UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

Z.100

Supplément 1

(05/97)

SÉRIE Z: LANGAGES DE PROGRAMMATION

Techniques de description formelle – Langage de description et de spécification (SDL)

Méthodologie du langage SDL+: Utilisation des diagrammes de séquences de messages MSC avec le langage SDL muni de l'ASN.1

Recommandation UIT-T Z.100 – Supplément 1

(Antérieurement Recommandation du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE Z

LANGAGES DE PROGRAMMATION

TECHNIQUES DE DESCRIPTION FORMELLE	Z.100–Z.199
Langage de description et de spécification (SDL)	**Z.100–Z.109**
Application des techniques de description formelle	Z.110–Z.119
Diagrammes des séquences de messages	Z.120–Z.129
LANGAGES DE PROGRAMMATION	Z.200–Z.299
CHILL: le langage de haut niveau de l'UIT-T	Z.200–Z.209
LANGAGE HOMME-MACHINE	Z.300–Z.499
Principes généraux	Z.300–Z.309
Syntaxe de base et procédures de dialogue	Z.310–Z.319
LHM étendu pour terminaux à écrans de visualisation	Z.320–Z.329
Spécification de l'interface homme-machine	Z.330–Z.399
QUALITÉ DES LOGICIELS DE TÉLÉCOMMUNICATION	Z.400–Z.499
MÉTHODES DE VALIDATION ET D'ESSAI	Z.500–Z.599

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

SUPPLÉMENT 1 À LA RECOMMANDATION UIT-T Z.100

MÉTHODOLOGIE DU LANGAGE SDL+: UTILISATION DES DIAGRAMMES DE SÉQUENCES DE MESSAGES MSC AVEC LE LANGAGE SDL MUNI DE L'ASN.1

Résumé

Le présent Supplément est publié sous forme de Supplément à la Recommandation Z.100 (SDL), mais il concerne également les Recommandations Z.105 (SDL avec ASN.1) et Z.120 (diagrammes MSC). Il décrit une méthodologie permettant d'utiliser ces langages en combinaison.

Le présent Supplément traite des sujets suivants:

1) termes et définitions ainsi que références sur l'utilisation et l'application du langage SDL;
2) aperçu général de la méthodologie (paragraphe 4);
3) descriptions plus détaillées des opérations suivantes:
   a) analyse des exigences (paragraphe 5);
   b) conception du projet et recherche de l'application (paragraphe 6);
   c) formalisation de l'application en SDL+ (paragraphe 7);
   d) questions relatives à la mise en œuvre (paragraphe 8);
   e) validation (paragraphe 9);
4) relation avec d'autres langages et techniques;
5) élaboration de la méthodologie pour la spécification des services.

Le présent Supplément n'est pas exhaustif. Il est destiné à être incorporé, par les utilisateurs du langage SDL+, dans leurs méthodologies globales et à être adapté à leurs systèmes d'application ainsi qu'à leurs exigences spécifiques. Ce Supplément, en particulier, ne couvre pas en détail les questions de calcul d'une mise en œuvre à partir de la spécification ou des essais des systèmes. Dans le cas des essais, il est prévu d'en faire l'objet partiel d'un document séparé, traitant de la production de tests pour les normes ou produits fondés sur le SDL+.

La méthodologie est un cadre qui doit être élaboré pour le contexte d'utilisation normale. Le présent Supplément comporte deux parties. La Partie I donne un aperçu général du cadre de travail (paragraphe 4), avec une description plus détaillée des parties principales dans les paragraphes 5, 6 et 7. Le cadre général est ensuite spécialisé dans la Partie II pour la spécification des services, à partir du paragraphe 12. Les parties indiquées aux paragraphes 5, 6 et 7 sont ensuite développées dans les paragraphes 13, 14 et 15, respectivement. Dans cette spécialisation, l'on a fait certains choix méthodologiques. Un grand nombre des détails indiqués dans les paragraphes 13, 14 et 15 pourront être utilisés lorsque le cadre général sera élaboré pour d'autres contextes et pour d'autres choix.

Source

Le Supplément 1 à la Recommandation UIT-T Z.100, élaboré par la Commission d'études 10 (1997-2000) de l'UIT-T, a été approuvé le 6 mai 1997 selon la procédure définie dans la Résolution n° 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'études 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 n° 1 de la CMNT.

Dans certains secteurs de la technologie 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

Supplément 1 à la Recommandation Z.100

MÉTHODOLOGIE DU LANGAGE SDL+: UTILISATION DES DIAGRAMMES DE SÉQUENCES DE MESSAGES MSC AVEC LE LANGAGE SDL MUNI DE L'ASN.1

(Genève, 1997)

1 Contexte

Le présent Supplément est destiné aux responsables de la méthodologie dans les domaines d'application des Recommandations de la série Z.100.

L'objectif de ce Supplément est d'offrir une méthodologie pour l'utilisation efficace des Recommandations de la série Z.100 sur les langages: Z.100, Langage de description et de spécification du CCITT (SDL, *specification and description language*) [1]; Z.105, Langage de description et de spécification combiné avec la notation de syntaxe abstraite numéro un (SDL/ASN.1) [2]; et Z.120, Diagramme de séquences de messages (MSC, *message sequence chart*) [3]. L'application de cette méthodologie devrait conduire à une spécification bien définie de l'application cible en langage SDL combiné avec la notation ASN.1 et avec les diagrammes MSC.

Utilisé avec la Recommandation Z.105, le langage SDL sert à définir le comportement, tandis que la notation ASN.1 sert à définir les données. Il est également possible d'utiliser le langage SDL pour définir des données, comme dans la Recommandation Z.100. Le présent Supplément couvre l'utilisation du langage SDL sans notation ASN.1 et avec notation ASN.1 comme dans la Recommandation Z.105.

Les diagrammes MSC sont utilisés pour décrire des séquences d'événements et ne couvrent normalement pas toutes les possibilités. Les diagrammes MSC sont utilisés pour décrire ce qui se passe pour des séquences particulières de stimuli, tandis que le langage SDL définit le comportement de tous les stimuli dans chaque état possible. Sans SDL, les diagrammes MSC ne donneront habituellement pas une description complète du comportement du système. Dans la méthodologie décrite par le présent Supplément, le langage SDL ne sera normalement pas utilisé sans diagrammes MSC. Le Supplément ne décrit pas une méthodologie générale pour l'utilisation des diagrammes MSC soit isolément soit avec des langages autres que SDL (avec ou sans notation ASN.1).

Etant donné que la méthodologie se fonde sur l'emploi du langage SDL avec la notation ASN.1 et avec les diagrammes MSC, le terme "SDL+" sera utilisé dans l'ensemble du présent Supplément dans le sens suivant: "langage SDL (Z.100) avec la notation ASN.1 (Z.105) et avec les diagrammes MSC (Z.120)".

L'on reconnaît qu'il existe de nombreuses façons d'utiliser le SDL+, selon les préférences de l'utilisateur, selon l'application cible, selon les règles internes de l'organisation, etc. La méthodologie décrite dans le présent Supplément est donc incomplète et nécessite donc une mise au point pour produire la méthodologie applicable pratiquement à un contexte particulier. Ce Supplément vise à aider les utilisateurs du langage SDL+ et à favoriser l'unification des usages ou applications de ce langage en tant qu'outil.

2 Définitions

Le présent Supplément définit les termes suivants:

2.1 **activité**: procédure spécifique dans un processus d'ingénierie, soutenue par une ou plusieurs méthode(s).

2.2 **agent**: objet qui modélise le comportement d'une entité dans l'environnement d'un système d'application.

2.3 **acteur**: personne (ou organisation) qui possède, dans l'environnement d'un système, un ou plusieurs rôle(s) d'interaction comportementale avec un système. Dans le contexte du présent Supplément, un acteur qui interagit avec les activités effectuées dans le système d'ingénierie est appelé *ingénieur*.

2.4 **Analyse**: activité par étapes permettant d'explorer les exigences collectées, d'organiser les informations qui y sont contenues et d'enregistrer ces informations sous un format reflétant cette organisation (voir le paragraphe 5).

2.5 **concept d'application**: concept se rapportant à un domaine d'application donné et au système qui est spécifié.

2.6 **domaine d'application**: domaine d'activité dans lequel fonctionnera le système spécifié.

2.7 **système d'application**: (terme abrégé en *système* lorsque l'attribut d'application peut être déduit du contexte): ensemble complexe de parties qui sert à fournir une fonctionnalité et d'autres caractéristiques. Dans le cadre du présent Supplément, il n'existe habituellement aucune distinction entre une spécification de système, une description de système et une instance de système réelle.

2.8 **ASN.1**: notation de syntaxe abstraite numéro un (Recommandations X.208 [4] et X.680 [5]).

2.9 **association**: relation de dépendance entre deux ou plus de deux classes (ou entre deux ou plus de deux objets) en notation de modélisation par objets ("Object Model Notation") (voir 13.2 et référence [11]). Cette relation est représentée par une ligne annotée entre symboles de classe (ou d'objet).

2.10 **comportement**: séquence d'actions avec des phases de stimulus et de réponse qui sont réalisées par un système qui peut changer d'état.

2.11 **classe**: description du modèle du système définissant un ensemble d'objets ayant des propriétés similaires (même structure et même comportement). Une classe représente un concept d'application.

2.12 **informations classifiées**: modèle selon lequel les exigences collectées sont structurées et définies conformément aux concepts qui ont été nommés et définis.

2.13 **prescriptions collectées**: ensemble des prescriptions collectées pour l'application à la suite d'une activité de Collecte de prescriptions.

2.14 **point de vue traitement**: point de vue d'un système et de son environnement qui est centré sur la décomposition du système afin d'en permettre la répartition (voir 4.1/X.903).

2.15 **point de vue conception**: point de vue d'un système et de son environnement qui est centré sur les fonctions requises pour soutenir ce système (voir le "point de vue ingénierie" au 4.1/X.903).

2.16 **Conception du projet**: activité consistant à mettre au point par étapes des spécifications partiellement informelles ou partielles à partir de différents points de vue et à différents niveaux de détail pour explorer des choix de conception d'ingénierie (voir le paragraphe 6).

2.17 **projets conçus**: esquisses pour le système étudié utilisant un ou plusieurs modèles décrivant partiellement le système.

2.18 **Documentation**: activité de sélection, de classement et d'enregistrement des informations sur un produit.

2.19 **point de vue entreprise**: point de vue d'un système et de son environnement qui est centré sur les objectifs, le domaine d'application et les politiques de ce système (voir 4.1/X.903).

2.20 **Formalisation**: activité de production par étapes d'une description SDL formelle d'un système (voir le paragraphe 7).

2.21 **description formelle en langage SDL+**: description en langage SDL+ conforme aux Recommandations Z.100 [1], Z.105 [2] et Z.120 [3], cohérente, non ambiguë, ne comportant aucun "texte informel" SDL, et décrivant de façon précise le système examiné.

2.22 **validation formelle**: recherche systématique d'une spécification (ou d'une mise en œuvre) pour déterminer si elle possède certaines propriétés souhaitables, se composant des opérations suivantes: contrôle de l'exactitude syntaxique et sémantique de la spécification (ou de la mise en œuvre) et contrôle de l'expression des exigences connues du système spécifié par la spécification (ou mise en œuvre).

2.23 **directive**: avis donné à l'utilisateur de la présente méthodologie au sujet de l'identification des décisions à prendre et éventuellement au sujet des raisons et moyens de prendre des décisions.

2.24 **description informelle en langage SDL+**: description en langage SDL+ issue de la conception de projet et s'écartant par un ou plusieurs aspects des critères de formalité, par exemple: pour le langage SDL, les déviations incluent l'ambiguïté, l'utilisation d'un texte informel et la non-conformité à la Recommandation Z.100 [1]; pour la notation ASN.1, les déviations incluent l'utilisation d'une structure de type "any" dans une description d'un type de données (les diagrammes MSC sont toujours censés être formels dans le cadre des contraintes imposées à leur fonction, qui consiste à fournir une trace des messages).

2.25 **point de vue information**: point de vue d'un système et de son environnement qui est centré sur la sémantique des informations et sur les activités de traitement de ces informations dans le système (voir 4.1/X.903).

2.26 **instruction**: déclaration impérative qui spécifie ce qui doit être exécuté dans le cadre d'une étape.

2.27 **méthode**: association d'une notation à des instructions, à des règles et à des directives régissant son emploi (voir [6] et [7]). Une *méthode* est un moyen systématique d'atteindre un objectif particulier. Une méthode est descriptive.

2.28 **méthodologie**: ensemble organisé et cohérent de méthodes et de principes, utilisé dans une discipline particulière. Aux termes d'un dictionnaire, une méthodologie est un système de méthodes et de principes utilisé dans une discipline particulière. Une méthode est une façon systématique de faire quelque chose ou encore les techniques ou les dispositions de travail pour un domaine ou un sujet particulier. Dans le contexte d'application du présent Supplément, les disciplines sont la mise au point de systèmes ou de protocoles de télécommunication, etc., appelés *systèmes d'application* (ou seulement *systèmes* pour plus de concision lorsque l'attribut d'application peut être déduit du contexte).

2.29 **modèle**: représentation de quelque chose qui possède une partie des propriétés et du comportement de cette chose.

2.30 **objet**: élément individuel qui fait partie d'une classe et qui possède un rôle bien défini dans le système, qui possède des propriétés, qui peut être géré et qui est caractérisé:

– par son état (selon toutes ses propriétés);

– par son comportement (selon la façon dont il agit et réagit);

– par son identité (selon laquelle il diffère des autres objets).

2.31 **orientation-objets**: technique de partitionnement du système étudié en objets séparés, de groupage des objets en classes et de mise en interaction de ces objets les uns avec les autres et avec l'environnement pour remplir les fonctions du système dans son ensemble.

2.32 **produit**: application (ou spécification) à réaliser, avec la documentation associée.

2.33 **prescription**: expression des idées à intégrer dans l'application développée.

NOTE – Le Guide ISO/CEI 2:1996, *Normalisation et activités connexes – Vocabulaire général*, contient la définition suivante:

prescription: disposition formulant les critères à remplir.

2.34 **Collecte des prescriptions**: activité d'évaluation initiale des prescriptions saisies et traitement des questions qui se posent à leur sujet dans le cadre de la mise au point d'une description formelle en langage SDL.

2.35 **bibliothèque de concepts**: enregistrement de concepts d'application qui sont disponibles pour utilisation à tout moment lors du processus de développement d'un système.

NOTE – La fourniture et l'exploitation d'une bibliothèque de concepts ne font pas partie du domaine d'application du présent Supplément.

2.36 **règle**: déclaration de principes visant à empêcher l'apparition de résultats non valides, invérifiables ou indésirables. La conformité est censée exister avec les règles exprimées par l'auxiliaire "doit/doivent" et est hautement recommandée avec les règles exprimées avec l'auxiliaire "devrait/devraient" (mais des exceptions existent).

2.37 **service**: ensemble de fonctions et de capacités offertes à un utilisateur par un fournisseur.

NOTE 1 – Dans cette définition, "l'utilisateur" et "le fournisseur" peuvent consister en une paire comme application/application, être humain/ordinateur, abonné/organisation (opérateur). Les différents types de service inclus dans cette définition sont le service de transmission de données et le service de télécommunication offert par une exploitation à ses clients ainsi que le service offert par une couche à d'autres couches dans un protocole stratifié.

NOTE 2 – **service SDL** (Z.100 [1]): autre moyen de spécifier une partie du comportement d'un processus (ou d'une machine à états finis de communication étendue).

2.38 **spécification**: détails et instructions décrivant les comportements, les structures de données, la conception, les constituants, etc. d'une mise en œuvre conforme à une application.

NOTE – **spécification SDL** (Z.100 [1]): définition des prescriptions d'un système. Une spécification se compose des paramètres généraux exigés du système et de la spécification fonctionnelle de son comportement requis.

2.39 **étape**: tâche autonome faisant partie d'une activité.

2.40 **système**: (voir *système d'application*).

2.41 **technique**: façon d'appliquer une méthode.

2.42 **point de vue technologie**: point de vue d'un système et de son environnement qui est centré sur le choix de la technologie utilisée dans ce système (voir 4.1/X.903).

2.43 **Essais**: combinaison de deux activités: *spécification de test* et *exécution de test*.

2.44 **Spécification des tests**: activité visant à produire un ensemble de tests à des fins particulières telles que le contrôle de la conformité d'une mise en œuvre avec une description formelle en langage SDL+ ou l'interopérabilité de différents produits.

2.45 **Exécution des tests**: activité de conduite des tests prescrits par une Spécification de test afin de produire un ensemble de résultats de test.

2.46 **outil**: moyen de faciliter l'accomplissement d'une séquence d'étapes en vue d'un objectif donné, habituellement (dans la présente méthodologie) mis en œuvre sous la forme d'un logiciel.

2.47 **type**: définition SDL d'un **type système**, d'un **type bloc**, d'un **type processus**, d'une **procédure** ou d'un **signal**.

2.48 **séquence d'utilisation**: séquence de transactions associées qui est exécutée par l'utilisateur d'un système en vue d'un résultat particulier.

2.49 **Validation**: processus, avec les méthodes, procédures et outils associés, permettant d'évaluer qu'une application est (ou qu'une norme peut être) pleinement mise en œuvre, est conforme aux normes et critères applicables à l'application (ou à la norme), et satisfait à l'objectif exprimé dans l'enregistrement des exigences sur lesquelles l'application (ou la norme) est fondée; et permettant d'évaluer, dans le cas d'une norme, qu'une mise en œuvre conforme à cette norme possède la capacité exprimée dans l'enregistrement des exigences sur lesquelles la norme est fondée.

2.50 **modèle de validation**: version détaillée d'une spécification ou d'une mise en œuvre, pouvant comporter des parties de son environnement, qui est utilisée pour effectuer une validation formelle.

3 Avantage de l'utilisation du langage SDL avec la notation ASN.1 et avec les diagrammes MSC

Il paraît évident que le succès d'un système dépend de la précision de sa spécification et de sa conception. Cela nécessite toutefois un langage de spécification approprié, répondant aux exigences suivantes (par souci de concision, le résultat de la spécification et de la conception du système est ici appelé *spécification*):

– un *ensemble de concepts* bien défini;

– des spécifications *univoques*, *claires*, *précises* et *concises*;

– une base pour *analyser* les spécifications quant à leur *complétude* et à leur *exactitude*;

– une base pour déterminer la *conformité* des mises en œuvre aux spécifications;

– une base pour déterminer la *cohérence* des spécifications les unes avec les autres;

– l'utilisation d'*outils* informatiques pour créer, tenir à jour, analyser et simuler des spécifications.

Un système peut faire l'objet de spécifications à différents niveaux d'abstraction. Une spécification est soit une base permettant de mettre au point des *mises en œuvre* soit un modèle (par exemple dans une norme) auquel une *mise en œuvre* peut être comparée. Une spécification devrait s'abstenir d'entrer dans les détails de mise en œuvre afin:

– de donner un aperçu général d'un système complexe;

– de différer des décisions de mise en œuvre;

– de ne pas exclure des mises en œuvre valides.

Contrairement à un programme, une spécification formelle (c'est-à-dire une spécification rédigée dans un langage de spécification formelle) n'est pas destinée à être exploitée sur un ordinateur. Servant non seulement de base pour mettre au point des mises en œuvre, une spécification formelle peut être utilisée pour assurer des communications précises et univoques entre personnes, en particulier pour passer des commandes et pour soumettre des offres.

L'utilisation du langage SDL+ permet d'analyser, d'étudier et de simuler des solutions en variante pour les systèmes à différents niveaux d'abstraction, ce qui est impossible en pratique au moyen d'un langage de programmation en raison des coûts et des délais. Le SDL+ offre à son utilisateur un ensemble bien défini de concepts, améliorant sa capacité à produire une solution à un problème et à étudier une solution.

3.1 Compréhension d'une spécification en SDL+

En langage SDL+, les concepts sont fondés sur des modèles mathématiques et sur la théorie de la signification. La façon d'appliquer les modèles du langage SDL+ à une application est indiquée ci-dessous.

Le domaine d'application d'un système s'interprète finalement en termes de concepts d'un langage naturel. Les individus acquièrent une compréhension de ces concepts en passant par un long processus d'apprentissage et d'expérience de la vie réelle. Dans la description d'une application en langage naturel, les phénomènes sont décrits tels qu'ils sont perçus par un observateur. La description en langage naturel du système fait appel à des concepts qui se déduisent directement de l'application et de la mise en œuvre du système.

Lorsqu'un système est spécifié au moyen du langage SDL+, la spécification ne fait directement usage ni des concepts de l'application ni de ceux de la mise en œuvre. En revanche, le langage SDL+ définit un *modèle* qui représente les propriétés significatives (principalement le comportement) du système. Pour comprendre ce modèle, il faut lui faire correspondre la compréhension intuitive de l'application en termes des concepts du langage naturel (voir la Figure 3-1). Ce mappage peut être fait de différentes façons: l'une consiste à choisir les noms des concepts introduits dans la spécification formelle qui présentent une bonne association avec ceux de l'application; une autre consiste à commenter le langage SDL+. Cela relève sensiblement de la même problématique que la compréhension d'un programme ou d'un algorithme qui résout un problème pris dans la vie courante.

Un modèle devrait posséder une bonne *puissance d'analyse* et une bonne *puissance d'expression* pour faciliter son mappage avec l'application. Malheureusement, la puissance d'analyse et la puissance d'expression sont généralement contradictoires: plus un modèle est expressif, plus il est difficile à analyser. Lorsque l'on conçoit un langage de spécification, il faut évidemment faire un compromis entre ces deux propriétés. De plus il faut insister sur le fait qu'un modèle représente toujours une vue simplifiée de la réalité et qu'il possède, par conséquent, toujours des limitations.

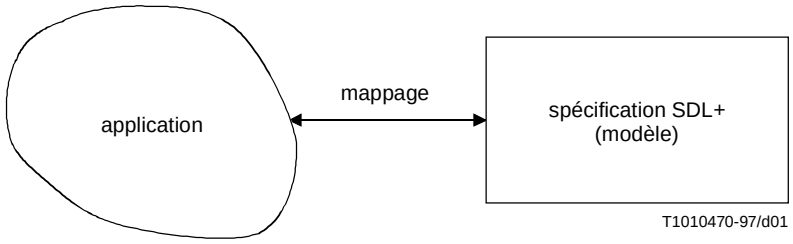


Figure 3-1/Suppl. 1 à la Rec. Z.100 – Comment comprendre une spécification en langage SDL+

3.2 Domaine d'application du langage SDL+

Bien que le langage SDL soit largement connu dans le secteur des télécommunications, mais il possède un domaine d'application plus vaste et est également utilisé dans d'autres industries. Le domaine d'application du SDL+ peut être caractérisé comme suit:

– *type de système:* en temps réel, interactif, réparti;

– *type d'information:* comportement et structure;

– *niveau d'abstraction:* du niveau général jusqu'au niveau détaillé.

Le langage SDL+ a été développé pour être utilisé dans les systèmes de télécommunication comprenant la communication de données; mais il peut en réalité être utilisé dans tous les systèmes en temps réel et interactifs. Il a été conçu pour la spécification du comportement d'un tel système, c'est-à-dire pour la coopération entre le système et son environnement. Il est aussi adapté à la description de la structure interne d'un système, de façon que le système puisse être développé et compris une seule partie à la fois. Cette caractéristique est essentielle pour les systèmes répartis.

Le SDL+ couvre différents niveaux d'abstraction, allant d'un large aperçu général jusqu'à la conception détaillée. On n'a pas voulu que ce soit un langage destiné à la mise en œuvre. La traduction plus ou moins automatique de spécifications SDL+ en langage de programmation est, cependant, possible dans de nombreux cas. La grandeur des changements à apporter à une spécification en SDL pour produire une mise en œuvre décrite en SDL peut être très réduite dans certains cas.

3.3 Relation avec la mise en œuvre

Normalement, une distinction est faite entre les langages *déclaratifs* et les langages *constructifs*. Le SDL est un langage constructif, ce qui signifie qu'une spécification en SDL+ définit un *modèle* qui représente des propriétés significatives d'un système (comme mentionné ci-dessus). Une question importante est de déterminer quelles doivent être ces propriétés significatives. Ce point est laissé ouvert en SDL+ car *c'est à l'utilisateur de décider quelles doivent être ces propriétés.*

Si une décision est prise de représenter seulement le comportement du système (tel qu'on le voit à sa frontière), alors on parlera normalement d'une *spécification*, et la structure donnée du modèle ne sera qu'une aide pour structurer la spécification en éléments gérables. Si la décision est prise de représenter aussi la structure interne du système, alors on parlera d'une *description*. C'est la raison pour laquelle la Recommandation Z.100 ne fait aucune distinction entre spécification et description.

Remarquer que la structure interne d'un système est normalement représentée en SDL par des *blocs*. Les *processus* et la signalisation interne à l'intérieur des *blocs* n'exigent pas de représenter la structure interne du système, et ainsi n'ont pas besoin d'être considérés comme des exigences d'une mise en œuvre, comme certaines personnes le considèrent à tort. La conformité d'une mise en œuvre à des spécifications est normalement traitée par les organismes de normalisation par une *règle de conformité explicite*. Cela est évidemment nécessaire aussi lorsqu'on utilise un langage constructif.

Etant donné que le langage SDL+ décrit la structure et le comportement, il est possible qu'une *description* en langage SDL+ représente précisément une mise en œuvre; inversement, une telle *description* précise peut être utilisée comme une description linguistique à partir de laquelle le logiciel d'exploitation sera automatiquement reconstruit. Le langage SDL+ peut donc servir aussi bien de langage de spécification abstrait (par exemple dans des normes pour protocoles) que de langage de mise en œuvre. L'utilisation d'un seul langage évite une éventuelle source d'erreur, qui est la traduction d'un langage à un autre. Une spécification et une mise en œuvre correspondante, rédigées en SDL+, peuvent présenter quelques différences notables qui proviennent de la prise en compte de certaines caractéristiques de mise en œuvre. Bien que le même langage soit utilisé à différents niveaux, certaines parties d'une spécification en SDL+ sont susceptibles de nécessiter des modifications directement utilisables dans une mise en œuvre décrite en SDL+.

La méthodologie du présent Supplément part du principe que l'ordre de grandeur des opérations d'ingénierie à effectuer pour modifier une spécification formelle détaillée (exécutable) en SDL+ afin d'obtenir une description de mise en œuvre est négligeable. Cette hypothèse n'est pas toujours vraie. Si la modification requise est importante, la méthodologie devra être étendue aux travaux additionnels de description de mise en œuvre. Les méthodes qui doivent être appliquées pour la réalisation varient en fonction des équipements concernés, des organisations mises en jeu, des emplacements physiques et de nombreux autres facteurs. La méthodologie produit du langage SDL+ qui est utilisé soit en tant que spécification soit en tant que description de mise en œuvre.

PARTIE I – MÉTHODOLOGIE GÉNÉRALE

4 Aperçu général des activités et description de la méthodologie

La méthodologie décrite dans le présent Supplément est un ensemble d'activités cohérent qui est utilisé pour l'ingénierie des systèmes afin de produire une définition de produit (sous forme d'une spécification ou d'une description de mise en œuvre). La définition du produit pourra être utilisée dans le cadre d'une norme, d'une spécification d'approvisionnement, en tant que spécification préalable à la mise en œuvre, code source d'une mise en œuvre ou base d'élaboration de tests pour un produit.

La méthodologie se compose d'activités. L'Analyse des exigences et la Formalisation en langage SDL+ constituent des *activités*. La même activité peut faire l'objet de plusieurs méthodes en variante. L'évaluation d'approches concurrentes et la sélection de l'une d'entre elles constituent donc une étape importante lors de l'élaboration de la méthodologie. La sélection doit toujours être fondée sur (entre autres éléments) les caractéristiques de l'application.

Une activité est une méthode ou un processus qui est régi par certains principes, qui accepte des entrées et produit des sorties (voir la Figure 4-1). La personne (ou l'organisation) qui effectue une activité peut être appelée "acteur" et peut être considérée comme remplissant différents "rôles" (ou fonctions). Dans le contexte d'application du présent Supplément, le terme "ingénieur" est utilisé au lieu du terme "acteur" parce que les activités relèvent de l'ingénierie. Pour assurer l'ingénierie au moyen du langage SDL+, les rôles qui sont remplis par les divers ingénieurs sont ceux d'analyste-système, de programmeur, d'ingénieur de tests, de concepteur de matériel, de gestionnaire de projet, d'opérateur d'équipement, etc. Ces rôles sont habituellement bien définis. Pour la mise au point de normes, les rôles d'ingénieur peuvent être ceux de responsable de l'application des normes, de rapporteur en normalisation, de concepteur de normes, de concepteur de suites de tests abstraites, d'organisme d'approbation des normes, etc. Une activité correspond

habituellement à un seul rôle principal (par exemple spécification ou spécificateur de système) mais interagit souvent avec d'autres rôles (client d'équipement, gestionnaire de produit, ingénieur de tests). Les différents rôles remplis par les ingénieurs sont hors du domaine d'application du présent Supplément. Un ingénieur joue souvent plusieurs rôles.

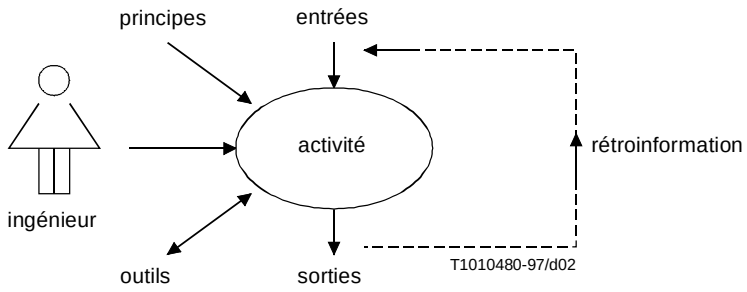


Figure 4-1/Suppl. 1 à la Rec. Z.100 – Une activité

En ingénierie des équipements, il est évident que les activités de commercialisation, de conception et de gestion se déroulent en parallèle. Elles sont aussi en relations de dépendance les unes avec les autres. Cela se vérifie souvent, même pour la création de normes. Les activités définies dans le présent Supplément ont des relations de dépendance mais peuvent aussi (dans les limites des contraintes imposées par ces relations) se dérouler en parallèle. L'activité de Documentation peut, par exemple, commencer dès qu'il existe une claire compréhension du domaine et des objectifs de l'application. Mais elle ne peut pas être réalisée avant qu'une description appropriée ait été rédigée en langage SDL+. A part les contraintes logiques imposées par la nécessité de recevoir des contributions d'autres activités, le présent Supplément ne prescrit pas la façon de programmer, d'organiser et de gérer les activités. Il n'existe aucun modèle "de cycle de vie".

En ingénierie des systèmes, il est généralement utile de pouvoir examiner le système sous différents angles. Différentes vues donnent différentes informations sur le système examiné. L'approche du traitement réparti ouvert (ODP, *open distributed processing*) [8] offre cinq vues différentes: entreprise, information, traitement, conception ("ingénierie") et technologie (voir la Figure 4-2).

NOTE – Dans le contexte d'application du traitement réparti ouvert, l'activité "ingénierie" concerne des questions de conception telles que les mécanismes de commande, la qualité de fonctionnement, la répartition, etc. Dans le présent Supplément, cette vue est nommée "conception" pour éviter une confusion avec le terme "ingénierie", qui est utilisé pour toutes les activités.

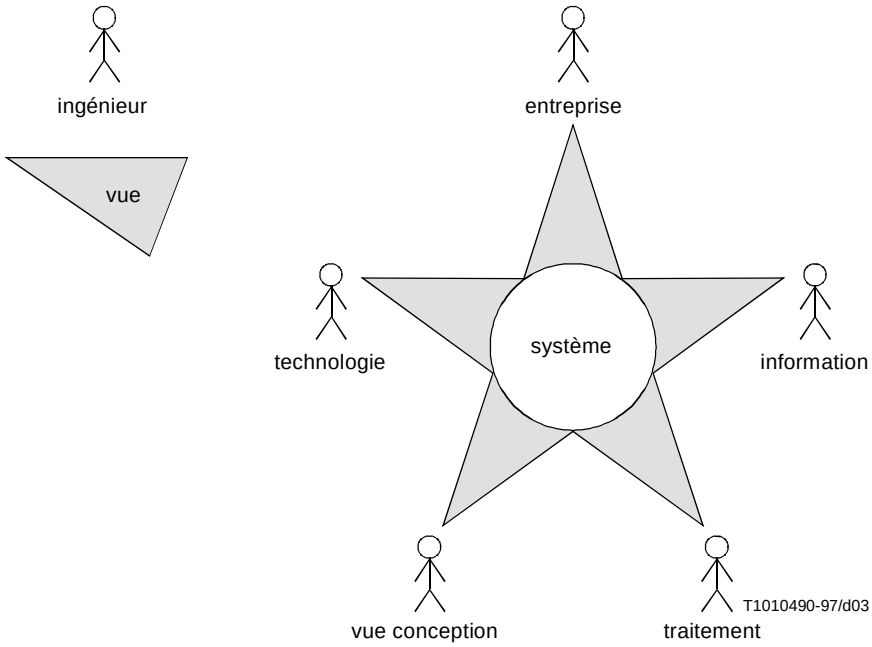


Figure 4-2/Suppl. 1 à la Rec. Z.100 – Les vues de traitement ODP d'un système

Une vue d'entreprise révèle les politiques, les actions et les cibles globales du système. Dans le contexte d'application du présent Supplément, la vue d'entreprise concerne les raisons, l'objectif et l'utilisation de l'application. Cette information devrait être donnée dans le cadre des prescriptions relatives à la nécessité de l'application. La vue d'entreprise est habituellement exprimée en langage naturel. Elle est matérialisée par des prescriptions et elle devrait être utilisée dans la description du domaine de l'application et pour la validation du produit.

Une vue d'information révèle les structures d'information, c'est-à-dire la façon dont les données élémentaires du système sont reliées les unes avec les autres. Dans le contexte d'application du présent Supplément, la vue d'information concerne les emplacements de mémorisation des données élémentaires, le nombre et la longueur des données élémentaires, les liaisons entre données élémentaires, et la cardinalité (c'est-à-dire la multiplicité) de ces relations (point à point, point à multipoint, multipoint à multipoint). Les sorties produites par les principales activités méthodologiques du présent Supplément contiennent des vues du système de type information.

Une vue de traitement révèle la façon dont les données élémentaires sont traitées à l'intérieur du système. Cette vue concerne la façon dont les informations sont transférées, consultées, transformées et gérées. La commande de traitement peut provenir de l'intérieur ou de l'extérieur du système. En association avec la vue d'information, la vue de traitement définit le comportement du système. Dans le contexte d'application du présent Supplément, la vue de traitement est décrite dans les processus SDL utilisant des données définies soit en notation ASN.1 soit en langage SDL. Le système est décomposé de façon qu'il puisse être réparti.

Une vue de conception révèle la façon dont le comportement du système est organisé et effectivement réparti pour tenir compte de l'environnement logistique. Des éléments tels que les caractéristiques de transmission, la répartition et la concomitance sont pris en compte dans cette vue, qui peut comporter la qualité de fonctionnement (performance) et la fiabilité. La conception est souvent un compromis entre différents éléments. Dans le contexte d'application du présent Supplément, le langage SDL+ peut comprendre des prescriptions de conception concernant la répartition des capacités et la concomitance. Ces prescriptions se retrouvent dans la structure (blocs et processus), dans le langage SDL+ et dans les annotations. Si ces éléments sont obligatoires pour l'application, la structure SDL doit correspondre aux prescriptions. Dans tous les autres cas, le langage SDL devrait être structuré de façon à être aussi clair que possible tout en facilitant aussi bien les tests que les validations.

Une vue de technologie se rapporte aux éléments technologiques contenus dans le système. Elle n'est applicable que si la description de la mise en œuvre est notablement différente de la spécification SDL+. La méthodologie du présent Supplément ne couvre pas ce cas.

Pour résumer ce qui précède, l'ingénierie d'une application implique différentes vues du système. Ces vues sont contenues dans différents modèles du système. Au cours du processus d'ingénierie, le système est graduellement construit par la production de ces modèles. Comme ces derniers appartiennent au même système, ils sont tous en relation les uns avec les autres.

Lorsque l'étape d'ingénierie d'un système commence, la connaissance relative à ce système s'inscrit généralement entre les deux extrêmes suivants:

– le système est médiocrement compris et les prescriptions ne sont définies que vaguement. La connaissance et l'expérience au sujet du système font défaut. Il n'existe pas de systèmes similaires à partir desquels les ingénieurs pourraient établir des analogies;

– le système est bien compris et bien défini. Il a déjà été mis en œuvre et la tâche d'ingénierie consiste à modifier un système existant. Les compétences sont à disposition, les modifications ne présentent aucune difficulté et les spécifications du système existant ont été créées en SDL+.

Dans le premier cas, une grande partie du travail d'ingénierie d'un système est la compréhension de ce qui est requis du système et de son comportement. Dans le deuxième cas, le travail consiste à apporter des modifications au SDL+ et à la documentation, ce qui peut être fait par création de certains types SDL+ nouveaux à ajouter aux types existants, ou peut-être par simple association nouvelle des types SDL+ existants afin de créer un nouveau système.

La méthodologie présentée dans le présent Supplément est applicable à l'ingénierie de définition-système d'une application, mais sans entrer dans les détails de mise en œuvre. C'est ce qu'on appelle la "spécification" de l'application. L'ingénierie requise est très semblable à celle qui est requise pour un système quelconque: elle peut s'inscrire à n'importe quelle distance des deux extrêmes énoncés ci-dessus. Les éléments de mise en œuvre ne sont pas traités en détail par cette méthodologie. Pour éviter d'alourdir le texte dans le reste du présent Supplément, le mot "système" impliquera "la définition-système de l'application".

La méthodologie est centrée sur les trois principales activités suivantes:

– l'Analyse, qui est l'organisation des prescriptions et l'identification des concepts (avec noms et définitions) pour le système;

– la Conception du projet, qui est la transformation d'informations classifiées en concepts de projet s'appliquant à une partie du système ou qui sont partiellement formels;

– la Formalisation, qui est l'expression de la spécification en termes de langage SDL formel, complété par des diagrammes MSC et par des notations ASN.1.

La méthodologie vise partiellement quatre autres activités. La première est la saisie des prescriptions, qui comprend la Collecte des prescriptions au début d'un projet. La deuxième est la Validation, qui est effectuée sur des spécifications formalisées, au fur et à mesure qu'elles sont produites. La troisième est la Documentation, qui traite de la sélection des spécifications du système en vue de leur archivage et de leur éventuelle réutilisation. La quatrième est la mise en œuvre, qui traite de la production d'un code exécutable pour le système cible. La Figure 4-3 illustre la méthodologie. L'activité de saisie des prescriptions n'est couverte que partiellement par le présent Supplément: la description qu'il contient ne vise que la Collecte des prescriptions. La Figure 4-3 ne montre pas la totalité des flux d'information.

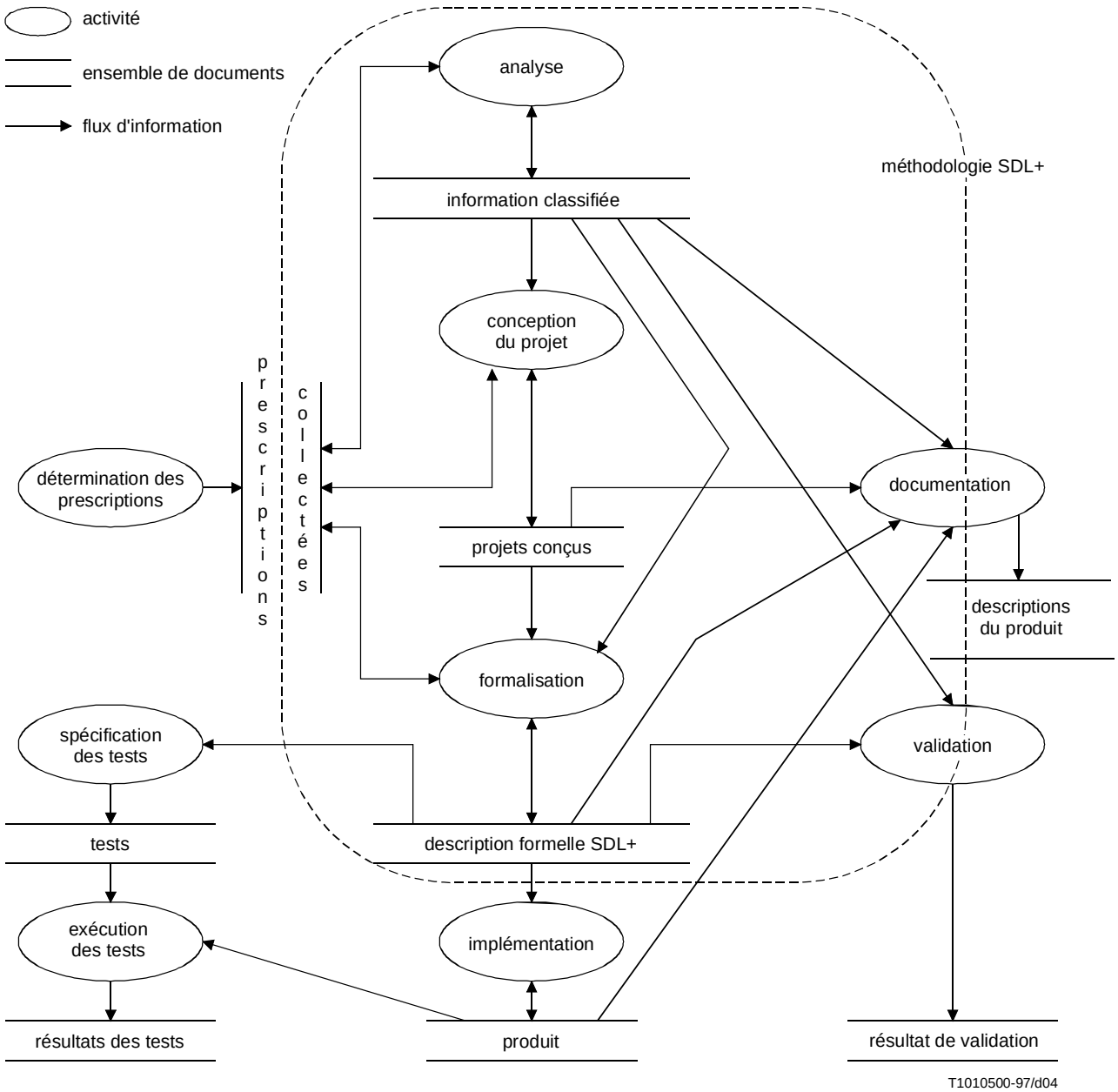


Figure 4-3/Suppl. 1 à la Rec. Z.100 – La méthodologie SDL+

L'ensemble complet des sept activités (Collecte des prescriptions, Analyse, Conception du projet, Formalisation, Validation, Documentation et Mise en oeuvre) est applicable lorsqu'un projet d'ingénierie de système commence par une médiocre compréhension de celui-ci. Dans la mesure où le système est bien défini, les activités d'Analyse et de Conception du projet peuvent toutefois être facultatives. Les ingénieurs qui utilisent cette méthodologie évaluent d'abord le degré de compréhension et de définition du système puis décident de l'activité qui devra servir de point d'entrée approprié dans la méthodologie. Pour faciliter ce choix, le présent Supplément offre des listes de critères permettant de commencer et d'achever les trois principales activités.

