UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

Z.100
Addendum 1
(10/96)

SÉRIE Z: LANGAGES DE PROGRAMMATION

Langage de description et de spécification (SDL)

Langage de description et de spécification du CCITT

Addendum 1

Recommandation UIT-T Z.100 – Addendum 1

(Antérieurement Recommandation du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE Z

LANGAGES DE PROGRAMMATION

Langage de description et de spécification (SDL)	**Z.100–Z.109**
Critères d'utilisation et d'applicabilité des techniques de description formelle	Z.110–Z.199
Langage de haut niveau de l'UIT-T (CHILL)	Z.200–Z.299
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
Divers	Z.400–Z.499

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

AVANT-PROPOS

L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'Union internationale des télécommunications (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 (Helsinki, 1er-12 mars 1993).

La Recommandation UIT-T Z.100, Addendum 1, que l'on doit à la Commission d'études 10 (1993-1996) de l'UIT-T, a été approuvée par la CMNT (Genève, 9-18 octobre 1996).

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 de télécommunications.

TABLE DES MATIÈRES

RÉSUMÉ

La présente Recommandation est un addendum à la Recommandation Z.100 (1993). Cette dernière, modifiée par la présente Recommandation, définit la version 1996 du langage de description et de spécification (SDL, *specification and description language*) du CCITT. Le présent addendum contient des corrections de la définition précédente du langage, ainsi que quelques ajouts et modifications du langage. Il n'y aura pas de nouvelle définition formelle de ces changements.

CONTEXTE

Au cours de la période d'études 1993-1996, un certain nombre d'erreurs mineures ont été détectées et ajoutées dans une "liste principale des corrections". Cette période d'études a été accompagnée de fortes demandes de stabilité du langage SDL, et cela a été respecté. Cependant, quelques modifications et ajouts ont été demandés par les utilisateurs. Ceux-ci ont été pris en compte. La publication d'un addendum au lieu d'une nouvelle Recommandation souligne la stabilité de cette version révisée du langage.

LANGAGE DE DESCRIPTION ET DE SPÉCIFICATION DU CCITT

ADDENDUM 1

(Genève, 1996)

1 Domaine d'application

Le présent addendum définit le langage de description et de spécification (SDL, *specification and description language*) du CCITT.

2 Références normatives

La présente Recommandation se réfère à certaines dispositions des Recommandations UIT-T et textes suivants qui de ce fait en sont partie intégrante. Les versions indiquées étaient en vigueur au moment de la publication de la présente Recommandation. Toute Recommandation ou texte étant sujet à révision, les utilisateurs de la présente Recommandation sont invités à se reporter, si possible, aux versions les plus récentes des références normatives suivantes. La liste des Recommandations de l'UIT-T en vigueur est régulièrement publiée.

[1] Recommandation UIT-T Z.100 (1993), *Langage de description et de spécification du CCITT*.

3 Abréviations

Le présent addendum utilise l'abréviation suivante:

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

4 Remarques générales concernant la liste de corrections à la Recommandation Z.100

Le présent addendum reflète la liste principale des corrections qui ont été accumulées durant la période d'études 1993-1996 et contient toutes les modifications qui ont été approuvées par la Commission d'études 10 de l'UIT-T lors de ses réunions. La liste principale des corrections possédait les catégories d'entrées suivantes:

- *corrections* et *clarifications:* considérées comme prenant effet immédiatement et décrites complètement;
- *modifications, points désengagés* et *extensions:* agréés par la Commission d'études 10 comme candidats à l'inclusion dans une révision de la Recommandation Z.100, mais soumis aux procédures d'approbation de toute l'UIT et prenant effet lors de l'approbation par l'UIT de Recommandations nouvelles ou révisées;
- *questions ouvertes* et *questions fermées:* représentent le statut courant des points examinés.

Certaines *modifications*, *extensions* ou *questions ouvertes* nécessitent d'être encore examinées.

La terminologie de maintenance du SDL, les directives et les procédures de demande de modification se trouvent au paragraphe 15 du présent addendum.

5 Corrections et clarifications à la Recommandation Z.100

Sous-paragraphe 2.2.2

Changer la règle de production de <path item> dans la *Grammaire textuelle concrète* page 17 pour ajouter l'option "<operator name> <exclamation>":

```
<path item> ::=
                    <scope unit kind>
                    { <name> | <quoted operator> | <operator name> <exclamation> }
```

Sous-paragraphe 2.2.2

Ajouter dans *Sémantique* à la fin du sous-paragraphe avant les deux derniers paragraphes:

> Ce qui suit clarifie les règles de visibilité des procédures, des services et des types de service (vers la fin de la page 19):
>
> 1) les procédures, services et types de service définis localement à un processus peuvent accéder et utiliser librement les variables et les signaux de ce processus;
>
> 2) les procédures, services et types de service définis globalement ne doivent pas utiliser de variables définies dans le contexte d'appel;
>
> 3) pour accéder et utiliser les entités définies localement au processus appelant, des paramètres correspondant au contexte doivent être utilisés.

Sous-paragraphe 2.4.5

Remplacer la dernière ligne de l'avant-dernier paragraphe de la *Grammaire textuelle concrète* (page 38, 1re ligne) par:

> Une procédure exportée définie dans le processus englobant doit être mentionnée par une entrée ou une sauvegarde dans un service et un seul.

Sous-paragraphe 2.4.5

Dans la partie de texte concernant les services, <valid input signal set> a été oublié. Ceci est contenu dans la représentation textuelle et dans la représentation graphique correspondante du diagramme des processus (<process text area>). La règle de production corrigée est:

```
<service text area> ::=
        <text symbol> contains {
                [ <valid input signal set> ]
                { <variable definition>
                        ....
                | <macro definition>  } *
                                }
```

Sous-paragraphe 2.4.6

Dans le corps de procédure, la représentation textuelle contient une erreur, vu que la représentation graphique contient la production intuitivement correcte. La production correcte est:

```
<procedure body> ::=
        [ <start> ]   { <state> | <free action> } *
```

Sous-paragraphe 2.7.5

Le premier paragraphe suivant les productions de la *Grammaire textuelle concrète* page 68 pour *Décision* doit être modifié comme suit (les modifications sont soulignées):

> Une partie <answer part> ou une partie <else part> dans une décision ou une option est respectivement une partie <answer part> ou une partie <else part> terminale si elle contient une <transition> dans laquelle une instruction <terminator statement> est spécifiée, ou si elle contient une chaîne <transition string> dont la dernière instruction <action statement> contient une décision ou une option terminale. Une <decision> ou une option <transition option> est respectivement une décision terminale ou une option terminale si chaque partie <answer part> et chaque partie <else part> dans son corps <decision body> est respectivement une partie <answer part> ou une partie <else part> terminale.

Le premier paragraphe de *Modèle* de *Décision* page 69 doit être modifié comme suit:

> Si une <decision> n'est pas une décision terminale, il s'agit donc d'une syntaxe dérivée pour une <decision> lorsque toutes les parties <answer part> non terminales et la partie <else part> si elle n'est pas terminale ont inséré à la fin de leur <transition> un branchement <join> vers la première instruction <action statement> qui suit la décision, ou si la décision est la dernière instruction <action statement> dans une chaîne <transition string> vers l'instruction <terminator statement> suivante.

Sous-paragraphe 4.10

Le modèle de l'entrée prioritaire est modifié. La sixième phrase du deuxième paragraphe de la page 105 est modifiée comme suit (les modifications sont soulignées):

> Les entrées prioritaires à l'état d'origine sont connectées au premier état, tandis que toutes les autres entrées, y compris les entrées prioritaires, sont connectées au second état (State 2). Toutes les entrées non prioritaires sont sauvegardées dans le premier état.

Sous-paragraphe 4.14

Le non-terminal <end> est supprimé dans la production corrigée (page 113):

```
<remote procedure save area> ::=
        <save symbol> ***contains***
        {[ <virtuality> ] procedure <remote procedure identifier list> }
```

Sous-paragraphe 5.3.1.11

(page 139) dans l'*Exemple* de *Modèle* la partie axioms doit être (supprimer une ")"):

```
    axioms
                Exor (a,b) == (a and (not b)) or ((not a) and b);
```

Sous-paragraphe 5.3.2

(page 147) Le non-terminal <sort> remplace extended sort dans la règle suivante:

```
<operator result> ::=
                returns [ <variable name> ] <sort>
```

Sous-paragraphe 5.4.6

La règle décrivant <external properties> devient:

```
<external properties> ::=
                alternative
                        <external formalism name> [ , <word> ] <end>
                        <external data description>
                [ endalternative ] <end>
```

Sous-paragraphe 6.1.2

(expression de type) dans la liste de cas dans la partie *Sémantique*, les mots "ou une définition de vue" du quatrième paragraphe doivent être supprimés car ils ne s'appliquent pas.

Sous-paragraphe 6.2

L'avant-dernier paragraphe de la *Grammaire textuelle concrète* (page 178) devient:

> La liaison d'un paramètre de contexte synonyme ou variable réel avec sa définition n'est pas résolue par le contexte mais par l'utilisation des règles de visibilité.

