méthode createCall et les paramètres associés sont décrits ci-après.

Méthode

`createCall()`

Cette méthode est utilisée pour créer un nouvel objet d'appel.

Paramètres

`appCall: in TIAppCallRef`

Paramètre spécifiant l'interface d'application pour les rappels automatiques en provenance de l'appel lancé.

`call: out TICallRefRef`

Paramètre spécifiant la référence de l'interface de l'appel lancé.

`callSessionID: out TSessionIDRef`

Paramètre spécifiant l'identificateur de la session de communication de l'appel lancé.

III.3.3.2 Langage IDL concernant la classe ICall

Un exemple de classe d'interfaces "ICall" est donné dans la Figure III.4.

Classe d'interfaces

<<Interface>> ICall
routeCallToDestination_Req(callSessionID : in TSessionID, responseRequested : in TCallResponseRequest, targetAddress : in TAddress, originatingAddress : in TAddress, originalDestinationAddress : in TAddress, redirectingAddress : in TAddress, appInfo : in TCallAppInfoArr) : TResult routeCallToOrigination_Req(callSessionID : in TSessionID, responseRequested : in TCallResponseRequest, targetAddress : in TAddress , originatingAddress : in TAddress, appInfo : in TCallAppInfoArr) : Tresult

Figure III.4/Q.65

Les méthodes routeCallToDestination_Req et routeCallToOrigination_Req ainsi que les paramètres associés sont décrits ci-après.

Méthode

routeCallToDestination_Req()

Cette méthode asynchrone demande le routage de l'appel (et des parties qui lui sont associées par leur nature même) vers l'entité d'arrivée, par l'intermédiaire d'une branche de communication passive (qui est implicitement créée).

Paramètres

callSessionID: in TSessionID

Paramètre spécifiant l'identificateur de la session de communication de l'appel.

responseRequested: in TCallResponseRequest

Paramètre spécifiant l'ensemble des événements observés qui conduisent à la production du paramètre **routeCallToDestination_Res**.

targetAddress: in TAddress

Paramètre spécifiant l'entité d'arrivée vers laquelle l'appel doit être acheminé.

originatingAddress: in TAddress

Paramètre spécifiant l'adresse de l'entité (appelante) de départ.

originalDestinationAddress: in TAddress

Paramètre spécifiant l'adresse primitive d'arrivée de l'appel.

redirectingAddress: in TAddress

Paramètre spécifiant la dernière adresse d'où l'appel à été réorienté.

appInfo: in TCallAppInfoArr

Paramètre spécifiant des informations relatives à l'application qui sont pertinentes pour l'appel (telles que la méthode d'alerte, le type de téléservice, les identités des services, les indicateurs d'interaction).

Méthode

routeCallToOrigination_Req()

Cette méthode asynchrone demande à la première entité appelante le routage d'un appel par l'intermédiaire d'une branche de communication de commande (qui est implicitement créée). L'objet d'appel doit déjà avoir été créé.

Paramètres

callSessionID: in TSessionID

Paramètre spécifiant l'identificateur de la session de communication de l'appel.

responseRequested : in TCallResponseRequest

Paramètre spécifiant l'ensemble des événements observés qui conduisent à la production du paramètre **routeCallToOrigination_Res()**.

targetAddress: in TAddress

Paramètre spécifiant l'entité d'arrivée vers laquelle l'appel doit être routé.

originatingAddress: in TAddress

Paramètre spécifiant l'adresse de l'entité (appelante) de départ.

AppInfo: in TCallAppInfoArr

Paramètre spécifiant des informations relatives à l'application qui sont pertinentes pour l'appel (telles que la méthode d'alerte, le type de téléservice, les identités des services, les indicateurs d'interaction).

III.4 Perspectives

L'Appendice III a permis d'expliquer comment l'outil que constitue le modèle UML est employé pour spécifier des services ou des applications. Nous avons ainsi atteint un point semblable à celui de la méthode alternative (à l'aide de modules SIB) où il faut mapper les résultats sur un protocole. Comme dans la démarche SIB, nous devrions maintenant aborder directement les diagrammes SDL afin de définir le système concernant lequel ces méthodes sont appelées. Deux options sont possibles ici:

1) nous pouvons introduire les résultats obtenus ci-dessus dans la machine SDL et produire le diagramme SDL approprié;

2) nous pouvons spécifier le langage IDL d'une manière qui est appropriée à l'implémentation, telle que celle qui consiste à décider d'utiliser une solution CORBA ou une autre solution. Après avoir effectué ce mappage direct, il est encore possible de mapper directement, à l'aide d'un moyen approprié, la solution IDL CORBA sur le diagramme SDL.

Ces deux options ne font pas l'objet de la présente Recommandation.

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

Genève, 2001