Ces listes de critères permettent également aux ingénieurs de suivre la méthodologie lorsque les travaux relatifs à un projet d'ingénierie-système contiennent des redondances et des itérations. En d'autres termes, la saisie des prescriptions peut encore avoir lieu pendant que les prescriptions existantes sont en cours de classification. Cette activité peut, en contrepartie, ne pas être terminée lorsque l'étape de Conception du projet commence. La Formalisation peut commencer dès que les ingénieurs ont une vision précise des interfaces du système et la première mouture de la documentation peut être produite pour une spécification dès que l'architecture globale est claire. De même, en formalisant une spécification en langage SDL+, les ingénieurs peuvent soulever des questions qui les obligeront à reclassifier ou à restructurer la description produite au cours de l'étape de Conception du projet.

Les paragraphes suivants décrivent brièvement les neuf activités en lesquelles cette méthodologie est subdivisée. Ces paragraphes définissent également le degré de flexibilité dont les ingénieurs disposent pour passer d'une activité à une autre. L'Analyse, la Conception du projet et la Formalisation sont regroupées en un seul paragraphe parce qu'elles sont étroitement liées. La Validation, la Spécification de test et l'Exécution de test sont également regroupées parce qu'un grand nombre des techniques d'essais sont également utiles pour la Validation.

Les activités d'Analyse, de Conception du projet, de Formalisation, de Mise en œuvre et de Validation sont également reprises dans les paragraphes 5, 6, 7, 8 et 9, respectivement.

4.1 Partie Collecte de la saisie des prescriptions

L'objet de la Collecte des prescriptions est de vérifier que les prescriptions sont logiquement décrites pour la spécification. Aucune indication n'est donnée dans la méthodologie quant à la création ou la Collecte de prescriptions; le point de départ est qu'un ensemble de prescriptions soit obtenu, sous une forme ou sous une autre. La méthodologie part du principe que les prescriptions collectées contiennent au moins une vue d'entreprise du système.

Dans cette activité, des critères minimaux sont donnés pour l'acceptation des prescriptions initiales. Certaines directives sont également données pour l'emploi de critères de qualité afin de déterminer que la Collecte des prescriptions a été effectuée. Celle-ci ne s'arrête pas nécessairement dès qu'une autre activité est entreprise. Certaines prescriptions peuvent nécessiter un raffinement à la suite d'une analyse plus fine du problème. Des changements de circonstances peuvent entraîner des changements de prescriptions et d'autres activités peuvent engendrer des questions conduisant à de nouvelles prescriptions ou à des modifications de prescriptions existantes. La Figure 4-3 montre les prescriptions collectées résultant des autres activités. Le flux d'information n'est pas montré pour la résolution d'une question au moyen de l'activité de saisie des prescriptions.

La production proprement dite des prescriptions par recherche, par ingénierie cognitive ou par d'autres méthodes, bien que non définie ici, se déroule néanmoins avant le début du développement. Par ailleurs, chaque fois que des ingénieurs s'interrogent sur des prescriptions, la présente méthodologie part du principe que les réponses ou décisions sont issues de cette activité. Les questions sur la complétude, la cohérence et autres attributs qualifiant les prescriptions peuvent se traduire par des modifications à des prescriptions existantes et par l'adjonction de nouvelles prescriptions à l'ensemble. Lorsque les questions ne peuvent pas être résolues dans le cadre de la méthodologie, il faut les résoudre par les phases extérieures à la présente méthodologie de l'activité de saisie des prescriptions.

On trouvera ci-dessous une liste de contrôle des critères minimaux que les prescriptions initiales doivent satisfaire. Si les prescriptions initiales ne sont pas adéquates, des solutions doivent être trouvées dans les activités de saisie de prescriptions extérieures au domaine de la présente méthodologie.

1) S'il existe un grand nombre de prescriptions initiales ou si celles-ci sont complexes, est-ce qu'un résumé des prescriptions est fourni?

2) Est-ce que les prescriptions initiales contiennent une déclaration d'objectif pour le système?

3) Est-ce que les prescriptions initiales décrivent les capacités du système?

4) Est-ce que la mise en œuvre est considérée comme possible étant donné l'état de la technologie actuelle ou les résultats à attendre des progrès technologiques?

Déterminer s'il existe un ensemble satisfaisant de prescriptions collectées relève d'un jugement subjectif car les prescriptions collectées ne sont (habituellement) pas sous une forme qui permette de formuler un jugement objectif. La liste de contrôle permettant de juger la réalisation de la Collecte de prescriptions est la suivante:

1) les prescriptions collectées sont censées être univoques, complètes et réalisables (ces attributs pouvant être ultérieurement démentis);

2) les prescriptions ne devraient pas être notoirement incorrectes, invérifiables, incohérentes sur le plan interne ou externe (ce qui ne garantit pas qu'elles soient correctes, vérifiables ou cohérentes mais ce qui évite d'aller plus loin s'il apparaît qu'une de ces prescriptions est fausse);

3) il est possible de mettre au point une description formelle en SDL+ sur la base des prescriptions collectées.

4.2 Analyse, Conception du projet et Formalisation

Dans la présente méthodologie, la modélisation des prescriptions est répartie sur trois activités. Les informations classifiées comprennent le point de vue entreprise et d'autres points de vue contenus dans les prescriptions collectées. Les informations classifiées forment un modèle dans lequel les prescriptions collectées sont structurées et définies conformément à des concepts (avec noms et définitions). Ces informations sont raffinées au cours de la phase de Conception du projet, afin de décrire les parties formelles du système à partir du point de vue information et les parties informelles du système à partir du point de vue traitement. Ces descriptions seront transformées, au cours de la Formalisation, en une spécification formelle qui donnera une description précise et complète du système selon les points de vue information, traitement et (si nécessaire) conception.

Analyse

Les prescriptions collectées sont structurées soit sur la base de la structure analytique des concepts utilisés pour des produits antérieurs soit sur la base d'une nouvelle structure élaborée spécialement pour le système à spécifier. Elles deviennent alors des informations classifiées, qui sont caractérisées par le fait que les prescriptions collectées (habituellement en langage naturel avec des diagrammes informels) ont été structurées en un modèle de concepts (avec noms et définitions).

Il n'existe aucun ensemble fixe de concepts ni aucune structure fixe. Lorsqu'il s'agit d'une nouvelle application, la bibliothèque de concepts est explorée pour détecter des enregistrements d'autres systèmes ayant des concepts et structures similaires.

Le résultat est une information plus structurée et une terminologie définie. Les idées sont clarifiées et élargies parce que l'ingénieur doit réfléchir au processus et faire des références aux prescriptions collectées.

Les notations utilisées pour structurer les informations peuvent être choisies de façon à fournir des moyens appropriés d'enregistrer les concepts et la structure propres aux prescriptions. Etant donné que celles-ci contiennent souvent des descriptions de séquences d'utilisation ou que de telles séquences sont construites en fonction des questions posées, des diagrammes de séquences de messages (MSC) peuvent être un outil approprié pour consigner une partie du comportement au cours de l'Analyse.

Conception du projet

Les informations classifiées (ou les prescriptions collectées) sont converties en conceptions de projet rédigées avec des notations qui ont une structure formelle et, habituellement, une syntaxe formelle. La principale caractéristique des conceptions de projet est qu'elles sont des projets incomplets du système. Un examen approfondi et une détermination du comportement détaillé du système ne sont pas des étapes essentielles parce que l'objectif est d'explorer des conceptions possibles. La sémantique des notations utilisées peut donc être informelle et les conceptions de projet peuvent autoriser un certain nombre de comportements différents. Le résultat est que la signification dépend de l'interprétation des mots utilisés et d'une compréhension mutuelle entre les ingénieurs qui utilisent les documents. Bien que cela soit souvent suffisant pour le point de vue entreprise et peut-être pour le point de vue information, ces interprétations ne sont pas suffisantes pour un point de vue traitement complet et précis.

Les notations utilisées peuvent être choisies de façon à convenir au système particulier, au langage de spécification formelle à utiliser (SDL+) et aux ressources disponibles (par exemple compétences, efforts et outils). Utilisé de façon informelle, le langage SDL+ est une notation appropriée à la conception de projets. Les diagrammes de séquences de messages sont toujours utilisés. La notation ASN.1 est utile, soit parce que les prescriptions collectées la comportent déjà soit parce que c'est une technique efficace pour décrire le point de vue information d'un système. Une notation de modèle par classes d'objets (de préférence celle qui a été utilisée dans l'Analyse) peut être utilisée pour modéliser les entités contenues dans le système ainsi que leurs relations.

Des améliorations et des corrections des concepts du système résulteront souvent de la Conception du projet. Il s'agira de prescriptions additionnelles qui seront donc représentées, sur la Figure 4-3, comme des rétroactions passant par les prescriptions collectées.

Formalisation

La description en SDL+ est issue des prescriptions collectées, des informations classifiées et des conceptions de projet. Au cours de la création de la description formelle en SDL+, les diagrammes SDL peuvent être considérés comme une conception de projet car ils contiennent souvent des parties informelles ou sont incomplets au sens des règles du langage SDL. Ces descriptions intermédiaires donnent néanmoins des vues utiles sur le système et en facilitent la compréhension.

Il y a parfois plusieurs descriptions SDL+ formelles. Une description abstraite est produite d'abord en tant que spécification formelle de niveau supérieur (sommet). Cette spécification est ensuite raffinée par étapes successives jusqu'au niveau requis pour l'application. Un raffinement complémentaire peut être effectué afin de produire une description qui pourra être interprétée aux fins de la validation. Un tel raffinement revient à produire une mise en œuvre à partir d'une spécification en SDL+.

La description formelle en SDL+ donne une vue traitement précise du système et (au besoin) une vue conception. Pour être formelle, la spécification ne doit contenir aucun "texte informel" (dans le sens SDL), de manière que le comportement ne soit déterminé que par la sémantique formelle du SDL+ et ne dépende pas d'une interprétation humaine.

L'information classifiée peut ne pas être produite si l'on a une claire compréhension des concepts mis en jeu et si ces concepts sont bien documentés. Toutefois, lorsque l'on commence à travailler sur les concepts, des malentendus apparaissent souvent et il peut alors être nécessaire de produire des informations classifiées. Les conceptions de projets peuvent également ne pas être produites si le comportement du système est bien compris. Dans ce cas, la description formelle en SDL+ peut être rédigée directement au moyen des prescriptions collectées, conformément aux étapes de la Formalisation: on remplace alors les données cognitives et expérimentales par des informations classifiées et des conceptions de projet. Cette méthode présente l'avantage d'être économique pour l'ingénieur ou le système particulier mais elle ne produit pas d'enregistrements pouvant servir ultérieurement à un autre ingénieur ou système. Dans la plupart des situations pratiques, il vaut mieux commencer par un squelette. De mauvaises approches peuvent être ignorées avant que beaucoup de travaux aient été consacrés à la Formalisation. Dans ce cas, l'option d'avoir du texte informel dans le SDL au cours de la Conception du projet est une caractéristique intéressante de ce langage.

Le résultat complet de l'approche contient les trois niveaux de description, alors que la description formelle en SDL ne constitue qu'une partie de cette description. Le résultat complet contient tout ce qui est connu au sujet du système.

4.3 Validation et Essais

Les deux activités relatives aux tests (leur Spécification et leur Exécution) sont groupées sous l'appellation *Essais*.

La différence entre Validation et Essais porte sur ce qui est comparé au cours de ces activités: dans le cas de la Validation, c'est surtout la description SDL+ qui est comparée au modèle d'informations classifiées produit par l'Analyse; tandis que dans le cas des Essais, c'est surtout avec un produit mis en œuvre que la description SDL+ est comparée. Aussi bien les Essais que la Validation concernent la détermination du fait que l'application possède les caractéristiques prévues. Les prescriptions collectées contribuent également aux activités d'Essais et de Validation (la Figure 4-3 ne montre que les principaux flux d'information).

Dans la présente méthodologie, la Validation comprend toutes les actions qui concernent le contrôle de la "validité" ou de la "valeur" des définitions en SDL+. La Validation possède deux aspects: l'évaluation de la conformité à des normes et à d'autres critères pour l'application; et l'évaluation de la conformité à l'objectif exprimé dans les prescriptions (en termes de capacités et de performance également). Dans ces deux cas, la Validation peut montrer qu'une définition SDL+ est non valide. Mais si aucun contrôle n'échoue, un jugement devra être porté pour déterminer si la définition SDL+ est valide. Le terme "vérification" (réservé pour prouver qu'une relation entre deux modèles est vraie) n'est pas utilisé.

Dans le cadre de l'activité de Validation, l'on utilise un modèle de validation formelle. Il s'agit soit de la définition SDL+ qui doit être validée soit d'un modèle qui en est issu et qui comporte des modèles partiels de l'environnement. Le modèle de validation est utilisé pour la validation formelle: recherche systématique de la validité, par exemple en contrôlant la grammaire SDL+ et en appliquant des tests élémentaires (cas de test). Des techniques de validation informelle, comme les incursions et les inspections avec une liste de contrôle peuvent également être appliquées: elles sont généralement nécessaires pour évaluer si l'objectif visé a été atteint.

La description formelle en SDL+ sert de base au modèle de validation. Etant donné que certaines constructions SDL permettent de définir des systèmes qu'il peut être difficile ou impossible à valider, il est parfois souhaitable d'imposer certaines contraintes quant aux constructions SDL+ utilisées en Formalisation, de manière que le système puisse être validé. Selon les exigences de validation, il est parfois nécessaire de compléter la description formelle en SDL+. Par exemple, de façon que le comportement du système puisse être validé dans le cas de processus à maximisation, le modèle de validation peut avoir besoin de restreindre le nombre d'instances simultanées d'un même processus SDL, jusqu'à un nombre inférieur à celui qui est indiqué par la description formelle SDL+.

La déduction des tests élémentaires pour le modèle de validation est semblable à la Spécification de test, sauf qu'il y a des objectifs de test différents. L'application des tests élémentaires à un modèle de validation est semblable à l'Exécution de test. En fait, beaucoup de tests et d'outils similaires peuvent être utilisés, aussi bien pour les Essais du modèle de validation que pour ceux d'un produit.

La forme la plus courante des Essais est celle des *tests de conformité*, qui consiste à évaluer, au moyen de tests, si un produit est conforme à sa spécification. Les tests de conformité constituent une étape importante dans le développement d'un produit parce qu'ils augmentent la confiance dans une interopérabilité correcte de systèmes ouverts, pour lesquels on considère la conformité à un protocole de communication ou à une spécification de service comme un préalable indispensable. La Recommandation Z.500 – Cadre pour les méthodes formelles de tests de conformité [9] – définit la signification de la conformité en cas d'utilisation de méthodes formelles comme le SDL+ pour spécifier un protocole ou un service de communication. Elle contient aussi un guide relatif à la production de tests assistée par ordinateur.

La Recommandation Z.500 définit un cadre d'utilisation de méthodes formelles pour les tests de conformité. Elle est destinée aux réalisateurs, testeurs et spécificateurs participant aux tests de conformité afin de les aider à définir la conformité et le processus de test d'une mise en œuvre par rapport à une spécification présentée sous la forme d'une description formelle. Elle est applicable lorsqu'une spécification formelle existe pour un protocole ou un service de communication, à partir de laquelle on pourra construire une suite de tests de conformité. Elle pourra aider à suivre le processus du présent Supplément et à développer des outils pour la production de tests assistée par ordinateur. La Recommandation Z.500 définit un cadre et ne prescrit aucune méthode particulière de production de tests élémentaires. Elle ne prescrit pas non plus de relation de conformité particulière entre une spécification formelle et une mise en œuvre. Elle complète les Recommandations X.290-X.294, Cadre général et méthodologie des tests de conformité d'interconnexion des systèmes ouverts pour les Recommandations sur les protocoles pour les applications du CCITT [10], qui sont applicables à une large gamme de produits et de spécifications, y compris les spécifications en langage naturel. La Recommandation Z.500 interprète les concepts des tests de conformité dans un contexte formel.

Des méthodes concernant d'autres formes de tests, tels que ceux d'interopérabilité, pourront utiliser comme base les méthodes de tests de conformité.

Les Essais ne sont pas autrement décrits dans le présent Supplément. Pour de plus amples informations, voir [9] et [10].

4.4 Documentation

Les activités d'Analyse, de Conception du projet et de Formalisation produisent des textes et diagrammes dont certains sont nécessaires pour spécifier les produits. L'objet de l'activité de Documentation est de sélectionner les descriptions essentielles dans les documents et d'organiser ces descriptions de façon que la spécification de l'application soit facile à comprendre, concise et précise. Cette activité est particulièrement importante si la spécification fait partie intégrante d'une norme ou d'un document d'approvisionnement.

La description formelle en SDL+ est toujours incluse dans la description d'un produit. Si plusieurs descriptions formelles SDL ont été produites, une seule devra être incorporée ou d'autres descriptions devront être clairement marquées en tant qu'abstractions du modèle final. D'autres descriptions sont souvent requises pour comprendre la description formelle SDL. Il s'agit des suivantes:

– conceptions de projet telles que diagrammes de séquences de messages;

– informations et concepts structurés dans les informations classifiées;

– textes ou diagrammes informels dans les prescriptions collectées;

– expressions logiques sur l'état du système;

– autres textes et diagrammes informels non produits dans le cadre de la présente méthodologie.

En général, la Validation est facilitée si les spécifications de comportement sont étayées par des informations statiques redondantes, telles que des expressions logiques associées aux états du système.

La description de produit est structurée selon un ensemble de principes.

4.5 Parallélisme des activités

Une activité ne peut pas commencer avant que les critères de démarrage correspondants aient été satisfaits. Une activité est terminée lorsque les critères de terminaison correspondants ont été satisfaits. S'il n'y a pas de contraintes (telles que la disponibilité d'énergie, d'outils ou de compétences), les activités peuvent se dérouler en parallèle. Il en découle logiquement que toutes les activités peuvent progresser en parallèle: c'est souvent ce qui se produit en pratique, en raison des fréquentes variations des exigences ou du contexte d'un système au fur et à mesure du développement de celui-ci. Il faut donc analyser les nouvelles prescriptions et modifier les conceptions de projet en même temps que la Formalisation se déroule. Il arrive aussi très fréquemment que l'on subdivise l'ingénierie d'un système en plusieurs étapes et que celles-ci en soient à différents stades d'avancement, à cause de contraintes en termes de ressources ou parce que certaines parties critiques du système sont mises à l'étude en premier afin de déterminer la faisabilité, la viabilité ou le coût d'une conception particulière. La méthodologie n'exige donc pas qu'une activité donnée soit terminée avant qu'une autre activité commence; elle exige seulement que les critères de démarrage soient satisfaits.

5 Analyse d'activité

L'Analyse est une approche progressive pour explorer les prescriptions collectées, pour organiser les informations qui y sont contenues et pour enregistrer ces informations sous un format qui reflète cette organisation. L'Analyse devrait être utilisée lorsqu'un domaine d'application ou un système à spécifier est médiocrement compris, ou lorsque les prescriptions collectées contiennent différentes descriptions du même concept d'application, ou pour donner des noms bien définis aux concepts d'application.

Au cours de l'Analyse, les informations issues des prescriptions collectées sont organisées en termes de concepts d'application. Au moins certains de ces derniers, tels que "primitive de service" et "unité de donnée de protocole" sont susceptibles d'avoir été définis au préalable et d'être bien connus. Il est naturel d'associer certaines des informations issues des prescriptions collectées à ces concepts d'application fondamentaux. De nouveaux concepts d'application, qui peuvent souvent être définis en termes de concepts existants, sont également identifiés et nommés pour remplacer des informations propres au système. L'ensemble des concepts d'application accumulés et les relations établies entre eux permettent aux ingénieurs de réfléchir au système, de communiquer des idées à son sujet et ainsi de mieux le comprendre. Une méthode orientée objets est bien adaptée à ce processus de modélisation des concepts d'application.

Les informations classifiées qui sont produites par cette activité fournissent une base pour les autres activités méthodologiques, en particulier la création de la spécification formelle en SDL+.

L'activité d'Analyse est ici décrite en termes généraux, mais pour la conduire effectivement sur des systèmes de toute dimension raisonnable, il convient d'utiliser des notations définies et les outils correspondants. Les techniques appropriées sont habituellement désignées sous le terme de "techniques d'analyse orientée objets". Ce sont par exemple (dans l'ordre alphabétique): la technique OMT [11], l'ingénierie OOSE [12] ou la notation SOON [6]. Le paragraphe 13 (partie II du présent Supplément) développe l'Analyse au moyen de l'une de ces méthodes, choisie arbitrairement.

5.1 Début de l'Analyse

Au début de l'Analyse, l'on doit passer en revue les objectifs, les informations disponibles et les choix à effectuer.

5.1.1 Objectifs de l'Analyse

Les objectifs de l'Analyse sont les suivants:

– résoudre toutes déficiences détectées dans les prescriptions collectées;

– établir les fondements des activités de Conception du projet et de Formalisation;

– identifier, nommer et définir les concepts d'application du système;

– produire les informations classifiées qui sont encore une spécification informelle afin de structurer les informations contenues dans les prescriptions collectées et de soutenir d'autre travaux de développement, d'exploitation et de maintenance sur le système.

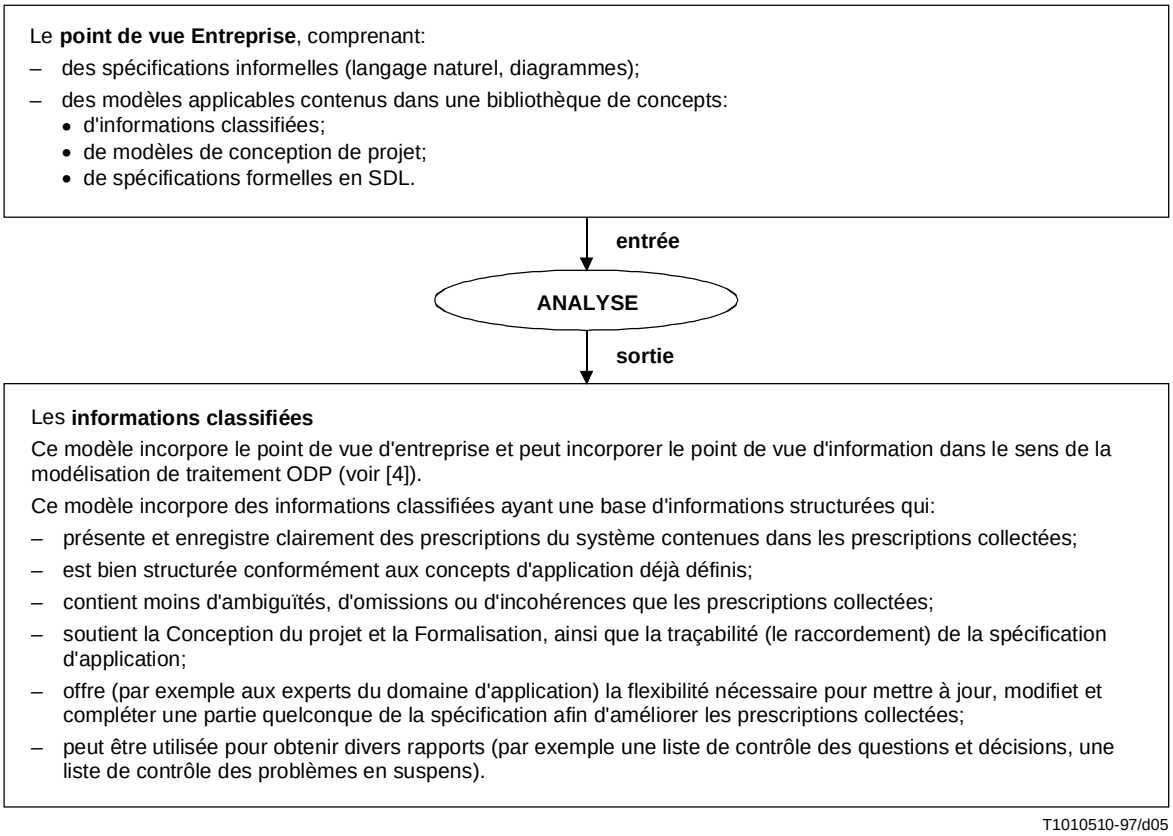


Figure 5-1/Suppl. 1 à la Rec. Z.100 – Entrée et sortie de l'activité d'Analyse

5.1.2 Détermination de l'opportunité d'omettre l'Analyse

NOTE 1 – Le présent sous-paragraphe définit la façon d'évaluer les informations et le degré de compréhension dont dispose l'ingénieur, avant que l'Analyse puisse être commencée. L'Analyse n'est pas toujours nécessaire pour le développement d'un système: la mise au point de la spécification peut alors procéder directement à la Conception du projet ou à sa Formalisation.

NOTE 2 – La méthodologie n'impose aucune séquence stricte quant à la chronologie de l'activité d'Analyse. Elle ne donne que des avis à l'ingénieur sur la façon et le moment d'utiliser l'activité d'Analyse.

Avant de commencer l'Analyse, l'ingénieur formule un jugement sur l'adéquation des concepts d'application contenus dans les prescriptions collectées et sur la compréhension qu'il en possède. Il décide ensuite si les prescriptions collectées sont décrites au niveau de détail qui est approprié à l'Analyse, à la Conception du projet ou à la Formalisation.

Les prescriptions collectées sont bien structurées et l'Analyse peut être omise lorsque les caractéristiques de la liste de contrôle suivante sont satisfaites:

1) il n'y a pas de capacités ou de données identiques à différents emplacements;
2) il est facile d'extraire les caractéristiques essentielles de chaque concept d'application pour le distinguer des autres et il existe un terme bien défini et unique pour chaque concept d'application;
3) les détails sont cachés s'ils ne font pas partie des caractéristiques essentielles des concepts d'application;
4) les concepts d'application se subdivisent en un ensemble de parties cohérentes qui peuvent être définies séparément;
5) les concepts d'application sont bien ordonnés. Il importe aussi bien de grouper des fonctions et données similaires que de décomposer logiquement les données en éléments plus petits.

5.2 Questions posées en cours d'Analyse

Dans la Figure 4-3, la flèche qui va de l'Analyse aux prescriptions collectées représente les questions qui se posent au cours du développement d'une application. Les questions relatives à la complétude, à la cohérence et à d'autres attributs qualitatifs des prescriptions peuvent donner lieu à des modifications des prescriptions existantes et à l'adjonction à l'ensemble de nouvelles prescriptions. Lorsque les questions ne peuvent pas être résolues dans le cadre de la méthodologie, elles doivent l'être par les parties extérieures à la présente méthodologie concernant l'activité de saisie des prescriptions.

5.3 Méthode de modélisation pour l'Analyse

Avant l'Analyse, les idées sur le système peuvent être:

– limitées: un nombre limité de concepts d'application sont connus de l'ingénieur;

– arbitraires: les ingénieurs créent et classifient des concepts d'application d'après leurs connaissances personnelles;

– spécifiques: chaque ingénieur possède un modèle mental, fondé sur son propre parcours;

– complexes: chaque ingénieur examine un certain nombre de points de vue dans son propre modèle.

L'Analyse utilise une méthode orientée objets pour résoudre ces problèmes. Cette méthode aide l'ingénieur:

– à fournir un modèle structuré, fondé sur une forme modulaire et donc stable qui est mieux compatible avec les modifications éventuelles (qui n'affecteront que certains des objets pris en compte dans le modèle);

– à subdiviser la complexité du problème en plus petites parties qui seront examinées séparément;

– à avoir une vue de la spécification qui soit commune avec celle de tous les ingénieurs participant au projet;

– à éviter les possibilités d'erreurs, d'ambiguïtés et d'incohérences;

– à garder trace des travaux.

En conclusion, pour définir le système, les ingénieurs possèdent un modèle qui se compose:

– d'une vue logique: pour décrire les éléments clés du système à spécifier;

– d'une vue dynamique: pour décrire le comportement individuel d'un objet ainsi que le comportement de tous les objets.

La vue logique est exprimée dans une notation de classe d'objets appropriée. La vue dynamique est habituellement exprimée en diagrammes MSC.

5.4 Etapes de l'Analyse

L'activité d'Analyse se compose de deux étapes: l'inspection et la classification.

5.4.1 Inspection

Dans l'étape d'inspection, un relevé systématique du contenu des prescriptions collectées est effectué. L'exécution de l'inspection permet à l'ingénieur d'évaluer si les connaissances ou compétences existant au sujet du domaine d'application sont suffisantes et si la spécification sera donc réalisable. Comme les recherches dans un domaine d'application peu connu peuvent être coûteuses, il est désirable d'identifier la nécessité d'une telle recherche aussitôt que possible dans le processus de spécification.

Les objectifs visés lors de l'exécution de l'inspection sont les suivants:

– se faire une idée générale du domaine d'application;

– évaluer l'adéquation des prescriptions collectées au domaine d'application;

– déterminer si le domaine d'application est couvert entièrement ou partiellement par les prescriptions collectées.

Les étapes d'inspection sont les suivantes:

1) collecter les données de source:

   a) compiler une liste de toutes les sources d'information. Une bibliographie, représentée sous forme de tableau créé par un progiciel de traitement de texte peut être suffisante;

   b) catégoriser chaque document en fonction de son apparente pertinence au domaine d'application;

2) inspecter chaque document, en commençant par le plus pertinent;

3) réévaluer la pertinence de chaque document au domaine d'application;

4) lire intégralement le document le plus pertinent, mais sans interruptions pour analyser les passages difficiles ou mal compris.

5.4.2 Classification

La classification se compose des opérations suivantes:

– identification des concepts d'application, de leur structure et de leurs divers aspects dans les prescriptions collectées;

– identification des informations redondantes ou manquantes dans les prescriptions collectées;

– établissement de connexions entre l'entrée dans l'Analyse et les informations classifiées.

La classification conduit à l'élaboration d'un modèle orienté objets logique décrivant dans le système:

– la structure des classes identifiées;

– la structure des objets identifiés dans les classes;

– les aspects associés à chaque objet dans une classe (aspects relatifs aux informations, aspects relatifs au comportement, aspects relatifs aux interfaces et aspects divers).

Le but de cette classification est d'identifier les classes et leurs objets jusqu'à un certain niveau de détail puis de s'assurer qu'ils sont convenablement nommés et définis.

La classification comporte 5 étapes:

1) identification et nomination de la partie du système à classifier;

2) identification et nomination des classes;

3) identification des structures d'objet et de leurs relations;

4) définition des classes et objets puis examen de leurs noms;

5) identification des aspects de chaque classe.

Les aspects relatifs au comportement des objets peuvent être saisis sous forme de diagrammes MSC. Il doit y avoir un seul objet (composite) qui représente le système et un ou plusieurs objets pour l'environnement. Il peut y avoir plusieurs cas séquentiels montrant, dans les diagrammes MSC, l'utilisation du système.

5.5 Conclusion de l'Analyse

Les résultats doivent être passés en revue afin de déterminer si l'Analyse peut être considérée comme complète: en d'autres termes, le critère de saut d'Analyse doit s'appliquer. Les activités de Conception du projet et de Formalisation peuvent être engagées avant qu'il soit conclu qu'une Analyse suffisante a été effectuée. Les critères de terminaison de l'Analyse sont donc différents des critères de lancement des activités de Conception ou de Formalisation du projet.

Les informations classifiées sont considérées comme une base d'informations structurées. Il doit y avoir une liste de contrôle permettant de déterminer si les informations classifiées sont suffisamment structurées en objets et si ces derniers sont suffisamment nommés et définis pour que l'on puisse terminer la Conception du projet et sa Formalisation. La liste de contrôle permet également de détecter et de traiter d'éventuelles incohérences et insuffisances dans la recherche des prescriptions collectées.

6 Conception du projet

L'objectif général de la Conception du projet du système est de mettre au point des spécifications partielles ou partiellement informelles à partir de différents points de vue et à différents niveaux de détail. La Conception du projet n'a pas besoin d'être complète ou strictement formelle mais elle doit permettre la recherche des différents modèles représentant les parties du système, de manière que l'ingénieur puisse faire des choix de conception.

Cette activité fait usage des prescriptions collectées et des informations classifiées, ainsi que des connaissances acquises au cours de l'Analyse et, le cas échéant, de modèles extraits de la bibliothèque de concepts.

La Figure 6-1 décrit l'entrée et la sortie de l'activité de Conception du projet.

L'ingénieur utilise et développe les modèles existants afin de produire des conceptions de projet fournissant un point de vue information plus formel et décrivant des points de vue traitement du système. Au besoin, des points de vue conception généraux peuvent être inclus. Les conceptions de projet permettent d'examiner de manière approfondie le comportement du système avant d'effectuer une formalisation systématique dans le cadre de l'activité correspondante. Dans une grande mesure, la Conception du projet et la Formalisation peuvent avoir lieu en parallèle, à condition que les conceptions de projet permettant la Formalisation aient été faites avant qu'elles soient nécessaires dans les étapes de Formalisation. Par exemple, les conceptions de projet produisent des séquences d'utilisation MSC typiques qui peuvent être utilisées lorsque des processus-squelettes sont produits pour la Formalisation.

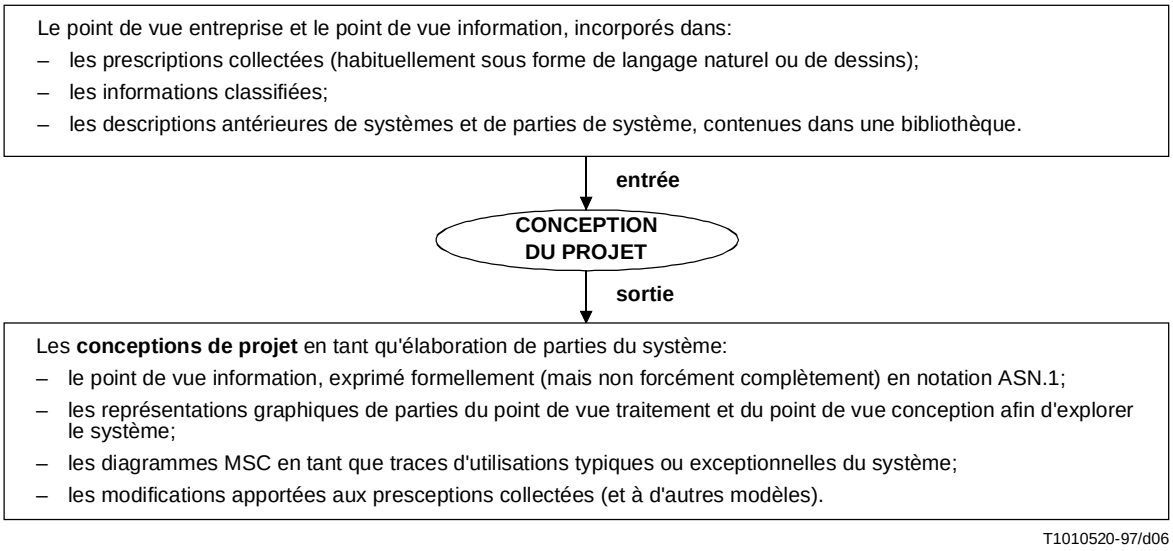


Figure 6-1/Suppl. 1 à la Rec. Z.100 – Entrée et sortie de la Conception du projet

Il existe une différence entre la façon dont les prescriptions collectées sont utilisées dans l'activité de Conception du projet et la façon dont elles sont utilisées dans l'activité d'Analyse:

– l'Analyse est centrée sur les noms (quels sont les objets présents?) contenus dans les prescriptions collectées afin de structurer et de définir des concepts produisant de façon informelle le point de vue entreprise et un point de vue information;

– la Conception du projet est centrée sur les verbes (que peuvent faire les objets?) contenus dans les prescriptions collectées (ou sur les verbes saisis dans les aspects comportementaux des informations classifiées) afin de calculer de manière formelle un point de vue traitement et, au besoin, un point de vue conception.

Les informations classifiées qui sont utilisées dans la Conception du projet sont soit le résultat de l'Analyse soit, si l'Analyse n'a pas été nécessaire, des descriptions équivalentes dans le cadre des prescriptions collectées. Bien que d'autres langages ou techniques puissent être utilisés pour la structuration lors de l'Analyse, le contenu de la structure sera probablement en langage naturel. La Conception du projet développe le point de vue information incorporé dans les informations classifiées afin de le rendre formel et donc de pouvoir utiliser les données à l'appui du comportement. Le point de vue information produit au cours de la Conception du projet est formel. Il est suggéré de le définir en notation ASN.1.

Etant donné que l'objectif final de l'activité de Formalisation est de produire un modèle et une description utilisant le langage SDL+, la Conception du projet utilise ce langage chaque fois que possible. La différence essentielle entre une conception de projet en SDL+ et un modèle formel est la couverture et la complétude. Aux fins de la Conception du projet, on peut se permettre de ne prendre en considération que les cas typiques et des parties limitées du système, de supposer que certains opérateurs de données sont bien définis et de ne pas tenir compte de certaines des règles imposées par les langages formels. Une conception de projet est une esquisse de la spécification, alors qu'un modèle formel doit être précis, univoque et interprétable.

De façon que tous les modèles produits soient localisables, les conceptions de projet sont associées aux informations classifiées et aux prescriptions collectées.

6.1 Début de la Conception du projet

Etant donné qu'il y a diverses raisons pour produire des conceptions de projet, les objectifs de la Conception du projet devraient être passés en revue et enregistrés. Les objectifs enregistrés deviennent des critères dans la liste de contrôle d'adéquation de la Conception du projet.

Les objectifs habituels sont les suivants (au choix):

1) fournir un diagramme général du système sous forme de documents de travail pour les ingénieurs impliqués;

2) identifier les parties critiques du système;

3) explorer la faisabilité des parties critiques du système;

4) déterminer un ensemble typique de séquences d'utilisation;

5) identifier les parties comportementales nécessaires pour assurer le service ou le protocole;

6) déterminer les parties comportementales nécessaires (les parties fonctionnelles du comportement qui seront normatives);

7) esquisser un modèle incorporant des parties normatives et des parties non normatives, les parties non normatives étant habituellement ajoutées de façon que le modèle puisse être interprété et que le comportement puisse être analysé;

8) clarifier et identifier les interfaces normatives et non normatives, aussi bien aux frontières qu'à l'intérieur du système décrit;

9) identifier les critères de recherche permettant de retrouver, dans une bibliothèque de concepts (en SDL+), les parties réutilisables.

Un objectif toujours applicable consiste à identifier les éléments manquants, les incohérences et les ambiguïtés dans les informations classifiées ou dans les prescriptions collectées. Les questions, les réponses et les sujets de complément d'étude associés aux informations classifiées sont enregistrés au cours de la Conception du projet. Il peut en résulter des modifications aux informations classifiées si une erreur d'ingénierie a été relevée, ou aux prescriptions collectées si celles-ci doivent être modifiées.

6.1.1 Détermination de l'opportunité d'omettre la Conception du projet

NOTE – Le présent sous-paragraphe définit la façon d'évaluer les informations dont dispose l'ingénieur et la compréhension que celui-ci possède du système avant le début de la Conception du projet. Celle-ci n'est pas forcément nécessaire pour la mise au point d'un système. Si ce n'est pas le cas, le développement peut passer directement à la Formalisation bien que, dans ce cas, des étapes de production de notation ASN.1 et de diagrammes MSC puissent être invoquées, le cas échéant, à partir de la Formalisation.

Avant le début de la Conception du projet, les informations classifiées sont évaluées afin de déterminer s'il est possible de sauter la Conception du projet et de passer à la Formalisation. La Conception du projet n'est pas forcément nécessaire si plusieurs critères de la liste de contrôle suivante sont satisfaits:

1) la catégorie du système est bien connue et bien comprise par les ingénieurs impliqués;

2) les ingénieurs impliqués sont expérimentés en SDL+;

3) les informations classifiées ont une relation claire et directe avec les interfaces du système et ont une structure de bloc en langage SDL;

4) les informations classifiées contiennent déjà des conceptions de projet en SDL+ pour au moins les cas typiques;

5) les informations classifiées contiennent déjà un point de vue information exprimé en notation ASN.1;

6) l'on ne prévoit aucune difficulté pour mettre en séquence ou pour structurer des données en SDL+;

7) l'on ne prévoit pas la nécessité de modéliser de multiples niveaux d'abstraction avant que la description formelle ait été produite en SDL+;

8) l'architecture dans laquelle le SDL+ doit s'intégrer n'impose aucune limitation ou contrainte nécessitant une recherche;

9) aucun problème de faisabilité ne se pose en SDL+ concernant la modélisation par machine à états finis.

6.1.2 Critères pour commencer la Conception du projet

Avant de commencer la Conception du projet, l'on doit disposer d'informations classifiées qui:

1) comportent un point de vue entreprise du système;

2) décrivent des concepts et des noms pour certaines parties du système;

3) donnent un point de vue information sur les principales parties du système.

6.1.3 Notations de conception de projet

Les notations sélectionnées dépendront de la technique d'analyse orientée objets qui sera employée pour l'Analyse (si cette activité n'a pas été omise) ou de la forme des informations d'entrée. Les notations suggérées pour la Conception du projet sont le SDL+ (utilisé informellement) et la notation utilisée pour les objets et les classes afin d'exprimer des relations plus détaillées entre les entités.

Les notations sélectionnées devraient offrir une base saine pour la validation et pour la spécification de tests de conformité. Les notations devraient être sélectionnées de façon à être utiles pour la Formalisation. C'est pourquoi le langage SDL+ devrait être utilisé chaque fois que possible. L'introduction d'une notation autre que SDL+ ou d'une notation utilisée dans les informations d'entrée devrait être évitée.

6.2 Etapes de la Conception du projet

Les étapes générales de l'activité de Conception du projet sont les suivantes:

1) constituer un modèle contextuel dans lequel le système est identifié et où l'environnement du système est détaillé, puis décrire les interfaces de communication et les autres relations que le système doit gérer;

2) créer ou compléter les diagrammes MSC qui décrivent les séquences d'utilisation, c'est-à-dire les séquences (protocoles) d'interaction typiques dans chaque couche des interfaces;

3) esquisser la structure du système et identifier les parties qui sont soumises aux prescriptions;

4) étendre les diagrammes MSC pour couvrir des situations moins courantes, situées aux limites des prescriptions et des pannes possibles;

5) détailler les informations contenues dans les interfaces et définir un modèle d'information pour le système;

6) définir un système prototype (non nécessairement formel ou exécutable) ou les parties du système qui peuvent gérer les séquences d'utilisation "intéressantes" (c'est-à-dire celles qu'il faut explorer).

Etant donné qu'un des objectifs de la Conception du projet est d'esquisser des spécifications pour explorer des approches, certains des modèles produits se révéleront inappropriés ou inapplicables. Lorsque les conceptions de projet ne conduisent pas directement à un résultat formalisé, la cause du problème est recherchée soit dans les prescriptions soit dans la conception; puis des variantes sont testées. Il en résulte une répétition d'une ou de plusieurs étapes. Il est inévitable que certaines des conceptions de projet produites soient abandonnées par la suite. Il convient de ne pas supposer que chaque conception de projet conduira directement à la description formelle finale en SDL. L'avantage est que les solutions qui ne sont pas concrétisables ou qui ne sont pas intéressantes peuvent être explorées puis abandonnées sans frais excessifs.

Le résultat de chaque étape devrait contenir des "liens" pour revenir aux informations classifiées et aux prescriptions collectées, ainsi qu'un enregistrement des questions soulevées par cette étape. A chaque question correspond une réponse ou une indication du fait que la question reste sans solution. Il est également probable que certaines conceptions de projet seront rejetées. Le motif du rejet de chaque conception de projet est enregistré.

6.3 Conclusion de la Conception du projet

Lorsque la Conception du projet a été effectuée, l'on dispose d'un ensemble de résultats comme indiqué ci-dessous. Ces résultats doivent être passés en revue afin de déterminer si l'activité de Conception du projet peut être considérée comme achevée.