Sous-paragraphe 6.3.2

La Recommandation Z.100 définit où sont autorisés les types virtuels au sein d'un bloc virtuel, et les contraintes implicites pour chaque cas. Quelques situations pas claires ont été signalées. Pour clarifier ce problème, les deux phrases suivantes sont proposées comme ajout au sous-paragraphe 6.3.2.

Ajout aux conditions statiques de la *Grammaire textuelle concrète:*

> Si **inherits** et **atleast** sont tous les deux utilisés, alors le type hérité doit être identique ou être un sous-type de la contrainte.
>
> Dans le cas d'une contrainte implicite, la redéfinition impliquant **inherits** doit être un sous-type de la contrainte.

6 Extensions mineures à la Recommandation Z.100

Sous-paragraphe 2.2.2

Ajouter à la fin du paragraphe qui commence par "Il est permis d'omettre ...":

> Compte tenu de la visibilité, si 1'<identifier> ne contient pas de <qualifier>, une clause <package reference clause> est considérée comme l'unité de portée englobante la plus proche de l'unité de portée à laquelle elle est associée et contient les entités visibles par le progiciel.

Sous-paragraphe 2.2.6

Un symbole supplémentaire de commentaire optionnel est ajouté, entraînant une modification de la *Grammaire graphique concrète:*

<comment area> ::=

{<comment symbol> | <comment symbol2>} ***contains*** <text>

is connected to <dashed association symbol>

En outre, ajouter après la définition du <comment symbol>:

<comment symbol2> ::=

7 Harmonisation des signaux et des entités distantes

7.1 Objectif

Les définitions de procédures distantes et les définitions de variables distantes servent au même but que les définitions de signaux: définir des primitives de communication. Cependant, comparées aux signaux, il y a en SDL-92 deux limites à l'utilisation des procédures distantes et des variables distantes:

1) elles ne peuvent pas être mentionnées sur les interfaces (acheminements de signaux, canaux et accès);

2) elles ne peuvent pas être utilisées en communication avec l'environnement du système.

En SDL-96 les procédures distantes et les variables distantes sont harmonisées avec les signaux. Ceci augmente le pouvoir d'expression de SDL, et permet une spécification des interfaces plus cohérente (et ainsi rend SDL moins compliqué).

La capacité à communiquer avec l'environnement par l'utilisation de procédures distantes est nécessaire pour deux raisons:

1) lors de la construction de grands systèmes il doit être possible de voir une partie du système comme indépendante, de façon à appliquer facilement les principes de test, de réutilisation et d'encapsulation. Cette partie pourrait être typiquement une sous-structure de bloc, car elle ressemble à un système complet, mais considérer une sous-structure de bloc comme un sous-système pose un problème majeur, car les sous-structures de bloc, contrairement aux systèmes, peuvent communiquer avec leur environnement en utilisant des procédures distantes et des variables distantes;

2) lors de l'emploi de SDL pour des architectures de calcul distribuées telles que celle définie par le consortium TINA, le nombre de serveurs (qui peuvent être des systèmes SDL) offrant des interfaces aux clients (qui peuvent aussi être des systèmes SDL) peut varier au cours de l'exécution. L'utilisation de SDL-92 dans ce domaine est difficile.

7.2 Solution

Les procédures distantes et les variables distantes peuvent être spécifiées dans les listes de signaux, de sorte que, comme les signaux, elles peuvent être spécifiées dans les canaux, acheminements de signaux (signal routes), accès (gates), ensembles de signaux et sauvegardes. Les unités de portée qui ont des procédures distantes et des variables distantes sur les accès, canaux, ou acheminements de signal qui leur sont connectés n'ont pas besoin d'inclure de spécifications <imported procedure specification> et de spécifications <imported variable specification>. Pour communiquer des entités distantes avec l'environnement, elles doivent être mentionnées sur les canaux connectés à l'environnement du système.

Aussi, la partie <basic input part> est harmonisée avec la partie <basic save part> puisque en SDL-96 la partie <basic input part> consiste en une liste d'éléments <signal list item>. En d'autres termes, un identificateur de liste de signaux peut être mentionné dans une entrée en SDL-96. Ensemble ces deux extensions évitent de devoir définir les productions de sauvegarde <remote procedure save> et d'entrée <remote procedure input>.

Si un identificateur dans un élément <signal list item> peut à la fois dénoter un signal et une procédure/variable distante selon les règles de visibilité, l'identificateur dénote le signal, à moins que l'identificateur soit précédé par le mot clé **procedure** (pour une procédure distante) ou **remote** (pour une variable distante).

7.3 Exemple

signal s1, s2(integer);

remote procedure rp;

signallist slist = s1, s2, **procedure** rp; /* The keyword **procedure** is not needed if no visible signal is named rp */

...

gate g **in with** (slist);

...

input (slist);

...

7.4 Modifications

Sous-paragraphe 1.3.2

Première ligne: remplacer "qu'ils reçoivent du" par "qu'ils échangent avec le".

Troisième ligne: remplacer "signaux" par "stimuli".

Sous-paragraphe 2.1

Deuxième ligne: remplacer "par des signaux" par "au moyen de signaux et de procédures/variables distantes".

Sous-paragraphe 2.4.1.2

Dans la production <entity kind> remplacer

| procedure

par

| [remote] procedure

Sous-paragraphe 2.4.1.2

Ajouter après la production d'<interface>:

> **procedure** est utilisée pour sélectionner à la fois les procédures (normales) et les procédures distantes. Si une procédure normale et une procédure distante ont le même nom <name>, **procedure** dénote la procédure normale. Pour forcer <definition selection> à dénoter la procédure distante, le mot clé **procedure** peut être précédé par **remote**.

Sous-paragraphe 2.4.2

Troisième ligne de *Sémantique:* remplacer "ne peut se faire qu'au moyen de signaux. A l'intérieur d'un système, ces signaux sont véhiculés dans des canaux" par "a lieu par des signaux, des procédures distantes et des variables distantes. A l'intérieur d'un système, ces signaux sont véhiculés dans des canaux explicites ou implicites".

Sous-paragraphe 2.4.4

Page 36, ligne 4 depuis le haut: remplacer "aucun signal" par "aucun stimulus".

Page 36, ajouter après le paragraphe commençant par "Les signaux reçus par les instances de processus ...":

> "Appeler et répondre à des appels de procédures distantes et accéder à des variables distantes correspondent aussi à des échanges de signaux, voir 4.13 et 4.14."

Sous-paragraphe 2.5.1

Première ligne de *Sémantique:* remplacer "signaux" par "signaux (y compris les signaux implicites résultant des échanges de procédures distantes et de variables distantes, voir 4.13 et 4.14)."

Sous-paragraphe 2.5.2

Première ligne de *Sémantique:* remplacer "signaux" par "signaux (y compris les signaux implicites résultant des échanges de procédures distantes et de variables distantes, voir 4.13 et 4.14)."

Sous-paragraphe 2.5.2

Ajouter un paragraphe dans *Sémantique:*

> Les procédures distantes et les variables distantes n'ont pas besoin d'être mentionnées dans les listes de signaux des acheminements de signaux et des canaux entre l'importateur et l'exportateur, mais si une variable distante est mentionnée sur un canal ou sur un acheminement de signal, alors elle doit être mentionnée sur tout le trajet de communication entre l'importateur et l'exportateur. Une procédure ou variable importée est mentionnée sortant de l'importateur, et une procédure ou variable exportée est mentionnée entrant vers l'exportateur. Si une procédure ou variable importée n'est pas mentionnée dans l'acheminement de signal sortant d'un processus ou service ou mentionnée dans les accès d'un type de processus ou d'un type de service, alors elle doit être spécifiée dans une spécification <imported procedure specification> ou <imported variable specification> dans ce (type de) processus ou (type de) service.

Page 47, 4e ligne du 2e paragraphe, remplacer:

> "des sorties <output> et" par "des sorties <output>, des définitions <exported variable definition>, des préambules <procedure preamble>, des spécifications <imported procedure specification>, des spécifications <imported variable specification> et".

Page 47, 3e ligne du 3e paragraphe, remplacer:

> "des sorties <output> et" par "des sorties <output>, des définitions <exported variable definition>, des préambules <procedure preamble>, des spécifications <imported procedure specification>, des spécifications <imported variable specification> et".

Page 47, 2e ligne du 5e paragraphe, remplacer:

> "et les sorties <output>" par "et les sorties <output>, les définitions <exported variable definition>, les préambules <procedure preamble>, les spécifications <imported procedure specification>, les spécifications <imported variable specification> et".

Page 47, 4e ligne du 6e paragraphe, remplacer:

> "des sorties <output> et" par "des sorties <output>, des définitions <exported variable definition>, des préambules <procedure preamble>, des spécifications <imported procedure specification>, des spécifications <imported variable specification> et".

Page 47, 3e ligne du 7e paragraphe, remplacer:

> "une sortie" par "les sorties, spécifications de procédures importées et spécifications de variables importées".