Les données contenues dans le système, ainsi que la structure et les interfaces de celui-ci, devraient être décrites et comprises suffisamment bien, de façon que:

1) les objectifs de la Conception du projet, tels qu'enregistrés au début de cette activité, soient atteints;

2) la Formalisation puisse être effectuée.

7 Formalisation

L'objectif général de la Formalisation est la production d'une spécification formelle pour le système à utiliser comme application et pour la Validation. L'on utilise toutes les informations classifiées ou tous les modèles de conception de projet représentés sur la Figure 7-1, toutes les connaissances acquises lors de l'Analyse et de la Conception du projet, et si possible des modèles extraits de la bibliothèque de concepts. Certains mappages sont suggérés dans les étapes détaillées de Formalisation (voir le paragraphe 15), entre les modèles issus des activités d'Analyse et de Conception du projet et la description formelle en SDL.

L'ingénieur formalise et produit des conceptions raffinées pour identifier, comprendre et analyser l'application à un niveau détaillé. Les travaux sont guidés par l'utilisation du langage SDL+. La Figure 7-1 décrit l'entrée et la sortie de l'activité de Formalisation.

Les informations classifiées qui sont référencées dans l'activité de Formalisation sont soit les résultats de l'Analyse ou (si celle-ci n'était pas nécessaire) les descriptions équivalentes qui sont contenues dans les prescriptions collectées. De même, les références aux conceptions de projet sont soit les résultats de l'activité de Conception du projet ou (si celle-ci n'était pas nécessaire) les descriptions équivalentes qui sont contenues dans les informations classifiées (ou dans les prescriptions collectées).

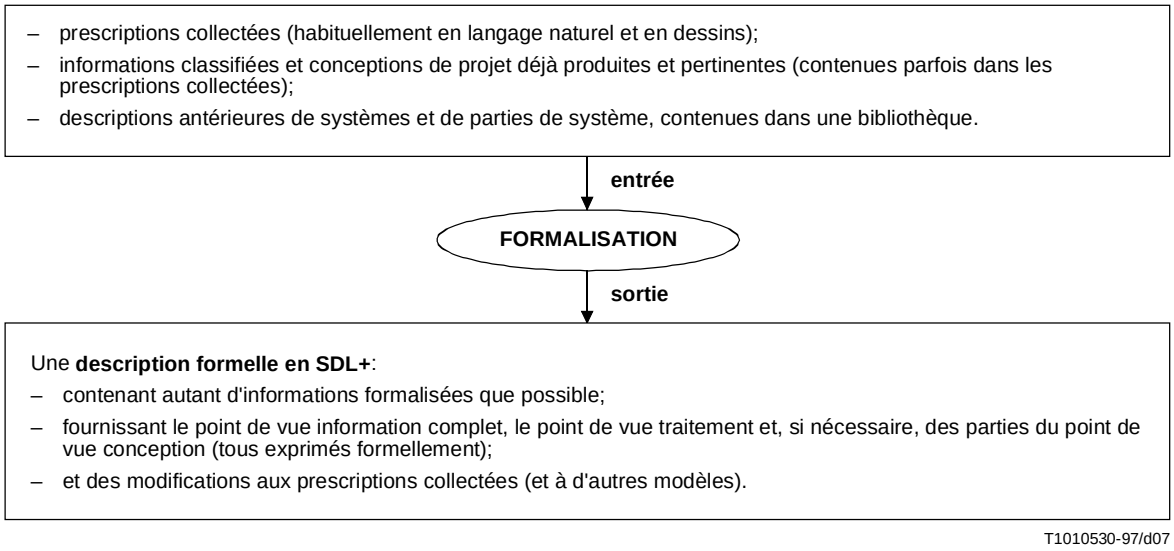


Figure 7-1/Suppl. 1 à la Rec. Z.100 – Entrée et sortie de la Formalisation

7.1 Début de la Formalisation

Au début de la Formalisation des objectifs, il faut passer en revue les entrées disponibles et l'utilisation des formalismes.

Les principaux objectifs de la Formalisation sont les suivants:

– production réelle de la description en SDL+;

– recherche du système guidée par l'utilisation du langage SDL+:

  au fur et à mesure que le langage SDL+ est produit, la compréhension augmente et la recherche progresse. Si l'activité d'Analyse ou de Conception du projet est omise, la recherche en cours de Formalisation est plus importante. Les directives associées aux étapes de Formalisation couvrent la recherche requise;

– cohérence et traitement des incohérences:

  la création des descriptions en langage SDL+ peut mettre en évidence des incohérences et des ambiguïtés dans les conceptions de projet, dans les informations classifiées et dans les prescriptions collectées. Les questions, les réponses et les points à étudier en liaison avec chacune de ces entrées sont enregistrés au cours de la Formalisation. Il peut en découler des modifications de certaines entrées et donc (à moins qu'une erreur d'ingénierie n'ait été faite dans la Conception du projet ou dans son Analyse) aux prescriptions collectées.

Avant le début de la Formalisation, les critères suivants doivent être satisfaits:

1) il doit y avoir des informations classifiées contenant un point de vue entreprise du système;
2) il doit y avoir des informations classifiées décrivant des concepts et des noms pour les principaux éléments du système;
3) il doit y avoir des informations classifiées contenant des descriptions d'interface pour les interfaces normatives;
4) il devrait y avoir des conceptions de projet contenant les principaux éléments structurels du système, comme le modèle fonctionnel utilisé dans les normes de la série Q.1200 [13];
5) il devrait y avoir des conceptions de projet contenant les principaux éléments d'un point de vue information du système.

7.2 Etapes de la Formalisation

Les étapes à suivre au cours de la Formalisation dépendront dans une certaine mesure des techniques utilisées pour l'Analyse ou pour la Conception du projet, ainsi que du contexte dans lequel le langage SDL+ sera utilisé. La spécification formelle sera exprimée en SDL+: les étapes ne dépendront donc pas de la notation utilisée. Les limitations de certains outils peuvent toutefois rendre impossible l'utilisation de certaines constructions SDL+, et il peut exister d'autres contraintes (des spécifications d'utilisateur par exemple) qui induisent des variations dans l'utilisation du SDL+. Il est cependant nécessaire de toujours définir la structure, le comportement et les données du système. Les étapes de la Formalisation peuvent viser ces définitions. Des étapes peuvent aussi être ajoutées pour exploiter les capacités typiques du langage SDL-92. Par ailleurs, si des diagrammes MSC n'ont pas déjà été produits au cours de la Conception du projet, ces diagrammes seront nécessaires pour faciliter la Formalisation SDL du processus.

Etant donné que le contexte a peu d'incidence, les étapes de Formalisation indiquées au paragraphe 15 peuvent être utilisées comme base pour un ensemble d'étapes dans un contexte quelconque. Ces étapes se subdivisent comme suit: structure, comportement, données, type et localisation. Les deux dernières étapes s'appliquent lorsque des parties du système peuvent être réutilisées. Lorsque ces étapes sont utilisées comme base pour un autre contexte, chacune d'elles devra être passée en revue, certaines pourront être supprimées et d'autres pourront être ajoutées.

Les informations classifiées et les conceptions de projet fournissent les éléments de compréhension et d'information nécessaires pour effectuer la Formalisation. Même si ces éléments ne sont pas mentionnés explicitement dans une étape particulière, ils sont toujours utilisés comme entrée. Les conceptions de projet devraient être exprimées en SDL+, auquel cas une partie de la Formalisation peut être triviale car certaines des descriptions peuvent ne nécessiter qu'un contrôle (et au besoin une modification) pour être correctes quant aux règles linguistiques et autres règles données pour chaque étape. Habituellement, de telles conceptions de projet sont cependant incomplètes et informelles, de sorte qu'elles ne couvrent qu'une partie du système. On trouve habituellement dans les directives une référence à l'utilisation des conceptions de projet et aux informations classifiées. Une partie des conceptions de projet (comme celles qui produisent des diagrammes MSC détaillés) sera probablement conduite en parallèle avec la Formalisation.

Le résultat de chaque étape comportera des "liens" vers les questions, décisions, prescriptions collectées, informations classifiées et conceptions de projet associées à cette étape. Cela permettra de remonter jusqu'à d'autres descriptions à partir de la description SDL.

7.3 Conclusion de la Formalisation

La Formalisation produit un modèle complet du système en SDL-92. Associé aux autres documents issus de l'Analyse et de la Conception du projet, ce modèle fournit une spécification complète du système. Il comporte le point de vue traitement du système et les aspects du point de vue conception du système qui doivent être intégrés à l'application.

Les étapes de Formalisation visent à produire un modèle exécutable sauf, le cas échéant, pour certaines sortes de données qui peuvent habituellement être rendues exécutables par utilisation interactive d'un outil logistique ou avec les données d'un langage de programmation. L'intérêt d'un modèle exécutable est que son exécution permet de démontrer la faisabilité et la fonctionnalité du système. Son inconvénient est que certains aspects du système doivent être définis explicitement pour que le système soit exécutable. Par exemple, les données transmises en transparence par le système doivent être modélisées de manière assez explicite (par exemple au moyen d'une chaîne de caractères) pour que le modèle puisse être exécuté. Mais, pour l'application, il peut s'agir du concept abstrait d'une unité de données.

Le modèle SDL+ est exécutable en sorte que l'exécution du modèle puisse être utilisée au cours de la Validation et que le modèle puisse être soumis à des tests de conformité montrant que ces tests sont compatibles avec le modèle.

Le résultat de la Formalisation devrait être contrôlé en fonction des critères suivants:

1) les attributs qualitatifs du système devraient être satisfaits par le SDL+ formel;
2) le SDL+ formel doit être conforme à la Recommandation Z.100 [1], à la Recommandation Z.105 [2] et à la Recommandation Z.120 [3];
3) la description formelle en SDL+ doit être compatible avec les conceptions de projet;
4) le langage SDL+ doit être conforme aux règles éventuellement formulées dans les étapes de Formalisation;
5) il doit avoir été démontré (par revue ou audit de projet approfondi et par exécution avec des séquences d'entrée définies) que la description formelle en SDL modélise de manière satisfaisante les capacités du système décrit par les prescriptions collectées.

8 Mise en œuvre

Pour certaines applications, l'importance des travaux d'ingénierie à effectuer pour transformer un modèle formel SDL+ en mise en œuvre utilisable peut être faible sinon triviale (par exemple la compilation du SDL+ pour le matériel cible). Mais, en théorie, le modèle SDL+ initial et la mise en œuvre correspondent à des niveaux d'abstraction tout à fait différents. Conformément à la méthodologie exposée dans le présent Supplément, la *description formelle en SDL* de la Figure 4-3 ne tiendra pas compte des prescriptions de mise en œuvre non fonctionnelles et des contraintes qui **ont** à être incluses dans le *produit* selon la même figure. Si la description formelle en SDL+ est très abstraite, il est alors possible de raffiner la description afin de produire une autre description formelle en SDL+ autorisant un plus grand nombre de prescriptions et de contraintes sur le produit. Lorsqu'il existe un grand nombre de contraintes ou que celles-ci sont importantes (s'appliquant par exemple à des questions de qualité de fonctionnement ou de répartition), le langage SDL+ peut être raffiné sélectivement afin de produire des systèmes fonctionnellement équivalents mais opérationnellement différents.

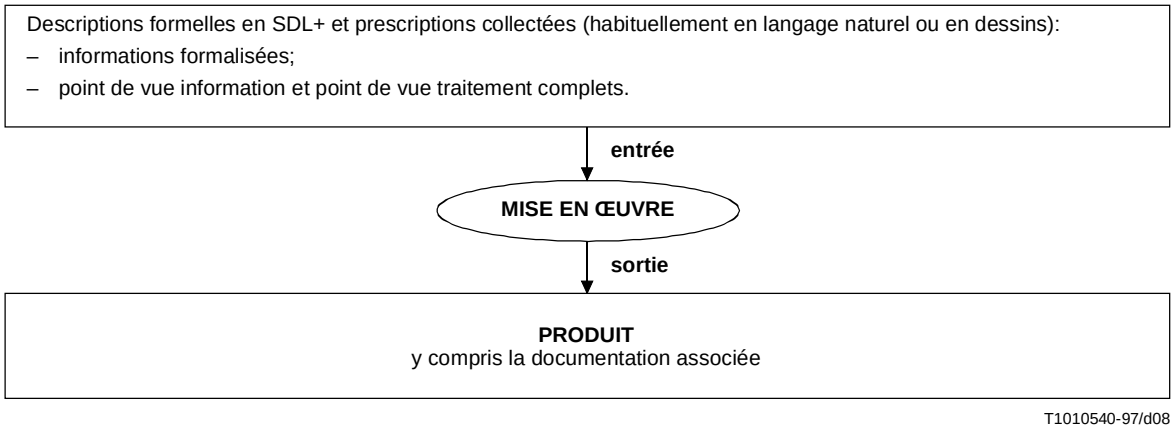


Figure 8-1/Suppl. 1 à la Rec. Z.100 – Entrée et sortie de la mise en œuvre

Il y aura certaines contraintes élémentaires (par exemple l'interfaçage avec le système sous-jacent) qui seront soit inexprimables en SDL+ ou exprimables plus efficacement dans un autre langage (par exemple en C++ ou en appels au système d'exploitation ou en assembleur). Il pourra même arriver qu'une partie du langage SDL ne soit pas mise en œuvre sous forme logicielle (dans le sens courant du terme). Ces situations n'impliquent pas que le projet SDL+ doive être complètement réécrit dans un autre langage car il existe des outils qui peuvent traduire automatiquement le SDL en langages de programmation tout en conservant le SDL pour décrire le système. Ces outils permettent d'insérer des liens vers des parties écrites en d'autres langages.

Comme la mise en œuvre dépend plus de l'application et des organisations que les autres activités, elle présente d'importantes variations. Seule une vue très générale sera donnée ici. Les principales étapes à prendre en compte sont les suivantes:

1) définir le point d'équilibre entre matériel et logiciel (architecture globale);

2) déterminer l'architecture du matériel pour soutenir le logiciel (architecture matérielle);

3) mettre au point l'architecture logicielle;

4) restructurer et raffiner la description SDL+ en ajoutant au besoin des descriptions pour définir le produit.

La description SDL+ peut être prise comme point de départ, de concert avec des aspects divers contenus dans les informations classifiées.

Les facteurs clés de l'architecture globale sont les suivants: prescriptions pour la répartition physique et pour les largeurs de bande de communication impliquées entre éléments répartis; prescriptions de qualité de fonctionnement y compris les situations de surcharge, les réponses chronologiquement critiques et le débit utile moyen; prescriptions de fiabilité et de redondance. Quelques choix devront être effectués au sujet de ce qui devra être retenu et de sa place, ainsi que de ce qui devra être mis en œuvre matériellement. Il est parfois nécessaire que l'ensemble du système soit sous forme logicielle. Il faudra examiner l'avantage économique des solutions matérielles (habituellement plus coûteux mais plus rapide) par rapport aux solutions logicielles. L'utilisation de modèles ou d'équipements matériels ou logiciels existants peut entrer en ligne de compte.

Il faut définir l'architecture matérielle dans les termes suivants: nombre de processeurs et d'autres unités de traitement, nombre de connexions et matériel de transmission associé, caractéristiques de qualité de fonctionnement des éléments matériels, comme la puissance de traitement et les temps de propagation sur les connexions physiques. Si le système matériel est défini dans les prescriptions, il importera de déterminer si ce système est réellement en mesure de soutenir le logiciel et de répondre aux contraintes de performance.

L'architecture logicielle est un développement d'une description en SDL+. Ce développement peut nécessiter une certaine restructuration d'une description en SDL en fonction de la répartition du SDL parmi les unités physiques. Cela peut nécessiter une transformation de certains blocs du système SDL en systèmes SDL coopératifs. Il faudra un système exécutable pour soutenir le SDL ainsi qu'un mécanisme (pouvant faire partie du système exécutable) pour convertir les événements externes en signaux SDL.

On trouvera des directives détaillées en [6].

9 Validation

La Validation a deux aspects:

– le contrôle du fait que la syntaxe et la sémantique d'une spécification sont correctes;

– le contrôle du fait que les prescriptions connues sont exprimées par la spécification.

Bien que la Validation soit fréquemment accomplie par examen d'expert, une des principales raisons d'exprimer une spécification en langage SDL+ est de permettre la validation assistée par ordinateur. Le premier aspect de la Validation est bien adapté à l'automatisation. La conformité aux règles du SDL+ garantit la cohérence interne d'une spécification et son absence d'ambiguïté involontaire. Les outils SDL+ automatisent le contrôle de la syntaxe et de la sémantique statique. Le contrôle de la sémantique dynamique nécessite habituellement l'exécution du modèle SDL+. Certains tests élémentaires seront nécessaires pour mettre le modèle à l'épreuve. La sémantique dynamique peut également être validée par des techniques telles que la recherche de l'espace des états.

Le deuxième aspect de la Validation a été plus difficile à automatiser. La preuve formelle qu'une spécification satisfait des prescriptions n'a jusqu'à maintenant été automatisée que pour des protocoles simples. Il est néanmoins possible d'améliorer, au moyen de techniques automatisées telles que la simulation et l'application de tests, le degré de certitude que la spécification répond aux prescriptions.

La définition d'un système en SDL+ ressemble dans une grande mesure à la rédaction d'un logiciel en langage de programmation. Comme les programmes logiciels, il faut soumettre la description formelle en SDL+ *à des tests*, car il est très probable qu'elle contiendra initialement des erreurs d'ingénierie et ne fournira pas les prescriptions connues. De telles erreurs apparaissent malgré l'application des contrôles en cours de création du SDL+, parce que l'ingénierie elle-même ne peut pas être totalement automatisée et que les êtres humains commettent des erreurs. L'application des tests dans le cadre de la Validation ressemble aux essais d'un programme concurrent. Mais, à la différence de celui-ci, qui est censé fonctionner sur une machine réelle dans un environnement réel, le système SDL+ s'exécute sur une machine abstraite avec des ressources normalement illimitées. A part cela, les essais du SDL+ sont analogues à ceux d'un logiciel. De tels essais contribuent aux deux aspects de Validation décrits plus haut.

Pour rendre le SDL+ exécutable sur une machine réelle de façon à pouvoir appliquer les tests, il est parfois nécessaire de modifier le modèle (et les tests) SDL+. Le modèle modifié est un modèle de validation. De même, une technique de non-exécution (telle que la recherche de l'espace des états) peut également, pour valider du SDL+ par examen automatisé, nécessiter qu'un modèle de validation modifié soit extrait du SDL+. La Validation par essais ou par application d'autres outils à des modèles de validation est appelée *validation formelle*.

Les diverses activités peuvent produire de nombreuses définitions, l'une raffinant l'autre ou décrivant le système d'un point de vue différent. Les définitions peuvent faire appel à différentes techniques de spécification, telles que le langage naturel, les diagrammes MSC, la notation ASN.1, la notation TTCN. Le processus de Validation garantit que les spécifications sont en harmonie les unes avec les autres. Il importe de noter ici que le SDL+ n'existe habituellement pas isolément mais est inséré dans un contexte et doit donc être validé dans le cadre de celui-ci.

On peut non seulement utiliser un certain nombre de notations différentes mais une définition de système peut faire référence à d'autres spécifications telles que des normes, sans les incorporer explicitement. Dans ces cas, il faut élaborer un ou plusieurs modèles de validation formelle avant de pouvoir contrôler l'exactitude par des moyens formels. Si une spécification contient des options, il faut en sélectionner un ensemble particulier afin de rendre le modèle exécutable. La Figure 9-1 montre le schéma général de Validation. Les cercles représentent des activités et les rectangles des documents.

La validation formelle révèle des erreurs que les développeurs font remonter jusqu'au modèle de validation ou jusqu'à la spécification. Une fois ce diagnostic posé, le modèle de validation est révisé ainsi que, si nécessaire, la spécification. Puis le modèle de validation est réexécuté.

La méthodologie décrite dans le présent sous-paragraphe ne couvre que l'extraction et l'exécution du modèle de validation, qui ne constitue qu'une partie de l'activité de Validation décrite dans la Figure 4-3. Certains des contrôles requis pour la validation sont couverts par l'Analyse, par la Conception du projet ou par la Formalisation. D'autres contrôles, non couverts par le présent Supplément, peuvent être extraits de l'application de techniques d'assurance de la qualité des logiciels (voir également la Recommandation Z.400 [14]).

9.1 Caractéristiques d'un modèle de validation

Un modèle de validation possède deux caractéristiques essentielles:

– le langage SDL+ est suffisamment détaillé pour que le modèle soit exécutable;

– il est pratique d'utiliser le modèle pour mettre à l'épreuve des cas aux limites (par exemple le nombre de circuits, d'appels et d'autres types de trafic est limité; la longueur des données élémentaires est limitée).

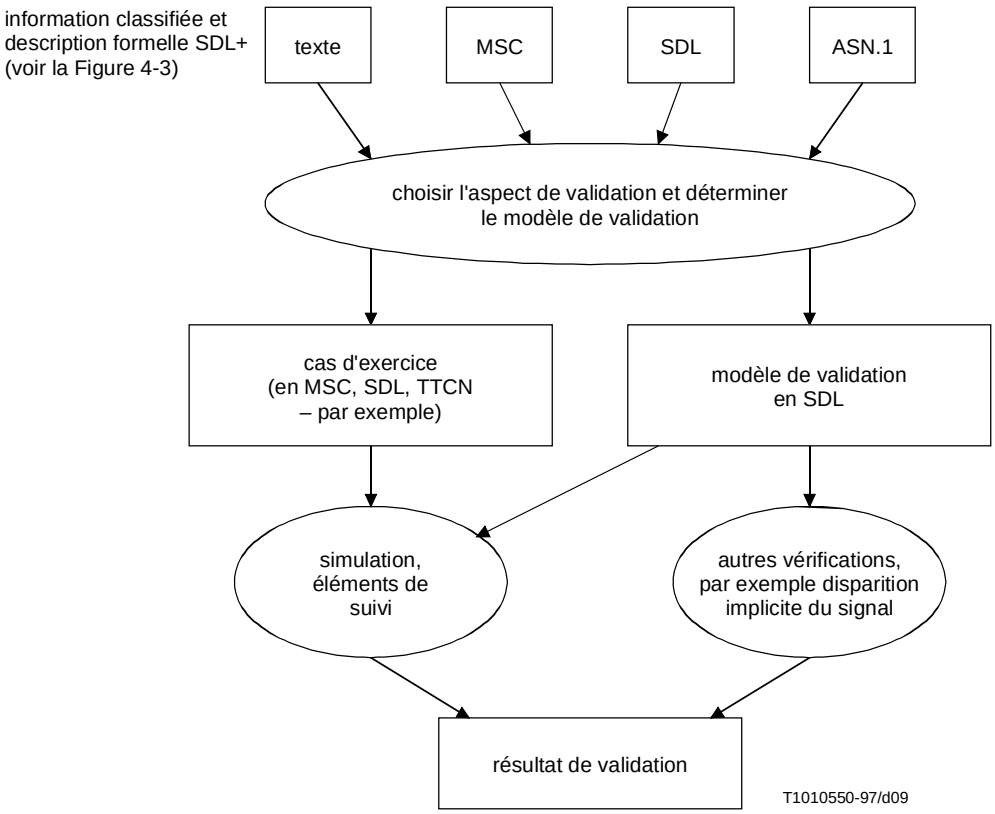


Figure 9-1/Suppl. 1 à la Rec. Z.100 – Schéma général de la Validation

9.2 Comparaison du modèle de validation avec le modèle formalisé

Si la méthodologie est utilisée en vue de produire une description en SDL+ pour une norme, c'est alors la description SDL+ contenue dans la norme (la norme formalisée) qui est recherchée.

Aussi bien le modèle de validation qu'une norme formalisée sont extraits du modèle formel SDL+ (le modèle formalisé) qui a été produit au cours de l'activité de Formalisation. L'hypothèse est que la norme formalisée est généralisée à partir du modèle formalisé. Par exemple, la norme formalisée peut omettre des éléments qui sont requis d'un point de vue technique mais qui ne font pas partie de la norme; ou bien la norme formalisée peut autoriser des options, dont au moins une doit être mise en œuvre pour rendre le modèle de validation exécutable. Il est souvent désirable qu'un modèle de validation comporte des éléments de l'environnement.

La relation entre le modèle de validation et la norme formalisée est que le modèle de validation se compose du SDL+ contenu dans la norme formalisée et de la modification minimale nécessaire pour rendre le modèle exécutable. Ces modifications peuvent être extraites du modèle formalisé, en admettant que celui-ci est exécutable. Dans certains cas, le modèle de validation peut être identique au modèle formalisé.

Si la méthodologie est utilisée pour produire une mise en œuvre, le modèle de validation peut devoir encore différer du modèle formalisé, soit parce que celui-ci a besoin d'un certain appui de mise en œuvre pour être exécutable, soit parce que la mise en œuvre a des limites trop vastes pour la Validation. Dans ce dernier cas, la Validation peut habituellement être effectuée par utilisation de limites plus serrées.

Si le modèle formalisé n'est pas exécutable, ou s'il est exécutable mais non validable, un modèle de validation distinct est extrait.

Instructions

1) définir de nouvelles limites pour le modèle de validation, y compris les éléments de l'environnement qui sont nécessaires à l'exécutabilité;

2) définir des limites pour les circuits, les appels et autres types de trafic;

3) identifier les cas d'exercice ou les observateurs.

Directives

La construction du modèle de validation peut se subdiviser comme suit:

1) identification du domaine d'application de la Validation;
2) choix d'un ensemble particulier d'options de mise en œuvre;
3) complément du système;
4) optimisation du modèle pour rendre le contrôle possible.

On trouvera des directives plus détaillées en [15].

9.3 Problèmes de définition de la validation d'une spécification

Au cours de la Validation, plusieurs modèles de validation peuvent être produits, selon les caractéristiques de l'application et les techniques automatisées qui ont été choisies pour exécuter le modèle. Par exemple, lorsqu'une spécification de protocole contenue dans une application est vraiment complexe, une option consiste à la partitionner et à valider chaque partie. De telles tactiques de simplification des spécifications sont nécessaires s'il faut valider une spécification complexe au moyen d'une recherche d'accessibilité. Celle-ci implique que tous les états possibles soient évalués au cours de l'exécution du modèle. Par ailleurs, un des avantages de la sélection d'une validation dans un espace d'états aléatoires, en présence d'une spécification de protocole complexe, est que le modèle de validation peut représenter le protocole entier.

10 Relation avec d'autres méthodes et modèles

La méthodologie contenue dans le présent Supplément peut être utilisée en combinaison avec d'autres méthodologies. Telles qu'elles sont habituellement appliquées, les méthodologies décrites ci-dessous donnent naissance à un ou à plusieurs modèles semi-formels. La plupart des modèles utilisent les conceptions de projet produites par l'activité de Conception du projet, mais avec des caractéristiques additionnelles issues de l'autre méthodologie. Par exemple, la méthodologie de la Recommandation I.130 [16] produit trois modèles: un modèle donnant le point de vue des utilisateurs, un autre montrant l'organisation des fonctions du réseau et un troisième modèle montrant les capacités de commutation et de signalisation à l'appui du modèle d'utilisateur.

10.1 Relation avec la méthode (d'étape 3) des Recommandations I.130/Q.65

Etant donné que la Recommandation I.130 est couramment utilisée comme base pour les étapes méthodologiques décrites en partie II.

L'utilisation des Recommandations I.130 [16] et Q.65 [17] n'a pas été limitée au domaine de la "méthode de caractérisation des services de télécommunication assurés sur un RNIS et des possibilités réseau d'un RNIS" comme indiqué dans le titre de la Recommandation I.130. Les étapes 1 et 2 servent à produire des normes pour les capacités de services, qui sont ensuite utilisées en étape 3 pour produire des normes de protocole destinées à soutenir ces services. Le langage SDL est utilisé dans les trois étapes de la méthode I.130:

– dans l'étape 1 – phase 1.3 de description dynamique d'un service, où "l'information est présentée sous forme d'un diagramme général du langage de description et de spécification (SDL)" (3.1/I.130);
– dans l'étape 2 – phase 2.3, où "les fonctions accomplies dans une entité fonctionnelle" sont "représentées sous la forme d'une spécification et d'un diagramme de langage de description (SDL)" (3.2/I.130);
– dans l'étape 3, où "les diagrammes SDL en provenance de l'étape 2 constituent la base" (3.3/I.130) et où "les Recommandations sur le SDL (Recommandations de la série Z.100)" sont décrites comme étant les "outils de la méthode, techniques (modèles) de description et les bibliothèques associées de matériel générique".

Il arrive quelquefois que ces trois descriptions en langage SDL apparaissent dans les normes et quelquefois une seule. Parfois, les différentes descriptions apparaissent dans différentes normes d'une même famille. C'est au groupe chargé de la norme qu'il appartient de prendre les décisions sur les questions de savoir quelles sont les descriptions qui devraient apparaître et dans quelles normes ainsi que ce qu'il faut normaliser (le point de vue de l'utilisateur, la mise en œuvre fonctionnelle du service et les protocoles et procédures d'accès et de jonction internodale): ces décisions sont hors du domaine d'application du présent Supplément, qui part du principe que chaque description SDL apparaît dans une norme distincte.

Lorsque la Recommandation I.130 a été rédigée pour la première fois, la Recommandation Z.120 [3] n'était pas encore une norme. Il convient donc de remplacer les "flux d'information" de la Recommandation I.130 par l'utilisation des diagrammes MSC.

L'approche I.130 est compatible avec la présente méthodologie. La Recommandation I.130 définit les descriptions qui devraient être fournies et la présente méthodologie précise la façon de produire ces descriptions. La description d'étape 1 pour la Recommandation I.130 peut être produite sur la base d'un diagramme contextuel développé en une simple description SDL pouvant être considérée comme une conception de projet pour l'entité fonctionnelle SDL d'étape 2. La spécification SDL d'étape 2 peut être produite par application de toutes les activités indiquées dans la présente méthodologie. La spécification SDL d'étape 3 peut être produite par réapplication de la présente méthodologie, les étapes 1 et 2 étant utilisées dans le cadre des prescriptions collectées.

La Recommandation Q.65 donne de plus amples détails sur la méthode d'étape 2. Dans la phase 2.1, un modèle fonctionnel est extrait afin de montrer la relation entre les entités fonctionnelles. Ce modèle devrait être redessiné sous forme de diagramme de système SDL avec représentation des entités fonctionnelles sous forme de types de bloc, des instances d'entité fonctionnelle sous forme de blocs et des relations sous forme de canaux. Etant donné que la spécification SDL contenue dans une norme devrait être complète, la version SDL du diagramme apparaîtra dans la norme. Il peut être utile d'inclure aussi dans celle-ci le modèle de l'entité fonctionnelle, qui ne comporte pas tous les renseignements contenus dans le diagramme SDL et qui donne donc une vue plus abstraite du système. Les diagrammes de flux d'information issus de la Recommandation Q.65 sont des diagrammes MSC ayant une instance d'entité fonctionnelle (bloc SDL) sur chaque axe d'instance.

Lorsqu'une description formelle SDL est produite pour l'étape 2, les actions d'entité fonctionnelle sont habituellement redondantes car les fonctions sont complètement décrites par le langage SDL. Ces descriptions d'entité fonctionnelle peuvent encore être utiles pour aider à expliquer le SDL et parce que certaines normes existantes les utilisent très souvent. Si ces descriptions sont utilisées, elles devraient être clairement marquées comme étant informatives.

10.2 Relation avec la modélisation en couches OSI

Le présent sous-paragraphe met à jour les données sur le SDL-92 qui avaient d'abord été publiées en [18], [19] et dans l'Appendice I/Z.100 [1].

Les concepts OSI utilisés dans le présent sous-paragraphe sont définis dans [20] et [21]. Les concepts clés sont expliqués ici pour rendre le sous-paragraphe plus autonome.

Un *service de couche* est offert par un *fournisseur de service* pour une couche donnée. Le fournisseur de service est une machine abstraite offrant un moyen de communication aux *utilisateurs* de la couche immédiatement supérieure. Le service est atteint par les utilisateurs aux *points d'accès au service* au moyen de *primitives de service*, comme le montre la Figure 10-1. Une primitive de service peut être utilisée pour la gestion de la connexion (connexion, déconnexion, réinitialisation, etc.) ou peut être un objet données (données normales ou données exprès). Il n'existe que quatre types de primitives de service:

– demande (de l'utilisateur au fournisseur);

– indication (du fournisseur à l'utilisateur);

– réponse (de l'utilisateur au fournisseur);

– confirmation (du fournisseur à l'utilisateur).

Figure 10-1/Suppl. 1 à la Rec. Z.100 – Un fournisseur de service en couche

Une *spécification de service* est un moyen de caractériser le comportement du fournisseur de service, à la fois localement (en fixant les séquences correctes de primitives de service transmises à un point d'accès au service), et de bout en bout (en fixant la relation correcte entre les primitives de service transmises à différents points d'accès du service). Une spécification de service ne concerne pas la structure interne du fournisseur de service; n'importe quelle structure interne fournie lors de la spécification du service n'est qu'un modèle abstrait pour décrire le comportement observable à l'extérieur du fournisseur du service.

Excepté pour la couche la plus haute, les utilisateurs d'un service en couche sont des *entités de protocole* pour la couche immédiatement supérieure, qui coopèrent pour augmenter les capacités du service de la couche en fournissant ainsi un service à la couche immédiatement supérieure. Cette coopération est conduite en respectant un ensemble prédéfini de règles de comportement et de formats de messages qui constituent un *protocole*. Conformément à cette optique, les entités de protocole de la couche (N) et le fournisseur de service (N-1) fournissent ensemble un raffinement pour le fournisseur du service (N) (voir la Figure 10-2).

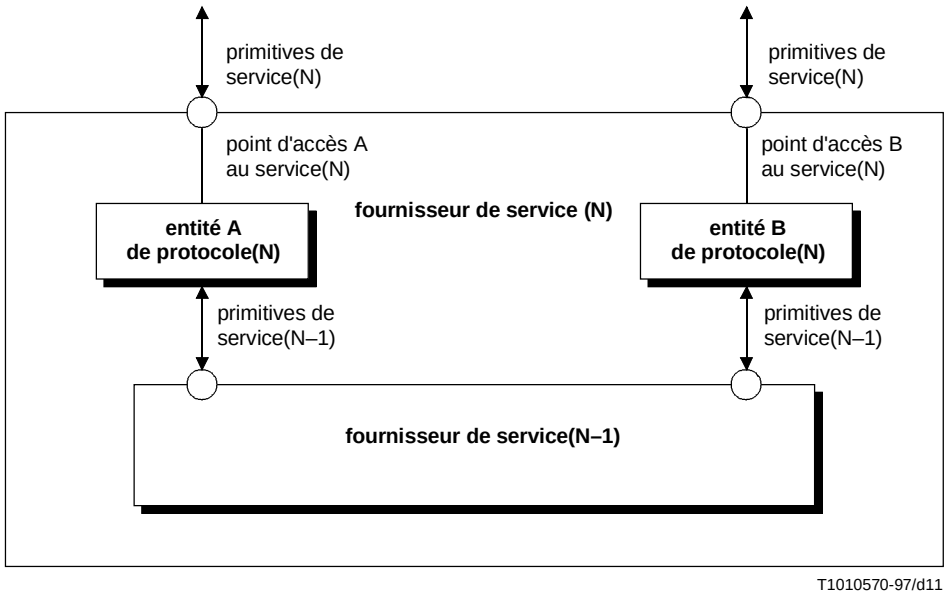


Figure 10-2/Suppl. 1 à la Rec. Z.100 – Raffinage du fournisseur de service de la couche (N)

Le raffinement du fournisseur de service (N) présenté à la Figure 10-2 peut naturellement être beaucoup plus compliqué. Par exemple, il peut y avoir des entités de protocole de relais (N) qui ne sont connectées à aucune entité de protocole de la couche (N+1). Par souci de concision, de tels cas ne sont pas considérés ici.

Les entités de protocole communiquent par échange d'*unités de données de protocole*. Celles-ci sont transférées en tant que paramètres des primitives de service de la couche sous-jacente. L'entité de protocole émettrice code les unités de données de protocole dans les primitives de service; l'entité de protocole réceptrice décode les unités de données de protocole à partir des primitives de service reçues. Un protocole se fonde sur les propriétés du fournisseur de service sous-jacent. Le fournisseur de service sous-jacent peut par exemple perdre, altérer ou modifier l'ordre de messages. En pareil cas, le protocole devrait contenir des mécanismes de détection et de correction d'erreur, de resynchronisation, de retransmission, etc. de façon à fournir un service fiable et généralement plus puissant à la couche immédiatement supérieure.

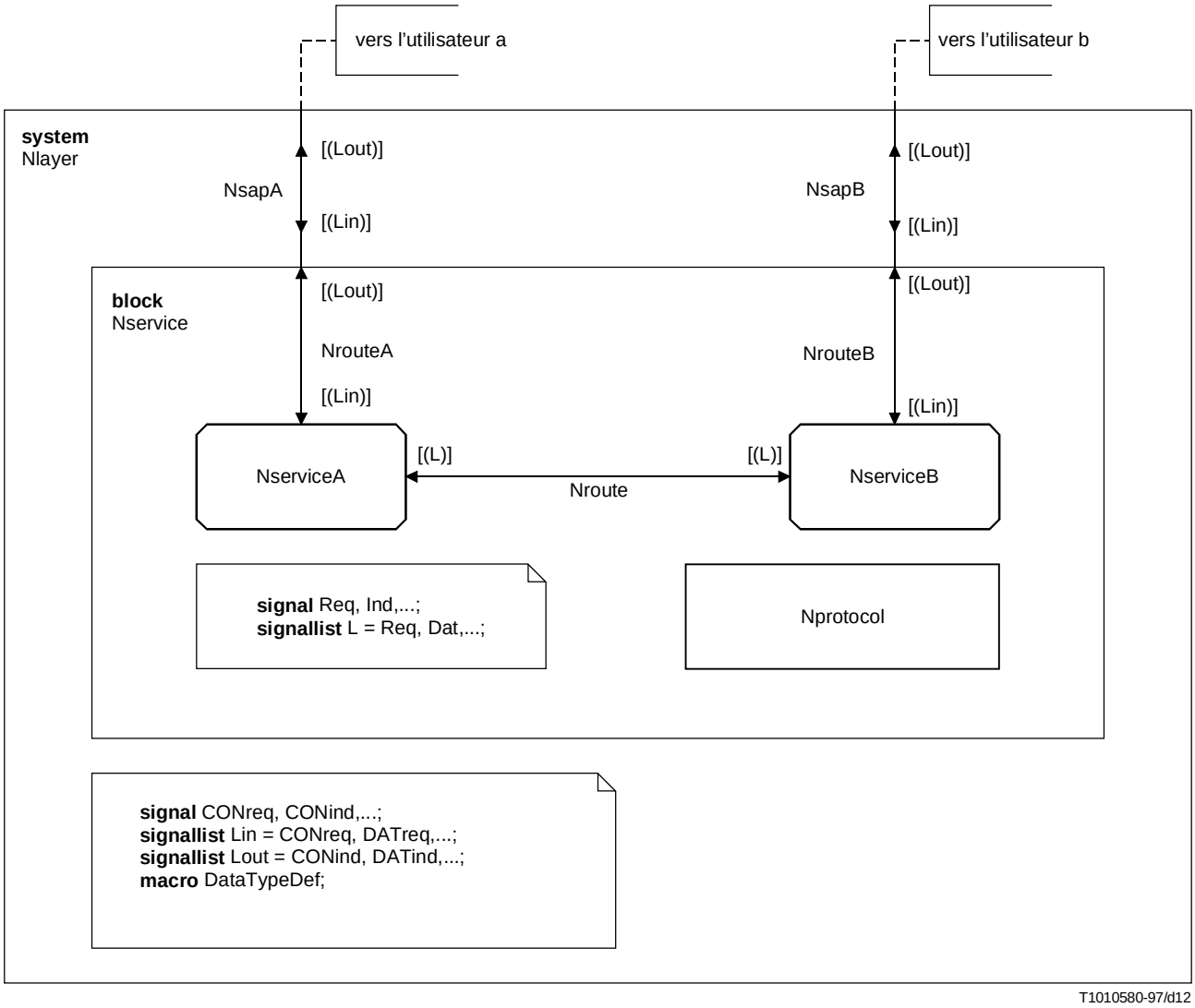
Les concepts architecturaux OSI peuvent être modélisés en SDL de différentes façons, dépendant principalement des aspects que l'on souhaite mettre en valeur. D'abord, une approche simple est décrite, ensuite d'autres approches sont esquissées comme variantes de cette approche de base.

Dans les exemples, la syntaxe graphique du SDL est utilisée autant que faire se peut. Noter cependant que, pour une raison pratique, certaines informations imposées par les règles syntaxiques peuvent être omises ou représentées par une suite de points (...) qui ne font évidemment pas partie de la syntaxe.

10.2.1 Approche fondamentale

Spécification de service

Une spécification de service pour la couche N peut être modélisée de manière directe comme un bloc *Nservice* contenant deux processus *NserviceA* et *NserviceB* (comme le montre l'Exemple 10-1).



Exemple 10-1/Suppl. 1 à la Rec. Z.100 – Spécification d'un service de couche (N) en SDL

Dans l'Exemple 10-1, les utilisateurs de ce service sont dans l'environnement du système et peuvent être considérés comme des processus capables de communiquer avec le système selon les termes de ce système.

Un point d'accès au service est représenté par un canal (*NsapA*, ou *NsapB*) transportant des signaux qui représentent les primitives du service. Un signal peut transporter des valeurs des sortes données dans la spécification du signal. Les spécifications de sorte (autres que celles des sortes prédéfinies) sont contenues dans la spécification distante de *macro DataTypeDef*, qui est omise ici car non pertinente pour l'objet du présent exposé.

Naturellement, dans le cas le plus général, il peut y avoir plus de deux processus concernés par la spécification du service. Par souci de concision, nous n'allons considérer ici que deux processus, un pour chaque point d'accès au service. L'exposé suivant s'applique aussi, cependant, au cas où il y a plusieurs processus pour un point d'accès au service.

Dans l'Exemple 10-1, on montre quelques exemples de spécifications de signaux. Comme quelques noms de signaux le suggèrent, on suppose qu'il s'agit d'un service orienté connexion. Dans le cas d'un service en mode sans connexion, un grand nombre de simplifications peuvent être faites. Cependant, ce cas ne sera pas traité plus amplement, pour des raisons de concision.

On considère ici, à la fois les aspects locaux et «de bout en bout» d'une spécification de service. Le comportement local s'exprime de façon indépendante par les processus *NserviceA* et *NserviceB*. Ces processus communiquent entre eux par des signaux (*Req, Ind*,...) qui sont internes au bloc et transportés par l'acheminement de signaux *Nroute*. Le comportement de bout en bout s'exprime par le mappage (effectué par chaque processus) entre les primitives de service et les signaux internes sur *Nroute*. Les processus *NserviceA* et *NserviceB* sont des images miroirs l'un de l'autre. Il y en a deux, au lieu d'un seul, pour permettre de modéliser fidèlement une situation de collision possible dans le fournisseur de service.

Le comportement non déterministe est une caractéristique inhérente au fournisseur de service parce que celui-ci peut refuser les tentatives de connexion et interrompre les connexions établies de sa propre initiative.

Il faut noter que la spécification du bloc *Nservice* contient seulement une référence aux processus *NserviceA* et *NserviceB*; ces processus sont spécifiés par des *spécifications distantes*, placées en dehors de la spécification du bloc; ils ne sont pas présentés ici parce qu'ils pourraient attirer l'attention du lecteur sur des caractéristiques nécessairement spécifiques d'un service donné.

Spécification de protocole

La spécification de protocole pour la couche N est modélisée par la sous-structure *Nprotocol* du bloc *Nservice*, voir l'Exemple 10-1.

Dans le *diagramme de blocs* de l'Exemple 10-1, une référence de sous-structure de bloc (un symbole de bloc contenant le nom *Nprotocol* de la sous-structure de bloc) a été introduite. La spécification de la sous-structure de bloc est donnée dans un *diagramme de sous-structure de bloc* distant (voir l'Exemple 10-2) contenant trois blocs: *NentityA*, *NentityB* et *N_1service*. Les deux premiers blocs représentent les entités de protocole (N), tandis que le bloc *N_1service* représente le fournisseur de service (N-1). La spécification de *N_1service* est analogue à la spécification de *Nservice* et ne figure pas dans le diagramme (car il s'agit d'une spécification distante).

Un bloc d'entité de protocole contient un ou plusieurs processus, selon les caractéristiques du protocole. Dans le cas présent, deux processus ont été choisis: *Ncom* et *Ncodex*. Le processus *Ncom* gère l'envoi et la réception d'unités de données de protocole, tandis que le processus *Ncodex* s'assure de la transmission des unités de données de protocole en utilisant le service sous-jacent. Théoriquement, les processus *Ncom* communiquent directement via un canal implicite *NChan* (transportant les unités de données de protocole); mais en réalité ils communiquent indirectement via le processus *Ncodex* et le fournisseur du service sous-jacent.

10.2.2 Approche en variante utilisant la sous-structure de canal

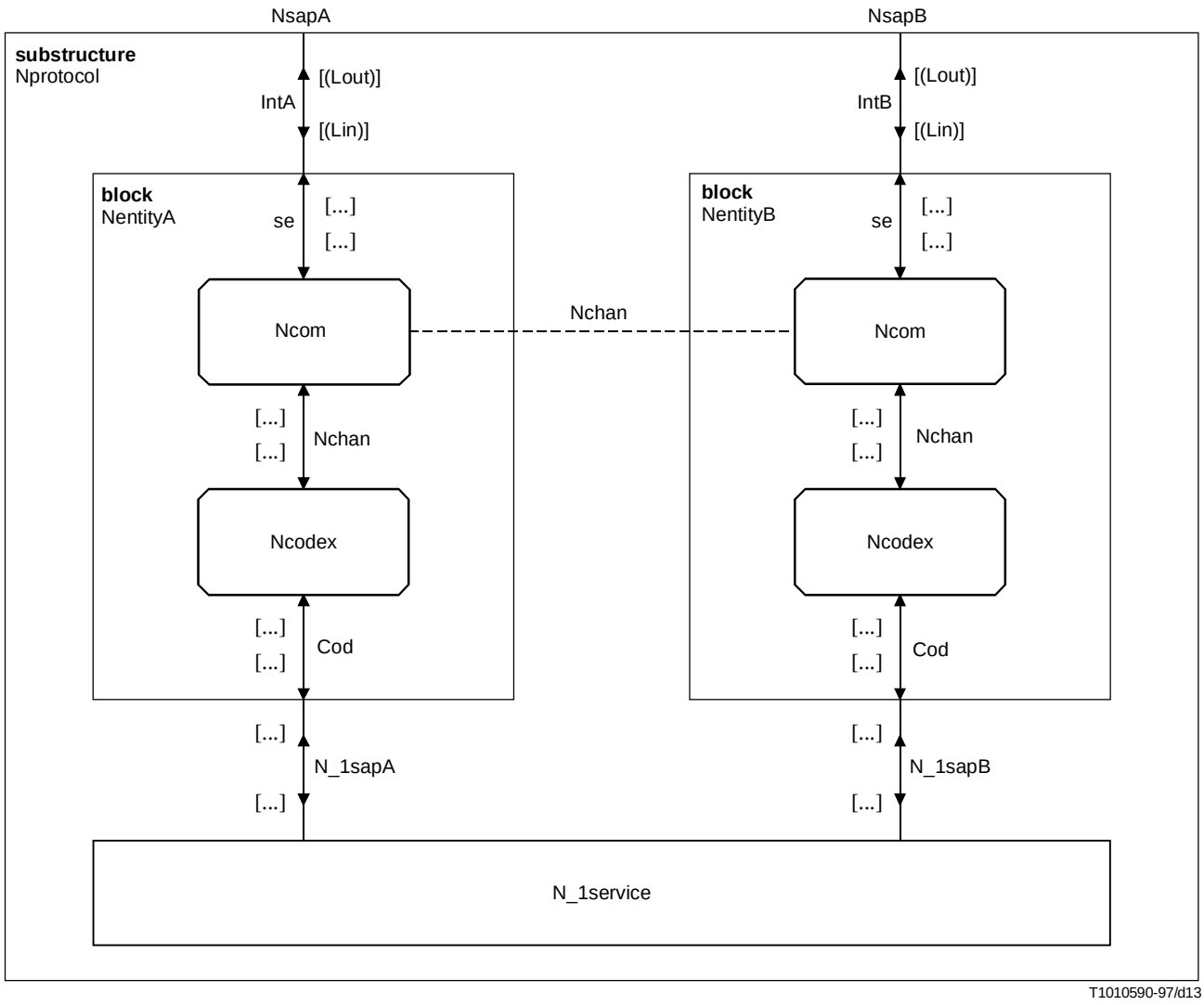
Cette approche s'obtient à partir de l'approche de base de l'Exemple 10-2 en groupant les processus différemment, en introduisant le canal réel *Nchan* et en utilisant une sous-structure de canal (voir l'Exemple 10-3). Le canal *Nchan* transporte les unités de données de protocole figurant dans la liste des signaux *Npdu*. Cette approche fait ressortir la vue du protocole et l'orientation en couches horizontales de l'OSI.

Noter que dans cette approche les blocs à l'intérieur de la sous-structure de canal ne représentent pas les entités de protocole et se superposent aux deux couches adjacentes. Les primitives de service sont cachées dans ces blocs et sont transportées dans les acheminements de signaux *N_1sapA*, *N_1sapB*, *N_2sapA*, *N_2sapB*, etc. Cependant, la plus haute couche choisie (N) doit être traitée séparément, comme indiqué dans l'exemple. Noter aussi que le diagramme de système (voir l'Exemple 10-1) n'est pas affecté par cette approche.

10.2.3 Architecture OSI symétrique

Lorsque l'architecture OSI est symétrique, ce qui est le cas lorsque les entités des deux côtés de l'architecture OSI sont les images miroirs l'une de l'autre, les spécifications de ces entités sont identiques, sauf le nom de l'entité. La spécification commune peut être donnée par instanciation d'un type, qui est le type de processus *Nservice* contenu dans le bloc *Nservice*. Ce type de processus, *Nservice*, est ensuite instancié en *NserviceA* et en *NserviceB* (voir l'Exemple 10-4). *x* et *z* sont les paramètres actuels correspondant aux portes (non représentées ici) du type de processus *Nservice*. Dans cet exemple, seul figure le diagramme de blocs (schéma fonctionnel) *Nservice* de l'Exemple 10-1. Noter que les spécifications de service sont toujours symétriques et que seules les spécifications de protocole peuvent être asymétriques.

Une autre approche consiste à ne représenter qu'un côté (voir l'Exemple 10-5), qui est une modification du diagramme du système de l'Exemple 10-1. Le canal *NsapB* a été remplacé par le canal *Nchannel*, transportant les signaux internes *L*. Ces signaux sont maintenant au niveau du système et leurs spécifications ont été déplacées en conséquence. Noter que cette approche ne peut pas être utilisée conjointement avec la sous-structure de canal (montrée à l'Exemple 10-3).



Exemple 10-2/Suppl. 1 à la Rec. Z.100 – Spécification d'un protocole de couche (N) en SDL

10.3 Relation avec l'architecture de la série de Recommandations Q.1200 (RI) et avec les modules SIB

L'architecture du "réseau intelligent" (RI) décrite dans la série de Recommandations Q.1200 [13] ne présente en elle-même aucune difficulté fondamentale de mappage au SDL. Mais il existe un certain nombre de manières différentes d'utiliser le langage SDL pour soutenir les modules indépendants des services (SIB, *service independent building blocks*). Toutefois, l'association du traitement d'appel de base à la logique de service est un problème de modélisation qu'il faut examiner lors de l'utilisation de l'architecture du réseau intelligent. La principale différence entre cette méthode et celle de la Recommandation I.130 est l'introduction d'un plan fonctionnel global au-dessus du plan fonctionnel (réparti) de l'utilisation des modules SIB.

Le plan fonctionnel réparti peut être modélisé de la même façon que l'étape 2 dans la Recommandation I.130.

Pour soutenir le réseau intelligent au moyen du langage SDL, il faut disposer d'un modèle SDL pour le plan fonctionnel global ainsi que:

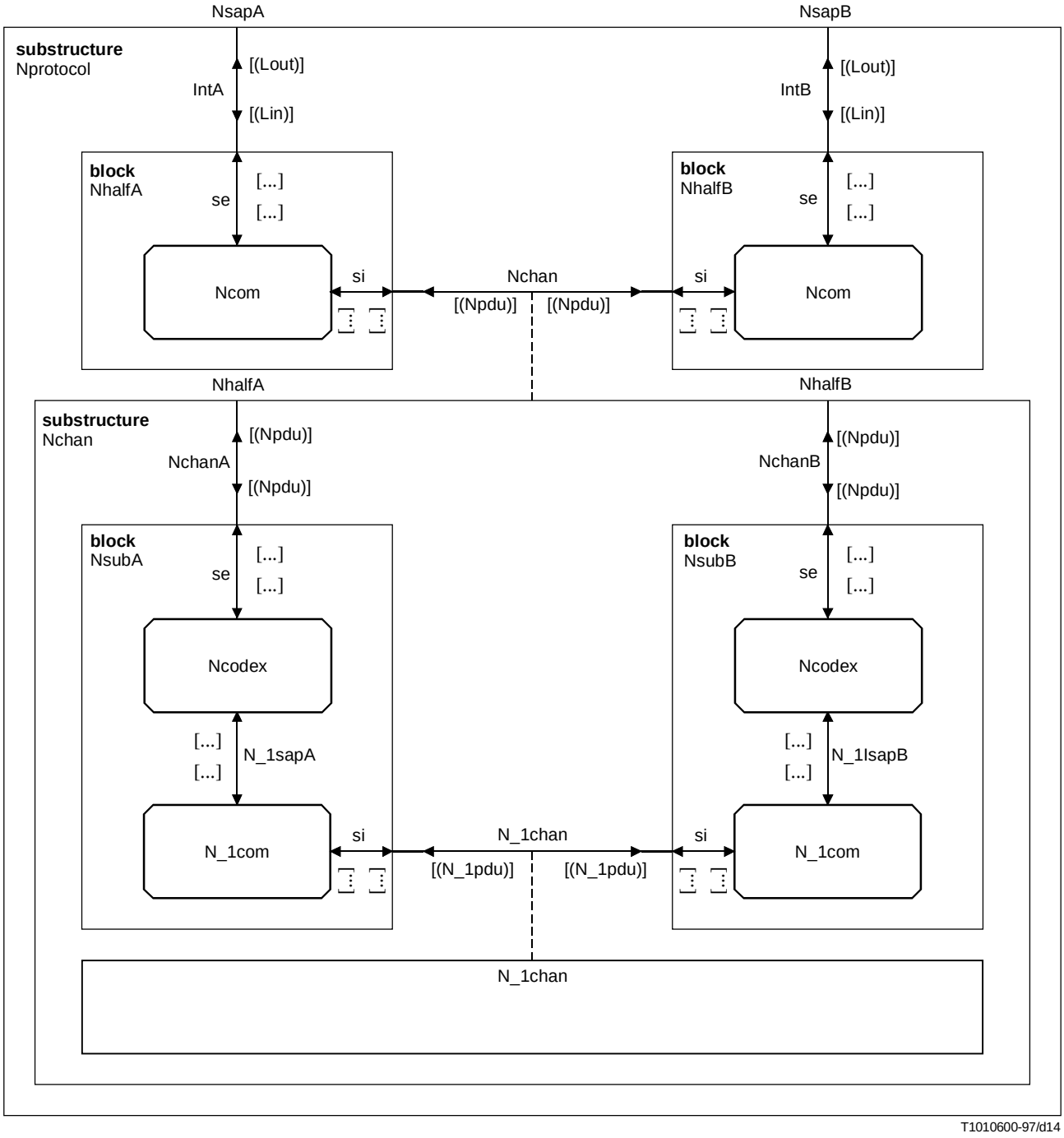
– d'un traitement d'appel de base (BCP, *basic call processing*) global et bien défini, de préférence en SDL formel mais au moins avec des interfaces clairement définies pour l'appel du premier module SIB dans une chaîne;

– d'un moyen permettant d'ajouter l'invocation d'un nouveau service au traitement BCP global;

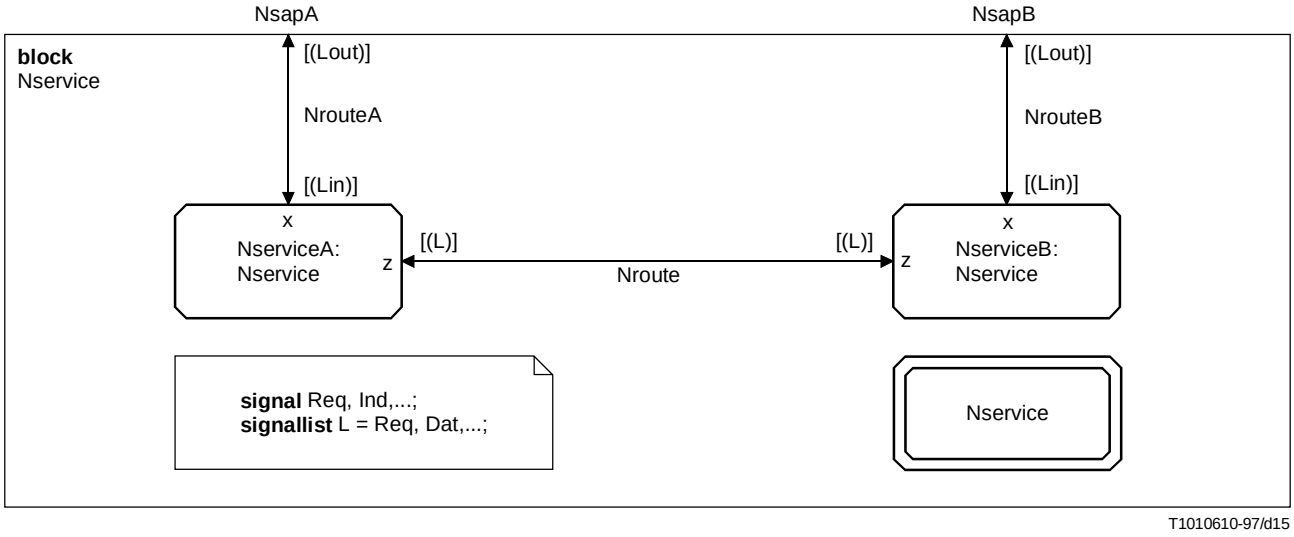
– d'un moyen permettant de spécifier les modules SIB en SDL et de les concaténer;

– d'un moyen d'utiliser le langage SDL de manière que le traitement BCP global et la logique de service puissent être répartis dans différentes normes.

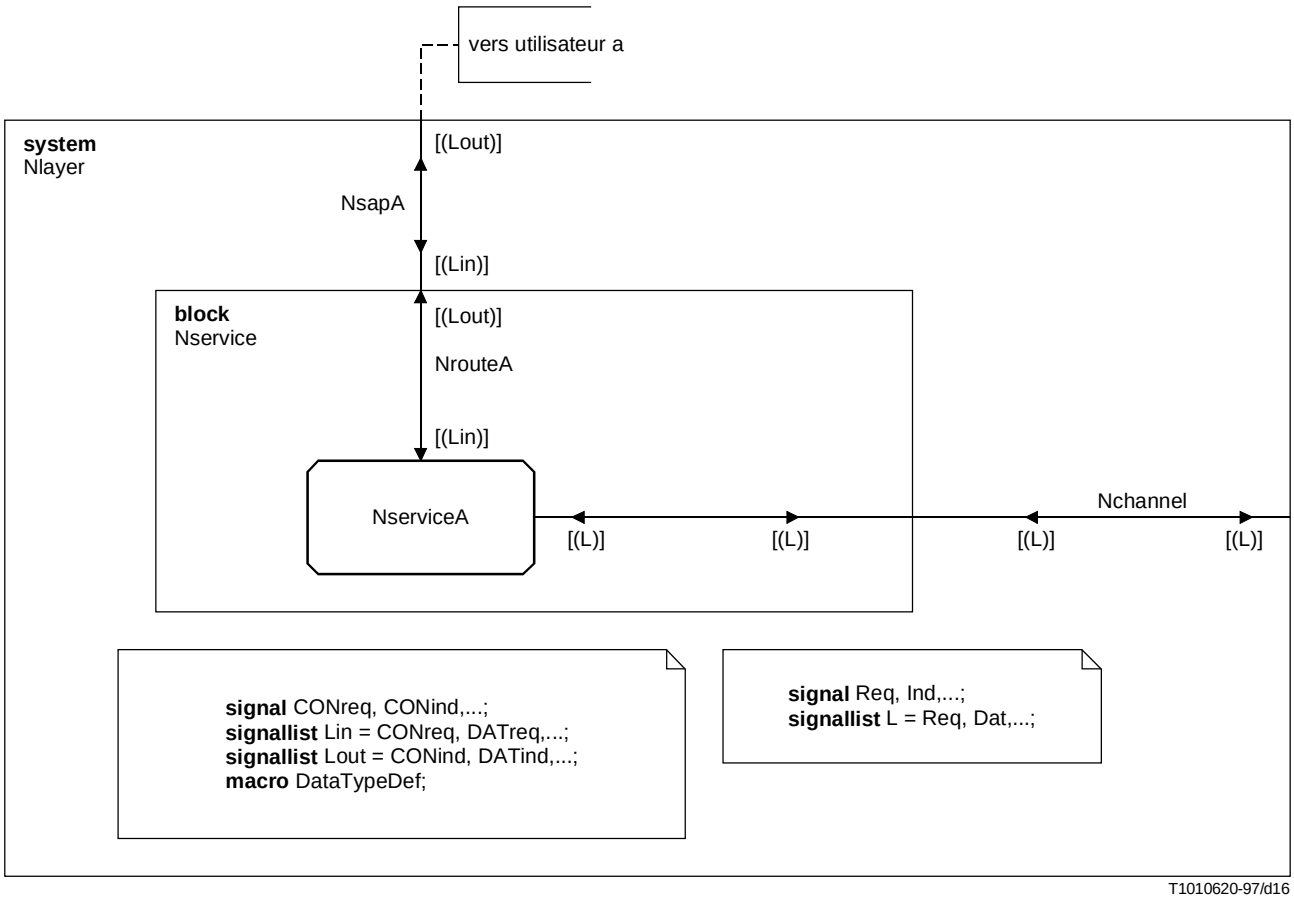
La version SDL-92 offre des mécanismes qui peuvent être utilisés pour soutenir l'approche du réseau intelligent.



Exemple 10-3/Suppl. 1 à la Rec. Z.100 – La vue de protocole de l'OSI



Exemple 10-4/Suppl. 1 à la Rec. Z.100 – Utilisation d'un type de processus pour représenter une architecture OSI symétrique



Exemple 10-5/Suppl. 1 à la Rec. Z.100 – Représentation d'un seul côté d'une architecture OSI symétrique

Pour appeler des modules SIB à partir du traitement BCP, celui-ci devrait être lui-même défini en tant que type de processus. Les points auxquels des modules SIB sont susceptibles d'être appelés devraient être contenus soit dans des procédures virtuelles soit dans des transitions virtuelles du BCP. L'appel du premier module SIB pourra ensuite être ajouté au traitement BCP par modification de la partie virtuelle du processus. La définition précise du processus dépend donc des interactions avec les services offerts par le BCP.

Les modules SIB peuvent eux-mêmes être modélisés de plusieurs façons. Ils peuvent être des procédures du traitement BCP mais dans ce cas ils ne peuvent pas fonctionner en parallèle avec le BCP ou les uns avec les autres. En variante, les modules SIB peuvent être des appels de procédure distante adressés à un ou plusieurs processus de logique de service, ou à un processus SDL. L'appel d'un module SIB peut être assuré par des appels de procédure distante ou par création d'un processus de module SIB ou par transmission de signaux. Les points de retour au traitement BCP nécessitent une attention spéciale parce qu'il peut y en avoir plusieurs. Cela donne à penser qu'une interface de procédure n'est pas obligatoirement appropriée, à moins que le retour de procédure soit toujours suivi d'une décision fondée sur une valeur retournée par cette procédure.

Les données globales et les données d'appel doivent être bien définies. Il faut un mécanisme pour traiter la base de données contenant ces données.

Une fois ces caractéristiques architecturales mises au point, on peut produire une description formelle en SDL pour le traitement BCP et pour les modules SIB, au moyen de l'architecture générale dans le cadre des prescriptions collectées et d'une contrainte sur la conception. La méthodologie de la présente norme pourra alors être suivie.

10.4 Relation avec les opérations distantes (opérations RO, *remote operations* et éléments ROSE) de la Recommandation X.219

Les opérations distantes selon la Recommandation X.219 [22] offrent un moyen d'associer une requête aux réponses possibles. Si une opération est synchrone (classe 1), elle peut être mappée sur une procédure distante en SDL. Si l'opération n'a jamais de réponse (classe 5), elle peut être mappée sur un signal SDL. Si l'opération est asynchrone et renvoie une réponse d'un type quelconque (succès ou échec), elle s'applique sur les signaux dans les deux sens d'un canal, le paramètre d'opération et les signaux de réponse étant associés informellement par des commentaires (ainsi que, implicitement, par le comportement du processus SDL). De même une association n'est mappée à aucun élément SDL explicite.

Etant donné que les éléments ROSE utilisent la notation ASN.1 pour définir les données, cette notation peut être utilisée directement pour exprimer les paramètres de signalisation.

Exemple

```
HoldMPTY ::= OPERATION
RESULT
ERRORS{
        IllegalSS_operation,SS_errorstatus, SS_incompatibility,
        FacilityNotSupported, SystemFailure }
```

peut devenir en SDL:

```
signal  HoldMPTY,
        IllegalSS_operation, SS_errorstatus, SS_incompatibility,
        FacilityNotSupported, SystemFailure;
```

ces signaux étant utilisés dans le canal à destination et en provenance du processus contenant l'opération.

10.5 Relation avec la Recommandation X.722 (objets GDMO)

NOTE – Bien que cela puisse être important pour la conception et éventuellement pour la normalisation des systèmes de télécommunication dans lesquels les ressources sont traitées comme des objets gérés, un complément d'étude est nécessaire pour convenir d'une méthode de combinaison des objets GDMO avec les normes relatives aux ressources.

Les objets GDMO [23] utilisent la notation ASN.1 comme base. Le modèle GDMO fournit le mot clé BEHAVIOUR (comportement), d'après lequel un objet peut être défini. Mais la définition se présente sous la forme d'une chaîne qui contient habituellement du langage naturel. Le comportement décrit n'est que celui de l'objet géré visé par la gestion. Normalement, l'objet possède un autre comportement, qui est normalisé par une norme de service ou de protocole. Les avantages procurés par la description en langage SDL du comportement de gestion d'un objet sont les suivants:

– le comportement de gestion de l'objet sera mieux défini et, pour des objets gérés complexes, ce comportement pourra être très complexe;

– cette description permet de mieux contrôler la cohérence de la description de gestion et de la description fonctionnelle;

– la description de gestion et la description fonctionnelle peuvent avoir quelques parties communes et une base permettant de les fusionner dans une mise en œuvre.

La base ASN.1 des objets GDMO offre une base pour définir l'objet géré en langage SDL.

11 Justification de l'approche

La méthodologie présentée dans le présent Supplément représente l'avis des experts contribuant à l'UIT-T sur l'utilisation du SDL+. Elle est fondée sur plusieurs années de recherches et d'expériences. Certaines des matières sur lesquelles ce Supplément est fondé peuvent être retrouvées en [24] et [25].

La méthodologie est spécifiquement orientée vers la production d'une spécification précise d'un système. Cette limitation est choisie pour quatre raisons:

1) un accord général existe sur le principe de formaliser les exigences en SDL;

2) il existe de nombreuses façons différentes de mettre en œuvre un produit à partir d'une spécification SDL;

3) le délai de commercialisation est souvent plus important que l'efficacité d'exécution et les outils SDL sont en mesure de produire un code logique à partir de descriptions de mises en œuvre qui sont proches des spécifications de produit;

4) la méthodologie peut être utilisée pour produire aussi bien des spécifications de produit que des spécifications à utiliser dans des normes (ou pour les commandes d'acquisition).

L'on prévoit que l'augmentation du nombre de normes rédigées de manière formelle en langage SDL permettra d'utiliser les descriptions SDL dans des normes qui seront utilisées comme base pour les acquisitions, pour la mise en œuvre des produits et pour leurs essais. Une attention spéciale est donc prêtée à la combinaison de la méthodologie du présent Supplément avec d'autres techniques utilisées pour rédiger des normes.

La méthodologie est définie en termes de SDL/GR. La plupart des utilisateurs préfèrent se servir de la forme SDL/GR et la trouvent plus facile à comprendre. Cette méthodologie peut être adaptée pour être utilisée avec les règles SDL/PR.

La méthodologie est présentée sous la forme d'un certain nombre d'activités et d'étapes à suivre, parce qu'un certain nombre d'utilisateurs du SDL ont demandé cette approche et parce que les étapes présentées dans l'Appendice I/Z.100 [1] ont reçu un accueil favorable de la part des utilisateurs. Ces étapes ont été élaborées et développées dans la partie II du présent Supplément. Les activités définies en partie I sont élaborées pour un cas particulier en partie II. La partie I définit donc un cadre général et la partie II définit un ensemble d'étapes utilisable.

La méthodologie n'utilise pas chaque élément du langage SDL et suggère, dans certains cas, que certains éléments ne soient pas utilisés. On ne devrait pas en déduire qu'il n'existe aucun cas approprié à l'usage de ces éléments. La méthodologie est générale et l'on s'attend à des divergences par rapport à des applications ou organisations particulières.

PARTIE II – ÉLABORATION DE LA MÉTHODOLOGIE GÉNÉRALE

12 Elaboration de la méthodologie pour la spécification d'un service

Les activités sont décrites au moyen d'un certain nombre d'*étapes* qui sont parfois subdivisées en *instructions* et en *directives*. Un certain nombre de *règles* s'appliquent au cours des étapes. Les règles sont exprimées soit par l'auxiliaire "doit/doivent" (shall) ou par l'auxiliaire "devrait/devraient" (should). L'auxiliaire "doit/doivent" est utilisé pour une règle qu'il faut observer afin de garantir qu'un système valide et bien construit est compréhensible, réalisable et testable. L'auxiliaire "devrait/devraient" est utilisé pour les règles qui peuvent être transgressées dans certaines circonstances. La différence entre une *directive* et une *règle* est que la directive donne un avis qui peut être ignoré sans conséquences sérieuses ou qui indique des options.

Les termes *étape, instruction, directive* et *règle* sont définis dans le paragraphe 2.

12.1 Méthodologie en trois étapes: étape 2 (Recommandation Q.65)

La relation entre la méthodologie en trois étapes des Recommandations I.130 [16] et Q.65 [17] et la méthodologie du présent Supplément est décrite au sous-paragraphe 10.1. On expliquera ci-après le mappage des étapes Q.65 aux constructions SDL.

Théoriquement, il devrait être possible d'utiliser les concepts structurels du SDL à l'étape 1 et les concepts comportementaux dans les étapes 2 à 4 avec introduction progressive des données. L'étape 5 reste en dehors du domaine d'application du SDL. L'argument général en faveur de l'utilisation du langage SDL dans les étapes 1 à 4 de la Recommandation Q.65 est principalement que l'adhésion à des normes accroît la lisibilité et rend possible l'utilisation d'outils.

On montrera ci-dessous comment les concepts issus de la plupart des étapes de la Recommandation Q.65 peuvent être appliqués au SDL.

Structure

Dans la Recommandation Q.65, l'étape 1 concerne le modèle fonctionnel, l'identification des entités fonctionnelles et leurs relations correspondant à l'usage des concepts structurels du SDL. La Figure 12-1 montre un exemple de modèle fonctionnel conforme à la Recommandation Q.65.

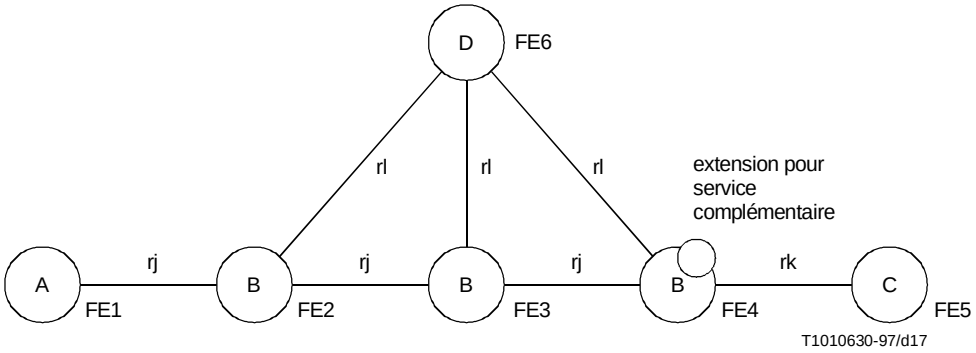


Figure 12-1/Suppl. 1 à la Rec. Z.100 – Exemple d'un modèle fonctionnel conforme à la Recommandation Q.65

Les éléments de la figure sont:

– les noms à l'intérieur des cercles (par exemple *A*): types d'entités fonctionnelles;

– les noms en majuscules adjacents aux cercles (par exemple *FE1*): noms des entités fonctionnelles (instances);

– les noms en minuscules entre les cercles (par exemple *rj*): relations entre les types d'entités fonctionnelles;

– le petit cercle ajouté: extensions pour service complémentaire.

Noter que ce modèle établit une nette distinction entre les types des entités fonctionnelles (par exemple *A*) et leurs instanciations (par exemple *FE1*) lorsque est décrite la structure du système. La Recommandation Q.65 mentionne également qu'il est commode de décrire les deux types d'entités fonctionnelles comme des sous-ensembles de la même entité fonctionnelle, si les deux types d'entités fonctionnelles ont beaucoup de points communs. Ces caractéristiques peuvent être exprimées au moyen des éléments **type** du SDL.

Des recherches ont montré que les "entités fonctionnelles" décrites dans la Recommandation Q.65 pouvaient être représentées par les concepts structurels du SDL de la façon suivante:

– modèle → système;

– entité fonctionnelle → bloc avec un processus;

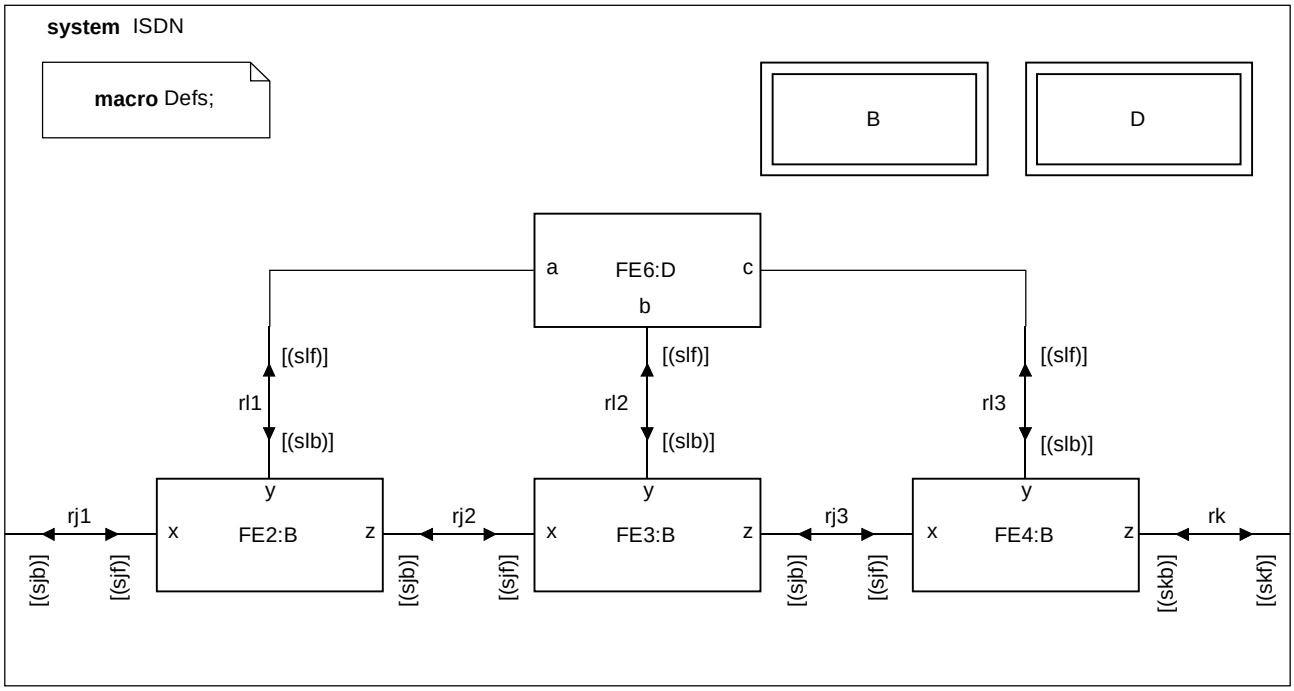
– relations entre entités fonctionnelles → canal.

De plus, en utilisant les constructions du genre **type**:

– type d'entité fonctionnelle → type de bloc.

Une différence entre le modèle fonctionnel conforme à la Recommandation Q.65 et les diagrammes de structure du SDL est que les diagrammes de structure du SDL modélisent normalement des systèmes ouverts.

L'Exemple 12-1 présente le modèle fonctionnel de la Figure 12-1 exprimé en SDL. Remarquer que les entités *FE1* et *FE5* sont absentes de l'intérieur du système. La communication avec ces entités fonctionnelles est modélisée comme une communication avec l'environnement. En laissant les entités *FE1* et *FE5* dans l'environnement, on anticipe le fait que l'on n'a pas l'intention de décrire le comportement de ces entités.



Exemple 12-1/Suppl. 1 à la Rec. Z.100 – Version SDL de l'exemple de la Figure 12-1

La distinction entre relation et type de relation peut être modélisée par l'usage de définitions de listes de signaux dans les diagrammes de structure du SDL. *B* et *D* sont des types de bloc.

Les avantages d'utiliser des diagrammes de structure du SDL résident principalement dans le fait que les diagrammes de structure du SDL contiennent les définitions de signaux, de types de données, etc. qui sont importantes pour les étapes suivantes de la Recommandation Q.65. Dans l'Exemple 12-1, ces définitions sont indiquées par la macro *Defs*.

Comportement

Le comportement est décrit dans les étapes 2 à 4 de la Recommandation Q.65. L'interaction entre entités fonctionnelles est décrite au moyen de diagrammes de flux d'information. A partir de ces derniers, un diagramme SDL (par exemple, un diagramme de processus) est produit pour chaque type d'entité fonctionnelle.

Les diagrammes de flux d'information correspondent aux diagrammes de séquences de messages (MSC) [3]. Les diagrammes de flux d'information sont produits pour les cas "normaux" et le SDL est ensuite utilisé pour décrire le comportement des entités fonctionnelles. Conformément à la méthodologie exposée dans le présent Supplément, cette description intervient dans les activités de Conception du projet et de Formalisation.

L'étape 4 de la Recommandation Q.65 concerne la définition d'un certain nombre d'actions de base pour chaque entité fonctionnelle. L'idée est alors de réutiliser ces actions de base dans différentes spécifications de service.

Les éléments de ce diagramme sont:

– les noms au sommet des colonnes (par exemple *FE1*): entités fonctionnelles;

– les noms en minuscules entre les cercles (par exemple *rj*): relations entre types d'entités fonctionnelles;

– les noms sur les flèches entre les colonnes (par exemple *ESTABLISH X req.ind*): flux d'information;

– les noms placés entre parenthèses et adjacents aux noms de flux d'information (par exemple *location*): information additionnelle transmise par le flux d'information;

– les noms dans les colonnes (par exemple *Action B, 100*): noms des actions de base des entités fonctionnelles;

– les nombres placés entre parenthèses [par exemple *(5)*]: étiquettes utilisées dans les diagrammes SDL.

L'Exemple 12-2 montre le diagramme MSC qui correspond au diagramme de flux d'information de la Figure 12-2. Le mappage entre les deux est évidemment:

– nom d'entité fonctionnelle → instance de processus (une instance de processus par bloc);

– flux d'information → message;

– informations additionnelles → paramètres de message;

– action de base de l'entité fonctionnelle → action;

– étiquettes des diagrammes SDL → commentaires.

Les diagrammes de flux d'information de la Recommandation Q.65 recommandés contiennent donc, fondamentalement, les mêmes informations que les diagrammes MSC. Les spécifications SDL peuvent être généralement vérifiées par rapport aux diagrammes MSC pendant une simulation et dans quelques cas simples (pas de création dynamique de processus, pas d'adressage dynamique des sorties). Ce contrôle peut être réalisé facilement, sans simulation. Les diagrammes MSC peuvent être dérivés des spécifications SDL, mais normalement un seul sous-ensemble du comportement ("le comportement normal") est exprimé dans un MSC, de sorte qu'il faut une intervention humaine pour déduire les MSC adéquats à partir d'une spécification SDL. Une autre possibilité est de déduire automatiquement des squelettes de diagrammes de processus SDL à partir des diagrammes de flux d'information. Voir 15.2.2, l'**étape B:2 – Processus squelettes**.

Données

Les **tâches** (***task***) et les **décisions** (***decision***) sont principalement informelles dans les diagrammes SDL existants pour le RNIS. Par conséquent, il est peu utile de définir des opérateurs pour les types de données. Les données élémentaires apparaissent principalement dans les constructions **entrée** (***input***), **sortie** (***output***) et **temporisateur** (***timer***) (voir l'Exemple 12-3).

Dans la plupart des cas, les types "booléens" *Boolean* et "entier" *Integer* sont suffisants, par exemple pour représenter la valeur de quelque fanion ou compteur. Le type "chaînes de caractères" *Charstring* est aussi largement utilisé parce qu'il permet de manipuler les paramètres de façon similaire à celle du texte informel.

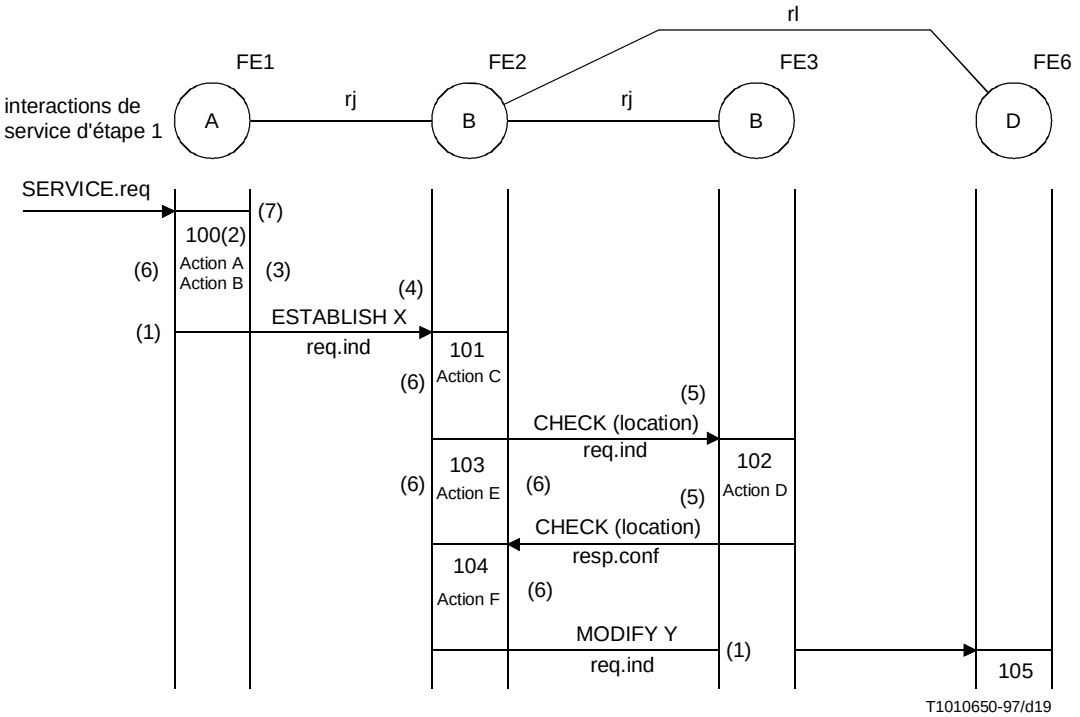
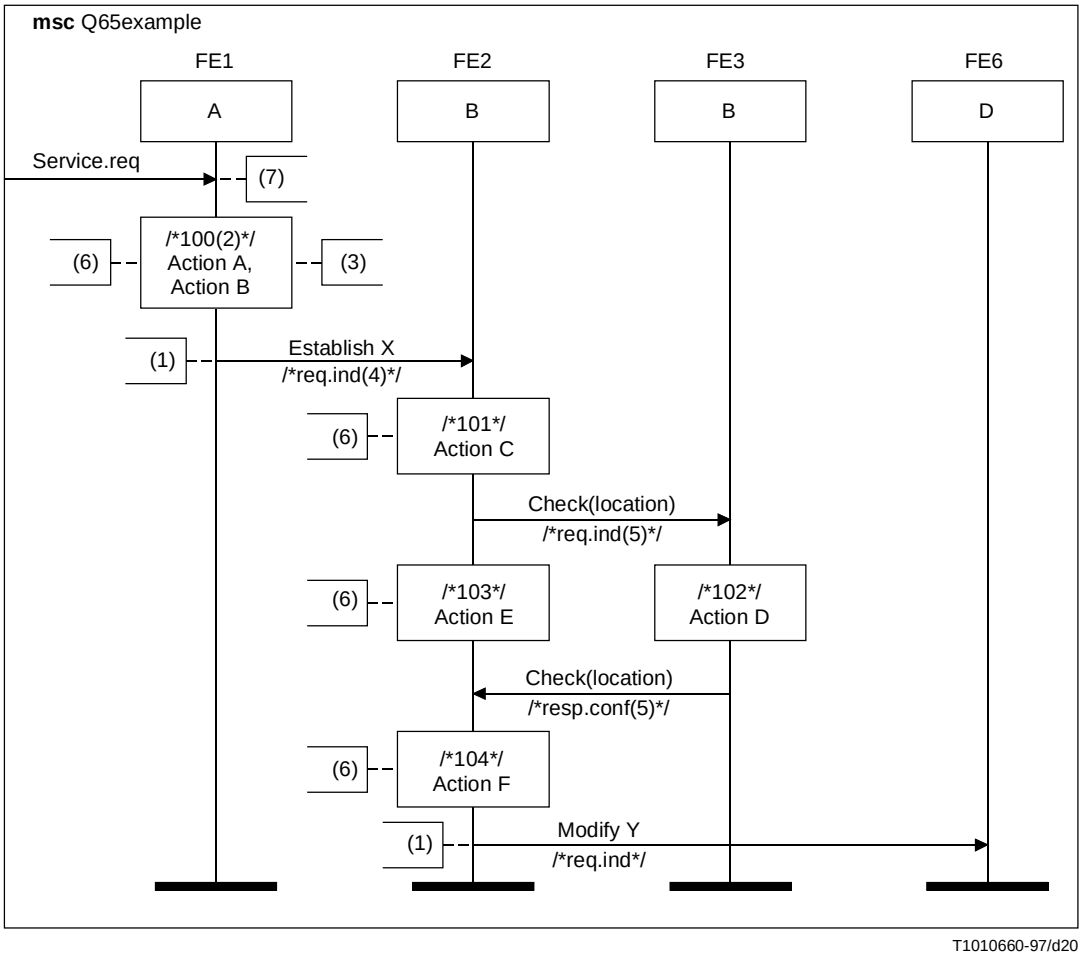
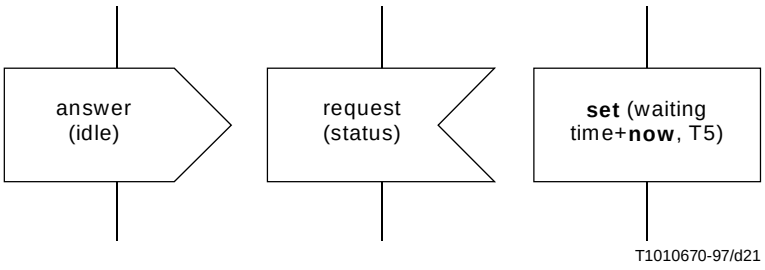


Figure 12-2/Suppl. 1 à la Rec. Z.100 – Exemple de diagramme de flux d'information extrait de la Recommandation Q.65



Exemple 12-2/Suppl. 1 à la Rec. Z.100 – Version MSC de l'exemple de la Figure 12-2



Exemple 12-3/Suppl. 1 à la Rec. Z.100 – Utilisation des données

L'utilisation des types énumérés devrait être utile. Un **newtype** ne comprenant que des littéraux correspond à un type de données énuméré (comme en Pascal):

newtype **Indication**
literals **idle, busy, congestion;**
endnewtype **Indication;**
...
signal **Check location (Indication) /*resp.conf */;**
...
output **Check location (busy) /*resp.conf*/;**
...