Page 47, 3e ligne du 8e paragraphe, remplacer:

"et les sorties <output>" par "les sorties <output>, les définitions <exported variable definition>, les préambules <procedure preamble>, les spécifications <imported procedure specification>, et les spécifications <imported variable specification>".

Sous-paragraphe 2.5.5

Remplacer "identificateurs de signaux" par "identificateurs de signaux, procédures distantes, variables distantes".

Sous-paragraphe 2.5.5

Remplacer <signal list item> et le paragraphe suivant de la *Grammaire textuelle concrète* par:

<signal list item> ::=

<signal identifier>

| **(**<signal list identifier>**)**

| <timer identifier>

| [**procedure**] <remote procedure identifier>

| [**remote]** <remote variable identifier>

La liste <signal list> établie en remplaçant tous les identificateurs <signal list identifier> dans la liste par la liste des identificateurs <signal identifier> ou <timer identifier> qu'ils désignent et en remplaçant tous les identificateurs <remote procedure identifier> et tous les identificateurs <remote variable identifier> par un des signaux implicites que chacun d'eux désigne (voir 4.13 et 4.14), correspond à un ensemble *Signal-identifier*-**set** dans la *grammaire abstraite*.

Un élément <signal list item> qui est un <identifier> désigne un identificateur <signal identifier> ou un identificateur <timer identifier> si cela est possible selon les règles de visibilité, sinon un identificateur <remote procedure identifier> si cela est possible selon les règles de visibilité, sinon un identificateur <remote variable identifier>. Pour forcer un élément <signal list item> à désigner un identificateur <remote procedure identifier> ou un identificateur <remote variable identifier> le mot clé **procedure** ou **remote** peut respectivement être utilisé.

Sous-paragraphe 2.6.4

Grammaire textuelle concrète supprimer <input part> et renommer <basic input part> en <input part>.

Remplacer la *Grammaire textuelle concrète* de <stimulus> par:

<stimulus> ::=

<signal list item> [**(**[<variable>] {**,**[<variable>]}***)**]

Un élément <signal list item> ne doit pas désigner un identificateur <remote variable identifier>, et s'il désigne un identificateur <remote procedure identifier> ou un identificateur <signal list identifier>, les paramètres du <stimulus> (y compris les parenthèses) doivent être omis.

Sous-paragraphe 2.6.4

Ajouter à *Modèle:*

Un <stimulus> dont l'élément <signal list item> est un identificateur <signal list identifier> est une syntaxe dérivée pour une liste de <stimulus> sans paramètres et insérée dans la liste <input list> ou dans la liste <priority input list> englobante. Dans cette liste, il y a une correspondance univoque entre les <stimulus> et les membres de la liste de signaux.

Sous-paragraphe 2.6.4

Ajouter après le premier paragraphe de la *Grammaire textuelle concrète:*

Dans la *Grammaire abstraite*, les <remote procedure identifier> sont aussi représentés comme des *Signal-identifiers.*

Sous-paragraphe 2.6.4

Supprimer <input part> et renommer <basic input part> en <input part>.

Sous-paragraphe 2.6.4

Supprimer <input area> et renommer <basic input area> en <input area>.

Sous-paragraphe 2.6.5

Premier paragraphe: remplacer "identificateurs de signaux" par "identificateurs de signaux et identificateurs de procédures distantes".

Sous-paragraphe 2.6.5

Supprimer <save part> et renommer <basic save part> en <save part>.

Sous-paragraphe 2.6.5

Supprimer <save area> et renommer <basic save area> en <save area>.

Sous-paragraphe 4.2.2

Supprimer <remote procedure input area> et <remote procedure save area> comme alternatives dans <any area>.

Sous-paragraphe 4.13

Remplacer <import expression> par:

<import expression> ::=

import (<remote variable identifier> [**,**<destination>] [**via** <via path>]**)**

Sous-paragraphe 4.13

Modèle: première ligne, remplacer "acheminés sur des canaux et des acheminements de signaux implicites." par "acheminés sur des canaux et des acheminements de signaux implicites ou explicites." Les canaux et les acheminements de signaux sont explicites si la variable distante a été mentionnée dans la liste <signal list> (sortante pour l'importateur et entrante pour l'exportateur) d'au moins un accès, canal ou acheminement de signaux connecté à l'importateur ou à l'exportateur. Quand une variable distante est portée par des canaux ou acheminements de signaux explicites, le mot clé **nodelay** situé dans la définition <remote variable definition> est ignoré.

Sous-paragraphe 4.14

Grammaire textuelle concrète: supprimer <remote procedure input transition> et <remote procedure save>.

Sous-paragraphe 4.14

Remplacer <remote procedure call body> par:

<remote procedure call body> ::=

<remote procedure identifier> [<actual parameters>]

[**to** <destination>] [**via** <via path>]

Sous-paragraphe 4.14

Grammaire textuelle concrète ajouter:

Quand la <destination> et le <via path> sont omis, il y a une ambiguïté de syntaxe entre <remote procedure call body> et <procedure call>. Dans ce cas, l'<identifier> contenu désigne un identificateur <procedure identifier> si cela est possible selon les règles de visibilité, sinon un identificateur <remote procedure identifier>.

Sous-paragraphe 4.14

Grammaire graphique concrète: supprimer <remote procedure input area> et <remote procedure save area>.

Sous-paragraphe 4.14

Dernier paragraphe avant *Modèle:* remplacer deux occurrences de <remote procedure save> par <save> et deux occurrences de <remote procedure input transition> par <input part>.

Sous-paragraphe 4.14

Modèle: première ligne, remplacer "acheminés sur des canaux et des acheminements de signaux implicites." par "acheminés sur des canaux et des acheminements de signaux implicites ou explicites." Les canaux et les acheminements de signaux sont explicites si la procédure distante a été mentionnée dans la liste <signal list> (la sortante pour l'importateur et l'entrante pour l'exportateur) d'au moins un accès, canal ou acheminement de signaux connecté à l'importateur ou à l'exportateur. Quand une procédure distante est portée par des canaux ou acheminements de signaux explicites, le mot clé **nodelay** situé dans la définition <remote procedure definition> est ignoré.

Sous-paragraphe 4.14

Page 114, a): remplacer deux occurrences de "**to** destination" par "**to** destination **via** viapath".

Sous-paragraphe 4.14

Page 114: supprimer le paragraphe commençant par "Une transition <remote procedure input transition> ou une zone <remote procedure input area> ...".

Supprimer **remote procedure input** et **remote procedure save** du glossaire.

8 Procédures externes et opérateurs externes

8.1 Objectifs

SDL est utilisé à la fois comme un langage de spécification et d'implémentation. Cependant dans certaines situations, SDL ne doit pas ou ne peut pas être utilisé. Par exemple:

- quand il est pratique de réutiliser du code existant, écrit dans un autre langage;
- pour s'interfacer avec un programme externe (par exemple l'API Windows), et que ce programme externe ne définit pas un lien du langage vers SDL (typiquement, les programmes externes nécessitent d'utiliser C++ ou Pascal, dans les cas où une interface externe est supportée);
- quand du code de bas niveau est nécessaire, comme pour développer des pilotes; ou
- quand on utilise des pointeurs.

En autorisant les appels en SDL de procédures et d'opérateurs développés dans un autre langage que SDL, il est possible d'utiliser SDL comme langage d'implémentation dans davantage de cas.

8.2 Solution

Il est proposé d'introduire des procédures et des opérateurs externes en SDL. Comme pour les synonymes externes et les données externes, la sémantique d'appel d'une procédure externe ne sera pas du domaine de SDL.

Quand SDL est utilisé comme langage de spécification, les procédures et les opérateurs externes peuvent être utilisés pour exprimer que l'effet d'un appel donné n'est pas du domaine de la spécification. Quand SDL est utilisé comme langage d'implémentation, les procédures et les opérateurs externes peuvent être utilisés pour interfacer vers d'autres langages.

Le format d'une procédure ou d'un opérateur externe doit être le même que le format d'une référence vers une procédure ou un opérateur, le mot clé **referenced** étant remplacé par **external**. Une grammaire graphique spécifique ne doit pas être introduite pour la définition de procédures externes (comme pour les procédures distantes), mais pour permettre un contrôle total des types, les types des paramètres formels et du résultat doivent être donnés dans la définition (ceci est déjà le cas pour les références d'opérateurs en SDL-92).

8.3 Exemple

SDL n'intégrant pas la possibilité d'ordonner les chaînes de caractères, une fonction de bibliothèque C pourrait être ajoutée:

procedure strcmp **fpar** op1, op2 Characterstring; **returns** Integer; **external**;

8.4 Modifications

Modifications au **sous-paragraphe 2.4.6**:

Ajouter <external procedure definition> comme alternative dans <procedure definition>, c'est-à-dire:

<procedure definition> ::=

<external procedure definition> |

<procedure preamble>

procedure {<procedure name> | <procedure identifier> }

...

endprocedure [<procedure name> | <procedure identifier>] <end>

Ajouter à *Grammaire textuelle concrète:*

<external procedure definition> ::=

procedure <procedure name>

[<procedure signature>] **external** <end>

Une procédure externe ne peut être mentionnée ni dans une expression <type expression>, ni dans un paramètre <formal context parameter>, ni dans une contrainte <atleast constraint>.