Les données élémentaires sont introduites dans les étapes 1 à 4 lors de la formalisation des spécifications: par exemple, les spécifications de signal appliquent des types de données. Les données peuvent être introduites à plusieurs niveaux de raffinement. Le bénéfice de l'introduction des données est naturellement qu'il permet de vérifier que les paramètres des entrées et des sorties, les questions et les réponses etc. sont cohérents dans l'ensemble de la spécification SDL.

13 Etapes d'Analyse

Supplément 1 à la Recommandation Z.100 (05/97)

L'activité d'Analyse se compose de deux étapes: l'Inspection et la Classification. Le domaine d'application n'a pas besoin d'être totalement inspecté avant de commencer l'étape de classification. Les concepts ou prescriptions d'application qui ont été inspectés peuvent être classifiés tandis que d'autres concepts ou prescriptions sont encore en inspection.

Les activités d'Inspection et de Classification aboutissent à la production de deux vues de l'application:

– une vue logique décrivant les éléments clés du système à spécifier;

– une vue dynamique décrivant le comportement de tous les objets.

La vue logique peut être donnée par une modélisation de relations entre composants (dite *modélisation par objets*). Cette vue correspond au point de vue information du système. Pour faciliter la réutilisation en vue d'inclure des composants de domaine externes ainsi que de créer de nouveaux composants de domaine réutilisables, on devrait utiliser des techniques d'analyse orientée objets (OOA, *object oriented analysis*) telles que la technique OMT [11], l'ingénierie OOSE [12] ou la notation SOON [6]. La réutilisation est grandement facilitée par le concept d'encapsulation des données et services ainsi que par le concept d'héritage. Ces deux concepts permettent de réutiliser sans modification des composants plus généraux ou des composants dont le comportement est proche de celui qui est attendu. La spécialisation de composants généraux ou l'adaptation de composants proches s'effectue par introduction des nouveaux composants appropriés. En conséquence, les techniques d'analyse OOA sont très utiles pour créer et réutiliser des composants de domaine (c'est-à-dire des composants appropriés à un domaine). Les étapes analytiques sont expliquées en détail dans l'approche de la technique OMT mais l'une des autres approches d'analyse OOA pourrait être utilisée.

Les diagrammes MSC sont parfaitement adaptés à la modélisation d'une vue (ou d'un comportement) dynamique, qui correspond à un point de vue traitement initial du système. Cette tâche consiste à définir des séquences d'utilisation attendues par l'utilisateur du système. Ces séquences d'utilisation seront généralement composées des scénarios typiques qui forment le comportement habituel du système, ainsi que des scénarios exceptionnels qui spécifient la réponse du système à des entrées erronées, à des pannes, etc. D'autres séquences d'utilisation peuvent être ajoutées afin de spécifier des aspects particuliers du comportement, par exemple lorsque le système entre en fonctionnement ou s'arrête. Cette modélisation est similaire à celle des utilisations élémentaires en [12].

Le présent paragraphe veille également à préciser certaines règles de cohérence qui doivent être satisfaites lorsque la modélisation par objets et la modélisation par séquences d'utilisation sont effectuées en parallèle.

L'activité d'Analyse doit être exécutée sans formulation d'hypothèses sur l'architecture d'application ou sur les détails de mise en œuvre. En particulier, les données doivent être modélisées conformément aux prescriptions issues seulement du point de vue de l'utilisateur et non pas du point de vue d'un ingénieur-systèmes qui tente d'optimiser la mise en œuvre du système. Il est utile, à ce propos, de disposer d'un modèle d'entreprise précisant clairement qui sont les utilisateurs et quels rôles ils jouent.

13.1 Etape d'inspection

Les différentes tâches constituant l'étape d'inspection sont présentées au 5.4.1. Ces tâches ne sont pas spécifiquement soutenues par les techniques d'ingénierie du système. Les ingénieurs peuvent, de préférence, utiliser un traitement de texte pour construire la bibliographie et rédiger les annotations de lecture.

13.2 Etape de classification pour la modélisation par objets

La modélisation par objets vise deux catégories d'objets:

– les objets physiques, qui représentent des entités physiques qui forment l'environnement du système ou qui sont utilisées par celui-ci pour fournir les services attendus;

– les objets logiques, qui représentent les informations consommées, produites ou mémorisées par le système, ou qui représentent des concepts d'application.

Les entités de modélisation par objets (physiques aussi bien que logiques) de l'environnement sont appelées *agents*. Un grand nombre des objets de télécommunication (comme les appels, les itinéraires, les comptes d'abonné) sont plus logiques que physiques. Cela est particulièrement vrai pour un système de spécification contenu dans une norme.

Les *associations* sont des relations de dépendance entre des objets tels que les suivants:

– connexions physiques entre objets physiques;

– relation producteur/consommateur entre objets (physiques ou logiques);

– relation d'utilisation entre objets (physiques ou logiques);

– liaisons d'*agrégation* entre objets (physiques ou logiques);

– liaisons d'héritage permettant la réutilisation d'objets plus généraux ou similaires.

Une *agrégation* est une association entre un objet composite et les objets composants, parfois appelée *raffinement*.

La modélisation n'est jamais influencée par le mode ou par le moment de la production ou de la transmission de point en point des informations. Elle n'est pas non plus influencée par le mode ou par le moment de l'interaction entre les entités physiques. Il s'agit d'un modèle statique (ou structurel).

Pour que le modèle soit général et puisse s'appliquer à différentes instances de l'application, des modèles de classe sont produits. Une classe d'objets X (par exemple un téléphone) définit les caractéristiques qui s'appliquent à tous les objets X (c'est-à-dire à tous les téléphones contenus dans l'exemple). Le modèle par classes montre les associations (relations) entre les classes. Dans une application réelle, les objets (instances des classes) doivent être conformes au diagramme de classe. Par exemple, si la classe "commutateur" est "raccordée à trois ou plus de trois" (association) objets de la classe "téléphone", il y aura toujours au moins trois téléphones pour un commutateur réel. Bien qu'il soit possible de dessiner des modèles d'instance d'objet, on n'y procède habituellement pas.

La notation utilisée est celle de la modélisation par objets (*object model notation*) de la technique OMT [11], dont les éléments de base sont représentés dans la Figure 13-1. La Figure 13-2 montre un diagramme partiel dans lequel le domaine d'application se rapporte à des opérations bancaires par cartes de paiement rechargeables.

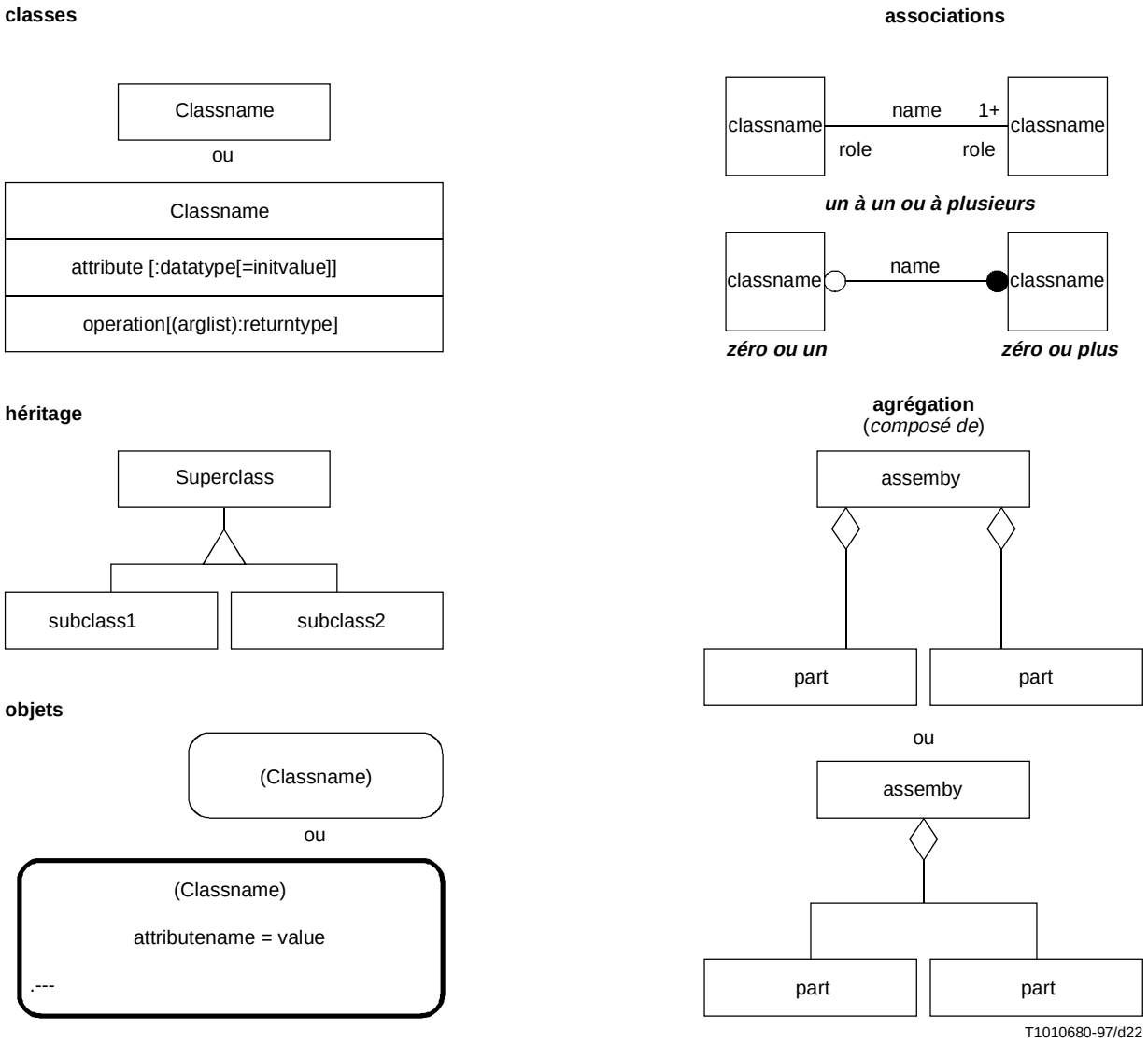


Figure 13-1/Suppl. 1 à la Rec. Z.100 – Eléments de base de la notation de modélisation par objets

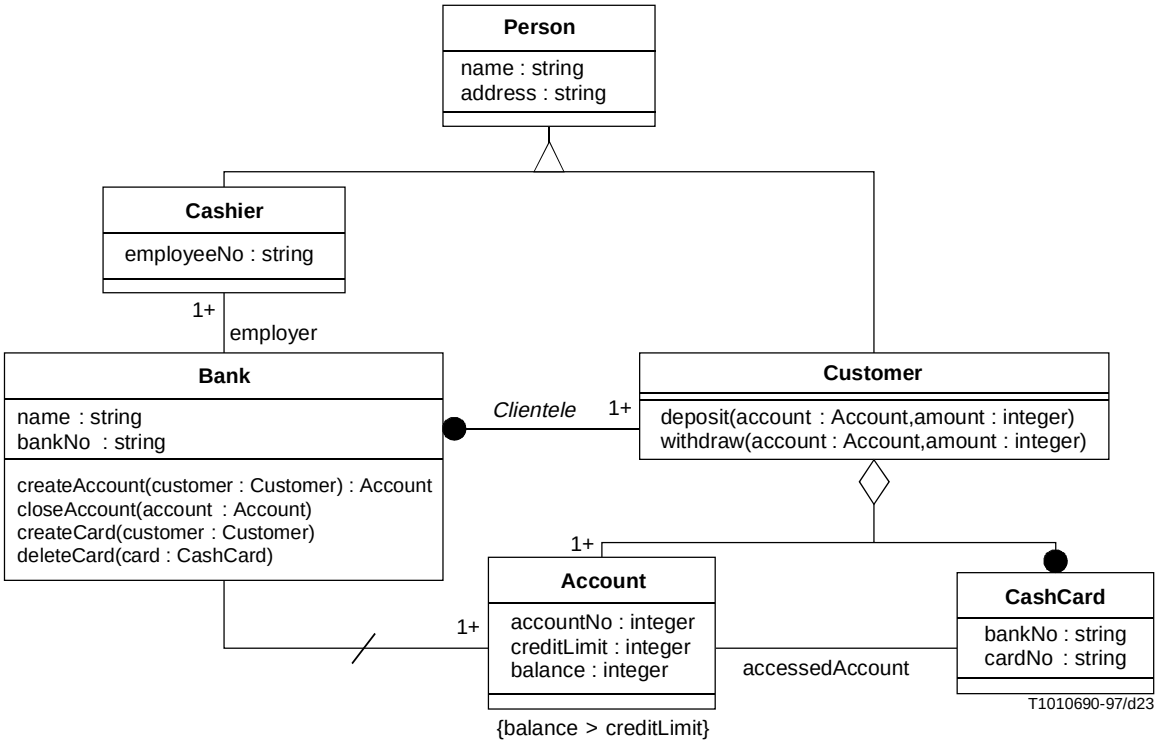


Figure 13-2/Suppl. 1 à la Rec. Z.100 – Exemple de notation de modélisation par objets

- classes (par exemple Banque): les classes contiennent des attributs (par exemple le nom et le numéro d'agence de la Banque) et des opérations (par exemple création ou fermeture de Compte bancaire); les attributs sont facultativement associés à un type et les signatures des opérations peuvent être déclarées;
- associations (par exemple association Clientèle entre Banque et Client): une association n'a pas d'orientation privilégiée; elle peut être nommée soit au moyen d'un nom unique soit par désignation de ses deux rôles (ceux des deux classes reliées par cette association); la multiplicité est indiquée pour chaque rôle de l'association;
- agrégations (par exemple entre Client et Compte ou entre Client et Carte de paiement): une agrégation est une association spéciale qui modélise un lien de raffinement; elle permet de décrire une structure hiérarchique;
- liens d'héritage (par exemple entre Personne et Caissier ou entre Personne et Client): il s'agit du soutien pour la réutilisation des propriétés de classe.

Les aspects informationnels d'une classe sont contenus dans les attributs de celle-ci, dans la multiplicité des associations qu'elle possède avec d'autres classes et dans les associations agrégatives contenant cette classe.

Les aspects interfaciaux d'une classe sont les associations de celle-ci avec d'autres classes qui ne sont pas des associations agrégatives, ainsi que les signatures des opérations de cette classe qui peuvent être invoquées par d'autres classes.

En général, les aspects de comportement ne sont pas détaillés par le modèle par classes produit au cours de l'activité d'Analyse: ils restent sous forme de texte dans les prescriptions collectées éventuellement associées (d'une manière ou d'une autre) aux descriptions de classe pertinentes. Les aspects divers d'une classe (aspects généraux tels que la flexibilité, la compatibilité, la facilité d'utilisation, la fiabilité, la sûreté, la sécurité; contraintes de conception; modes de panne; qualité de fonctionnement; etc.) sont également laissés sous forme de texte. Cependant, il convient que, dans la mesure du possible, les textes pour les aspects comportementaux et divers soient associés aux classes produites, de façon que l'on puisse déterminer quel aspect est traité par quel objet.

Pour gérer la complexité d'un diagramme de classe et pour en préserver la lisibilité, un diagramme est structuré en modules. Ces derniers représentent habituellement différentes vues du système, comme celle de l'utilisateur, celle de l'environnement physique, celle d'un service à fournir par le système. La collection de tous ces modules constitue le diagramme de classe complet. Les classes peuvent être (ce qui est souvent le cas) représentées sous forme de plusieurs modules. Par exemple, dans le domaine des opérations par cartes de paiement, les modules importants sont: la description d'un client du point de vue bancaire (compte, carte de paiement, etc.), le réseau de distributeurs automatiques (point de vue du groupe bancaire), la composition d'un distributeur en termes d'équipements. Il y a aussi un module décrivant quels sont les entités physiques et les concepts mis en jeu lors de l'exécution d'une transaction bancaire.

Instructions

1) identifier et nommer les classes, et les créer dans le modèle;
2) identifier et nommer les associations et les agrégations entre classes;
3) identifier et nommer les attributs de classe;
4) identifier et nommer les opérations de classe;
5) organiser et simplifier les classes au moyen des héritages;
6) grouper les classes en modules;
7) vérifier que des chemins d'accès existent pour les interrogations habituelles et que les données requises existent;
8) itérer et raffiner le modèle.

Directives

Identification des classes

L'analyse des prescriptions d'application conduit à l'identification des objets physiques et logiques. Généralement, les classes sont d'abord identifiées par examen des noms utilisés dans les documents en mode texte collectant les prescriptions. Il est parfois utile de créer un dictionnaire de données énumérant toutes les classes, assorti d'une description copiée ou extraite des prescriptions saisies. En variante, l'on peut faire appel à des liens hypertextuels vers les prescriptions collectées. Il sera important de déterminer la frontière entre les classes situées à l'intérieur du système et entre les classes situées à l'extérieur du système. Cette distinction peut se refléter dans les noms donnés à des agents extérieurs au système. Il y a lieu de donner un nom au système.

Nomination des classes (associations, attributs et opérations)

Le choix des noms est une décision importante: il y a souvent plusieurs termes dans les prescriptions collectées pour une classe d'objets (ou pour des associations, des attributs ou des opérations). Un mauvais choix de nom peut donner lieu ultérieurement à des confusions. Les noms choisis devraient cependant être réexaminés plus tard et être éventuellement modifiés étant donné que la compréhension du système sera meilleure une fois que certains travaux d'ingénierie auront été effectués.

Spécificité des classes

Il y a lieu que les classes ne soient extraites que des prescriptions collectées et qu'elles n'introduisent pas de détails architecturaux ne faisant pas partie des prescriptions collectées. En d'autres termes, les seules contraintes architecturales à prendre en considération concernent les conditions dans lesquelles le système doit fonctionner. Par conséquent, les classes représentant des entités physiques devraient être limitées aux entités qui concernent le domaine d'application et non la mise en œuvre de l'application. Les entités logiques devraient représenter des concepts d'application couramment acceptés. L'utilisation d'une bibliothèque de concepts aidera, en association avec l'expérience générale, à déterminer si une classe est appropriée. Par exemple, une base de données constituée d'une liste de clients avec leurs noms, leurs adresses et la liste des produits qu'ils ont achetés peut être modélisée par la classe "Client" avec les deux attributs "nom" et "adresse" ainsi que par la classe "produit" sans créer de troisième classe représentant la base de données. Les classes redondantes (qui représentent le même concept, la même entité physique ou qui contiennent les mêmes attributs) doivent être supprimées. Les classes vides, c'est-à-dire sans attributs et sans lien précis avec les autres classes du modèle doivent également être supprimées.

Identification des associations

Toute relation de dépendance entre classes est modélisée sous la forme d'une association. Celle-ci peut représenter une liaison physique, une relation producteur/consommateur ou une relation d'utilisation (les agrégations seront analysées plus loin). Les associations sont initialement identifiées par l'examen des verbes (qui "relient" des noms, par exemple "emploie" ou "accède à"). La plupart des associations sont binaires, c'est-à-dire qu'elles ne mettent en jeu que deux classes (il existe une notation pour les associations multiples, le cas échéant). Il ne faut pas trop se préoccuper de la nomination des associations car il peut y en avoir beaucoup qui correspondent à une possession ("détient", "possède") et à des connexions ("raccordée à", "en liaison avec", "s'adresse à") qui n'ont pas besoin de noms car le modèle par classes rend l'association évidente.

Identification des agrégations

Les associations qui représentent un lien de possession ou de composition sont habituellement mieux modélisées sous la forme d'agrégations. Une agrégation ajoute une nouvelle contrainte, par rapport à une association simple: le comportement de l'objet exerce une forte influence sur le comportement de ses objets agrégés: la création et la suppression de l'objet initiateur de l'agrégation et à la suppression des objets agrégés. Les agrégations sont créées par transformation d'associations nommées "partie de", "composée de", "élément de", etc.

Spécificité des associations et des agrégations

Les associations redondantes doivent être supprimées ou être explicitement marquées comme étant redondantes (par la notation d'association "dérivée", voir dans la Figure 13-2 l'association entre "Account" et "Bank"). Les interactions entre classes sont modélisées sous la forme d'associations seulement si elles représentent des propriétés structurelles du domaine d'application au lieu d'un événement transitoire. La multiplicité (nombre d'éléments mis en jeu de chaque côté) des associations devrait être convenablement définie.

Identification des attributs appropriés

Les attributs sont des propriétés des classes. Ils servent à saisir les données définies dans les prescriptions collectées, telles que les noms, les adresses, les numéros de carte, etc. Les détails de conception ou de mise en œuvre tels que les identificateurs de processus (Pid) doivent être éliminés. Les attributs complexes (données structurées, données sans limitation de longueur, etc.) peuvent être transformés en classes (de données). Les attributs peuvent être associés à des types mais il y a lieu d'éviter des descriptions de type trop compliquées si elles correspondent à des détails de mise en œuvre. Les attributs ne peuvent pas pointer sur d'autres classes: il faut alors utiliser des associations.

Identification des opérations appropriées

Les classes sont enrichies par l'adjonction d'opérations. Celles-ci ne sont pas faciles à identifier si l'on n'effectue que la modélisation par objets. La modélisation de la vue dynamique des prescriptions aide beaucoup à identifier les opérations que les classes doivent fournir. Par conséquent, les ingénieurs doivent modéliser la vue dynamique parallèlement à la vue logique. Les opérations correspondent à des services (y compris l'accès aux données et leur production) fournis par la classe à l'environnement ou à des classes auxquelles elle est reliée par des associations. Les signatures d'opérations peuvent aussi être définies (paramètres formels et valeur à retourner).

Utilisation des héritages

Les classes qui ont une structure partiellement commune (attributs ou associations) peuvent être réorganisées par l'introduction d'une nouvelle classe qui encapsule cette substructure commune. Les classes initiales deviennent alors des sous-classes spécialisées de la superclasse. L'introduction de superclasses peut aussi faire suite à la réutilisation de classes déjà définies, disponibles dans le domaine d'application. Il convient cependant d'éviter d'avoir de nombreux niveaux d'héritage.

Groupage des classes en modules

Les modules sont créés afin de représenter des points de vue significatifs dans le domaine d'application. Les classes sont groupées en modules. Une classe peut être présente dans plusieurs modules, ce qui est le cas général car les points de vue ne sont pas des partitions du système. Pour améliorer la lisibilité des diagrammes, l'ingénieur ne devrait pas créer plus de douze classes dans un même module. Selon la complexité de chaque classe, le nombre d'attributs, le nombre d'opérations, le nombre d'associations, le nombre idéal de classes pour un module est compris entre 4 et 8.

Contrôle des chemins d'accès et de la couverture des données

Le modèle par classes peut être contrôlé en fonction des prescriptions. Deux éléments doivent être contrôlés: l'aptitude à la navigation dans le modèle pour accéder aux informations et la couverture des données. Si une demande de navigation requise n'est pas soutenue, même indirectement par un ensemble d'associations, il faut introduire de nouvelles associations. Le modèle dynamique est très utile pour contrôler la complétude des chemins d'accès car il montre quelles entités interagissent les unes avec les autres et quelles informations sont consultées par quelle entité (informations transmises par des messages). Si une donnée élémentaire n'est pas présente dans le modèle sous forme d'attribut, d'ensemble d'attributs ou de résultat d'opération, il faut introduire de nouveaux attributs ou de nouvelles opérations.

Raffinement du modèle

La modélisation par objets est un processus itératif. Le diagramme de classe initial est très proche des prescriptions collectées (noms sous forme de classes, verbes sous forme d'associations, etc.). Par itérations successives, le modèle est réorganisé de façon à être non ambigu, non redondant, facile à comprendre et complet par rapport aux prescriptions collectées. L'identification des attributs et des opérations sera en particulier raffinée par d'ultérieures itérations. Un jugement d'ingénierie devra être formulé sur le degré de complétude que le modèle devrait avoir. La modélisation dynamique conduit à un enrichissement du modèle par objets, surtout en termes d'opérations et d'associations de classe. Les itérations dans le modèle par objets devraient donc être effectuées en parallèle avec les itérations dans le modèle dynamique. Un tel raffinement peut être considéré comme plus approprié à l'activité de Conception du projet mais comme celle-ci est facultative, il peut être considéré comme faisant partie de la Classification.

Règle 1 – Les noms au pluriel ne doivent pas être utilisés pour nommer des classes (il n'y a qu'une seule classe et plusieurs instances).

Règle 2 – Le modèle par objets doit contenir, au moyen de classes, tous les agents actifs dans un domaine de l'environnement du système.

Règle 3 – Le modèle par objets doit contenir toutes les données exprimées par les prescriptions collectées, au moyen d'attributs ou d'opérations.

Règle 4 – Le modèle par objets doit satisfaire à toutes les prescriptions en termes de navigation dans les objets, au moyen d'associations et d'agrégations.

Résultat: le résultat est un modèle par objets cohérent et complet, exprimant sous forme de diagramme la vue logique des prescriptions collectées.

13.3 Etape de classification pour la modélisation par séquences d'utilisation

Les objectifs de la modélisation par séquences d'utilisation sont de saisir et de classifier le comportement attendu du système en réponse aux stimuli issus de son environnement, sans formuler d'hypothèses sur la façon dont ces stimuli sont considérés à l'intérieur du système ni sur la façon d'élaborer les réponses. Le comportement est classifié au moyen d'un ensemble de diagrammes MSC habituellement composé de séquences d'utilisation typiques (qui représentent en fait la mission du système) et de séquences d'utilisation exceptionnelles qui décrivent la robustesse attendue. D'autres séquences peuvent être ajoutées pour préciser des états particuliers du système (démarrage, arrêt, etc.).

Etant donné que la modélisation par séquences d'utilisation vise à classifier, au cours de l'Analyse, le comportement attendu du système, il n'est pas nécessaire de produire des diagrammes de transition d'états. Ces diagrammes seront créés lors des étapes de Conception du projet ou de Formalisation.

Les séquences d'utilisation décrites dans la présente étape ne tiennent pas compte de l'architecture interne du système. Les seuls éléments mis en jeu sont les suivants: le système représenté en tant qu'instance unique et tous les agents mis en jeu lors de la communication avec le système. Ces agents correspondent aux entités dynamiques telles que l'utilisateur ou le dispositif interagissant avec le système. Par ailleurs, lorsque l'architecture du système est esquissée dans la Conception du projet, les séquences d'utilisation produites lors de l'Analyse peuvent être raffinées en conséquence pour produire des séquences d'utilisation qui reflètent les composants internes du système.

L'une des principales difficultés rencontrées consiste à gérer convenablement le grand nombre de diagrammes MSC à créer. L'ensemble des diagrammes MSC produits couvre tous les scénarios contenus dans les prescriptions collectées. Mais il y a lieu d'éviter les redondances afin d'avoir un ensemble minimal. A cette fin, il convient que les ingénieurs prêtent une attention particulière à la structure du document contenant les diagrammes MSC.

Instructions

1) identifier les séquences d'utilisation, typiques et exceptionnelles, à créer;
2) décomposer les séquences d'utilisation complexes en séquences plus petites et indiquer la façon dont les séquences d'utilisation de niveau inférieur sont regroupées par les diagrammes MSC de haut niveau (HMSC, *high-level MSC*);
3) pour chaque séquence d'utilisation de niveau inférieur, dessiner le diagramme MSC en représentant sous forme d'instances (barres verticales) le système et tous les agents impliqués;
4) décrire sous forme de messages dans les diagrammes MSC la chronologie attendue (sur la base des prescriptions) des flux de commande et de données;
5) s'assurer que toutes les séquences d'utilisation, typiques et exceptionnelles, collectées dans les prescriptions, sont couvertes par le document MSC;
6) s'assurer de la cohérence du modèle dynamique décrit par le document MSC, au moyen du modèle par objets.

Directives

Identification des séquences d'utilisation requises

Les séquences d'utilisation à produire sont identifiées d'après les prescriptions qui concernent les aspects comportementaux du système. Il arrive souvent que des séquences particulières soient recherchées ("Que se passerait-il si...?") mais n'aient pas été prises en compte dans les prescriptions collectées. Dans ce cas, les ingénieurs peuvent décider d'une séquence appropriée et la confirmer en posant une question au sujet des prescriptions collectées.

Organisation des séquences d'utilisation

Habituellement, le comportement prévu d'un système est un ensemble de variantes d'un certain nombre de séquences nominales. Ces variantes représentent toutes les exceptions qui peuvent se produire et qui devraient être prises en considération lorsque les séquences nominales sont mises en exploitation.

De façon à éviter toute redondance entre les séquences décrites, à faciliter la réutilisation de parties de séquences, le comportement prévu devrait être subdivisé en petits diagrammes MSC réutilisables. En MSC-96, il existe plusieurs manières de combiner les diagrammes MSC:

– les références des MSC servent à renvoyer (à partir d'un diagramme MSC) à d'autres diagrammes MSC;

– les séquences d'utilisation complexes devraient être décrites par des diagrammes MSC de haut niveau, contenant des graphes orientés renvoyant à d'autres diagrammes MSC;

– la combinaison de diagrammes MSC peut également être décrite en termes de diagrammes MSC contenus dans des références MSC, par exemple "MSC1 **seq** (MSC2 **alt** MSC3)";

– de légères variations, lorsque par exemple il y a un choix entre des messages en variante, sont mieux représentées par des expressions en coupure de ligne.

Les diagrammes MSC simples devraient être expressément nommés afin de donner une idée de la façon dont ils pourraient être combinés. Par exemple: "no_answer_from_third_party" ou "caller_hangs_up".

Identification des instances d'un diagramme MSC

Les instances de diagrammes MSC apparaissant dans une séquence d'utilisation devraient être des instances d'agents définies comme des classes dans le modèle par classes et dans le système d'application. Généralement les instances MSC correspondent à des entités physiques. Dans l'environnement, les objets logiques correspondent à des éléments passifs et sont transmis comme paramètres des messages MSC.

Le système apparaît dans un diagramme MSC sous la forme d'un trait vertical. Il peut y avoir dans le diagramme MSC autant d'instances du système qu'il y a d'instances concomitantes en communication les unes avec les autres dans le monde réel. Mais il est plus habituel de ne prendre en considération qu'un seul système. Par exemple, dans le cas d'un réseau de systèmes interconnectés, le diagramme MSC contiendra plusieurs instances du système. De même, un agent peut avoir plusieurs instances dans un MSC si cette situation se produit dans la réalité. Différentes instances du même agent devraient être expressément nommées en tant que telles de façon que le diagramme MSC soit compréhensible. Par exemple "Caller" et "Called" dans un MSC mettant en jeu deux abonnés en dialogue. On limite habituellement le nombre d'instances contenues dans le MSC au nombre minimal requis pour montrer le comportement dans la réalité.

Il n'est pas nécessaire, en Analyse, de décrire formellement les moyens (canaux de communication et identification d'instance) utilisés par les agents et par le système pour entrer en interaction. Les ingénieurs doivent considérer que les instances peuvent être atteintes indépendamment, sans problème d'identification ou de connexion.

Identification et formulation des messages

Les messages correspondent habituellement à des opérations de classe dans le modèle par objets. Leurs paramètres correspondent habituellement aux attributs de classe ou aux résultats des opérations de classe.

Pour satisfaire aux critères d'accessibilité des informations, il devrait y avoir dans le modèle par objets des chemins (directs ou indirects) entre les instances MSC qui sont en dialogue. Cependant, les associations du modèle par objets ne doivent pas nécessairement se traduire par des messages des diagrammes MSC.

Contrôle de la complétude du modèle dynamique

Chaque scénario possible devrait être couvert par un diagramme MSC, de manière que le modèle dynamique soit normalement terminé lorsque tous les scénarios possibles ont été couverts. Mais la complétude est habituellement impossible à obtenir parce que le nombre de séquences d'utilisation possibles est très grand. Les itérations sur le modèle dynamique seront arrêtées lorsque les ingénieurs considéreront qu'un ensemble raisonnable de séquences d'utilisation a été produit. Ce choix est subjectif et dépendant du système.

Contrôle de la cohérence du modèle dynamique par rapport au modèle par objets

Le modèle dynamique doit être contrôlé par comparaison avec le modèle par objets et les informations manquantes doivent être ajoutées au modèle par objets afin de respecter les règles suivantes:

1) les instances MSC doivent correspondre au système ou à des objets d'agent (instances de classes);

2) les messages doivent correspondre aux opérations de classe. Habituellement, un message envoyé par une instance de "a" à une instance de "b" est déclaré dans la classe "b" par l'indication que "b" fournit ce service à "a". Mais parfois, une opération associée peut aussi être déclarée dans la classe "a" par l'indication que "a" fournit ce service à d'autres classes qui ne souhaitent pas accéder directement à "b" (par exemple parce que "b" fournit un service de niveau trop bas);

3) les paramètres des messages devraient correspondre aux attributs de classe ou aux résultats des opérations de classe. Les attributs devraient être déclarés dans la classe correspondant au message ou déclarés dans d'autres classes rendues accessibles par des associations;

4) pour deux instances MSC en dialogue, leurs classes correspondantes doivent être connectées dans le modèle par objets, soit directement soit indirectement par des associations.

La cohérence entre les deux modèles est évaluée au cours des dernières itérations sur les modèles. Il est tout à fait courant d'enrichir le modèle par objets d'opérations extraites des messages de diagramme MSC correspondants.

Règle 5 – Pour montrer les séquences de messages échangés entre le système et son environnement en fonction du temps indiqué dans les diagrammes, il faut utiliser la notation MSC.

Règle 6 – Les séquences d'utilisation doivent montrer le système comme un ensemble en interaction avec les agents environnementaux, sans tenir compte de l'architecture interne du système.

Résultat: un modèle par séquences d'utilisation (diagrammes MSC) cohérent avec le modèle par objets, qui saisit sous forme de diagrammes la vue dynamique des prescriptions collectées.

14 Etapes de conception de projet

Les étapes détaillées aux 14.1 à 14.6 sont présentées dans un ordre qui convient pour lancer ces étapes, mais celles-ci peuvent être exécutées dans un ordre quelconque, sauf si les informations issues d'un certain modèle sont utilisées dans un autre modèle. Ces étapes peuvent facilement être exécutées en parallèle parce que chacune d'elles est centrée sur un type de modélisation différent. Chaque modèle peut s'appliquer à la totalité ou à une partie du système.

Comme le nombre de conceptions de projet nécessaires varie notablement, il est possible de n'indiquer que les conceptions de projet qui sont susceptibles d'être plus utiles à la Formalisation que d'autres. C'est ce qui est résumé dans le Tableau 1.

Tableau 1/Suppl. 1 à la Rec. Z.100 – Utilité des conceptions de projet lors de la formalisation

Modèle de conception de projet	Utilité
modélisation des relations entre composants	il est parfois utile de déterminer combien il y a de blocs, de processus ou d'éléments de données.
diagrammes de flux de données ou de commande	un diagramme contextuel est souvent utile. La hiérarchisation des diagrammes est utile en ASN.1.
structuration des informations	cette opération doit être effectuée, soit lors de la Conception du projet soit lors de la Formalisation.
modélisation par séquences d'utilisation	les séquences d'utilisation sont normalement utilisées pour la conception de projet. Elles sont utiles pour les parties informatives de l'application.
modélisation comportementale du processus	ce modèle n'est produit qu'occasionnellement. Il sert de base aux diagrammes formels.
modélisation par aperçu général d'états	ce modèle est utile pour les processus de contrôle mais il n'a pas besoin d'être dans l'application.

L'utilisation de tableaux binaires pour modéliser la vue information peut paraître intéressante au premier abord mais elle a des inconvénients notables car ces tableaux confondent les problèmes de définition de la structure informationnelle avec le codage des informations. Les tableaux binaires ont tendance à banaliser la finalité de chaque donnée élémentaire. Des données élémentaires importantes ne sont parfois pas étiquetées d'un nom significatif dans des tableaux binaires parce qu'elles ne sont représentées que par un seul élément binaire ou par quelques bits. Les tableaux binaires ne sont pas flexibles, en partie parce qu'ils confondent signification et structure dans le codage: une modification du moyen de transport peut nécessiter un changement de codage mais sans autre modification sémantique. Les tableaux binaires peuvent parfois être mal compris si l'on ne voit pas clairement si le bit de numéro le plus élevé (ou le plus à gauche) est de poids fort ou de poids faible, aligné à gauche ou à droite, transmis en premier ou en dernier. La présentation des tableaux binaires n'est pas normalisée. Les tableaux binaires ne supportent pas la composition, la recherche analytique et le contrôle systématique, qui sont bien supportés par les outils pour l'ASN.1 et le SDL.

Règle 7 – Les tableaux binaires ne doivent pas être utilisés pour modéliser le point de vue information.

NOTE – Les tableaux binaires peuvent être utilisés pour décrire le codage.

14.1 Modélisation d'une relation entre composants

L'objectif de cette étape est de compléter le modèle par classes produit en Analyse afin de décrire l'architecture interne du système. Ce modèle est utilisé pour représenter des flux de données ou de commande, des informations, des séquences d'utilisation et pour la Formalisation. Les détails ajoutés au modèle par classe correspondent à l'architecture interne du système: les aspects informationnels et la conception des associations. Cette modélisation est toutefois considérée comme faisant partie de l'Analyse si la Conception du projet n'est pas incluse en tant qu'activité. Il existe deux moyens d'effectuer cette modélisation: en tant que raffinement du modèle de classes à partir de l'Analyse ou par remplacement de classes par des classes en variante plus détaillées.

Instructions

1) définir les sortes de données des attributs de classe ainsi que les opérations de classe déclarées dans le modèle par objets de l'Analyse;

2) concevoir les associations de classes.

Ces activités conduisent à un enrichissement du modèle par classes en introduisant de nouveaux attributs, de nouvelles associations, de nouvelles liaisons d'héritage ou de nouvelles classes.

Directives

Choix du type des attributs et des opérations

Dans l'Analyse, les sortes de données utilisées pour les attributs et pour les opérations devraient soit être des sortes simples soit être des noms sans raffinement complémentaire. Des listes, tableaux ou structures complexes ne devraient pas avoir été utilisés. Si elles relèvent de l'Analyse, c'est-à-dire si elles saisissent des concepts d'application, elles devraient être modélisées en tant que classes et associations. Sinon, les données informationnelles élémentaires mais complexes sont négligées. Pour la Conception du projet, les sortes de données nécessaires doivent être définies plus précisément, mais la description de toutes les sortes de données contenues dans le modèle par classes pourrait être un effort redondant par rapport à d'autres étapes telles que la modélisation des flux de données ou la modélisation des structures informationnelles. Les sortes de modélisation des données devraient donc être limitées aux attributs et aux opérations les plus importants pour contribuer à la modélisation par flux de données sans travaux superflus.

En technique OMT, les sortes de données ne sont déclarées qu'au niveau du modèle et ne sont donc pas accessibles par toutes les classes; des sortes de données ne peuvent pas être déclarées localement à une classe. Les attributs sont associés aux types soit par création de nouvelles sortes de données globales soit par adjonction d'agrégations ou d'associations si les structures de données complexes correspondantes contiennent des classes ou accèdent à des classes. De nouvelles classes peuvent également être ajoutées au modèle si les structures de données complexes contiennent plus d'informations que celle qui est accessible (au moyen des attributs) dans le modèle par objets de l'Analyse.

L'adjonction de sortes de données pour des paramètres et pour des résultats d'opération peut également conduire à une réorganisation des hiérarchies d'héritage afin de faciliter la redéfinition d'opérations. Les signatures des opérations devraient être conformes aux hiérarchies d'héritage.

Conception des associations

Les associations définies dans le modèle par objets de l'Analyse sont bilatérales et permettent d'accéder aux instances de classe sans connaître la façon d'y procéder. La conception de projet devrait expliquer comment accéder aux instances. Généralement, pour chaque association, un sens privilégié devrait être choisi. Pour la classe des destinataires de cette association, "l'attribut clé" devrait être choisi parmi ceux qui existent (ou devrait être créé s'il n'existe pas encore). Cet attribut clé identifie une instance de cette classe parmi les autres instances. De nouvelles classes peuvent aussi être ajoutées pour résoudre le problème des associations à cardinalités dans les deux sens.

Règle 8 – Le modèle par objets est complété par l'utilisation de la même notation que celle qui a été utilisée pour l'Analyse.

14.2 Modélisation des flux de données et de commande

Instructions

1) produire un diagramme contextuel;

2) décomposer le bloc SDL contenu dans le diagramme contextuel en sous-blocs utilisant la structure d'agrégation et de décomposition de niveau supérieur extraite des informations classifiées (c'est-à-dire du modèle de classe);

3) répéter la décomposition pour les sous-blocs jusqu'à ce que le comportement de chaque bloc ne puisse plus être notablement subdivisé ou qu'il soit évident que le bloc correspond à un processus SDL unique.

Directives

Le diagramme contextuel est un diagramme de système SDL qui contient un bloc unique représentant le comportement du système. Les canaux à destination et en provenance de ce bloc représentent les flux d'information entre le système et les agents situés dans l'environnement. Dans les informations classifiées, on peut identifier ces flux d'après les associations établies entre le système et des agents situés dans l'environnement et d'après les événements MSC entre instances de système et instances d'agent. Ces flux représentent **l'interface** du système. Il y a un seul canal par flux d'information identifié séparément (habituellement un par instance d'agent dans le diagramme MSC). S'il est requis que deux ou plus de deux flux d'information soient fusionnés dans l'environnement, ces flux sont représentés avec le même point de connexion à la frontière du système SDL établissant la liaison avec l'environnement. Les signaux ne sont pas associés aux canaux mais le flux d'information est déduit des descriptions en langage naturel.

S'il n'existe pas de classe dans les informations classifiées pour le système, ce sont les classes situées au niveau supérieur (et non pas des parties d'autres classes, autrement dit les classes non agrégées) et qui ne sont pas des agents (classes d'objets situées dans l'environnement) qui peuvent être considérées comme étant agrégées pour former le système.

Les sous-blocs qui possèdent des aspects **informationnels** importants et peu ou pas du tout de comportement spécifique (autre que le stockage des informations) sont assortis de la notation "données". Il ne devrait jamais y avoir de connexion directe entre deux "blocs de données" de ce type. Les canaux sont identifiés d'après les informations d'interface. Les canaux à destination et en provenance des "blocs de données" indiquent les flux de données. Les canaux entre d'autres blocs (de comportement) indiquent les flux de commande. Le comportement de ces blocs devrait faire l'objet d'une description. Si celle-ci n'existe pas, il faut en rédiger une en langage naturel.

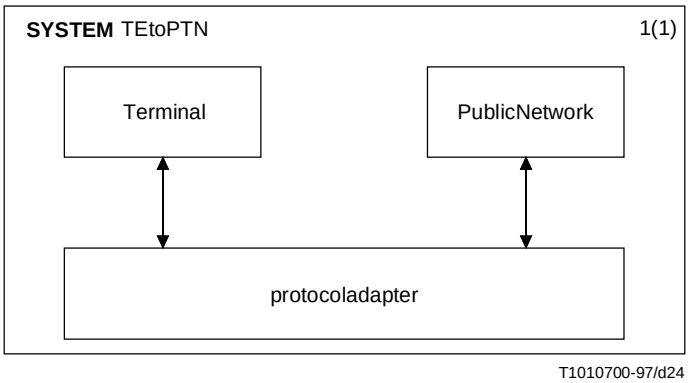
Ces descriptions font appel au langage SDL pour montrer des flux de données et de commande; mais elles ne montrent pas le détail des signaux ni le comportement du processus. Cette approche permet d'explorer diverses décompositions et de mettre l'accent sur les flux de données et sur les répertoires de données. Chaque fois qu'un "bloc de données" possède des connexions avec plusieurs blocs, il y a lieu de reconsidérer la décomposition car les données devraient être contenues dans un même processus du modèle formalisé.

Il convient d'examiner l'architecture acceptée qui a été utilisée dans le domaine d'application. Par exemple, une architecture qui sépare des flux d'information peut être une prescription, de manière qu'il y ait différents plans de protocole: "le plan d'utilisateur", "le plan de commande" et le "plan de gestion".

Il arrive souvent que le contexte entier ne puisse être spécifié au moyen d'une approche descendante depuis le sommet. Dans ce cas, il peut être nécessaire d'appliquer une certaine compilation de bas en haut à partir des spécifications de processus.

Règle 9 – Tout diagramme contextuel doit être en langage SDL.

Exemple



NOTE – Pour le diagramme contextuel, la description SDL n'a pas besoin d'être complète: dans cet exemple, les canaux n'ont pas de nom et les signaux ne sont pas définis.

Figure 14-1/Suppl. 1 à la Rec. Z.100 – Diagramme contextuel pour l'adaptateur de protocole

14.3 Modélisation de la structure des informations

Les structures informationnelles contenues dans le système sont modélisées au moyen de la notation ASN.1 afin d'élaborer le point de vue information à partir du modèle de classe dans le cadre des informations classifiées. Même si le codage et la structure des informations sont mélangés dans les prescriptions collectées, le codage devrait être examiné séparément de la structure informationnelle. Un modèle logique du système peut être produit sans codage. Celui-ci n'est nécessaire que lorsque l'on examine la contrainte de correspondance de la structure informationnelle avec les bits. Si les règles de codage par défaut sont satisfaisantes, le codage proprement dit peut être omis.

Instructions

1) identifier les flux d'information contenus dans les diagrammes de flux de données et de commande les moins abstraits (les plus détaillés) ou, si la Formalisation est déjà en cours, les canaux issus des blocs contenant des processus et les routes de signalisation contenues dans ces blocs;

2) identifier la structure des aspects informationnels associés aux flux d'information et définir un nom de type ASN.1 explicite pour chaque sorte de données qui ne fait pas partie d'une agrégation (messages de données de niveau supérieur);

3) identifier et nommer le niveau immédiatement supérieur de la structure dans les messages de données de niveau supérieur, en choisissant le même nom explicite de type ASN.1 si la même information est utilisée à plusieurs endroits;

4) définir le type ASN.1 de niveau supérieur dans les termes des noms de type de niveau immédiatement supérieur et répéter les deux dernières étapes jusqu'à ce que chaque partie soit un type ASN.1 simple;

5) identifier la structure par rapport aux aspects informationnels des blocs les plus internes (aussi bien de comportement que de données) dans les diagrammes de flux de données et de commande les moins abstraits;

6) identifier et nommer les sortes de données contenues à l'intérieur de ces blocs de manière similaire aux données pour interfaces, mais en réutilisant les types ASN.1 définis pour ces interfaces;

7) identifier toutes valeurs des types ASN.1, auxquelles il est fait référence par un nom; définir les noms ASN.1 de ces valeurs à partir d'un type ASN.1 simple en réutilisant, au besoin, les noms des valeurs pour former des noms ASN.1 composés;

8) identifier les aspects des blocs les plus internes d'après le comportement; identifier les opérateurs nécessaires pour les sortes de données et les enregistrer sous la forme de commentaires de la notation ASN.1.

Directives

Habituellement, il existe des sortes agrégées de données pour différents messages, appelées souvent unités de données de protocole (PDU, *protocol data unit*). Leur structure peut être décrite ou définie dans les prescriptions collectées et dans le modèle de classe contenu dans les informations classifiées.

Les types ASN.1 intermédiaires sont choisis de façon qu'ils puissent être utilisés dans différents messages. En particulier, lorsque des informations sont converties d'un protocole en un autre, il y a souvent quelques éléments de données communs qui sont transférés d'une interface à une autre par l'intermédiaire de messages.

L'agrégation correspond aux types composés en notation ASN.1, de sorte que l'on utilisera, lors de la décomposition des types, les mots clés SEQUENCE, SEQUENCE OF, SET, SET OF et CHOICE.

En plus de la recherche des types définis dans le cadre de cette étape, il y a lieu de consulter la bibliothèque de "types utiles" ASN.1 et les définitions ASN.1 utilisées pour les interfaces des normes associées à l'application.

Au cours du développement, il est parfois commode d'utiliser la structure ASN.1 "any". Lorsque les définitions de types ASN.1 sont complètes, la structure "any" devrait avoir été remplacée par un nom spécifique, si possible défini extérieurement.

Si des tableaux binaires ont été fournis dans les prescriptions collectées, il conviendra de récrire ces tableaux en notation ASN.1. Le codage de base ASN.1 (X.209 [10]) n'est pas toujours approprié à un protocole: il peut par exemple ne pas être assez efficace. Dans ce cas, on fait appel au codage explicite. Une forme simplifiée des tableaux binaires peut être requise dans la documentation afin de décrire le codage.

Des données en langage SDL peuvent être utilisées à la place de la notation ASN.1, si:

– l'ASN.1 n'est pas requise pour les définitions d'interface;

– le codage de base ASN.1 n'est pas à utiliser;

– le langage SDL offre un bon appui aux sortes de données requises.

Tableau 2/Suppl. 1 à la Rec. Z.100 – Notation ASN.1 pour l'établissement d'une commande d'appel en partage de ressources radioélectriques

```
CC-SETUP::=     SEQUENCE {
        pd      ProtocolDiscriminator;
        ti      TransactionIdentifier;
        mt      MessageType;
        pi      PortableIdentity OPTIONAL;          -- présent seulement si "appel entrant" est mis en œuvre
        fdi     FixedIdentity OPTIONAL;             -- présent seulement si "appel entrant" est mis en œuvre
        bs      BasicService OPTIONAL;              -- présent seulement si "appel entrant" est mis en œuvre
        ia      IWU_Attributes OPTIONAL;            -- obligatoire si le service de base indique "autre"
                                                    -- non autorisé si le service de base indique "attributs par
                                                    -- défaut"
        ri      RepeatIndicator OPTIONAL;           -- si ri apparaît avant les attributs d'appel et non après,
                                                    -- cela indique une liste prioritaire d'attributs d'appel
                                                    -- pour négociation

        cla     SEQUENCE SIZE(0..3) OF
                        CallAttribute OPTIONAL;

        ri      RepeatIndicator OPTIONAL;           -- si ri apparaît après les attributs d'appel et non avant


        cnal    SEQUENCE OF ConnectionAttribute OPTIONAL;
        cri     CipherInfo OPTIONAL;
        cni     ConnectionIdentity OPTIONAL;
        fy      Facility OPTIONAL;
        pi      ProgressIndicator OPTIONAL;         -- non autorisé si le signal est envoyé de P à F
        dy      Display OPTIONAL;                   -- non autorisé si le signal est envoyé de P à F
        kp      KeyPad OPTIONAL;                    -- non autorisé si le signal est envoyé de F à P
        sl      Signal OPTIONAL;                    -- non autorisé si le signal est envoyé de P à F
        fea     FeatureActivate OPTIONAL;           -- non autorisé si le signal est envoyé de F à P
        fei     FeatureIndicate OPTIONAL;           -- non autorisé si le signal est envoyé de P à F
        np      NetworkParameter OPTIONAL;          -- non autorisé si le signal est envoyé de F à P
        tc      TerminalCapability OPTIONAL;        -- non autorisé si le signal est envoyé de F à P
        eec     EndToEndCompatibility OPTIONAL;     -- obligatoire si le système utilise le débit LU6 (V.110/I.30)
        rp      RateParameters OPTIONAL;            -- obligatoire pour l'établissement d'un appel dans le
                                                    -- service d'adaptation de débit
        td      TransitDelay OPTIONAL;
        ws      WindowSize OPTIONAL;
        cgpn    CallingPartyNumber OPTIONAL;
        cdpn    CalledPartyNumber OPTIONAL;
        cds     CalledPartySubaddress OPTIONAL;
        sc      SendingComplete OPTIONAL;           -- obligatoire si l'on dispose de toutes les informations
                                                    -- nécessaires pour l'établissement de l'appel
        iti     IWU_to_IWU OPTIONAL;
        ip      IWU_Packet OPTIONAL
        }
```

S'il n'y a pas de raison de préférer les données en langage SDL, il convient d'utiliser des données en notation ASN. 1.

Le codage explicite est différé aussi longtemps que possible. Il peut parfois être effectué après la Formalisation. Le codage des messages ne modifie pas le comportement de la description formelle SDL mais s'applique aux interfaces de façon que les motifs binaires correspondent aux utilisations dans l'environnement.

Règle 10 – Il convient d'utiliser, chaque fois que possible, les "types utiles" de la notation ASN.1 et d'autres types déjà normalisés.

Règle 11 – Le codage doit être effectué séparément de la structuration et de la nomination des types.

Exemple

Voir le Tableau 2.

14.4 Modélisation par séquences d'utilisation

La modélisation par séquences d'utilisation a été commencée au cours de la phase d'Analyse par saisie du point de vue du seul utilisateur du système. A ce stade, l'on dispose d'un modèle SDL des flux de commande et de données dont la description servira de base à l'établissement d'un modèle détaillé par séquences d'utilisation.

Les tâches additionnelles visent principalement les opérations suivantes: l'utilisation d'entités SDL au lieu des classes du modèle par objets, l'utilisation de signaux SDL au lieu des événements associés au modèle par objets, l'utilisation des protocoles réels au lieu des échanges abstraits modélisés dans l'Analyse. Par exemple, dans le modèle de l'Analyse, une information produite par une entité extérieure a été modélisée sous la forme d'un unique message abstrait, tandis que lors de la Conception du projet, cette information sera modélisée par l'ensemble des interactions qui se produisent dans la réalité. Ces tâches additionnelles conduisent également à l'introduction de nouveaux diagrammes MSC détaillant les échanges abstraits.

Etant donné que les diagrammes MSC sont souvent utilisés pour supporter les étapes comportementales au cours de la Formalisation, la production de ces diagrammes peut être nécessaire dans le cadre de la Formalisation. Dans ce cas, l'on utilise cette étape, en combinaison avec l'étape du 13.3, pour produire les diagrammes MSC.

Instructions

1) suivre les instructions 1) à 5) du 13.3 pour produire les diagrammes MSC mais modéliser dans le système des instances MSC qui correspondent à des entités SDL (blocs ou processus ou services) internes au système;

2) contrôler la cohérence entre les diagrammes MSC de ce niveau et entre ceux qui sont contenus dans les informations classifiées;

3) produire (facultativement) des diagrammes MSC pour des interactions de niveau inférieur à l'intérieur des descriptions SDL et contrôler la cohérence interne des diagrammes MSC aux niveaux supérieurs.

Directives

Il convient que les interfaces avec l'environnement (ou les niveaux supérieurs) utilisent les zones latérales du diagramme. Si toutefois il y a au moins trois interfaces avec l'environnement, il est plus pratique de les représenter sous forme d'axes d'instance.

Il peut y avoir plusieurs instances de la même classe dans un diagramme de séquence de messages. Par exemple, il peut y avoir deux axes d'instance qui représentent chacun des instances différentes d'une commande d'appel ou d'interfaces avec l'environnement.

S'il n'y a qu'une seule instance de chaque classe ou de chaque élément SDL, les noms des instances doivent être, sur les diagrammes, les mêmes que ceux des classes ou éléments SDL correspondants. S'il y a plusieurs instances, celles-ci doivent avoir des noms uniques et la classe (ou l'élément SDL) est alors indiquée d'après sa sorte. Par exemple, dans la Figure 14-2, PTNX est une sorte d'instance et PTNX_A est un nom d'instance.

Habituellement, les messages ont déjà été identifiés lorsque les diagrammes MSC sont dessinés, soit en tant qu'aspects de comportement et d'interface des classes correspondant aux instances contenues dans les informations classifiées, soit en tant que signaux utilisés en SDL. La séquence des messages peut être identifiée d'après les aspects de comportement et d'interface des classes contenues dans les informations classifiées.

Il n'est pas essentiel de subdiviser les diagrammes de séquences de messages en diagrammes MSC simples, référencés à partir d'un même diagramme MSC principal. Mais cette opération est utile lorsque de nombreuses séquences ont le même "établissement" ou la même "libération". Lorsque le diagramme MSC se rapporte à des diagrammes MSC simples et que des conditions sont imposées, ces diagrammes devraient recevoir des noms significatifs tels que "appel_en_cours".

Le choix d'un ensemble logique de séquences est subjectif et fonction du système. Il n'est généralement pas possible de dessiner toutes les séquences de messages possibles pour un système particulier car leur nombre est très grand.

Règle 12 – Pour montrer dans des diagrammes la séquence des messages échangés entre des parties du système (avec l'environnement) en fonction du temps, on doit utiliser la notation des diagrammes MSC.

Exemple

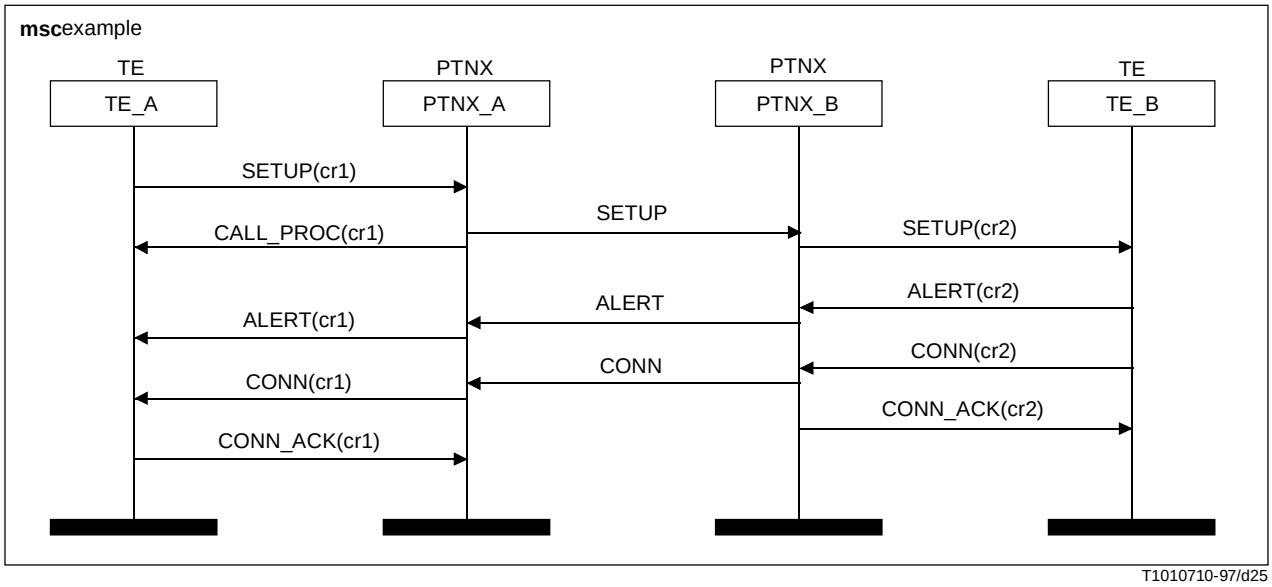


Figure 14-2/Suppl. 1 à la Rec. Z.100 – Envoi en bloc, appel établi

14.5 Modélisation du comportement d'un processus

Cette modélisation fait appel à l'usage informel de diagrammes de processus SDL afin de schématiser le comportement de processus. Au cours de l'activité de Conception du projet, cette étape n'est généralement utilisée que pour des processus critiques. Le comportement d'autres processus est déduit des séquences d'utilisation et d'autres descriptions. L'ensemble des signaux peut ne pas être défini formellement et l'on utilise un "texte informel" SDL plutôt que des expressions formelles pour formuler des tâches, des décisions et des paramètres.

Etant donné que cette modélisation peut être effectuée d'une manière similaire à la Formalisation, elle n'est pas décrite en détail dans la présente étape. La différence essentielle entre la Formalisation et la modélisation de processus lors de la Conception du projet est le niveau de formalité. Les diagrammes SDL de la Conception du projet peuvent être informels, incomplets, hétérogènes et non valides au point de vue SDL. Les règles de la Formalisation ne sont pas nécessairement applicables, ce qui n'empêche pas les diagrammes de fournir des vues utiles de l'application et de permettre de faire des économies.

14.6 Modélisation d'aperçu général d'états

Cette modélisation fait appel aux diagrammes SDL auxiliaires pour fournir des aperçus généraux du comportement du système. Ces diagrammes sont des types suivants: diagrammes arborescents, diagrammes d'aperçu général d'états et diagrammes matriciels de signaux/états (voir I.9.2/Z.100 à I.9.4/Z.100 [1]).

Il peut aussi être utile d'annoter les états contenus dans l'aperçu général (ou dans les schémas SDL du 14.5) avec les propriétés de ces états, exprimées sous la forme d'expressions logiques. Ces expressions peuvent être utiles lors de la définition des objectifs de test.

15 Etapes de Formalisation

NOTE – Dans toutes les étapes où un système, un bloc, un processus ou une procédure est défini, il convient de toujours rechercher un type existant qui pourrait être utilisé ou modifié. Cette directive est placée ici afin d'éviter sa répétition dans chaque étape où elle s'applique.

Si la notation de modélisation par objets (technique OMT) a été utilisée au cours de l'Analyse, le Tableau 3 donne des directives générales pour effectuer les transformations en SDL correspondantes.

Tableau 3/Suppl. 1 à la Rec. Z.100 – Transformations en SDL de la notation OMT de modélisation par objets

Notation de modélisation par objets	Transformation SDL possible
classe	bloc (type), processus (type), service (type), déclaration de données (ou nouvelle sorte newtype ou syntype)
classe abstraite	type de bloc, type de processus, type de service, nouvelle sorte (newtype) (ou syntype ou générateur) (de données)
relation d'instanciation	ligne de création de processus
attribut (de classe autre que de données)	déclaration de données d'une variable locale
attribut (d'une classe de données)	champ de structure ou deux opérateurs (lecture, écriture)
type d'attribut, type de paramètre	sorte de données de la construction SDL, dérivée de l'attribut
opération	procédure (locale, exportée, importée), définition de signal, opérateur
paramètre d'une opération	paramètre de (procédure ou de signal ou d'opérateur)
héritage simple	héritage
association, liaison	route de signal, canal
nom d'association	nom de (route de signal ou de canal)
multiplicité des agrégations	nombre d'instances de (blocs ou processus), tableau (de données)
nom de rôle	porte, nom de variable
agrégation	structure pour blocs et processus, variables locales dans un processus, deux opérateurs (ou une construction) à l'intérieur de données

Les constructions de la notation de modélisation par objets selon la technique OMT, non mentionnées ci-dessus, ne sont pas utilisées habituellement.

On peut trouver d'autres directives en [26] et [27].

Les étapes de l'activité de Formalisation se divisent en groupes pour Structure (étapes S), Comportement (étapes B), Données (étapes D), Types (étapes T) et Localisation (étapes L). Un groupe d'étapes ne doit pas être terminé avant le début d'un autre groupe. Il est courant que des travaux soient en cours dans plusieurs groupes en même temps. Par exemple, les étapes D peuvent être utilisées pour définir les paramètres des signaux en parallèle avec les étapes S ou les étapes B. De même, au fur et à mesure du raffinement du système, il est possible d'appliquer les étapes S à un même bloc pendant qu'un autre bloc est soumis aux étapes B, D ou T. Pour tirer pleinement parti des types exprimés en SDL-92, les étapes T et L devraient être appliquées en même temps que les étapes S et B.

15.1 Etapes relatives à la structure (étapes S, *S-steps*)

Les étapes relatives à la structure servent à définir en SDL les interfaces externes et internes du système ainsi que le partitionnement du système en blocs SDL.

15.1.1 Etape S:1 – Frontières et environnement du système

Instructions

1) identifier les frontières entre le système à décrire et son environnement;

2) trouver un nom approprié pour le système;

3) dessiner un diagramme de système SDL avec un nom identifié; expliquer de manière informelle le système et sa relation avec l'environnement, dans un commentaire inséré dans le diagramme SDL du système.

Directives

Cette étape peut paraître triviale mais l'importance de sa préparation ne devrait pas être sous-estimée. L'activité d'Analyse ou de Conception du projet, effectuée avant cette étape, aidera à choisir la frontière correcte ainsi qu'un nom approprié. Ne pas procéder à cette étape si l'on possède un niveau de compréhension suffisant des prescriptions pour le système.

Le nom du système et sa description commentée peuvent probablement être extraits directement d'une information classifiée. Le nom du système et sa description commentée apparaîtront dans la description formelle SDL: ils devraient donc être appropriés au système complet.

Les informations classifiées peuvent contenir des descriptions de classes, aussi bien à l'intérieur qu'à l'extérieur du système. Il peut y avoir une seule classe qui soit aisément identifiable comme étant appropriée au système SDL; ou il peut être nécessaire d'identifier un ensemble de classes qui sont mises en interface pour former le système.

Lorsque les informations classifiées contiennent un aspect comportemental pour chaque classe et qu'aucune interface n'a été identifiée comme étant liée à l'environnement, le système SDL peut être un système clos sans interfaces extérieures. Dans ce cas, le système SDL décrit le comportement ou l'objectif de l'application ainsi que les classes qui communiquent avec elle. Ces classes sont informatives.

Un "diagramme contextuel", produit au moyen d'une conception de projet en SDL informel (voir 14.2) et d'un diagramme MSC, peut être utile pour déterminer la frontière entre le système SDL et l'environnement. Lors de l'examen de la conformité d'un "diagramme contextuel" établi par conception de projet SDL, il convient de prêter particulièrement attention au nom du système et au commentaire descriptif. On étudiera également le degré d'intégration de l'environnement dans la description formelle SDL. Les interfaces normatives pour l'application peuvent devoir être des interfaces internes du système SDL.

Les diagrammes MSC pour le système sont souvent produits sous forme de conceptions de projet une fois cette étape franchie. Si des diagrammes MSC ont déjà été produits, il convient d'établir une distinction entre les axes correspondant aux entités considérées comme appartenant au système et les axes pour les entités appartenant à l'environnement extérieur. Ces dernières n'ont en effet aucune description formelle dans le système SDL (ou ne sont fournies qu'en tant que parties informatives du système). Il est possible de considérer le système comme étant composé de systèmes SDL en communication: mais, dans ce cas, le système devrait être décrit en tant que système SDL unique avec des blocs représentant les parties communicantes. Les systèmes SDL communicants peuvent être pris en considération lorsque les prescriptions collectées s'appliquent à un ensemble d'applications associées.

Voir s'il existe des types utiles dans une bibliothèque, se rapportant au système ou aux interfaces avec l'environnement, et pouvant être intégrés dans un module SDL.

Si le système comporte des temporisateurs, il faut définir l'unité de temps. Normalement, une unité sera égale à 1 ms ou à 1 s, ces valeurs étant appropriées. Il est également utile de définir des synonymes SDL pour des multiplicateurs utiles, tels que "secondes" et "heures".

Règle 13 – La frontière du système doit être identifiable par des communications assurées seulement au moyen d'un échange de messages discrets.

Règle 14 – La donnée relative à l'unité de temps devrait être enregistrée dans un commentaire du diagramme de système.

Résultat: le résultat est un nom significatif pour le système et une description contenue dans un commentaire.

15.1.2 Etape S:2 – Communications discrètes du système

Instructions

1) identifier le flux d'informations en termes de messages discrets échangés entre le système et son environnement;
2) modéliser ces messages par des signaux définis dans le système;
3) indiquer la relation entre chaque signal et des objets extérieurs au système;
4) indiquer la fonction de chaque signal dans un commentaire accompagnant la définition du signal;
5) grouper dans une liste (souvent tous) les signaux associés pour un même objet et un même sens;
6) inclure les définitions et listes de signaux dans le diagramme du système.

Directives

Les signaux à destination et en provenance de l'environnement peuvent habituellement être identifiés dans les informations classifiées comme étant des aspects d'interface en classe(s) extérieure(s). Les aspects informationnels associés décrivent le contenu des signaux.

Le "diagramme contextuel" ou les diagrammes MSC au niveau du système permettent d'identifier les signaux. Le nombre de signaux peut être diminué par qualification des signaux au moyen de paramètres (voir l'étape D:1). Les signaux pour communications externes sont définis au niveau du système et correspondent à des événements discrets entre le système et son environnement. Ces événements peuvent être bien définis dans le point de vue information, sous la forme d'unités de description de protocole ou de flux d'information.

Si la méthode des trois étapes (voir 13.1) est utilisée comme cadre et qu'une description d'étape 1 a été produite, les signaux nécessaires pour communiquer avec l'utilisateur sont ceux qui sont nécessaires pour communiquer avec l'environnement. Un comportement particulier est parfois attendu de l'utilisateur, auquel cas une partie (ou la totalité) du comportement de l'utilisateur peut être modélisée à l'intérieur du système.

Un système qui comprend le sujet de l'application et tous les objets communicants est un système fermé qui ne possède aucun signal pour communiquer avec son environnement.

Lorsqu'un signal paramétré est introduit, il convient d'identifier ou de nommer une sorte de données correspondante. La sorte de données devrait correspondre à un type ASN.1 nommé dans le point de vue information.

Les définitions de signaux et celles de la liste de signaux sont généralement trop vastes pour comporter un symbole textuel en première page SDL du diagramme de système contenu dans la documentation. Des pages supplémentaires devraient être ajoutées au diagramme de système pour introduire ces descriptions textuelles. Il convient de ne pas insérer dans le diagramme SDL de longues déclarations sur la relation entre les signaux et les classes externes, ou des descriptions contenant des dessins informels. De telles déclarations devraient faire l'objet de références à partir de la spécification SDL. Des déclarations plus courtes devraient être insérées, avec les définitions SDL appropriées.

NOTE – Il est recommandé d'insérer les parties textuelles du SDL dans des diagrammes SDL. Le fait d'insérer le texte SDL d'un symbole textuel dans un diagramme identifie clairement le texte comme étant SDL et identifie aussi le diagramme SDL auquel ce texte se rapporte.

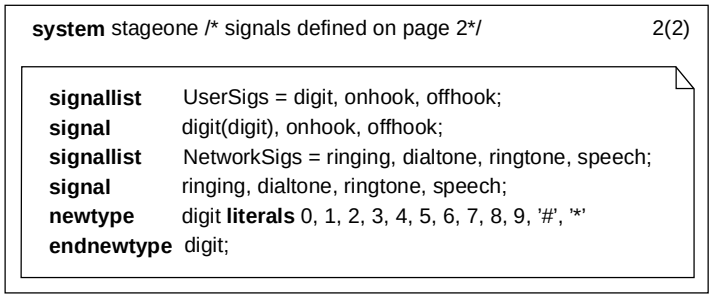
Pour faciliter la documentation, les définitions des listes de signaux sont placées avant celles des signaux figurant dans ces listes. Les définitions des données sont placées après celles des signaux. Les définitions sont groupées en éléments logiques, les définitions associées formant un seul symbole textuel.

Règle 15 – Les définitions textuelles d'un diagramme devraient être placées dans des symboles textuels de diagramme.

Règle 16 – Si des définitions textuelles occupent plus de 50 % d'un diagramme, celui-ci devrait être subdivisé en deux ou plusieurs pages où les définitions textuelles forment des groupes logiques dans des symboles textuels distincts.

Résultat: le résultat est "l'alphabet" du système. Les signaux définis au niveau du système communiquent avec l'environnement bien qu'ils puissent devoir être raffinés ultérieurement.

Exemple



NOTE – La notation ASN.1 n'a pas été utilisée pour le signal "digit" (chiffre de numérotation) parce qu'il est prévu que les noms des chiffres soient 0, 1, 2, etc.

Figure 15-1/Suppl. 1 à la Rec. Z.100 – Liste de signaux issus de l'étape S:2

15.1.3 Etape S:3 – Parties extérieurement identifiables

Instructions

1) identifier les principales parties contenues dans le système et les représenter sous la forme de blocs du système;

2) trouver un nom approprié pour chaque bloc et décrire informellement celui-ci, ainsi que ses relations avec son environnement (sa structure englobante), dans un commentaire contenu dans le bloc.

NOTE – Les étapes S:3 à S:6 sont utilisées de façon récurrente lorsqu'un bloc est décomposé en sous-blocs. Lorsque les étapes sont réappliquées, le "système" qui est examiné est en fait un bloc SDL et le terme "système" devrait être considéré comme remplacé par l'expression "bloc englobant". La différence entre un bloc et le système est que les signaux (listes de signaux et données) utilisés pour communiquer avec la structure environnante sont définis **à l'extérieur** du bloc pour un bloc et **à l'intérieur** du système pour un système.

Directives

Les informations classifiées contiennent des classes composantes qui correspondent aux blocs contenus (connectés par des canaux) ainsi que des définitions de type en SDL. Les classes composantes du système (ou ses blocs si cette étape est utilisée par récurrence) correspondent aux blocs et aux canaux. Certaines classes sans aspect comportemental peuvent représenter des objets contenus dans l'environnement.

Les éventuelles prescriptions architecturales, telles que la séparation du système en plans protocolaires distincts, ont pour objet de faciliter l'identification des éléments constituants du système.

Les blocs internes sont pris en charge au cours de cette étape et les canaux au cours de l'étape S:4. Les blocs sont des objets qui contiennent un aspect comportemental avec des états internes. Ces derniers peuvent correspondre aux données décrites dans l'aspect informationnel. L'ingénieur doit déterminer si une classe doit être décrite en SDL comme un type de bloc, un type de processus, une procédure, un signal, un temporisateur, ou une sorte de données. Dans la plupart des cas, le choix d'un type SDL correspondant à une information classifiée est évident. Mais dans certains cas il peut dépendre de la présentation abstraite prévue pour la spécification formelle. Par exemple, on peut décrire le comportement d'un objet en termes d'états (ou de processus, de procédure) ou en termes d'axiomes (d'opérateur ou d'axiomes contenus dans des définitions de nouvelles sortes). Dans ce cas particulier, une description en termes d'états est recommandée, sauf si le comportement correspond clairement à des fonctions abstraites de données.

La structure contenue dans les conceptions de projet peut aussi être utilisée pour déterminer la structure des blocs.

Les blocs délimitent le domaine de visibilité. C'est pourquoi les signaux, les sortes et les types utilisés seulement dans un bloc devraient être définis à l'intérieur de ce bloc, de façon que les informations soient cachées aux niveaux supérieurs et rendent ainsi ces niveaux plus faciles à comprendre. Ce principe s'applique également aux étapes S:6 et S:7 pour ces éléments. Il est exprimé dans la Règle 18 en tant que principe général pour le masquage d'informations.

Un bloc est "informatif" s'il ne possède aucun comportement qui doit être normatif. Le but d'un bloc informatif est de rendre le système complet, de façon que son fonctionnement:

– puisse être compris grâce à son usage et à son interaction avec ces parties informelles;

– soit exécuté et analysé en vue d'un produit de meilleure qualité et d'une meilleure validation d'appui.

Un bloc ne peut être informatif que si tous les blocs, processus et procédures (y compris les procédures distantes) qu'il englobe sont informatifs. Une fois qu'un bloc (ou un processus ou une procédure) est marqué comme étant informatif, cela implique que tous les blocs, processus et procédures englobés sont informatifs. Dans ce cas, il n'est pas nécessaire de marquer ces blocs, processus et procédures comme étant "informatifs".

Lorsqu'il existe un grand nombre de blocs directement englobés dans le système (ou dans un bloc), certains de ces blocs sont regroupés et encapsulés dans un bloc comme dans l'étape S:6.

NOTE – Le terme "informatif" n'a pas le même sens que le terme "informel". En description SDL formelle, ce sont les deux parties, informative et normative, qui devraient être formelles (c'est-à-dire exprimées formellement).

Règle 17 – Il ne doit pas y avoir plus de cinq blocs au niveau du système (ou directement englobés dans un bloc).

Règle 18 – Une définition doit avoir le plus petit domaine d'application qui englobe tous les usages de l'élément défini.

Règle 19 – Si un bloc (un processus ou une procédure) est informatif et n'est pas lui-même englobé dans un bloc informatif (un processus ou une procédure), ce bloc doit être assorti de l'annotation "informatif" dans le diagramme qui y fait référence ou dans son diagramme référencé ou aux deux endroits.

Résultat: le résultat est un diagramme contenant un ensemble de blocs et éventuellement l'identification des types pour ces blocs.

Exemple

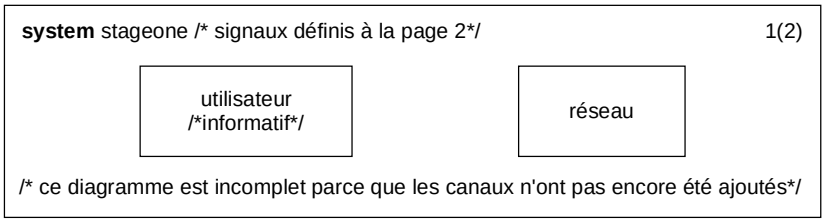


Figure 15-2/Suppl. 1 à la Rec. Z.100 – L'utilisateur et le réseau pour une description d'étape 1 selon la Rec. I.130 [7]

15.1.4 Etape S:4 – Trajet de communication entre parties

Instructions

1) identifier les canaux nécessaires entre les blocs et la frontière du diagramme, et entre les blocs à l'intérieur du diagramme;

2) pour chaque canal, identifier le ou les sens de communication;

3) associer une liste de signaux à chaque sens du canal;

4) choisir un nom de liste de signaux associé à la fonction et à l'usage de la communication.

Directives

La description relative à la façon dont les blocs internes d'un diagramme sont connectés fait partie des aspects d'interface de la classe qui correspond au diagramme dans les informations classifiées. Les canaux peuvent être représentés explicitement comme des classes contenues dans les informations classifiées, en particulier s'il y a des prescriptions spécifiques au sujet du canal. Un canal et sa ou ses listes de signaux associées définissent la signature d'une interface.

Les listes de signaux mentionnées à l'étape S:2 correspondent aux extrémités d'un canal à la frontière du système. Chaque point de la frontière représente une communication avec un objet externe différent (un canal ou un bloc externe au système SDL). Un ou plusieurs canaux peuvent connecter un tel point aux blocs indiqués dans le diagramme du système SDL. Chaque canal regroupe certains flux du comportement du système dans le "diagramme contextuel" contenu dans les conceptions de projet. Des prescriptions réalistes de modélisation, telles que des interfaces indépendantes, sont prises en considération.

Si le système contient un seul bloc, il est connecté à l'environnement par des canaux. Sinon, les blocs situés au niveau du diagramme sont également connectés par des canaux, selon les flux d'information échangés entre les blocs. Il n'est pas recommandé qu'un diagramme de blocs ne contienne qu'un seul bloc.

Des cas de communication typiques, représentés dans des diagrammes MSC, aident à identifier les canaux.

Les effets spéciaux qui dépendent du délai ou du non-délai de chaque canal ne devraient normalement pas être utilisés. Si un seul canal est utilisé entre deux blocs, les signaux envoyés à un seul bloc arrivent dans le même ordre à l'autre bloc. Les canaux sont censés avoir un délai, à moins qu'il n'y ait des prescriptions prescrivant une communication sans délai.

Si la communication sur un canal a besoin d'avoir des messages spécifiques avec un format et un codage spécifiques, conformes aux informations classifiées et aux prescriptions collectées, ce canal est "normatif". Les canaux qui sont internes au système et qui n'identifient pas une interface particulière pour des essais de produit ou d'autres fins sont habituellement informatifs. La communication par canaux informatifs contribue au comportement du système mais un autre système peut avoir des canaux ou des communications différents ayant le même comportement.

Règle 20 – Il ne devrait y avoir qu'un seul canal entre deux blocs.

Règle 21 – Chaque canal normatif doit être assorti du commentaire "normatif".

Résultat: le diagramme est défini avec un ensemble de trajets de communication correspondant aux interfaces externes et internes entre blocs directement englobés.

Exemple

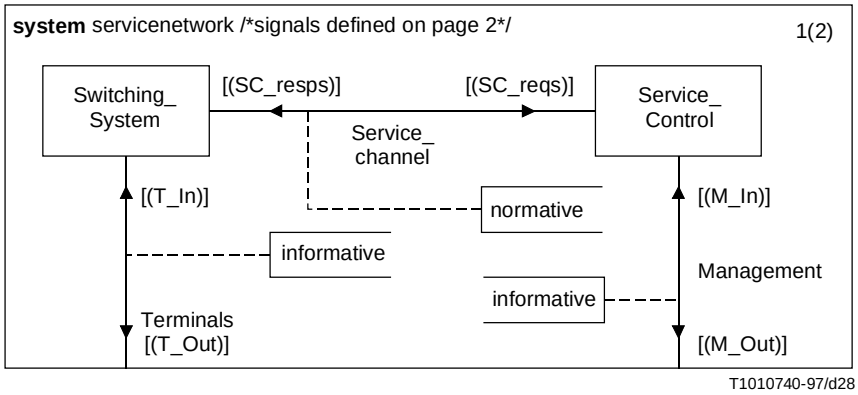


Figure 15-3/Suppl. 1 à la Rec. Z.100 – Modèle de réseau fournisseur de services

15.1.5 Etape S:5 – Signaux d'association aux trajets de communication

Instructions

1) pour chaque nom de liste de signaux, identifier les signaux appropriés;
2) nommer et définir chaque nouveau signal.

Directives

L'association de signaux aux listes de signaux utilisées entre les blocs peut nécessiter la définition de nouveaux signaux dans le diagramme contenant ces blocs.

La description correspondante se trouve dans les aspects d'interface contenus dans les informations classifiées ou (si les canaux sont décrits comme des objets partiels) dans les informations sur le signal, données par le canal correspondant dans les informations classifiées.

Si une conception de projet a été produite pour le diagramme SDL, il y a lieu de réexaminer chaque liste de signaux associée à un sens de transmission. Contrôler également les messages entre les axes appropriés des diagrammes MSC.

Voir s'il y a lieu de redéfinir la liste de signaux associée à un canal aux niveaux supérieurs, au moyen de la nouvelle liste de signaux. Cela peut améliorer la structure et la clarté du SDL. La définition de la liste de signaux doit se trouver à l'intérieur du diagramme de niveau le plus élevé dans lequel elle est utilisée.

Bien que le langage SDL autorise la référence directe à des signaux dans un symbole de liste de signaux (c'est-à-dire […]) pour un canal dans un diagramme, il n'est pas recommandé d'énumérer les signaux dans le symbole. La liste des signaux est habituellement requise à plusieurs endroits et il est plus facile de la modifier si elle n'est définie qu'une seule fois dans une liste de signaux.

Règle 22 – Un maximum de trois signaux (ou de listes de signaux) devrait être énuméré dans un symbole de liste de signaux, au lieu d'utiliser une liste de signaux attachée au canal.

Règle 23 – Les listes de signaux, les signaux et les données utilisés dans la communication externe d'un système devraient être définis dans un ou plusieurs symboles textuels distincts des autres définitions.

Résultat: les signaux sont associés à des noms de listes de signaux (c'est-à-dire, indirectement, à des canaux).

15.1.6 Etape S:6 – Masquage et sous-structuration d'informations

Instructions

1) examiner tour à tour chaque bloc contenu dans le diagramme courant et décider si ce bloc doit contenir d'autres blocs ou des processus;
2) si le bloc doit contenir d'autres blocs, appliquer récursivement à ce bloc la séquence d'étapes S:3 à S:6 en le considérant comme un (sous-)système autonome et en introduisant les nouveaux signaux, blocs, canaux et listes de signaux;
3) si le bloc ne doit pas contenir d'autres blocs, appliquer l'étape S:7 pour subdiviser ce bloc en processus.

Directives

Si le système est grand, certains blocs peuvent être considérés comme formant un système autonome et peuvent être encore subdivisés conformément aux règles indiquées pour le système. La description qui en résulte comporte alors des blocs emboîtés. Chaque bloc peut alors être élaboré comme décrit ci-dessus. Cette étape peut être issue de l'encapsulation de classes dans les informations classifiées. Une classe dont le comportement est l'aspect principal et qui n'est pas resubdivisée est susceptible d'être un processus et l'objet qui l'englobe directement est donc un bloc. Une classe qui contient des classes resubdivisées est habituellement un bloc à subdiviser.

On ne voit pas toujours clairement si le contenu d'un bloc devrait être des blocs ou des processus. Dans ce cas, une subdivision en blocs est d'abord tentée. S'il est difficile de subdiviser le bloc, celui-ci est probablement un processus.

NOTE – Un bloc de type "**block** inner" directement englobé dans un bloc de type "**block** outer" est une notation abrégée pour un "**block** inner" directement englobé dans une "**substructure** outer" elle-même englobée dans un "**block** outer" vide par ailleurs. Voir la Figure 15-4.

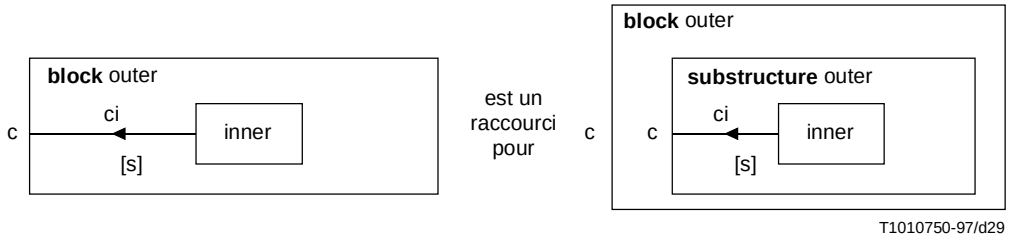


Figure 15-4/Suppl. 1 à la Rec. Z.100 – Notation abrégée d'englobement de blocs

Dans certains cas, un processus devra être encapsulé dans un bloc pour s'intégrer à un diagramme.

Le langage SDL permet également à un bloc de contenir à la fois des descriptions de processus et une substructure contenant d'autres blocs. Il peut être utile d'utiliser cette capacité au cours de la Conception du projet mais il n'est pas recommandé de l'employer pour la Formalisation.