Il n'existe pas de grammaire graphique correspondante pour une définition <external procedure definition>.

Ajouter à *Sémantique:*

Une procédure externe est une procédure dont le corps <procedure body> n'est pas contenu dans la description SDL. Conceptuellement, une procédure externe est censée être munie d'un corps <procedure body> et être transformée en définition <procedure definition> au cours de la transformation générique du système, voir sous-paragraphe 4.3.1.

Sous-paragraphe 5.3.2

Ajouter <external operator definition> comme alternative dans <operator definition>, c'est-à-dire:

<operator definition> ::=

<external operator definition> |

operator {<operator identifier> | <operator name> }

...

endoperator [<operator identifier> | <operator name>] <end>

Ajouter à *Grammaire textuelle concrète:*

<external operator definition> ::=

operator <operator name>

[<operator signature>] **external** <end>

Il n'existe pas de grammaire graphique correspondante pour <external operator definition>.

Ajouter à *Sémantique:*

Un opérateur externe est un opérateur dont le comportement n'est pas contenu dans la description SDL. Conceptuellement, un opérateur externe est censé être muni d'un comportement et être transformé en définition <operator definition> au cours de la transformation générique du système, voir sous-paragraphe 4.3.1.

9 Extensions et harmonisation des canaux et des acheminements de signaux

Les points concernant les extensions et l'harmonisation des canaux et des acheminements de signaux sont:

1) autoriser les canaux à être optionnels;
2) simplifier la spécification des canaux et des acheminements de signaux;
3) canaux/acheminements de signaux dont les deux extrémités sont connectées à la même unité de portée.

9.1 Autoriser les canaux à être optionnels

Comme pour les acheminements de signaux, spécifier les canaux pour une unité de portée donnée est optionnel en SDL-96.

Les raisons de cette extension sont:

- dans les systèmes contenant beaucoup de blocs, les canaux réduisent la lisibilité et rendent difficile la répartition sur plusieurs pages d'un système/sous-structure;
- pour des systèmes ouverts client/serveur, l'accent est davantage mis sur les interfaces offertes/ consommées par le client/serveur que sur les liens de communication (qui changent dynamiquement).

9.2 Simplification de la spécification des canaux et des acheminements de signaux

Aussi lorsque des canaux/acheminements de signaux sont spécifiés pour une unité de portée donnée, il est permis d'abandonner une partie de ses informations 'redondantes', ce qui simplifie les diagrammes:

- *Nom du canal/acheminement de signal*

  Souvent, le nom du canal ou de l'acheminement de signal est référencé nulle part. En particulier, ceci est le cas lors de la connexion à des instances basées sur des types car dans un tel cas il n'y a pas de connexion du canal vers l'acheminement de signal et l'identificateur de l'accès est utilisé dans **output via** au lieu du canal/acheminement de signal.

- *Listes de signaux*

  Souvent les listes de signaux d'un canal/acheminement de signal peuvent être obtenues à partir de ses extrémités. Ceci est le cas en particulier lors de la connexion d'instances basées sur des types et lorsque l'on connecte des instances 1 vers 1.

9.3 Canaux/acheminements de signaux connectés à la même unité de portée à ses deux extrémités

Autoriser un canal ou un acheminement de signal à être connecté à la même unité de portée à ses deux extrémités est un besoin déjà reconnu pour SDL-96 car il était noté comme problème ouvert pour SDL-92.

En SDL-96, les canaux et acheminements de signaux unidirectionnels (pas les bidirectionnels) peuvent être connectés à la même unité de portée à leurs deux extrémités (qui ne doit pas être **env**).

De tels canaux et acheminements de signaux doivent être unidirectionnels afin d'éviter les problèmes d'ambiguïté relatifs aux ensembles de blocs, sous-structures de canaux et connexions à des instances sans type.

9.4 Modifications

Ces modifications permettent:

- de connecter les deux extrémités d'un canal ou d'un acheminement de signal au même bloc, ensemble de processus ou service;
- d'omettre les canaux dans une unité de portée;
- d'omettre le nom d'un canal ou d'un acheminement de signal dans la définition d'un canal ou d'un acheminement de signal;
- d'omettre une ou les deux listes de signaux dans un canal ou un acheminement de signal;
- de mentionner le même canal ou acheminement de signal à plusieurs endroits sur la frontière d'un bloc, processus ou service.

Sous-paragraphe 2.4.4

Grammaire textuelle concrète: l'avant-dernier paragraphe est remplacé par:

> "L'utilisation de l'ensemble <valid input signal set> est précisée au 2.5.2, *Grammaire textuelle concrète*, et au 7.3."

Sous-paragraphe 2.4.5

Grammaire textuelle concrète: le dernier paragraphe est remplacé par:

> "Les ensembles complets <valid input signal set> (voir 7.3.1) de services valides à l'intérieur d'un processus doivent être disjoints. De plus, deux services dans le même processus ne doivent pas consommer des instances du même type de signal de temporisation."

Sous-paragraphe 2.5.1

Grammaire abstraite: dans le premier paragraphe après les règles de grammaire, remplacer la dernière phrase par:

> "Si les deux extrémités sont le même bloc, le canal doit être unidirectionnel (c'est-à-dire le second *Channel-path* dans *Channel-definition* doit être absent)."

Sous-paragraphe 2.5.1

Grammaire textuelle concrète: dans la règle de grammaire de <channel definition>, remplacer "<channel name>" par "[<channel name>]".

La règle de grammaire de <channel path> devient:

> <channel path>::=
>
> **from** <channel endpoint> **to** <channel endpoint>
>
> [**with** <signal list>] <end>

Le paragraphe suivant doit être ajouté immédiatement après les règles de grammaire:

> "Il est nécessaire de spécifier le nom <channel name> d'arrivée si le nom <channel name> d'origine est spécifié. Si le nom <channel name> d'arrivée n'est pas spécifié, le canal ne peut pas être référencé par son nom."

Le paragraphe suivant doit être ajouté immédiatement avant la *Grammaire graphique concrète:*

> "Il est permis de connecter un canal aux deux sens d'un accès bidirectionnel."

Sous-paragraphe 2.5.1

Grammaire graphique concrète: dans la règle de grammaire de <channel definition area>, "<channel name>" est remplacé par "[<channel name>]", et la première occurrence de "<signal list area>" est remplacée par "[<signal list area>]".

Dans le premier paragraphe après les règles de grammaire, la première occurrence de "une zone <signal list area>" est remplacée par "au plus une zone <signal list area>".

Sous-paragraphe 2.5.1

Sémantique: dans le premier paragraphe, remplacer la première phrase par:

> "Une définition *Channel-definition* est un moyen de transport pour les signaux (incluant les signaux implicites résultant de l'échange de procédures et variables distantes, voir 4.13 et 4.14)."

Supprimer le dernier paragraphe.

Sous-paragraphe 2.5.1

Modèle: remplacer tout le texte par:

> "*Noms de canaux implicites:* si le nom <channel name> est omis d'une définition <channel definition> ou d'une zone <channel definition area>, le canal est nommé implicitement et de façon unique, à moins que le canal soit associé à une sous-structure. Dans ce cas, le nom <channel name> est le même que le nom <channel substructure name>.

Canaux et liste de signaux implicites sur des canaux: si un système, un type de système, une sous-structure de bloc ou une sous-structure de canal ne contient pas de canaux explicites, des canaux implicites sans listes de signaux sont obtenus. Par la suite, des canaux sans liste de signaux explicites sont remplis avec des signaux, des procédures distantes et des variables distantes. Les détails de ceci sont décrits dans 7.3.3 et 7.3.5.

Si un système, un type de système, une sous-structure de bloc ou une sous-structure de canal contient des canaux explicites pour des signaux, mais pas de canaux explicites pour des procédures distantes et des variables distantes, des canaux implicites sont obtenus, capables de véhiculer ces procédures et variables distantes. Les détails de ceci sont décrits dans 7.3.4.

Les chemins <channel path> définis explicitement doivent posséder des listes de signaux non vides après ces transformations.

Canaux connectés à des ensembles de blocs basés sur des types: un canal dont les deux extrémités sont les accès d'une définition <textual typebased block definition> représente des canaux individuels depuis chacun des blocs de cet ensemble vers tous les blocs de cet ensemble, incluant le bloc lui-même. Tout canal bidirectionnel résultant connectant un bloc de l'ensemble au bloc lui-même est séparé en deux canaux unidirectionnels.

Un canal dont une extrémité est un accès d'une définition <textual typebased block definition> représente des canaux individuels vers ou depuis chaque bloc de l'ensemble.

Les canaux individuels de chaque canal original ont tous la même propriété de retard que le canal d'origine."

Sous-paragraphe 2.5.2

Grammaire abstraite: le deuxième paragraphe après les règles de grammaire est remplacé par:

"Si les deux points extrêmes sont le même ensemble de processus ou service, l'acheminement de signal doit être unidirectionnel (c'est-à-dire le second *Signal-route-path* dans *Signal-route-definition* doit être absent)."

Le paragraphe suivant doit être ajouté immédiatement avant la *Grammaire graphique concrète:*

"Un acheminement de signal peut connecter ensemble les deux directions d'un accès bidirectionnel."

Sous-paragraphe 2.5.2

Grammaire textuelle concrète: dans la règle de grammaire de <signal route definition>, "<signal route name>" est remplacé par "[<signal route name>]".

La règle de grammaire de <signal route path> devient:

<signal route path>::=

from <signal route endpoint> **to** <signal route endpoint>

[**with** <signal list>] <end>

Le paragraphe suivant est ajouté immédiatement après les règles de grammaire:

"Si le nom <signal route name> n'est pas spécifié, l'acheminement de signal ne peut pas être désigné par un nom."

Le texte suivant doit être ajouté avant *Sémantique:*

"Les règles suivantes s'appliquent si la définition <block definition> ou la définition <block type definition> qui englobent directement une définition <process definition> ne contiennent aucune définition <signal route definition>, ou si certains des chemins <signal route path> menant à la définition <process definition> ne contiennent pas de listes <signal list>:

- si la définition <process definition> ne contient pas de services, elle doit contenir un ensemble <valid input signal set> explicite;
- si la définition <process definition> contient des services, chaque définition <service definition> soit doit contenir un ensemble <valid input signal set> explicite, soit tous les chemins <signal route path> menant à la définition <service definition> doivent contenir des listes <signal list> explicites.

Une définition <service definition> doit contenir un ensemble <valid input signal set> explicite si la définition <process type definition> ou la définition <process definition> englobante ne contient pas de définition <signal route definition>, ou si certains des chemins <signal route path> menant à la définition <service definition> ne contiennent pas de liste <signal list>.

Si un ensemble <valid input signal set> d'une définition <process type definition>, <process definition>, <service type definition> ou <service definition> est spécifié explicitement, ce qui suit s'applique:

- l'ensemble <valid input signal set> doit être un sous-ensemble de l'ensemble <valid input signal set> complet (voir 7.3.1);

- dans le cas d'un type de processus ou de service, l'ensemble <valid input signal set> n'a pas besoin de mentionner les signaux qui sont mentionnés explicitement sur les accès entrants du type de processus ou de service. Dans le cas d'un ensemble de processus ou d'un service, l'ensemble <valid input signal set> n'a pas besoin de mentionner les signaux qui sont mentionnés explicitement sur les acheminements de signal menant à l'ensemble de processus ou au service;

- l'ensemble <valid input signal set> n'a pas besoin de mentionner les procédures distantes ou les variables distantes.

Si une procédure distante ou une variable distante n'est mentionnée dans aucun acheminement de signal sortant d'un ensemble de processus ou d'un service, ni dans aucun accès sortant d'un ensemble de processus ou d'un service, alors elle doit être mentionnée dans une spécification <imported procedure specification> ou dans une spécification <imported variable specification> de cet ensemble de processus, type de processus, service ou type de service."

Sous-paragraphe 2.5.2

Grammaire graphique concrète: dans la règle de grammaire <signal route definition area>, "<signal route name>" est remplacé par "[<signal route name>]", et la première occurrence de "<signal list area>" est remplacée par "[<signal list area>]".

Dans le second paragraphe après les règles de grammaire, la première occurrence de "une zone <signal list area>" est remplacée par "au plus une zone <signal list area>".

Sous-paragraphe 2.5.2

Sémantique: au premier paragraphe, la première phrase est remplacée par:

"Une définition *Signal-route-definition* est un moyen de transport pour les signaux (incluant les signaux implicites résultant de l'échange de procédures et variables distantes, voir 4.13 et 4.14)."

L'avant-dernier paragraphe est remplacé par:

"Lorsqu'une instance de signal est envoyée à une instance du même ensemble d'instances de processus, l'interprétation du nœud *Output-node* implique soit que le signal est appliqué directement dans l'accès d'entrée du processus destinataire, soit que le signal est envoyé à travers un acheminement de signaux qui connecte l'ensemble d'instances de processus à lui-même."

Le paragraphe suivant est ajouté:

"Une procédure distante ou une variable distante sur un acheminement de signaux est mentionnée comme sortant d'un importateur et rentrant vers un exportateur."

Sous-paragraphe 2.5.2

Modèle: tout le texte est remplacé par:

"*Noms implicites d'acheminement de signaux:* si le nom <signal route name> est omis dans une définition <signal route definition> ou dans une zone <signal route definition area>, l'acheminement de signaux est nommé de manière implicite et unique.

Acheminements de signaux implicites et listes de signaux implicites sur des acheminements de signaux: si un bloc, un type de bloc, un ensemble de processus (contenant des services) ou un type de processus (contenant des services) ne contient pas d'acheminements de signaux explicites, des acheminements de signaux implicites sans listes de signaux sont obtenus. Par la suite, les acheminements de signaux sans liste de signaux explicites sont remplis avec des signaux, des procédures distantes et des variables distantes. Les détails de ceci sont décrits dans 7.3.3 et 7.3.5.

Si un bloc, un type de bloc, un ensemble de processus (contenant des services) ou un type de processus (contenant des services) contient des acheminements de signaux explicites pour des signaux, mais pas d'acheminements de signaux explicites pour des procédures distantes et des variables distantes, des acheminements de signaux implicites sont obtenus, capables de véhiculer ces procédures et variables distantes. Les détails de ceci sont décrits dans 7.3.4.

Les chemins <signal route path> définis explicitement doivent posséder des listes de signaux non vides après ces transformations."

Sous-paragraphe 2.5.3

Grammaire textuelle concrète: le texte suivant est ajouté immédiatement après les règles de grammaire:

"Aucun canal ni acheminement de signaux ne doit être mentionné plus d'une fois dans une connexion <channel to route connection> ou une connexion <signal route to route connection>. Aucun acheminement de signaux ne doit être mentionné après le mot clé **and** dans plus d'une connexion <channel to route connection> ou d'une connexion <signal route to route connection> d'une unité de portée donnée.

Deux connexions <channel to route connection> ou <signal route to route connection> d'une unité de portée donnée doivent – avant le mot clé **and** – soit mentionner le même ensemble de canaux/acheminement de signaux, soit ne pas avoir de canaux/acheminement de signaux en commun."

Sous-paragraphe 2.5.3

Grammaire graphique concrète: le dernier paragraphe est supprimé.

Le paragraphe suivant est ajouté à la fin du sous-paragraphe:

"NOTE – A cause de la construction **connect**, un canal ou un acheminement de signaux (externe) qui peuvent être anonymes dans la version graphique de la spécification, peuvent devoir être nommés dans la version textuelle correspondante. Ceci est tout à fait analogue au cas des zones <merge area> dans les dessins de processus, services ou procédures.

Un outil qui convertit la version graphique d'une spécification en une version textuelle devrait ainsi être capable de générer des noms de canaux et d'acheminements de signaux implicites."

Sous-paragraphe 2.5.3

Le nouveau sous-paragraphe *Modèle* est ajouté:

"Si plus d'une connexion <channel to route connection> ou <signal route to route connection> d'une unité de portée mentionnent les mêmes canaux/acheminements de signaux avant le mot clé **and**, ces connexions sont fusionnées."

Sous-paragraphe 3.2.2

Grammaire textuelle concrète: l'avant-dernier paragraphe est remplacé par:

"Si une définition <block substructure definition> contient des définitions <channel definitions> (toutes avec des listes <signal list> explicites) et des définitions <textual typebased block definition>, chaque accès (contenant seulement des signaux) des définitions <block type definition> des définitions <textual typebased block definition> doit être connecté à au moins un canal.

Si une définition <block substructure definition> contient des définitions <channel definitions> (certaines ou toutes avec des listes <signal list> implicites ou mentionnant des procédures distantes ou des variables distantes) et des définitions <textual typebased block definition>, chaque accès des définitions <block type definition> des définitions <textual typebased block definition> doit être connecté à au moins un canal.

Les connexions <channel connection> doivent remplir des conditions statiques similaires aux connexions <channel to route connection> et aux connexions <signal route to route connection> (voir 2.5.3, *Grammaire textuelle concrète*)."

Le texte suivant est ajouté immédiatement avant la *Grammaire graphique concrète:*

> "Un chemin <channel path> explicite sans liste de signaux explicite doit avoir une extrémité à un bloc qui a à la fois une version non partitionée et partitionée. Dans ce cas, et après que des chemins <channel path> et des chemins <signal route path> sans liste <signal list> ont été obtenus (7.3.3 et 7.3.4), la liste <signal list> dans le chemin <channel path> peut être remplie (7.3.5) de deux manières différentes:
>
> - en tant que partie d'une séquence (de chemins <channel path> et de chemins <signal route path> sans listes <signal list> explicites) menant vers ou hors de la version *non partitionée*;
> - en tant que partie d'une séquence (de chemins <channel path> et de ...) menant vers ou hors de la version *partitionée*.
>
> La liste <signal list> résultante doit contenir les mêmes signaux, procédures distantes et variables distantes dans les deux cas."