L'application récurrente de cette étape produit un certain nombre de niveaux. Même si le système est complexe, il ne devrait pas y avoir plus de trois niveaux de blocs. Si chaque niveau a trois à quatre blocs avec deux à cinq processus dans chaque bloc-feuille, il y aura plus de 100 processus dans l'ensemble du système SDL.

Les blocs englobés devront être représentés sous la forme de références de bloc plutôt que de diagrammes de blocs parce que les diagrammes à imbrication directe deviennent rapidement trop grands et trop complexes pour pouvoir être gérés ou compris. Le même principe s'applique à tous les diagrammes (de blocs, de processus, de procédure ou de type).

S'il y a plus de cinq définitions évidentes de processus dans un bloc, celui-ci devrait être subdivisé en plusieurs blocs contenant moins de processus. Les interactions entre axes de diagramme MSC correspondant à des processus peuvent aider à identifier des "grappes" naturelles de processus pour chaque bloc non partitionné.

Il y a lieu de dessiner un diagramme arborescent pour les blocs.

Règle 24 – Les diagrammes devraient être imbriqués par références plutôt que par englobement direct.

Règle 25 – Le nombre de canaux issus de chaque bloc ne devrait pas dépasser cinq.

Résultat: le résultat est un ensemble de diagrammes de blocs et un arbre de blocs dont la racine est le système et dont chaque feuille est un bloc non resubdivisé en blocs.

15.1.7 Etape S:7 – Constituants de bloc

Instructions

1) identifier les objets séparés dont le comportement doit être subdivisé en processus pour le bloc choisi à l'étape S:6; définir ces objets comme étant les processus de ce bloc;

2) trouver un nom approprié pour chaque processus et le décrire informellement, ainsi que ses relations avec son environnement (qui est le bloc englobant), dans un commentaire situé à l'intérieur de la référence au processus;

3) définir, pour chaque processus, son nombre initial et son nombre maximal d'instances;

4) utiliser des trajets de signaux pour connecter les ensembles de processus à des canaux situés à la frontière du bloc.

Directives

Les trajets de signaux dans le bloc peuvent être déterminés de façon analogue aux canaux dans les étapes S:3 et S:4.

Si un processus a été encapsulé dans un bloc "fictif", le bloc non subdivisé contient une description de processus d'un seul type seulement et la relation avec son environnement est déterminée d'après l'interface externe du bloc.

Examiner les modèles par objets contenant des associations entre classes correspondant à des processus. La multiplicité propre (1 à 1, 1 à plusieurs, plusieurs à plusieurs) de l'association aide à définir le nombre initial et le nombre maximal d'instances pour ces processus.

Règle 26 – Pour chaque bloc, au moins un processus doit avoir un nombre initial d'instances supérieur à zéro, de façon qu'il puisse créer d'autres instances dans ce bloc.

Règle 27 – Le nombre de définitions de processus dans chaque bloc ne devrait pas être supérieur à cinq.

Résultat: identification des processus à l'intérieur des blocs-feuilles de l'arbre de blocs.

15.1.8 Etape S:8 – Signaux locaux dans un bloc

Instructions

1) identifier les signaux entre les processus locaux du bloc;
2) définir au niveau du bloc tous ceux de ces signaux qui sont additionnels (non définis comme étant externes au bloc);
3) identifier dans le bloc toute procédure de processus importée ou exportée, ainsi que la définition de procédure distante correspondante; définir ces signaux au niveau du bloc (ou du type de bloc).

Directives

La description correspondante se trouve dans les aspects d'interface internes de la classe correspondant au bloc englobant. Si les chemins du signal sont décrits sous la forme d'objets partiels, des informations sur ce signal ou sur son chemin sont données par le chemin du signal local correspondant dans les informations classifiées. Dans le cas spécial d'un bloc "fictif" (bloc contenant un seul processus), il n'y aura pas de signaux additionnels au niveau du bloc.

Le langage SDL autorise trois manières (autres que la transmission de signal) pour utiliser dans un processus la valeur contenue dans les variables de données d'un autre processus: par vue-extraction, par importation-exportation et par appel d'une procédure distante située dans l'autre processus. Tous ces mécanismes laissent la possibilité de concevoir un système contenant des erreurs inhérentes. Ils doivent être utilisés avec précaution et il est recommandé de simuler leur interaction. Des procédures distantes sont permises. La vue-extraction peut conduire à un comportement non déterministe. L'importation-exportation peut être remplacée par des appels de procédure distante.

Règle 28 – La vue-extraction ne doit pas être utilisée.

Résultat: le résultat est une définition de chaque signal et de la procédure distante au niveau du (type de) bloc.

15.2 Etapes relatives au comportement (étapes B, *B-steps*)

Ces étapes consistent à décrire le comportement de composants en termes de communication mais sans fournir de définition formelle des données couvertes par les étapes de données. Le présent sous-paragraphe décrit ces étapes pour un processus (**process**) SDL, mais aussi bien ces étapes que les étapes de données sont également d'application générale à la définition du comportement d'une procédure (**procedure**) SDL.

15.2.1 Etape B:1 – Ensemble de signaux d'un processus

Instructions

Identifier l'alphabet d'entrée dans le processus, appelé signalset (ensemble de signaux) du processus. A cette fin, on identifie:

– tout signal qui peut être reçu par le processus (soit une "annonce" en terminologie de traitement ODP), auquel le processus peut envoyer ou ne pas envoyer une réponse;
– toute procédure exportée du processus. Il s'agit d'un cas où le processus remplit le rôle de serveur dans un modèle client-serveur (soit une "interrogation" en terminologie de traitement ODP), où le signal correspondant est implicite mais où le nom de la procédure distante peut être considéré comme faisant partie de l'alphabet.

Directives

Bien que l'ensemble de signaux puisse habituellement être simplement dérivé du diagramme de blocs englobant, le but de cette étape est de passer en revue l'ensemble de signaux en ne prenant en compte que le processus courant. S'il est décidé de modifier l'ensemble de signaux à partir d'autres processus, les diagrammes de blocs correspondants doivent être modifiés. Les processus qui envoient les signaux auront besoin d'être mis à jour (probablement à un autre moment), s'ils ont déjà été formalisés. Une source de signaux non représentée sur le diagramme de blocs est le processus d'envoi de signaux vers la même source ou vers d'autres instances de la même définition de processus.

L'ensemble de signaux est utilisé lors de la détermination du comportement du processus dans chaque état de l'étape B:4. Il est suggéré de conserver un enregistrement de l'ensemble de signaux si cette opération ne peut pas être effectuée automatiquement au moyen des outils utilisés pour le SDL.

Ce sont les signaux du processus englobant qui sont utilisés pour une procédure; mais de nouveaux signaux peuvent être identifiés: ils doivent alors être ajoutés à l'ensemble de signaux du processus (identifié dans cette étape mais pour le processus englobant).

Résultat: le résultat est la définition de l'ensemble de signaux et l'ensemble de procédures exportées pour chaque processus.

15.2.2 Etape B:2 – Processus squelettes

Instructions

1) produire des diagrammes MSC pour au moins les cas d'utilisation "typiques" si ces derniers n'ont pas été produits sous forme de conceptions de projet;
2) produire un processus squelette par mappage des diagrammes MSC, en ne tenant compte que des utilisations "typiques":
   - 2.1 à partir du symbole de départ du processus, construire une arborescence des états sur la base des changements "normaux" d'état du processus;
   - 2.2 construire l'arbre de processus par développement du niveau de chaque état sur la base de chaque entrée qui est consommée mais non ignorée dans le diagramme MSC en la faisant suivre d'une transition (y compris les sorties et les créations) vers les différents états;
   - 2.3 identifier un état comme étant différent des autres états s'il possède un ensemble de signaux différent qu'il consomme ou qu'il conserve, ou s'il a une réponse différente à un signal;
   - 2.4 inclure la supervision temporelle (**set** et **reset**) ainsi que l'entrée de temporisation correspondante;
   - 2.5 lors de la représentation graphique de l'arborescence, indiquer les endroits où le processus revient au même état et transformer l'arbre en graphe;
3) représenter ce graphe sous la forme d'un diagramme de processus;
4) déterminer si le processus possède deux ou plus de deux ensembles intersectés d'interfaces pour différentes phases comportementales du processus et si ces ensembles peuvent être entrelacés; puis, si les comportements sont indépendants, subdiviser le processus en plusieurs sous-processus.

NOTE – Si les comportements sont couplés par l'utilisation de données communes, la subdivision du processus en sous-processus nécessite qu'une ou plusieurs procédures distantes accèdent aux données. Il est également possible de subdiviser un tel processus en services SDL.

Directives

Les diagrammes MSC sont produits dans le cadre de l'activité de Conception du projet.

Les diagrammes MSC peuvent conduire semi-automatiquement à un processus squelette. L'ordre des événements sur l'axe vertical du diagramme MSC correspond à un ordonnancement des événements dans un processus pour lequel un squelette peut être produit. Pratiquement, on effectue cela en prenant des cas d'utilisation typiques dans le diagramme MSC puis en rédigeant la définition du processus d'après ces cas d'utilisation (les "cas simples et courants"). La création dynamique de processus peut aussi être déduite du diagramme MSC.

La supervision temporelle peut être indiquée sur les diagrammes MSC. Elle sert à modéliser une durée, à superviser la libération d'une ressource et à surveiller les réponses provenant de sources peu fiables.

Quelquefois, les diagrammes MSC montrent un message adressé à un processus, qui nécessite une réponse fondée sur des informations non accessibles au processus ou non contenues dans le message. Si le processus demandeur possède ces informations, la solution consiste généralement à s'assurer qu'elles sont insérées dans un paramètre de signal (et à

mettre à jour les définitions, le cas échéant). Il peut s'agir du message de demande ou d'un message antérieurement associé. Si les informations se trouvent dans un autre processus ou sont extérieures au système, le processus doit communiquer avec le détenteur de ces informations au moyen de signaux ou d'une procédure distante. Mettre à jour les diagrammes MSC ou les marquer comme étant incomplets et simplifiés.

Un processus squelette est censé représenter le comportement essentiel d'un processus mais non pas ce processus complet. D'autres conceptions de projet (et les diagrammes MSC) devraient être comparées au processus squelette afin de contrôler la cohérence du processus avec ces autres conceptions.

Deux processus se trouvant respectivement dans les états N et M, peuvent être combinés en un seul processus ayant les états N × M. La subdivision d'un tel processus produit donc deux sous-processus beaucoup plus simples et donc plus faciles à comprendre.

Lorsqu'un processus (ou une procédure) est informatif, il peut y avoir des transitions spontanées commençant par l'absence d'état **none**, qui modélisent le comportement de l'utilisateur ou d'autres événements imprévisibles (voir par exemple la Figure 15-5).

Règle 29 – Les transitions spontanées ne devraient pas être utilisées dans les parties normatives de la norme.

Règle 30 – Le diagramme MSC doit soit refléter correctement le traitement des messages par le système SDL ou doit être clairement annoté de la façon et de la raison de leurs différences avec le comportement SDL.

Résultat: le résultat est un processus squelette ayant la forme d'un diagramme d'aperçu général d'états comprenant la temporisation, avec les diagrammes MSC normaux et compatibles.

Exemple

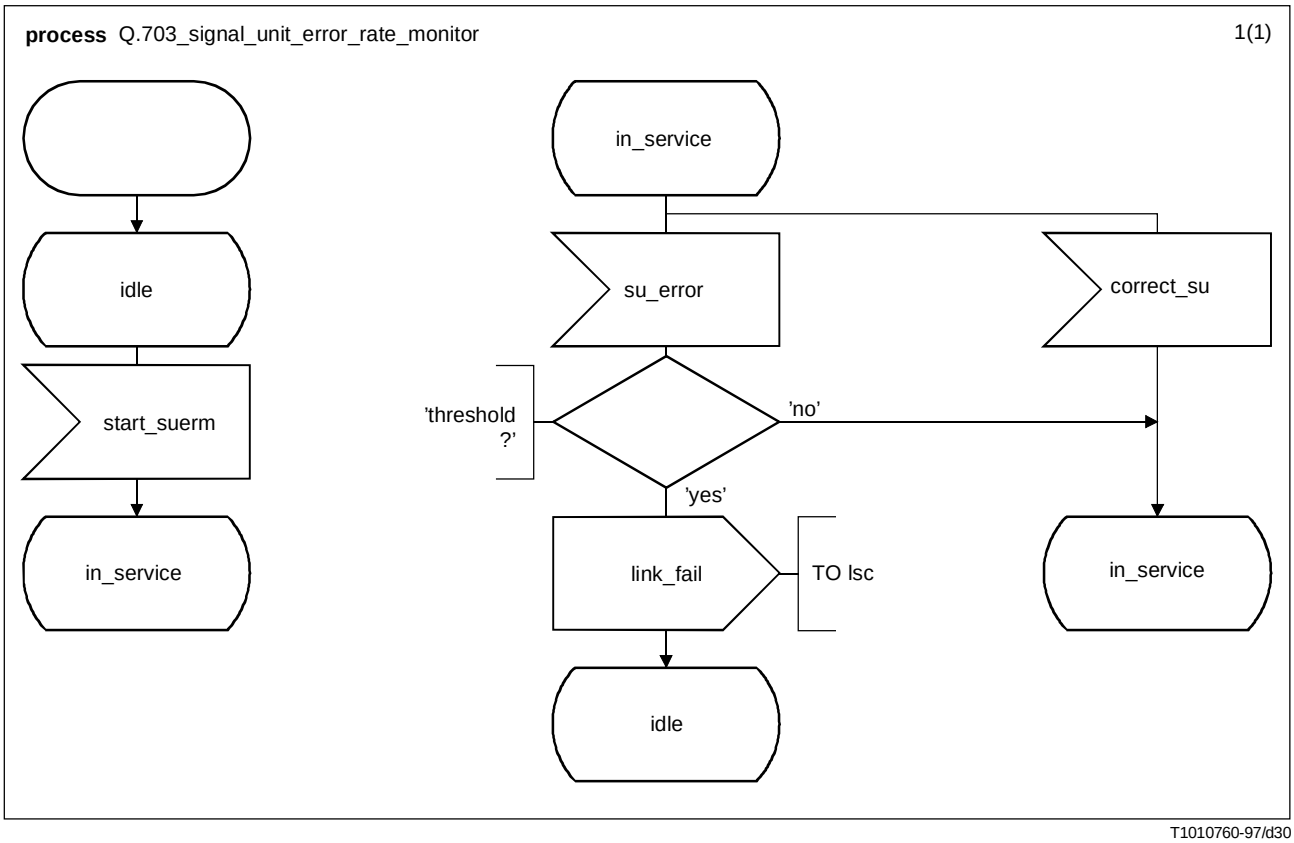


Figure 15-5/Suppl. 1 à la Rec. Z.100 – Exemple de processus squelette

15.2.3 Etape B:3 – Processus informels

Instructions

1) identifier les combinaisons d'utilisations;
2) identifier les informations mémorisées par le processus et examiner si cela est implicite dans les états du processus ou s'il faut des données internes;
3) utiliser ces informations pour définir les actions internes de chaque processus;

4) ajouter aux transitions des tâches ou des procédures et éventuellement d'autres décisions; mais n'utiliser que des textes informels dans les tâches et les décisions;

5) si une procédure est utilisée, lui donner un nom significatif et utiliser (ultérieurement) les étapes B:1 à D:9 pour la définir.

Directives

Utiliser les diagrammes MSC existants et produire de nouveaux diagrammes MSC pour identifier les combinaisons d'utilisations.

Déterminer s'il y a lieu d'utiliser une tâche ou une procédure pour décrire le traitement des informations dans une transition, bien que ce traitement puisse être modifié ultérieurement. Utiliser une procédure s'il est prévu que ces informations soient extraites d'un autre processus ou si la tâche est susceptible d'être complexe ou de dépendre de plusieurs valeurs de données enregistrées; sinon, choisir une tâche. Si une procédure est choisie, il peut être judicieux d'en examiner les paramètres (nombre et sortes).

Il est parfois évident que les mêmes actions sont effectuées à plusieurs points du processus. Ces actions peuvent être collectées et transformées en procédure.

Résultat: le résultat est un processus informel qui couvre tous les cas décrits dans le diagramme MSC.

Exemple

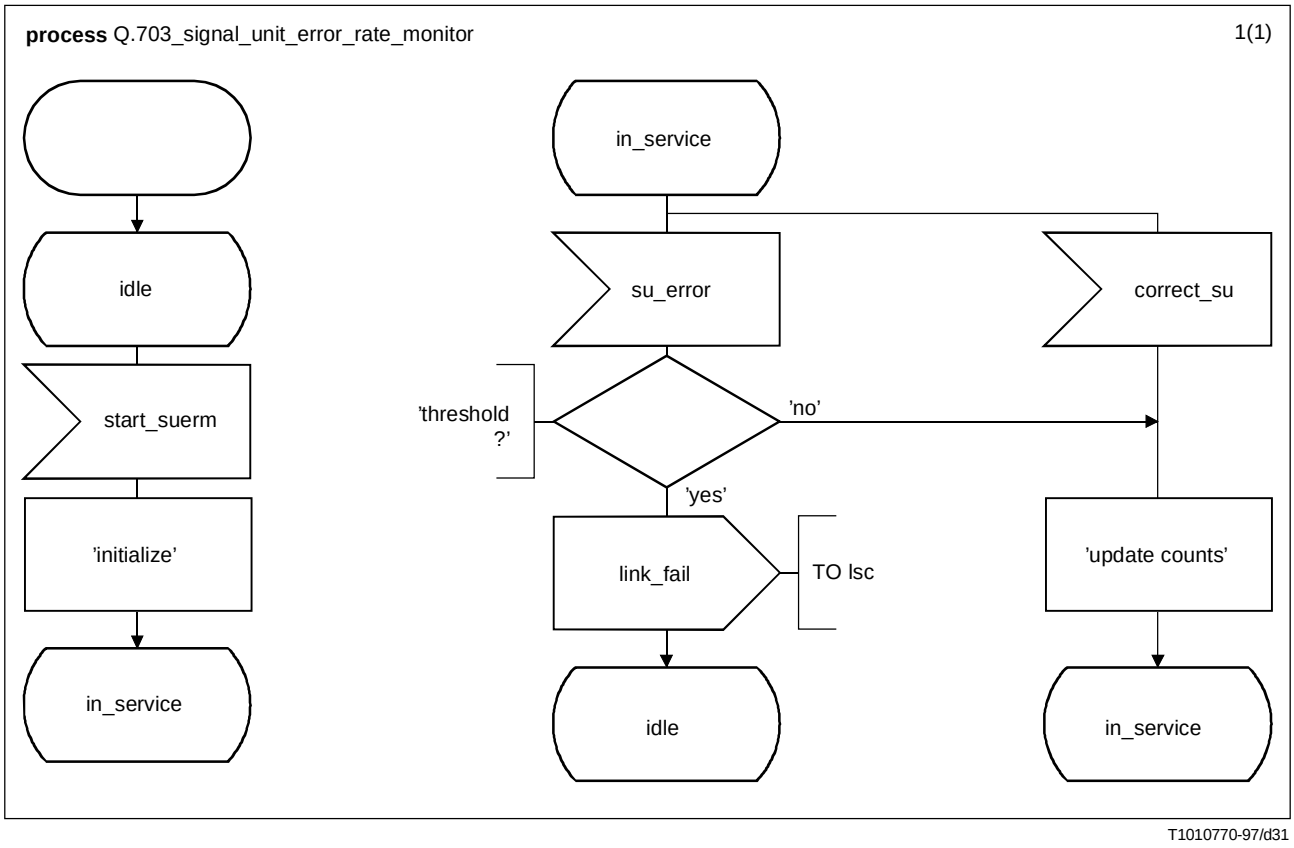


Figure 15-6/Suppl. 1 à la Rec. Z.100 – Version informelle du même exemple que dans l'étape B:2

15.2.4 Etape B:4 – Processus complets

Instructions

1) identifier à chaque état et pour chaque élément de l'ensemble de signaux (signal ou procédure distante) si le signal (ou l'appel de procédure distante) est introduit par le processus ou sauvegardé;

2) si l'élément est introduit, déterminer la transition effectuée (il peut s'agir d'une transition de retour implicite au même état);

3) poursuivre l'instruction 1) et l'instruction 2) jusqu'à ce qu'il n'y ait plus d'états à la fin d'une transition qui n'aient pas été examinés, de façon que tous les éléments de l'ensemble de signaux aient été examinés pour chaque état;

4) analyser d'abord chaque processus puis les combinaisons en remontant jusqu'au système SDL entier afin de contrôler l'absence de propriétés indésirables et revoir au besoin la conception pour les éviter.

Directives

Une matrice signaux/états (voir I.9.4/Z.100 [1]) peut être utilisée pour contrôler l'action effectuée pour chaque signal dans chaque état. Lorsqu'un nouvel état est identifié, la matrice est étendue. Cette matrice peut également faciliter l'identification du moment où deux des états définis conduisent au même comportement et peuvent être combinés, ou du moment où deux signaux ont les mêmes états suivants. Dans ces cas, il est parfois possible de réduire le nombre de signaux ou d'états.

Un diagramme d'aperçu général d'états (voir I.9.3/Z.100 [1]) peut être représenté graphiquement pour donner un aperçu général du comportement du processus. Si celui-ci n'est pas destiné normalement à se terminer, chaque état devra généralement être accessible à partir de tout autre état; mais ce ne sera pas toujours nécessaire car il peut y avoir des états initiaux pour le démarrage du processus et des états finals pour sa terminaison.

Il existe une limite au nombre de recherches réalisables sur un même processus. Les résultats des recherches sont d'autant plus intéressants que l'on peut analyser plus de processus "complets" en combinaison dans un bloc ou en tant que système entier. Les propriétés indésirables qui peuvent être détectées (telles que les états inaccessibles, les blocages à froid et à chaud) dépendront de la complexité du système et des outils disponibles.

Pour les explorations, on peut partir du principe que chaque décision contient la construction SDL "**any** value" (toute valeur).

La recherche d'autre chose qu'un processus simple est difficile sans outils pour produire l'espace des états puis pour le contrôler. Même les outils ont des difficultés avec de grands nombres de processus ou avec des processus très complexes. Il ne s'agit donc pas d'une étape triviale mais la recherche à ce stade économise beaucoup de temps perdu et d'efforts si une reconception est jugée nécessaire: c'est là l'un des principaux avantages de l'utilisation du langage SDL.

Résultat: à ce stade, la définition du processus (de la procédure ou du service) est complète mais informelle.

Règle 31 – Tous les états contenus dans un processus doivent être accessibles dès le début de ce processus.

Règle 32 – Une procédure qui est exportée par un processus (pour utilisation comme procédure distante) ne doit pas être sauvegardée dans chaque état de ce processus.

Règle 33 – Chaque signal reçu par le processus devrait avoir au moins une entrée conduisant à une transition non vide.

15.3 Etapes relatives aux données (étapes D)

La fonction des étapes relatives aux données est de fournir une définition formelle des données utilisées dans les processus. Sans données formelles, une décision quelconque quant au comportement et aux opérateurs utilisés dans des expressions donne des résultats incertains.

Les deux premières étapes (D:1 à D:2) concernent les valeurs d'interface. Les trois étapes suivantes (D:3 à D:5) concernent des variables contenues dans des processus. Les étapes restantes (D:6 à D:9) concernent la définition formelle des nouvelles sortes de données. Ces dernières étapes sont requises si de nouvelles sortes de données ont de nouveaux opérateurs. C'est pourquoi les étapes D:6 à D:9 ne sont qu'occasionnellement nécessaires et devraient être inutiles si la notation ASN.1 a été utilisée pour définir les données.

L'étape D:1 s'applique au système dans son ensemble, tandis que les étapes D:2 à D:9 peuvent s'appliquer à chaque processus ou à chaque procédure.

Règle 34 – Les sortes de données utilisées doivent être définies au moyen du langage SDL combiné avec la notation ASN.1 (comme défini dans la Recommandation Z.105 [2]).

15.3.1 Etape D:1 – Paramètres des signaux

Instructions

1) identifier les valeurs à acheminer par des signaux, en commençant par les signaux au niveau du système;

2) rechercher des sortes de données prédéfinies afin de représenter les valeurs identifiées;

3) étendre les définitions des signaux avec les sortes de données;

4) identifier et définir, au besoin, de nouvelles sortes de données.

Exemple

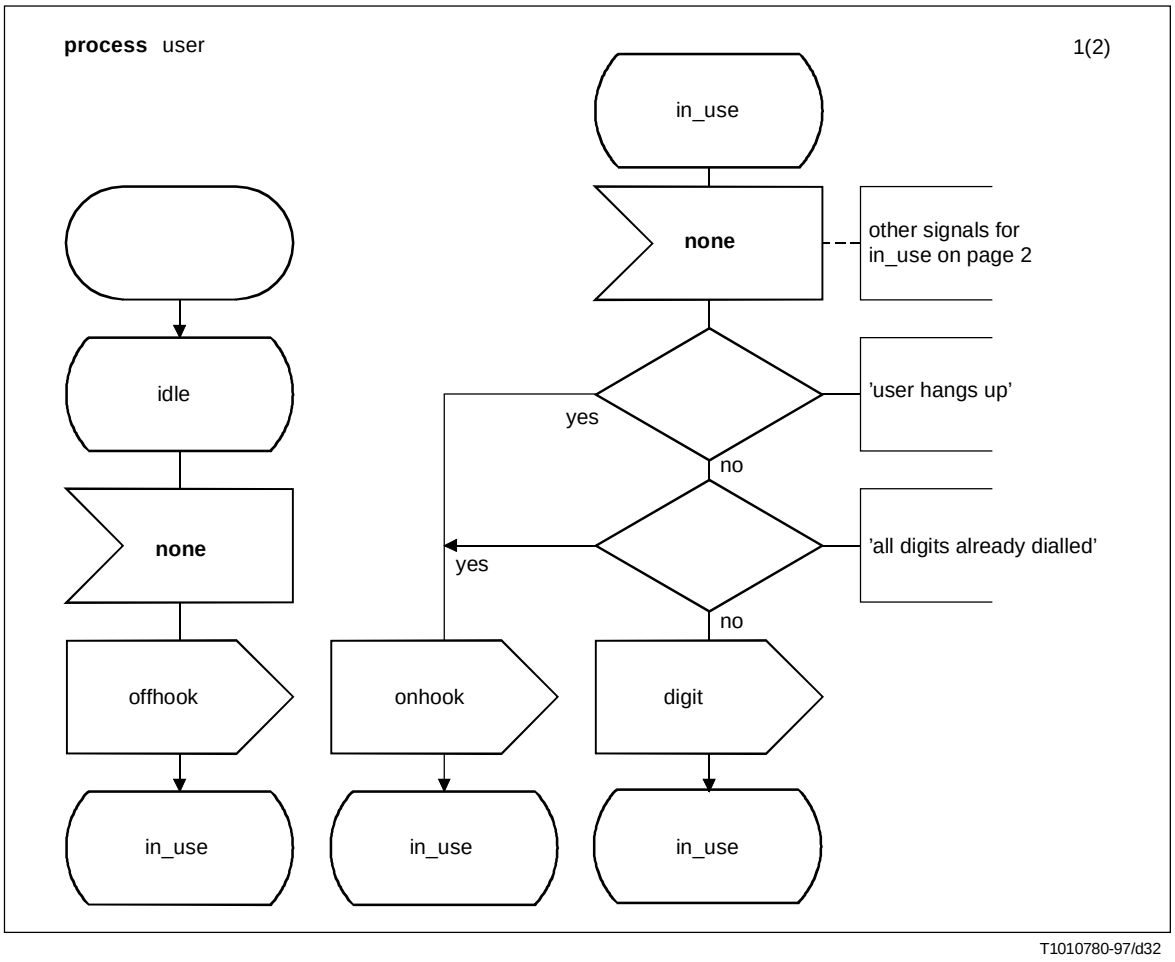


Figure 15-7/Suppl. 1 à la Rec. Z.100 – Partie d'un processus complet mais informel

Directives

La description de la communication prévue, figurant dans les informations classifiées, constitue une source pour les noms de sortes de données.

La notation ASN.1 permet de décrire formellement les données utilisées pour les signaux. Par conséquent, la sorte de données correspondante peut être utilisée directement. Si la notation ASN.1 n'a pas été utilisée, une décision doit être prise pour savoir s'il convient de définir les sortes de données en notation ASN.1 ou en langage SDL. La notation ASN.1 devrait être utilisée si le codage des données est important ou si les données se trouvent à une interface avec l'environnement. Le langage SDL peut être utilisé si aucun codage particulier n'est requis et si la sorte de données n'est utilisée que dans le système SDL.

Règle 35 – Si des tableaux binaires ont été utilisés pour définir des données, celles-ci doivent être converties en ASN.1 ou en SDL.

Règle 36 – Les noms des sortes de données existantes ne devraient pas être utilisés pour de nouvelles sortes de données, même si elles ont des domaines de visibilité différents.

NOTE – Si les sortes de données ont le même domaine de visibilité, les règles du langage SDL ne permettent pas que les noms soient identiques.

Résultat: les signaux ont des paramètres contenant les sortes définies de données correspondantes.

Exemple

Conformément à la Figure 15-5, le signal "digit" peut être étendu à "digit(digit)", où la sorte de données est définie comme indiqué dans la Figure 15-1.

15.3.2 Etape D:2 – Paramètres des processus et procédures

Instructions

1) identifier les sortes de données requises pour les paramètres du processus (ou de la procédure);

2) indiquer le rôle de chaque paramètre dans un commentaire figurant en tête du processus (ou de la procédure).

Directives

Dans un processus, un paramètre est traité comme une variable. La seule différence entre un paramètre de processus et une variable ayant une valeur par défaut est qu'un paramètre de processus peut recevoir une valeur différente pour chaque instance du processus. Normalement, cette valeur est l'identité d'une autre entité, par exemple un identificateur Pid ou un numéro d'équipement.

Les informations classifiées et les conceptions de projet concernant la création et la suppression donnent des indications sur le rôle des paramètres à transmettre à un processus lors de sa création. L'invocation d'une procédure peut également correspondre à une création (et le retour de procédure à une suppression).

Règle 37 – Un paramètre de processus ne devrait pas contenir l'identificateur Pid parent car celui-ci est disponible en tant que mot ***parent****.*

Résultat: les paramètres du processus (ou de la procédure) ont été formalisés.

15.3.3 Etape D:3 – Paramètres d'entrée

Instructions

1) ajouter aux entrées des paramètres conformes aux définitions de signal;

2) définir les variables selon les nécessités pour recevoir les valeurs d'entrée;

3) indiquer dans un commentaire le rôle de chaque variable.

Directives

Contrôler que chaque paramètre défini dans une définition de signal est utilisé dans au moins une entrée ou est requis pour la communication avec l'environnement.

Règle 38 – Un paramètre de signal ne devrait pas contenir l'identificateur Pid de l'expéditeur car cette identification est disponible en tant que mot ***sender****.*

Résultat: les interfaces d'entrée ont été formalisées.

15.3.4 Etape D:4 – Transitions formelles

Instructions

1) remplacer le texte informel, se trouvant dans les tâches, dans les décisions et dans les réponses, par des attributions formelles, des expressions formelles, des expressions d'étendue formelle et par des appels de procédure; identifier les nouveaux opérateurs éventuellement utilisés dans les expressions formelles à ajouter aux sortes de données dans l'étape D:6;

2) définir des variables et synonymes additionnels, selon les besoins;

3) définir des procédures additionnelles, selon les besoins;

4) ajouter des paramètres aux appels de procédure conformément aux définitions de procédure.

Directives

Le texte informel assortissant antérieurement les symboles peut être utile sous forme de commentaires joints aux symboles.

Si la valeur d'une fonction utilisée dans une expression ne dépend que des paramètres réels, le choix existe de transformer cette fonction en opérateur ou en procédure. Si le résultat dépend d'autres données, ce doit être une procédure. Si le comportement de la fonction dépend de données issues d'un autre processus, il peut être opportun d'en faire une procédure distante ou de décider de la façon d'obtenir ces informations. L'obtention d'informations peut nécessiter d'ajouter d'autres paramètres à des signaux existants et de mémoriser dans la procédure ces informations ou cette communication, éventuellement avec d'autres signaux.

La définition d'une procédure est traitée de la même manière qu'un processus dans les étapes B:1 à D:9, y compris les possibilités d'appel interne d'une procédure et d'imbrication d'une procédure dans une autre procédure.

Lorsque le comportement normatif est indépendant de certaines données dans des processus ou procédures informatifs, la construction **any** peut être utilisée pour créer des valeurs aléatoires pour ces données ou des décisions aléatoires dans des processus ou procédures informatifs. De cette façon, les parties informatives peuvent modéliser des situations dans lesquelles les données et le comportement sont imprévisibles. Des transitions spontanées sont utilisées si c'est l'instant où un événement se produit ("quand") qui est imprévisible (voir 15.2.2) et une valeur de type **any** est utilisée si c'est l'événement qui est imprévisible.

*Règle 39 – Une valeur ou une décision qui n'est pas déterministe (utilisant **any**) ne doit pas apparaître dans une partie normative du système sauf si elle est explicitement marquée comme étant informative.*

*Règle 40 – Une valeur ou une décision qui n'est pas déterministe (utilisant **any**) doit toujours être assortie d'un commentaire expliquant comment le choix est effectué.*

Résultat: à la fin de cette étape, le texte informel a été éliminé.

Exemple

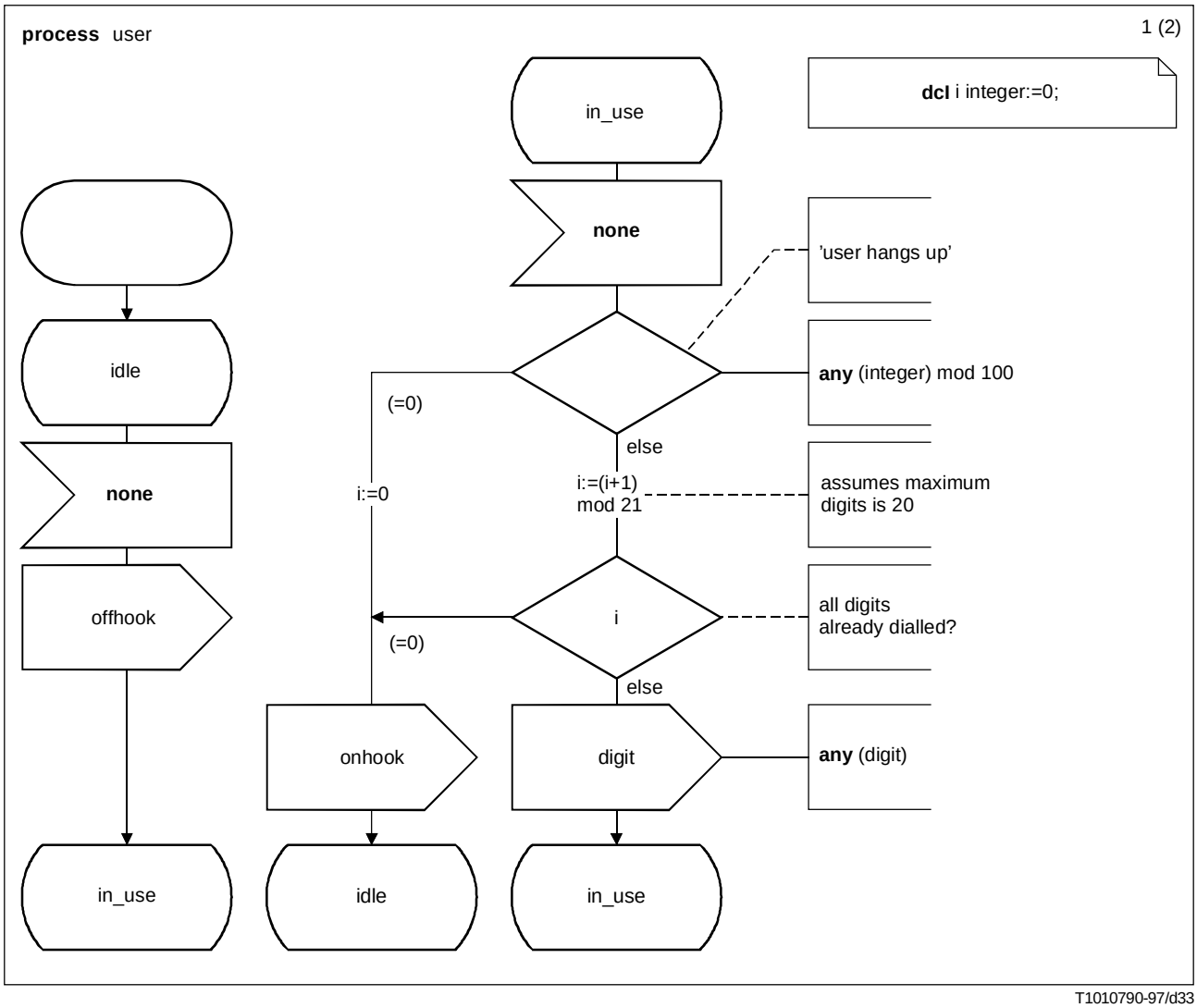


Figure 15-8/Suppl. 1 à la Rec. Z.100 – Partie d'un processus complet mais informel pour un utilisateur

Cet exemple formalise le processus utilisateur de la Figure 15-5. Il illustre également l'utilisation de la construction **any**.

15.3.5 Etape D:5 – Paramètres de sortie et de création

Instructions

1) ajouter des expressions aux sorties utilisant des variables et des synonymes;

2) ajouter des paramètres concrets pour créer des actions.

Directives

Contrôler que chaque paramètre envoyé dans une sortie est utilisé dans au moins une entrée possible ou est requis pour la communication avec l'environnement. Par rapport au contrôle de l'utilisation d'un paramètre de définition de signal à l'étape D:3, il s'agit ici de vérifier qu'un paramètre d'instance de signal peut être utilisé.

Règle 41 – Si un paramètre est omis dans une sortie, il ne doit pas y avoir d'entrée correspondante qui attende une valeur pour ce paramètre.

Résultat: la description SDL est maintenant formelle, sauf pour le comportement des expressions, qui dépend des types de données.

15.3.6 Etape D:6 – Signatures de données

Instructions

1) identifier et définir toutes les sortes de données et de valeurs (synonymes) à définir;
2) si certaines valeurs sont identifiées comme étant dépendantes de l'installation réelle du système, les exprimer au moyen de synonymes externes;
3) pour chaque sorte de données requise, créer une nouvelle sorte (newtype) ou un syntype avec un nom significatif ou définir un type ASN.1;
4) identifier d'éventuels opérateurs nouveaux à définir;

   NOTE – S'il n'y a pas de nouveaux opérateurs, le processus de formalisation est terminé.
5) pour les nouvelles sortes (newtypes) contenant de nouveaux opérateurs, énumérer les signatures (les littéraux et les opérateurs avec les sortes de paramètres) puis identifier (dans un commentaire) un ensemble d'opérateurs et de littéraux pouvant servir à représenter toutes les valeurs possibles (les "constructeurs").

Directives

Des synonymes externes peuvent être utilisés pour:

– fournir des dimensions pour le nombre de processus, le nombre de données élémentaires dans un tableau, etc.;
– effectuer des choix parmi des options de transition ou parmi des éléments à sélectionner offrant une capacité facultative et une variante comportementale.

Hériter autant que possible des données prédéfinies Z.100 [1], de "types utiles" ASN.1 et des données prédéfinies Z.105 [2]. L'objectif est d'éviter d'avoir à définir le comportement pour de nouveaux opérateurs. Si de nouveaux opérateurs sont définis, ils devraient l'être au moyen d'une définition d'opérateur.

On peut éviter l'introduction de nouveaux opérateurs:

– en utilisant les définitions ASN.1 telles que définies dans la Recommandation Z.105 [2];
– en utilisant une structure (**struct**) pour les enregistrements;
– en utilisant un générateur (**generator**) prédéfini tel qu'un tableau ou une chaîne;
– en utilisant un ensemble (**syntype**) si l'ensemble des valeurs de données est destiné à devenir un sous-ensemble d'une sorte de données et à être compatible avec cette sorte;
– en donnant à une sorte un nouveau nom avec **syntype** si le but est d'introduire un nom plus significatif;
– en donnant à une sorte un nouveau nom avec **newtype** et **inherits all** si le but est d'avoir une sorte de données ayant les mêmes propriétés qu'une sorte de données existante, mais sans être compatible avec cette sorte existante;
– en utilisant un générateur (**generator**) défini par l'utilisateur, ce qui évite de définir des axiomes au moyen d'autres générateurs.

Les opérateurs sont habituellement énumérés dans la nouvelle sorte (newtype) pour la sorte de données correspondant au résultat de l'opérateur. Il arrive qu'un opérateur produise une valeur de sorte de données qui est définie dans un contexte différent, par exemple un opérateur entier ou booléen. Dans ce cas, l'opérateur est énuméré dans le cadre de la nouvelle sorte (newtype) (d'une) des sortes paramétriques de données.

Un ensemble d'opérateurs et de littéraux qui peut être utilisé pour représenter toutes les valeurs possibles est un ensemble de constructeurs de la sorte de données. Ces constructeurs seront utilisés dans des étapes ultérieures si (et seulement si) il est nécessaire de définir des propriétés d'opérateur au moyen d'axiomes SDL. L'ensemble des constructeurs n'est

habituellement pas unique ni toujours évident. L'ensemble choisi devrait être juste suffisant pour définir toutes les valeurs, de manière que si un constructeur est supprimé, l'ensemble des valeurs sera différent. Le choix peut être difficile!

Bien que cette étape produise une description formelle SDL, si de nouveaux opérateurs sont introduits, la capacité peut être différente de ce qui avait été prévu, parce que les sortes de données définies par l'utilisateur possèdent parfois des valeurs de type "too many". L'étape D:7 corrige ce défaut mais fait dépendre l'interprétation du texte informel. Les étapes D:8 et D:9 rendent la description formelle.

Résultat: à ce stade, le système est complètement défini sur le plan formel et est complet si aucun opérateur nouveau n'a été introduit.

15.3.7 Etape D:7 – Description informelle de données

Instructions

Ajouter aux définitions de nouvelles sortes (newtypes) des axiomes informels sous la forme de texte informel.

Directives

Les noms des opérateurs devraient correspondre à leur fonction et donc en faciliter la description.

Bien que le résultat soit en langage SDL correct, seules les propriétés statiques peuvent être complètement analysées. La raison en est que les propriétés dynamiques dépendent de l'interprétation des axiomes, ce qui ne peut être fait qu'informellement. Dans un environnement d'appui réel, certaines caractéristiques ou hypothèses supplémentaires peuvent rendre ce niveau de description suffisant.

Résultat: les axiomes informels fournissent un enregistrement du but visé par la définition d'une nouvelle sorte (newtype) sans le définir formellement.

15.3.8 Etape D:8 – Description formelle de données

Instructions

1) pour chaque signature d'opérateur, ajouter une définition d'opérateur, si possible sous la forme d'un diagramme d'opérateurs;

   NOTE – Si tous les opérateurs peuvent être définis de cette façon, la formalisation est terminée.

2) si certains des opérateurs ne peuvent pas être définis par des diagrammes d'opérateurs, formaliser les axiomes en remplaçant les axiomes informels par des axiomes formels indiquant les propriétés essentielles des opérateurs;

3) utiliser le texte des axiomes informels en tant que commentaires des axiomes formels.

Directives

Les opérateurs peuvent être définis algorithmiquement au moyen d'un diagramme d'opérateurs, ou être définis axiomatiquement. La forme (algorithmique) du diagramme d'opérateurs est recommandée car elle est plus facile à comprendre. La méthode axiomatique est décrite ci-dessous.