Sous-paragraphe 3.2.2

Modèle: le paragraphe suivant est ajouté immédiatement après celui existant:

> "Si plus d'une connexion <channel connection> d'une sous-structure de bloc mentionnent les mêmes canaux avant le mot clé **and**, ces connexions <channel connection> sont fusionnées."

Sous-paragraphe 3.2.3

Grammaire textuelle concrète: le paragraphe immédiatement après la règle de grammaire de <channel substructure definition> est remplacé par:

> "Le nom <channel substructure name> après le mot clé **substructure** peut seulement être omis si la définition <channel definition> possède un nom <channel name> explicite. Dans ce cas, le nom <channel substructure name> est le même que le nom <channel name>."

Dans l'avant-dernier paragraphe: le texte ", et pour chaque point d'extrémité on doit avoir exactement une connexion <channel endpoint connection>" est supprimé.

Le texte suivant est inséré entre les deux derniers paragraphes:

> "Un canal peut contenir un chemin <channel path> sans liste de signaux explicite et être associé à une sous-structure de canal. Dans ce cas, et après que des chemins <channel path> et des chemins <signal route path> implicites sans liste <signal list> ont été obtenus (7.3.3 et 7.3.4), la liste <signal list> dans le chemin <channel path> peut être remplie (7.3.5) de trois différentes façons:
>
> - en tant que séquence (de chemin <channel path> et de chemin <signal route path> sans liste <signal list> explicites) "contournant" la sous-structure de canal;
> - en tant que séquence (de chemin <channel path> et ...) menant dans la sous-structure de canal;
> - en tant que séquence (de chemin <channel path> et ...) menant hors de la sous-structure de canal.
>
> La liste <signal list> résultante doit être la même dans les trois cas."

Sous-paragraphe 3.2.3

Grammaire graphique concrète: dans la règle de grammaire de <channel substructure diagram>, "+" est remplacé par "*".

Le paragraphe situé immédiatement après cette règle de grammaire est remplacé par:

> "Une occurrence d'identificateur <block identifier> ou **env** identifie une extrémité du canal partitioné. L'occurrence est placée hors du symbole <frame symbol> près de l'extrémité de certains ou de tous les sous-canaux associés au symbole <frame symbol>. Un symbole <channel symbol> dans le symbole <frame symbol> et qui lui est connecté indique un sous-canal."

Sous-paragraphe 3.2.3

Modèle: dans le paragraphe b), la dernière phrase est remplacée par: "représentées par une connexion <channel connection> dans laquelle l'identificateur <block identifier> ou **env** a été remplacé par le nouveau canal approprié."

Sous-paragraphe 3.3

Grammaire textuelle concrète: le texte suivant est ajouté au sous-paragraphe:

> "Dans chaque *partie délimitée par une limite de porte* (ce terme est expliqué en 7.3.4) d'une spécification, tous les canaux et toutes les listes de signaux sur les canaux doivent être explicites si la partie utilise les signaux à différents niveaux de raffinement."

Sous-paragraphe 4.13

Grammaire textuelle concrète: le premier paragraphe après les règles de grammaire est remplacé par:

> "L'utilisation de **nodelay** est décrite en 7.3.4 et 7.3.5."

Le quatrième paragraphe après les règles de grammaire est remplacé par:

> "Une variable distante mentionnée dans une expression <import expression> doit être dans l'ensemble complet de sortie (7.3.1) d'un type de processus, ensemble de processus, type de service ou ensemble de service englobant."

Sous-paragraphe 4.13

Modèle: dans le premier paragraphe, les deux premières phrases sont remplacées par:

> "Une opération d'import est modélisée par un échange de signaux définis implicitement."

Après le troisième paragraphe, le suivant est ajouté:

> "Sur chaque canal ou acheminement de signaux qui mentionne la variable distante, la variable distante est remplacée par *xQUERY*. Pour un tel canal ou acheminement de signaux, un nouveau canal ou acheminement de signaux est ajouté dans la direction opposée; ce canal ou acheminement de signaux transporte le signal *xREPLY*. Dans le cas d'un canal, le nouveau canal possède la même propriété de retard que celui d'origine."

Sous-paragraphe 4.14

Grammaire textuelle concrète: le premier paragraphe après les règles de grammaire est remplacé par:

> "L'utilisation de **nodelay** est décrite en 7.3.4 et 7.3.5."

Le quatrième paragraphe après les règles de grammaire est remplacé par:

> "Une procédure distante mentionnée dans un appel <remote procedure call> doit être dans l'ensemble complet de sortie (7.3.1) d'un type de processus, ensemble de processus, type de service ou ensemble de service englobant."

Sous-paragraphe 4.14

Modèle: dans le premier paragraphe, les deux premières phrases sont remplacées par:

> "Un appel de procédure distante est modélisé par un échange de signaux définis implicitement."

Après le deuxième paragraphe, le suivant est ajouté:

> "Sur chaque canal ou acheminement de signaux qui mentionne la procédure distante, la procédure distante est remplacée par *pCALL*. Pour chacun de ces canaux ou acheminements de signaux, un nouveau canal ou acheminement de signaux est ajouté dans la direction opposée; ce canal ou acheminement de signaux transporte le signal *pREPLY*. Dans le cas d'un canal, le nouveau canal possède la même propriété de retard que celui d'origine."

Sous-paragraphe 6.1.1.3

Sémantique: le 2e paragraphe est supprimé.

Sous-paragraphe 6.1.1.4

Sémantique: le 2e paragraphe est supprimé.

Sous-paragraphe 6.1.4

Grammaire textuelle concrète: dans le 3e paragraphe, ligne 4, le texte "si l'ensemble des signaux" est remplacé par "si l'ensemble des signaux (s'il est spécifié)".

Les 4e et 5e paragraphes sont remplacés par:

> "Si le type dénoté par <base type> dans une définition <textual typebased block definition> ou dans une définition <textual typebased process definition> contient des acheminements de signaux, les règles suivantes s'appliquent: pour chaque combinaison de (accès, signal, direction) définie par le type, le type doit contenir au moins un des acheminements de signaux qui – pour la direction donnée – mentionne **env** et l'accès et soit mentionne le signal soit n'a pas de liste <signal list> explicite associée. Dans le dernier cas, il doit être possible de dériver que l'acheminement de signaux peut transporter le signal dans la direction donnée.
>
> Si le type contient des acheminements de signaux mentionnant des procédures distantes ou des variables distantes, une règle similaire s'applique.
>
> Si une définition <block substructure definition> dans le type de bloc dénoté par <base type> dans une définition <textual typebased block definition> contient des canaux, les règles suivantes s'appliquent: pour chaque combinaison de (accès, signal, direction) définie par le type de bloc, le type doit contenir au moins un canal qui – pour la direction donnée – mentionne **env** et l'accès et soit mentionne le signal soit n'a pas de liste <signal list> explicite associée. Dans le dernier cas, il doit être possible de dériver que le canal peut acheminer le signal dans la direction donnée.
>
> Si le type de bloc contient des canaux mentionnant des procédures distantes ou des variables distantes, une règle similaire s'applique.
>
> Il est autorisé d'utiliser les chemins de communication explicites pour les signaux "normaux", mais il faut utiliser les chemins de communication implicites pour les procédures distantes et les variables distantes, même si la spécification définit explicitement et/ou utilise des accès qui mentionnent des procédures distantes ou des variables distantes.
>
> Il est permis de spécialiser les accès implicites rpv_gate explicitement dans un sous-type. Une spécialisation explicite rpv_gate est traitée comme la spécialisation de n'importe quel autre accès (par exemple, il est permis d'ajouter des signaux "normaux" à un rpv_gate).
>
> Dans un sous-type d'un type avec des chemins de communication explicites pour les signaux, mais avec des chemins de communication implicites pour des procédures distantes et des variables distantes, il est permis d'ajouter des chemins <channel path> ou des chemins <signal route path> explicites qui ne contiennent pas de liste <signal list> explicite, ou qui mentionnent des procédures distantes ou des variables distantes."

Sous-paragraphe 6.1.4

Modèle: le texte qui suit est ajouté au début:

> "*Accès implicites pour des procédures distantes et des variables distantes:* un accès implicite appelé rpv_gate est obtenu pour chaque type de bloc, type de processus et type de service qui (directement ou à travers des instances contenues) exporte des procédures distantes ou des variables distantes, et qui ne possède pas d'accès explicites mentionnant ces procédures distantes ou ces variables distantes. Les détails de ceci sont décrits dans 7.3.2.
>
> Après cette transformation, chaque procédure distante ou variable distante exportée ou importée doit être mentionnée sur au moins un accès (dans la direction appropriée) du type."

Le titre "*Transformation des accès:*" est ajouté au début du premier paragraphe du texte existant.

Sous-paragraphe 7

Un nouveau sous-paragraphe est ajouté:

7.3 Insertion d'accès implicites, de canaux, d'acheminements de signaux et de liste de signaux

Les transformations ci-dessous sont appliquées dans l'ordre.

7.3.1 Ensembles complets d'entrées et de sorties

<valid input signal set> complets: l'ensemble <valid input signal set> complet d'une définition <process type definition>, <process definition>, <service type definition> ou <service definition> est l'ensemble des signaux, procédures distantes et variables distantes qui:

- apparaissent dans l'ensemble <valid input signal set> complet du supertype, s'il existe, du type de processus ou de service (dans le cas d'une définition <process type definition> ou <service type definition>); ou
- sont mentionnés sur les accès entrants du type de processus ou de service (dans le cas d'une définition <process type definition> ou <service type definition>), ou sont mentionnés explicitement sur les acheminements de signaux menant à l'ensemble de processus ou au service (dans le cas d'une définition <process definition> ou <service definition>); ou
- sont exportés par les définitions de procédures et les définitions de variables dans le type de processus, l'ensemble de processus, le type de service ou le service (seulement pour les procédures distantes et les variables distantes).

Dans le cas d'une définition <process type definition> ou d'une définition <process definition> contenant des services, l'ensemble <valid input signal set> complet contient en outre les signaux, procédures distantes et variables distantes qui sont contenus dans les ensembles <valid input signal set> complets de ces services.

Ensembles d'entrées et de sorties complets: l'ensemble d'entrées complet d'une définition <process type definition>, <process definition>, <service type definition> ou <service definition> est son ensemble <valid input signal set> complet. L'ensemble de sorties complet d'une définition <process type definition>, <process definition>, <service type definition> ou <service definition> est l'ensemble de signaux, procédures distantes et variables distantes qui:

- apparaissent dans l'ensemble de sorties complet du supertype, s'il existe, du type de processus ou de service (dans le cas d'une définition <process type definition> ou <service type definition>); ou
- sont mentionnés sur les accès sortants du type de processus ou de service (dans le cas d'une définition <process type definition> ou <service type definition>), ou sont mentionnés explicitement sur les acheminements de signaux venant de l'ensemble de processus ou du service (dans le cas d'une définition <process definition> ou <service definition>); ou
- sont mentionnés dans les nœuds de sortie de l'ensemble de processus ou du service (dans le cas d'une définition <process definition> ou <service definition>) (seulement pour les signaux); ou
- sont mentionnés dans les spécifications de procédures importées et de variables importées dans le type de processus, ensemble de processus, type de service ou service (seulement pour les procédures distantes et variables distantes).

Dans le cas d'une définition <process type definition> ou d'une définition <process definition> contenant des services, l'ensemble complet de sorties contient en outre les signaux, procédures distantes et variables distantes qui sont contenus dans l'ensemble complet de sorties de ces services.

NOTE – Les ensembles complets d'entrées et de sorties décrits ici n'incluent pas les signaux de temporisation ou les signaux définis implicitement.

7.3.2 Accès implicites

Accès implicites pour des procédures distantes et des variables distantes: une définition <gate definition> implicite est ajoutée pour chaque type de bloc, type de processus et type de service qui:

- directement ou à travers des instances contenues, exporte ou importe des procédures distantes ou des variables distantes; et
- n'a pas d'accès explicite mentionnant ces procédures distantes ou variables distantes.

L'accès dénoté par cette définition <gate definition> a le nom rpv_gate, possède une partie entrante si le type exporte au moins une procédure distante ou une variable distante, et possède une partie sortante si le type importe au moins une procédure distante ou une variable distante. Chaque partie mentionne les procédures distantes et variables distantes qui communiquent dans la direction correspondante.

L'accès implicite rpv_gate (s'il est présent) et les chemins de communication implicites spécialement pour les procédures distantes et les variables distantes, sont ajoutés au supertype avant la fusion du contenu du supertype avec le sous-type.

Si le type hérite d'un tel accès implicite d'un autre type et exporte ou importe de nouvelles procédures distantes ou variables distantes, la nouvelle définition <gate definition> implicite contient le mot clé **adding** et mentionne les nouvelles procédures distantes et variables distantes.

7.3.3 Canaux et acheminements de signaux implicites

Dans la description ci-dessous, il est assumé que lorsque des canaux ou acheminements de signaux implicites sont obtenus, des constructions **connect** implicites sont aussi obtenues dans les cas appropriés.

Canaux implicites: si une définition <system definition> ou une définition <system type definition> ne contient pas de définition <channel definition> explicite, des définitions <channel definition> implicites sans liste <signal list> sont obtenues de telle sorte que chaque paire de "channel endpoints" possible est connectée dans chaque direction possible. Un "channel endpoint" signifie ici:

- une définition <block definition>;
- un accès d'une définition <textual typebased block definition> textuelle;
- l'environnement du système.

Cependant, aucune définition <channel definition> implicite ne connecte une définition <block definition> avec elle-même; plutôt, un bloc dénoté par une définition <block definition> communique avec lui-même via des chemins de communication implicites *internes*.

Ainsi, si une définition <block substructure definition> (d'une définition <block definition> ou <block type definition>) ou une définition <channel substructure definition> ne contient pas de définition <channel definition>, des définitions <channel definition> implicites sont obtenues de façon similaire. Ici, un "channel endpoint" a la même signification qu'au-dessus, ou une des suivantes:

- un canal extérieur connecté à l'unité de portée englobante (dans le cas d'une définition <block definition>);
- un accès de l'unité de portée englobante (dans le cas d'une définition <block type definition>);
- le canal "englobant" (dans le cas d'une définition <channel substructure definition>).

Les sous-structures imbriquées sont ainsi traitées récursivement de la même façon.

Tous les canaux implicites sont unidirectionnels et retardants.

Acheminements de signaux implicites: si une définition <block definition> ou une définition <block type definition> ne contient pas de définition <signal route definition> explicite, des définitions <signal route definition> implicites sans liste <signal list> sont obtenues de telle sorte que chaque paire de "signal-route endpoints" possible est connectée dans chaque direction possible. Un "signal-route endpoint" signifie ici:

- une définition <process definition>;
- un accès d'une définition <textual typebased process definition>;
- un canal connecté à l'unité de portée englobante (dans le cas d'une définition <block definition>);
- un accès de l'unité de portée englobante (dans le cas d'une définition <block type definition>);
- l'environnement du système (dans le cas d'une définition <block definition> définie comme un système).

Cependant, aucune définition < signal route definition> implicite ne connecte une définition <process definition> avec elle-même; plutôt, un ensemble de processus dénoté par une définition <process definition> communique avec lui-même via des chemins de communication *internes* (si décomposés en services) ou sans aucun chemin de communication (si non décomposé en services).

Tous les acheminements de signaux implicites sont unidirectionnels.

Ainsi, si une définition <process definition> ou une définition <process type definition> contient des services, mais pas d'acheminements de signaux, des définitions <signal route definition> implicites sont obtenues de façon similaire.

Canaux et acheminements de signaux implicites pour les procédures distantes: si une unité de portée contient des canaux ou acheminements de signaux explicites pour des signaux, mais pas de canaux ou acheminements de signaux explicites pour des procédures distantes et variables distantes, des canaux/acheminements de signaux implicites sont obtenus pour les procédures distantes et variables distantes seulement, dans l'unité de portée en question. Le sous-paragraphe 7.3.4 décrit dans quelles conditions précises et comment cela se produit.

7.3.4 Canaux et acheminements de signaux implicites pour des procédures distantes et des variables distantes

Dans ce qui suit, le terme *frontière d'accès* est adapté. Une frontière d'accès (gate-boundary) est la frontière d'une définition <system type definition>, <block type definition>, <process type definition> ou <service type definition>.

Une spécification SDL est donc constituée de parties délimitées par des frontières d'accès. Ces parties sont appelées *gate-boundary-delimited* dans la suite du texte.

Dans chaque partie gate-boundary-delimited, des canaux ou acheminements de signaux implicites sont obtenus, particulièrement pour les procédures distantes ou les variables distantes, si les conditions suivantes sont remplies (c'est-à-dire dans chaque partie gate-boundary-delimited, les procédures distantes et les variables distantes sont traitées de la même façon que les signaux "normaux", sauf si les conditions suivantes sont remplies):

- tous les chemins <channel path> et les chemins <signal route path> (il peut n'y en avoir aucun) dans la partie gate-boundary-delimited contiennent des listes <signal list> explicites;
- aucune liste <signal list> dans les chemins <channel path> ou les chemins <signal route path> de la partie ne mentionne une procédure distante ou une variable distante.

Dans ce cas, tous les canaux et acheminements de signaux implicites dans l'unité de portée affectée sont obtenus de façon similaire à celle décrite dans 7.3.3 et 7.3.5, mais avec la modification suivante: chaque chemin <channel path> ou <signal route path> implicite obtenu selon les règles de 7.3.3 est remplacé par deux ou trois chemins <channel path> ou <signal route path>. La liste <signal list> du chemin <channel path> ou du chemin <signal route path> "original" implicite et obtenue selon les règles de 7.3.5, est partagée en deux ou trois chemins <channel path> ou <signal route path> résultants comme suit:

- un chemin (path) est rempli seulement avec les signaux "normaux". Si le chemin est un chemin <channel path>, il est retardant. Ce chemin n'est *pas* obtenu si l'unité de portée englobante contient déjà des canaux ou des acheminements de signaux pour des signaux "normaux";
- un chemin est rempli seulement avec les procédures distantes et variables distantes qui ont été définies *sans* l'attribut **nodelay**. Si le chemin est un chemin <channel path>, il est retardant;
- un chemin est rempli seulement avec les procédures distantes et variables distantes qui ont été définies *avec* l'attribut **nodelay**. Si le chemin est un chemin <channel path>, il n'est *pas* retardant.