Bien que la méthode axiomatique ait une meilleure base mathématique et soit utilisée dans le cadre des Recommandations Z.100 [1] et Z.105 [2], il est plus difficile d'écrire des axiomes corrects que de construire un diagramme d'opérateurs qui atteint le but recherché. De nombreux ingénieurs semblent trouver la définition des types de données par axiomes difficile et sujette à erreurs. Les axiomes sont difficiles à manipuler au moyen d'outils, de sorte qu'un bon appui en termes d'outils est plus difficile à trouver. Il convient donc de ne pas utiliser les axiomes sans en avoir l'expérience ou sans les indications d'un utilisateur expérimenté. Pour appliquer un opérateur en langage SDL, il suffit d'avoir une définition de sa signature.

Pour spécifier les propriétés d'un opérateur au moyen d'axiomes, on spécifiera d'abord les propriétés des constructeurs. On obtiendra ainsi les valeurs possibles de la sorte. On spécifiera ensuite des équations pour les opérateurs et littéraux restants. Il est relativement aisé de déclarer les propriétés essentielles mais il est généralement difficile de rendre les axiomes complets et de s'assurer qu'ils ne se contredisent pas (voir l'étape D:9).

Ces axiomes peuvent être considérés comme étant informatifs. A moins que l'on n'envisage d'utiliser les axiomes comme base d'une certaine mise en œuvre automatisée, on peut faire comme si la formalisation était terminée en plaçant un commentaire dans les axiomes pour indiquer que ceux-ci sont informatifs, qu'ils n'ont pas été démontrés et qu'ils peuvent être incomplets. Dans ce cas, l'étape D:9 doit être omise.

Règle 42 – Les axiomes SDL ne devraient pas être utilisés pour définir des propriétés d'opérateur.

Résultat: le résultat est que certaines (mais probablement pas la totalité) des propriétés des types de données ont été définies, si bien que, formellement, chaque sorte possède un grand nombre (habituellement infini) de valeurs en plus de celles qui avaient été prévues.

15.3.9 Etape D:9 – Formalisation complète de données

NOTE – Cette étape ne devrait être utilisée que dans des circonstances exceptionnelles.

Instructions

Ajouter des axiomes (ou des définitions d'opérateur) aux définitions des nouvelles sortes (newtypes) de données, jusqu'à ce qu'elles soient complètes (c'est-à-dire jusqu'à ce que toutes les expression contenant des opérateurs et des littéraux non constructeurs puissent être récrites dans des expressions ne contenant que des opérateurs et littéraux constructeurs).

Directives

Les directives détaillées pour cette étape sont données au I.5/Z.100 [1].

Eviter de définir des constructeurs (c'est-à-dire des opérateurs qui créent des valeurs d'une sorte de données). S'assurer plutôt qu'il existe un littéral pour chaque (et pour toute) valeur de sorte. En admettant que l'on n'utilise pas d'inégalités (**noequality**), chaque littéral défini possède une valeur distincte de celle de tout autre littéral.

Si de nouveaux opérateurs sont requis pour faire complètement usage de sortes définies dans des unités de visibilité englobante, il y a lieu de définir l'opérateur dans une nouvelle sorte (newtype) fictive.

Ne jamais modifier (intentionnellement ou non!) le nombre de valeurs d'une sorte en utilisant des axiomes dans la nouvelle sorte (newtype) d'une autre sorte.

Résultat: si toutes les étapes précédentes ont été complètement suivies, on disposera à ce stade d'une spécification complètement formelle en langage SDL, comportant des définitions complètes et univoques pour toutes les sortes de données utilisées dans la spécification.

15.4 Etapes relatives au type (étapes T, *T-steps*)

Les étapes relatives au type servent à définir les éléments du système en termes de types de façon à pouvoir les réutiliser.

Pour tirer le meilleur parti de l'application de ces étapes, il est recommandé de les appliquer, dans les très grands systèmes, en parallèle avec les étapes précédentes. La raison en est que le développement d'une branche hiérarchique du système peut faire apparaître des types qui pourront être réutilisés dans une autre branche.

Les étapes les plus importantes sont T:1 et T:2. Les étapes T:3 à T:5 peuvent être considérées comme facultatives. Une bonne connaissance de la version SDL-92 et de l'orientation-objets est recommandée avant d'utiliser les étapes T:3 à T:5. Dans certains cas, on peut faire appel à la paramétrisation décrite à l'étape L:2, au lieu de la spécialisation.

15.4.1 Etape T:1 – Identification des types SDL

Instructions

1) modifier les définitions d'instance utilisées dans le système pour les types;

2) utiliser le même type pour deux instances lorsque celles-ci (par exemple des blocs) ont le même comportement.

Directives

Cette étape permet de reconnaître clairement les types et de supprimer quelques redondances dans le système SDL. Une situation typique est celle dans laquelle un système est défini "de bout en bout" et où la terminaison à chaque extrémité est décrite par des blocs ayant le même comportement. La définition doit être répétée pour les deux blocs à moins qu'un type de bloc ne soit utilisé, auquel cas la définition de ce type de bloc peut être utilisée pour les deux blocs.

La définition d'un type de système est facultative mais permet de réutiliser une telle définition pour construire des normes analogues. Cela peut être particulièrement utile en présence d'une série de normes associées.

L'adjonction de types de bloc ou de processus implique celle de portes pour identifier la façon dont la connexion à la définition utilisant le type correspond aux trajets de communication de ce type.

Un processus ou bloc qui n'est utilisé qu'une seule fois devrait initialement être modélisé comme type parce qu'il sera possible, dans une phase ultérieure, de remplacer ce modèle par un type fondé sur un type existant. Bien qu'il soit plus facile de produire initialement des diagrammes sans utiliser de types, l'introduction de types permet une réutilisation et donc une grande économie de descriptions répétées et d'efforts d'ingénierie.

Il convient de noter que toutes les définitions de procédure et de signal sont des définitions de type.

Règle 43 – Deux blocs (ou processus) qui modélisent des instances du même objet doivent être modélisés par des définitions de bloc (processus) fondées sur la définition d'un type commun de bloc (processus).

Résultat: types définis pour le système, pour les blocs et pour les processus.

Exemple

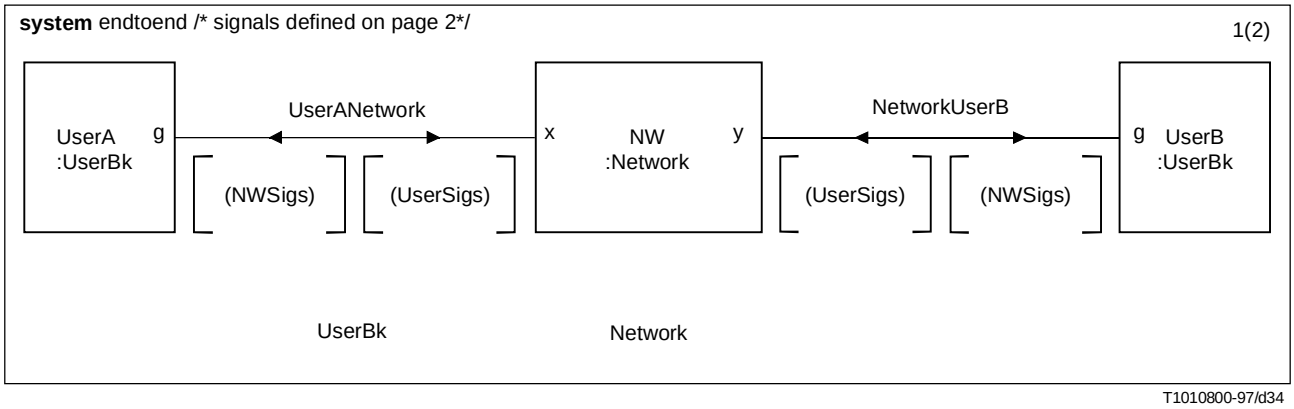


Figure 15-9/Suppl. 1 à la Rec. Z.100 – Définition de système utilisant des types

15.4.2 Etape T:2 – Spécialisation de types par adjonction de propriétés

Instructions

1) identifier les cas où un même type possède tout le comportement d'un autre type lorsque ce second type peut recevoir un certain stimulus (signal ou appel de procédure distante) et peut aussi gérer des stimulations additionnelles;

2) définir le premier type comme un sous-type qui hérite (**inherits**) les propriétés du second type et juste ajouter (**adding**) les propriétés permettant de gérer les stimulations additionnelles.

Directives

Un sous-type est un type créé par spécialisation d'un autre type (appelé supertype du sous-type). La spécialisation peut être effectuée de deux manières: soit par adjonction de propriétés au supertype soit par modification, dans le sous-type, de certaines des propriétés du supertype. Dans la présente étape, on recourt à la première méthode.

Les propriétés qui peuvent être ajoutées ne sont limitées que par celles qui ont déjà été définies pour le type, de manière que les propriétés ajoutées ne contredisent pas la définition héritée. Des canaux et des blocs peuvent être ajoutés à un type de système ou de bloc englobant des blocs. Des routes de signal, des définitions de (type de) processus, des procédures et des définitions de signal peuvent être ajoutées à un type de bloc englobant des processus. De nouvelles transitions pour de nouveaux signaux peuvent être ajoutées à un type de processus et les transitions peuvent conduire à de nouveaux états.

Résultat: un nouveau sous-type ayant un comportement étendu, mais qui est incompatible avec le comportement précédent.

Exemple

Deux processus (suerma et suerm_supertype) sont identiques sauf que l'un d'entre eux (suerma) accepte un signal supplémentaire (abort_call). Lorsque ce signal supplémentaire d'abandon d'appel est introduit, il réinitialise le processus à l'état de repos quel que soit son état précédent. Le type du second processus peut être un sous-type du premier processus (voir la Figure 15-10).

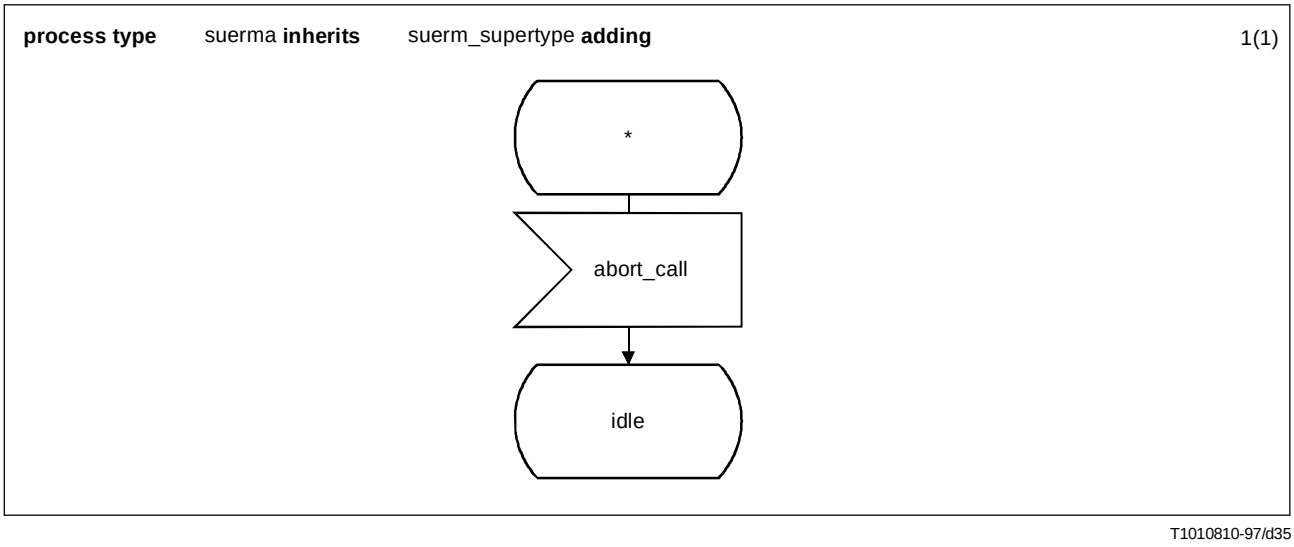


Figure 15-10/Suppl. 1 à la Rec. Z.100 – Sous-type créé par adjonction d'une propriété

15.4.3 Etape T:3 – Généralisation de types par virtualisation

Instructions

1) s'il n'y a pas de types ayant presque la même structure interne et le même comportement, mais que ces types gèrent des situations légèrement différentes (parce qu'ils sont "similaires"), l'étape est complète;

2) pour chaque ensemble de types "similaires" [comme dans l'instruction 1)], appliquer les instructions 3) à 6);

3) identifier la partie commune et la spécifier en tant que supertype général, à utiliser pour les autres types qui deviennent des sous-types lorsque l'étape T:5 est appliquée;

4) si le supertype général est un type de bloc, identifier les instances (blocs et processus) qui doivent être différentes dans ses sous-types, les convertir en instances fondées sur des types et définir ceux-ci comme des types virtuels contenus dans le supertype général;

5) si le supertype général est un type de processus ou une procédure, identifier les séquences d'actions partielles qui devraient être adaptables:

   5.1 s'il y a des transitions complètes, les transformer en transitions virtuelles;

   5.2 sinon, remplacer les séquences d'actions partielles par des appels de procédure et définir les procédures virtuelles correspondantes;

   5.3 s'il y a d'autres procédures dans le supertype général, examiner si ces procédures devraient être virtuelles;

6) pour chaque type virtuel contenu dans le supertype général, identifier ou définir un type qui possède la structure interne et les paramètres appropriés à toutes les redéfinitions et utiliser ce type comme contrainte sur le type virtuel comme détaillé dans l'étape T:4.

NOTE – Le supertype défini par cette étape n'est pas utilisé dans le système SDL tant que d'autres étapes ne sont pas appliquées.

Directives

Un supertype général possède quelques parties redéfinissables. Ces parties sont identifiées comme étant virtuelles. Si le type est utilisé sans redéfinition de ces parties, les définitions contenues dans les parties virtuelles sont applicables. Les définitions des parties virtuelles dans le type général sont donc choisies de manière qu'elles modélisent la variante la plus commune.

Veiller à ne pas trop généraliser des types. La généralisation présente également un danger lorsque deux types ont un comportement fonctionnel similaire mais des concepts très différents, parce qu'il peut être difficile de comprendre la fonction du supertype général lorsqu'il est appliqué à chaque cas. Le supertype général devrait reprendre les caractéristiques essentielles d'un concept en termes de comportements non virtuels.

Une transition virtuelle permet de redéfinir l'ensemble de la transition. On peut transformer certaines parties de la transition en procédures virtuelles, de façon que les parties communes soient fixes et que l'ensemble de la transition puisse ne pas être virtuel.

Règle 44 – La partie commune d'un supertype devrait constituer au moins 70 % du total de chaque sous-type.

Résultat: un type généralisé par la définition de certains des types ou transitions locaux comme étant virtuels.

Exemples

Un cas typique est celui de deux processus qui sont identiques sauf pour la transition d'un signal vers un état. Le supertype est alors la définition de processus dans laquelle cette transition est soit virtuelle soit contient un appel d'une ou de plusieurs procédures virtuelles.

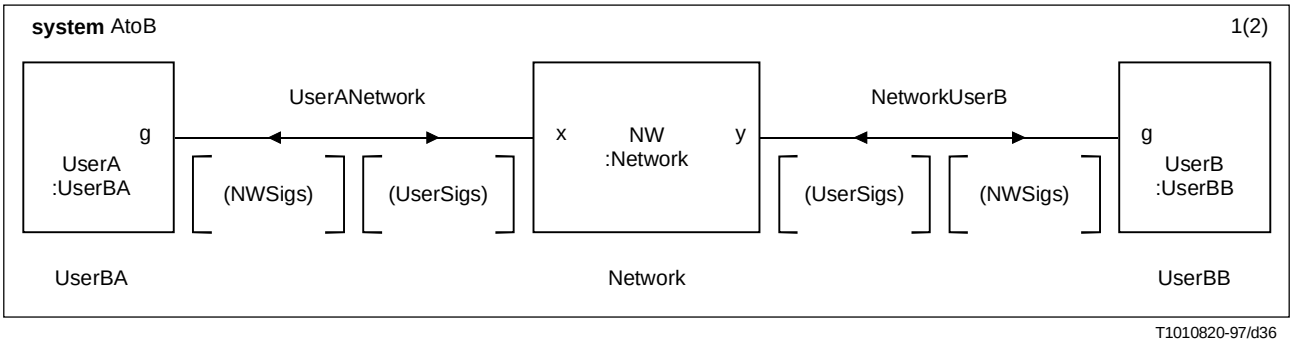


Figure 15-11/Suppl. 1 à la Rec. Z.100 – Le système AtoB

A titre d'autre exemple, supposons que le système "AtoB" représente une communication de A à B par un réseau. Les extrémités A et B sont des blocs presque identiques, contenant chaque fois un seul processus, celui d'usager appelé. Le processus d'utilisateur est légèrement différent dans chaque cas. Ce supertype général peut être un type de bloc (UserBk).

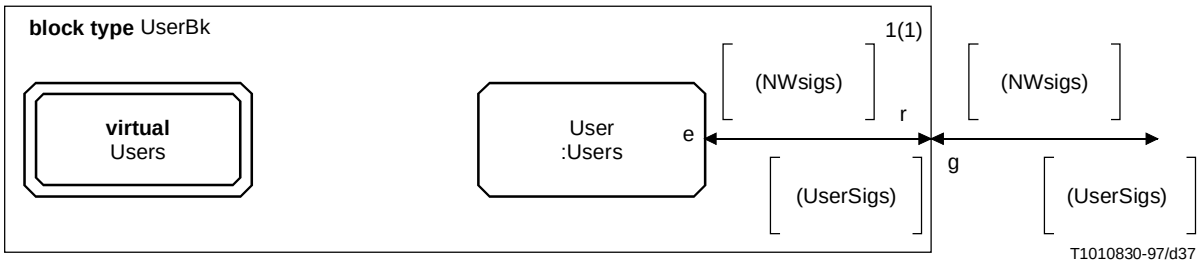


Figure 15-12/Suppl. 1 à la Rec. Z.100 – Le supertype de bloc pour les "utilisateurs"

Etant donné qu'il n'y a qu'un seul processus dans chaque bloc et que ce processus a un comportement différent dans chaque cas, ce processus est fondé sur le type de processus virtuel (**virtual**) "Users". Les types de bloc UserBA et UserBB peuvent être définis comme héritant le bloc UserBk du supertype général et comme redéfinissant le processus virtuel "Users".

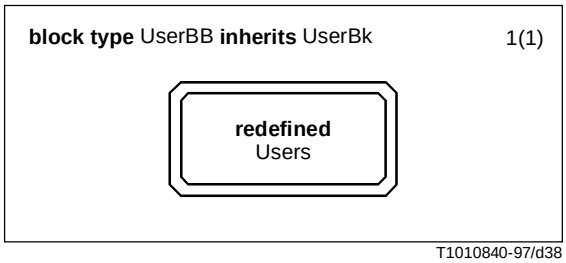


Figure 15-13/Suppl. 1 à la Rec. Z.100 – Définition du bloc UserBB au moyen du bloc UserBk du supertype général

15.4.4 Etape T:4 – Restriction de types virtuels

Instructions

1) déterminer les propriétés d'un type virtuel qui doivent s'appliquer à toutes les redéfinitions de ce type virtuel;

2) définir (ou identifier) un type qui possède ces propriétés (appelé "type de contrainte");

3) ajouter le "type de contrainte" **atleast** aux définitions des types virtuels (**virtual**) dans le supertype général;

4) définir un type redéfini (**redefined**) (fondé sur le type de contrainte virtuel) pour chaque instance d'utilisation du type virtuel.

Directives

Un supertype général englobe des types virtuels qui peuvent être différents chaque fois que le supertype général est utilisé pour définir un sous-type. Il faut définir le type virtuel englobé dans le supertype général ainsi qu'un type (correspondant au type virtuel) englobé dans chaque sous-type (du type général). Le type de contrainte:

– définit les propriétés communes du type virtuel contenu dans le supertype général ainsi que chacune des redéfinitions du type virtuel;

– contraint les définitions de façon que chacune d'elles soit un sous-type du type de contrainte et doive donc hériter du type de contrainte.

Si la définition du type virtuel englobé dans le sous-type est la même que celle du type de contrainte, c'est la définition par défaut et la redéfinition dans le sous-type est omise.

Règle 45 – Le type de contrainte doit fixer les propriétés:

– *des paramètres d'une procédure virtuelle de façon que tous les appels aient le même nombre de paramètres du même type;*

– *du paramètre, des portes et du comportement commun d'un processus virtuel;*

– *des portes et des blocs ou processus internes connectés à la porte pour un bloc virtuel.*

Résultat: une définition de type (le "type de contrainte") pour chaque type virtuel dans le supertype général, et un sous-type (du type de contrainte) pour chaque comportement réel.

Exemple

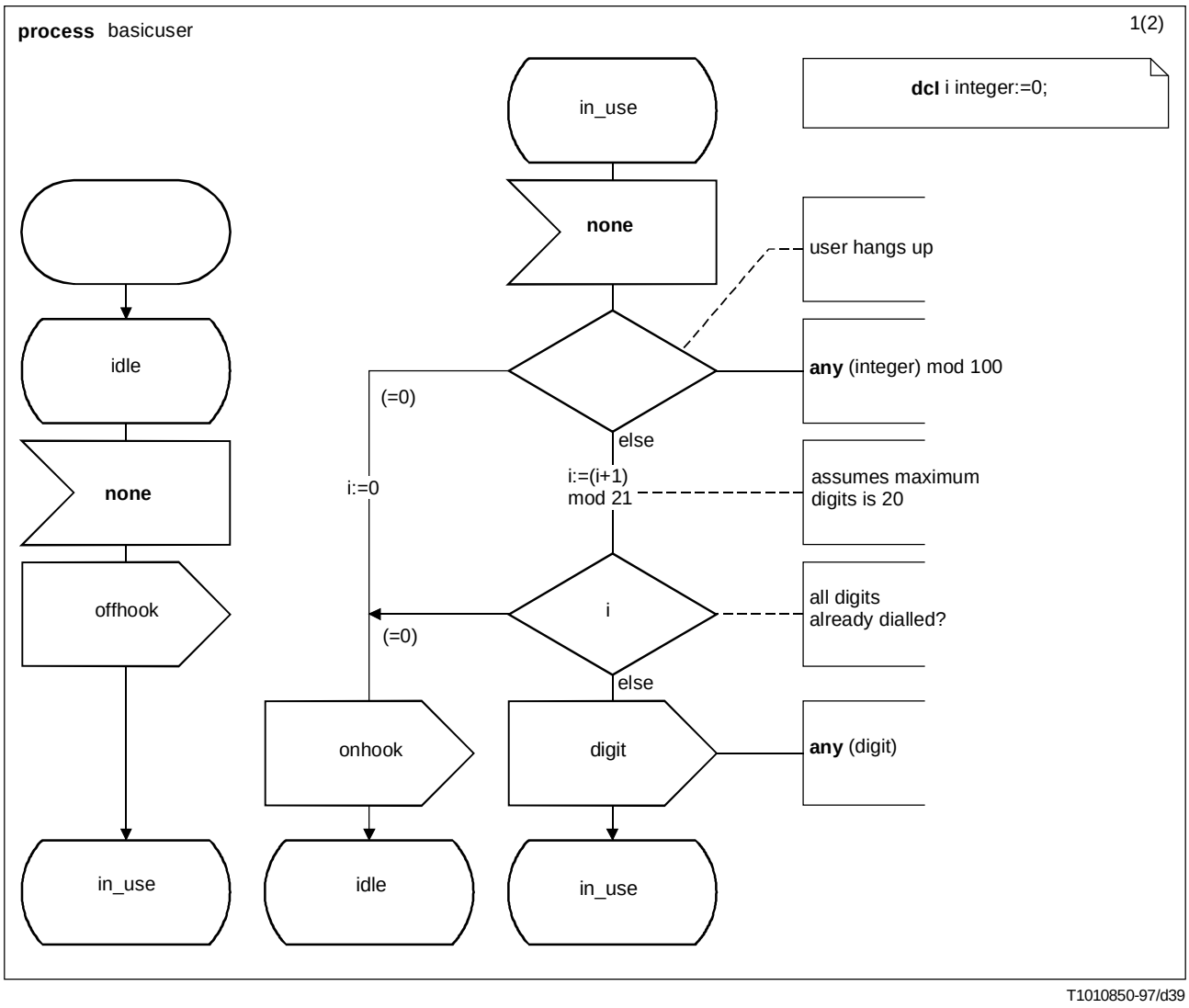


Figure 15-14/Suppl. 1 à la Rec. Z.100 – Le type de processus basicuser

L'on part du principe que, chaque fois que le bloc *UserBk* est utilisé, le type de processus *Users* possède au moins les propriétés d'un utilisateur de base (*basicuser*). La définition du type de processus virtuel pour le processus *Users* dans le type *UserBk* du supertype général est alors définie comme étant "**atleast** basicuser".

15.4.5 Etape T:5 – Spécialisation de types par redéfinition

Instructions

1) remplacer chaque sous-type identifié dans l'étape T:3 par un type qui hérite (**inherits**) du supertype général, et dans ce sous-type;

2) redéfinir les transitions virtuelles pour le contexte;

3) redéfinir les types virtuels pour le contexte en utilisant les redéfinitions (**redefinitions**) de l'étape T:4.

Directives

Cette étape est nécessaire pour produire un type qui décrit le comportement requis à partir des types plus abstraits qui ont été produits par des étapes T précédentes.

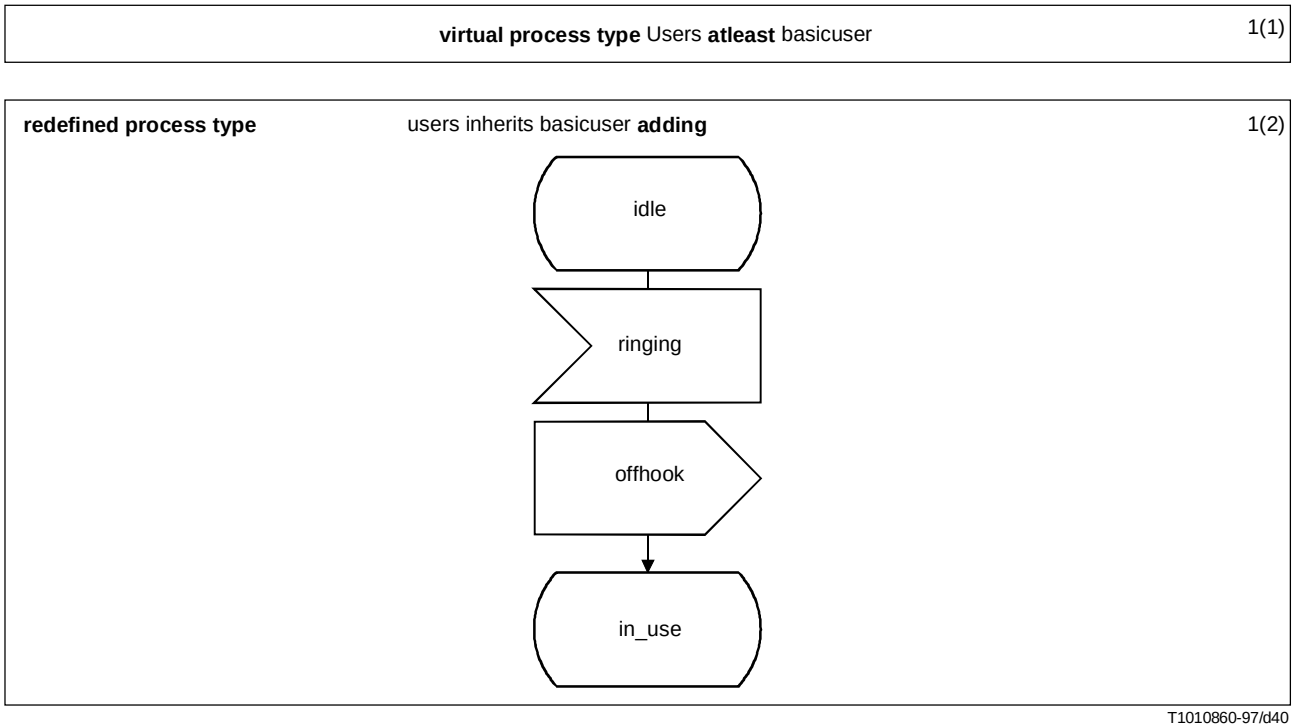
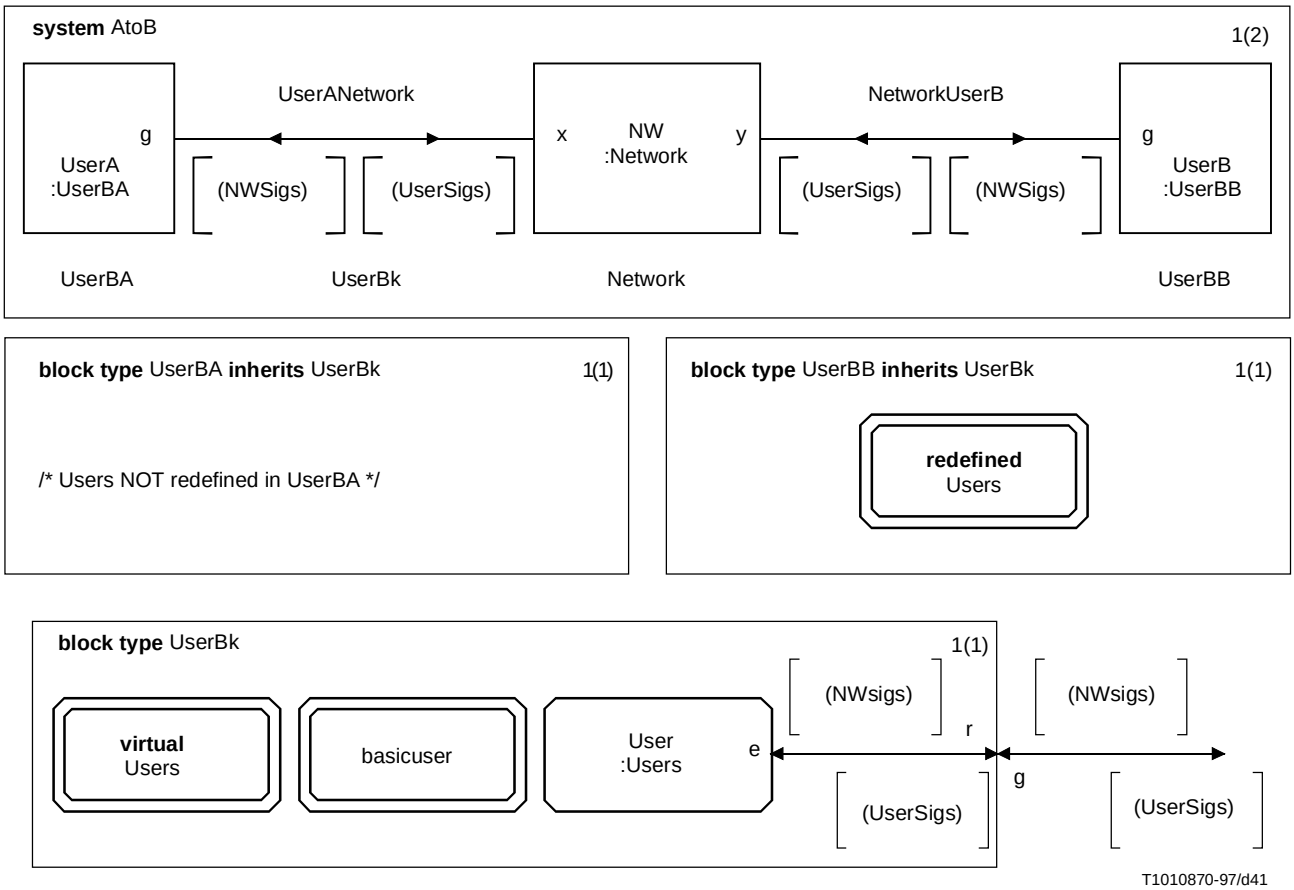


Figure 15-15/Suppl. 1 à la Rec. Z.100 – Le type de processus virtuel Users et sa redéfinition

Résultat: un système fondé sur des définitions de type.

Exemples



NOTE – Etant donné que le bloc UserBA est le même que le bloc UserBk, le bloc UserBA, peut être éliminé. De même, étant donné que le processus *basicuser* n'est utilisé que dans le bloc *UserBk*, la définition (virtuelle) du bloc *Users* pourrait aussi définir le type de contrainte.

Figure 15-16/Suppl. 1 à la Rec. Z.100 – Le système AtoB (avec un type de bloc du supertype général: UserBk)

15.5 Etapes de localisation (étapes L, *L-steps*)

La fonction des étapes de localisation est d'assurer que les types sont définis aux meilleurs emplacements afin d'éviter une définition inutile de nouveaux types.

15.5.1 Etape L:1 – Types non paramétrés

Instructions

1) trouver l'unité de visibilité appropriée pour la définition de chaque type;
2) transférer chaque type (et les définitions associées) dans cette unité de visibilité ou (si possible) dans un composant;
3) pour chaque nouveau composant, ajouter une utilisation (**use**) dans le symbole littéral du diagramme de système faisant référence à ce composant.

Directives

Un type qui ne dépend d'aucune définition peut être transféré dans un composant. Veiller à identifier les types ou définitions mutuellement dépendants qui peuvent être transférés ensemble dans un composant.

On peut en même temps passer en revue l'unité de visibilité pour d'autres définitions. Chaque définition est placée de manière qu'elle puisse être réutilisée à différents endroits mais aussi avec la plus petite visibilité pour tous ses usages. Il est judicieux de transférer dans un composant les définitions des signaux pour le système car cela rend celui-ci plus proche d'un bloc.

Un type doit être local s'il dépend de définitions ou d'autres types dans la même unité de visibilité (en supposant qu'il ne soit ni approprié ni possible de transférer toutes les définitions associées). Un type doit toujours être local si sa visibilité doit être restreinte au contexte local. La nécessité de restreindre la visibilité d'un type apparaît souvent à la suite de la redéfinition d'un type virtuel avec le même nom. Un type devrait être local s'il n'est significatif que dans ce contexte.

Il est utile de placer dans un composant un ensemble de définitions, de façon à pouvoir le réutiliser dans différents systèmes. Mais le SDL-92 ne permet de joindre des références de composant qu'au diagramme de système. L'utilisation d'un composant contredit donc la limitation de la visibilité locale de définitions. La localisation dans des composants est examinée plus en détail dans l'étape L:3.

Règle 46 – Le domaine de visibilité prévu pour l'utilisation d'un composant doit être ajouté sous forme de commentaire à la clause faisant référence à ce composant.

Résultat: toutes les définitions possèdent un domaine de visibilité locale approprié ou sont définies dans un composant avec un commentaire sur leur domaine de visibilité prévu.

15.5.2 Etape L:2 – Définition des paramètres contextuels

Instructions

1) identifier le moment où deux ou plus de deux définitions sont identiques sauf pour l'utilisation de certains items nommés (comme des signaux ou des procédures);
2) déterminer les contraintes (plus petite définition commune) qui s'exercent sur les types des items nommés (par exemple un signal peut être contraint à avoir deux paramètres sous forme d'entiers) et identifier ou définir un type à utiliser dans une clause **atleast**;
3) définir une nouvelle définition de type ayant comme paramètres contextuels les items nommés et les clauses de contrainte **atleast** sur les paramètres;
4) remplacer les usages des types presque identiques par des usages du nouveau type avec des paramètres contextuels, de manière que les paramètres réels soient les items nommés initialement;
5) placer une référence à la nouvelle définition de type dans l'unité de visibilité qui englobe tous les usages.

Directives

Les paramètres contextuels sont ceux des types qui sont remplacés par des paramètres réels qui sont eux-mêmes des définitions d'items. Le remplacement est statique et intervient avant l'interprétation du système SDL. Les paramètres réels indiqués dans le contexte d'utilisation du type définissent un type dont les paramètres ont été remplacés.

Les autres paramètres en SDL (par exemple des paramètres de processus, de procédures ou de signaux) sont remplacés par les paramètres réels qui sont des valeurs. Ce remplacement est dynamique et intervient lors de l'interprétation du système. Les valeurs de ces paramètres sont transmises aux instances du type.

L'utilisation de paramètres contextuels est parfois une variante à la virtualisation de certains éléments d'un type de façon qu'il puisse être réutilisé. Dans le cas d'un signal, d'un temporisateur, d'une variable, d'un synonyme et de paramètres contextuels de sorte, il n'y a aucun choix: le langage SDL ne possède pas de types virtuels pour ces items. Dans le cas des procédures, il est recommandé de faire appel à une procédure virtuelle si la procédure réelle est toujours locale. Il est recommandé d'utiliser un paramètre contextuel de procédure si la procédure réelle est parfois plus globale. Pour un processus, il est préférable d'utiliser un paramètre contextuel de processus contenu dans un bloc, si les types de processus pour les paramètres réels peuvent être utilement définis comme étant extérieurs aux blocs. Le système et les blocs ne peuvent être des paramètres contextuels.

Tableau 4/Supp. 1 à la Rec. Z.100 – Utilisation de paramètres contextuels

	Peut être un paramètre contextuel	**Peut avoir un paramètre contextuel**	**Peut être un type virtuel**
système, bloc	non	oui	oui
processus	oui	oui	oui
procédure	oui	oui	oui
signal, temporisateur	oui	oui	non
sorte	oui	oui	non
variable, synonyme	oui	non	non

Il n'y a parfois aucune contrainte à exercer sur le paramètre contextuel, auquel cas la clause **atleast** est omise.

Règle 47 – Les types possédant des paramètres contextuels ne devraient être utilisés que lorsque la compréhension du système est augmentée par la réutilisation de concepts.

Résultat: un type moins dépendant du contexte est inclus dans le système et utilisations de ce type en remplaçant deux ou plus de deux définitions dépendant du contexte.

15.5.3 Etape L:3 – Définition des modules

Instructions

1) identifier les groupes d'items associés qui sont spécifiques du système et qui peuvent être répartis dans un composant approprié à un usage général;

2) définir un composant dont les items identifiés sont représentés par des types;

3) mémoriser certaines informations permettant de trouver le composant lors de recherches de types appropriés;

4) mémoriser dans la bibliothèque l'utilisation du composant;

5) chaque fois qu'un composant est réutilisé, enregistrer son usage dans la bibliothèque et ouvrir un fichier particulier pour tous les modifications éventuellement apportées à ce composant afin de le rendre réutilisable.

Directives

Les composants sont particulièrement utiles lorsqu'il existe un ensemble de systèmes SDL associés, comme des descriptions de différents services complémentaires dans différentes normes associées. L'on part du principe que les composants sont mémorisés dans une bibliothèque de concepts pour utilisation future.

Lors de la recherche de définitions dans la bibliothèque de concepts, les informations classifiées sont utilisées comme clés de recherche élémentaires pour trouver dans la bibliothèque des correspondances avec des concepts proches. Pour constituer une collection de types utiles à un domaine d'application, il est recommandé de faire appel à un composant plutôt que d'englober ces types dans (par exemple) un type de bloc puis à spécialiser celui-ci pour chaque utilisation. Un type de bloc est plus approprié pour un élément fonctionnel du système.

Un composant enregistré en bibliothèque est parfois proche de ce qui est requis mais ne correspond pas tout à fait aux informations classifiées. Dans ce cas, le composant se trouvant dans la bibliothèque peut être modifié afin de couvrir le nouveau système. D'autres utilisations du composant devraient être vérifiées avant de modifier celui-ci, afin d'éviter d'introduire des incompatibilités avec ces cas.

Bien que la manipulation d'un composant ne soit pas complètement définie dans la norme SDL, l'environnement d'appui est censé la prendre en charge de façon à la mettre à la disposition de tout autre composant ou de toute autre spécification de système.

Résultat: des composants de définitions de type pour réutilisation dans la bibliothèque et utilisation dans le système SDL.

16 Références

[1] Recommandation UIT-T Z.100 (1993), *Langage de description et de spécification du CCITT*.

[2] Recommandation UIT-T Z.105 (1995), *Langage de description et de spécification combiné avec la notation de syntaxe abstraite numéro un.*

[3] Recommandation UIT-T Z.120 (1996), *Diagramme de séquences de messages (MSC).*

[4] Recommandation X.208 du CCITT (1988) (équivalant à ISO/CEI 8824:1990), *Spécification de la syntaxe abstraite numéro un (ASN.1).*

[5] Recommandation UIT-T X.680 (1994)/Amd. 1 (1995) (équivalant à ISO/CEI 8824-1:1996), *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification de la notation de base*.

[6] BRÆK (R.), HAUGEN (Ø.): Engineering Real Time Systems, Prentice-Hall, 1993.

[7] OLSEN (A.) *et al*: Systems Engineering Using SDL-92, North Holland, 1994.

[8] Recommandations UIT-T X.900-X.905 (à paraître), *Technologies de l'information – Traitement réparti ouvert* (ISO/CEI 10746).

[9] Recommandation UIT-T Z.500 (1997), *Cadre général méthodes formelles appliquées aux tests de conformité*.

[10] Recommandations UIT-T X.290-X.292, *Cadre général et méthodologie des tests de conformité OSI pour les Recommandations sur les protocoles pour les applications de l'UIT-T* (ISO/CEI 9646).

[11] RUMBAUGH (J.) *et al*: Object-Oriented Modeling and Design, Prentice-Hall, 1991.

[12] JACOBSEN (I.) *et al*: Object-Oriented Software Engineering, Addison-Wesley, 1992.

[13] Recommandation UIT-T Q.1200 (1993), *Structure des Recommandations de la série Q sur le réseau intelligent.*

[14] Recommandation UIT-T Z.400 (1993), *Structure et format des manuels de qualité pour le logiciel de télécommunication.*

[15] HOGREFE (D.): Validation of SDL Systems, Computer Networks and ISDN Systems, North Holland, 1996.

[16] Recommandation I.130 du CCITT (1988), *Méthode de caractérisation des services de télécommunication assurés sur un RNIS et des possibilités réseau d'un RNIS*.

[17] Recommandation UIT-T Q.65 (1997), *Méthode fonctionnelle unifiée de caractérisation des services et des capacités des réseaux.*

[18] BELINA (F.), HOGREFE (D.), TRIGILA (S.): Modelling OSI in SDL (in Turner: Formal Description Techniques), North Holland, 1988.

[19] BELINA (F.), HOGREFE (D.), SARMA (A.): SDL with Applications from Protocol Specification, Prentice Hall, 1991.

[20] Recommandation UIT-T X.200 (1994), *Technologies de l'information – Interconnexion des systèmes ouverts – Modèle de référence de base: le modèle de référence de base*.

[21] Recommandation UIT-T X.210 (1993), *Technologies de l'information – Interconnexion des systèmes ouverts – Modèle de référence de base: conventions pour la définition des services de l'interconnexion de systèmes ouverts.*

[22] Recommandation X.219 du CCITT (1988), *Opérations distantes: modèle, notation et définition du service*.

[23] Recommandation X.722 du CCITT (1992), *Technologies de l'information – Interconnexion des systèmes ouverts – Structure des informations de gestion: directives pour la définition des objets gérés*.

[24] REED (R.) (editor): Specification and Programming Environment for Communication Software, North Holland, 1993.

[25] OLSEN (A.) *et al*: Systems Engineering Using SDL-92, North Holland, 1994.

[26] WITASZEK (D.) *et al*: A Development Method for SDL-92 Specifications based on OMT, in SDL '95 with MSC in CASE, *Proceedings of the Seventh SDL Forum*, North Holland, 1995.

[27] GUO (F.), MACKENZIE (T.W.): Translation of OMT to SDL-92, in SDL '95 with MSC in CASE, *Proceedings of the Seventh SDL Forum*, North Holland, 1995.

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 pour données et communication entre systèmes ouverts
Série Z	**Langages de programmation**