7.3.5 Listes de signaux implicites sur des canaux et des acheminements de signaux

Listes de signaux implicites sur des canaux et des acheminements de signaux: la spécification contient maintenant un nombre de séquences de chemins <channel path> et de chemins <signal route path>, chaque séquence ayant les propriétés suivantes:

- chaque chemin <channel path> ou <signal route path> (excepté le dernier) de la séquence est connecté au chemin <channel path> ou <signal route path> suivant;
- aucun des chemins <channel path> ou des chemins <signal route path> de la séquence ne contient de liste <signal list>;
- l'"extrémité" d'origine du premier chemin <channel path> ou <signal route path> est l'une des suivantes:
  - un nombre de chemins <channel path> ou <signal route path>, dont certains possèdent des listes <signal list> explicites,
  - un accès,
  - une définition <process definition> ne contenant pas de services,
  - une définition <service definition>,
  - l'environnement du système.

L'"extrémité" d'arrivée du dernier chemin <channel path> ou <signal route path> satisfait une condition similaire.

Les listes <signal list> dans les chemins <channel path> et <signal route path> dans les séquences sont maintenant remplies de façon à pouvoir véhiculer "autant" de types de signaux, d'appels de procédure distante et de demandes d'importation que possible. Cette dérivation utilise les ensembles d'entrées et de sorties de toutes les définitions <process definition> et <service definition> aux "extrémités" des séquences.

Ainsi, chaque séquence contribue pour l'intersection de l'ensemble de signaux, procédures distantes et variables distantes suivant, à toutes les listes <signal list> de la séquence:

- l'ensemble d'origine de la séquence (sans tenir compte si l'"extrémité" d'origine est l'environnement du système);
- l'ensemble de destination de la séquence (sans tenir compte si l'"extrémité" d'arrivée est l'environnement du système);
- l'ensemble de signaux, procédures distantes et variables distantes qui sont visibles tout au long de la séquence entière.

Lors du remplissage d'une procédure distante ou d'une variable distante sur un canal donné, la propriété delay/**nodelay** de la procédure distante ou de la variable distante est ignorée à moins que le canal ait été obtenu selon les règles de 7.3.4. Si un chemin <channel path> ou <signal route path> a été obtenu selon les règles de 7.3.4, la liste <signal list> sur ce chemin est remplie selon les règles additionnelles de 7.3.4.

Maintenant tous les chemins <channel path> et <signal route path> implicites dont les listes de signaux sont vides sont supprimés. Toutes les constructions **connect** qui deviennent ainsi "orphelines" sont aussi supprimées.

10 Utilisation d'un bloc ou d'un processus ou d'un service comme système

En SDL, un système est vu comme une entité indépendante qui communique avec son environnement par des signaux. Cependant, il existe souvent plus d'une vue. Considérons par exemple un ordinateur. Il peut être considéré comme un système par lui-même bien défini, mais il peut aussi être un composant d'un système plus grand (par exemple un réseau d'ordinateurs). Ainsi il est utile de permettre à un service, un processus ou un bloc d'être considérés comme un système à part entière. Cette possibilité est aussi utile pour des systèmes simples constitués, par exemple, d'un seul processus.

Une définition <system definition> peut ainsi être un seul (peut-être basé sur un type) bloc, processus ou service. Notez qu'une extension séparée décrite ailleurs permet à une unité de portée de posséder une clause <use clause> qui signifie qu'une telle définition <system definition> peut encore être basée sur des entités définies dans un progiciel.

Tous les acheminements de signaux/connexions à des canaux sont laissés pour le niveau le plus externe de processus/bloc.

10.1 Modifications

Sous-paragraphe 2.4.1.1

Grammaire concrète ajouter: <package list> et <system definition> ne peuvent pas être tous les deux omis. La liste <package list> peut être seulement spécifiée si la définition <system definition> est une définition <textual system definition> ou un diagramme <system diagram>.

Sous-paragraphe 2.4.1.3

Modifier la production <system definition> en:

<system definition> ::=

<textual system definition>

| <system diagram>

| <block definition>

| <block diagram>

| <textual typebased block definition>

| <graphical typebased block definition>

| <process definition>

| <process diagram>

| <textual typebased process definition>

| <graphical typebased process definition>

| <service diagram>

| <textual typebased service definition>

| <graphical typebased service definition>

Sous-paragraphe 2.4.1.3

Ajouter à *Modèle:*

1) Une définition <system definition> étant une définition <service definition> ou une définition <textual typebased service definition> est une syntaxe dérivée pour une définition <process definition> ayant le même nom que le service, contenant des acheminements de signaux implicites et contenant la définition <service definition> ou la définition <textual typebased service definition> comme seule définition.

   Une définition <system definition> étant un diagramme <service diagram> ou une définition <graphical typebased service definition> est une syntaxe dérivée pour un diagramme <process diagram> ayant le même nom que le service, contenant des acheminements de signaux implicites et contenant le diagramme <service diagram > ou la définition <graphical typebased service definition > comme seule définition.

2) Une définition <system definition> étant une définition <process definition> ou une définition <textual typebased process definition> est une syntaxe dérivée pour une définition <block definition> ayant le même nom que le processus, contenant des acheminements de signaux implicites et contenant la définition <process definition> ou la définition <textual typebased process definition> comme seule définition.

   Une définition <system definition> étant un diagramme <process diagram> ou une définition <graphical typebased process definition> est une syntaxe dérivée pour un diagramme <block diagram> ayant le même nom que le processus, contenant des acheminements de signaux implicites et contenant le diagramme <process diagram> ou la définition <graphical typebased process definition> comme seule définition.

3) Une définition <system definition> étant une définition <block definition> ou une définition <textual typebased block definition> est une syntaxe dérivée pour une définition <system definition> ayant le même nom que le bloc, contenant des canaux implicites et contenant la définition <block definition> ou la définition <textual typebased block definition> comme seule définition.

   Une définition <system definition> étant un diagramme <block diagram> ou une définition <graphical typebased block definition> est une syntaxe dérivée pour un diagramme <system diagram> ayant le même nom que le bloc, contenant des canaux implicites et contenant le diagramme <block diagram> ou la définition <graphical typebased block definition> comme seule définition.

Pour les deux conditions suivantes, le contenu de sous-structure contenu dans un diagramme <block diagram> ou une définition <block definition> est considéré comme directement contenu dans le diagramme <block diagram> ou dans la définition <block definition>:

Une zone <channel definition area> ou une zone <signal route definition area> pour un canal ou acheminement de signaux contenu directement dans une définition <system definition> ne peut pas être *associée à* des identificateurs <channel identifiers> ou à des identificateurs <external signal route identifiers>.

L'unité de portée contenue directement dans une définition <system definition> ne peut pas contenir de connexion <channel to route connection>, <channel connection> ou <signal route to route connection>.

11 Expressions d'état

11.1 Objectif

En liaison avec l'utilisation d'un état astérisque, il est souvent nécessaire d'avoir accès à l'information concernant l'identité de l'état dans lequel on est le plus récemment entré. Une telle possibilité fait partie de SDL-96.

11.2 Solution

Un nouvel opérateur impératif **state** a été introduit. Il renvoie une valeur chaîne de caractères contenant l'orthographe de l'état dans lequel on est le plus récemment entré.

11.3 Exemple

state *; /* Say covering among others the state named state1 */

input *;

...

decision state;

(state1) : **task** 'do something for state1';

else : **task** 'do something else';

enddecision;

11.4 Modifications

Sous-paragraphe 5.4.4

Ajouter une alternative dans <imperative operator>:

<imperative operator> ::=

<now expression>

| ...

| <anyvalue expression>

| <state expression>

Ajouter un nouveau **sous-paragraphe 5.4.4.7**:

Concrete textual grammar

<state expression> ::=

state

Modèle

<state expression> est une syntaxe dérivée pour un littéral Charstring qui contient l'orthographe en minuscules du nom de l'état dans lequel on est le plus récemment entré, dans la plus proche unité de portée englobante. Si un tel état n'existe pas, <state expression> renvoie la chaîne vide ('). La construction est transformée en même temps que <dash nextstate> (voir 7.1 étape 18).

12 Utilisation étendue des progiciels

12.1 Objectif

Couramment, les progiciels peuvent seulement être utilisés dans des progiciels et au niveau système. Le désavantage de ceci est que toutes les définitions d'un progiciel seront visibles dans toute la portée du système. Souvent, il est nécessaire d'inclure des définitions à un niveau plus bas. Par exemple, il est fréquent qu'un certain type de données ne soit nécessaire que dans un seul processus du système, mais que ce type soit cependant tellement général qu'il doive être réutilisé dans d'autres systèmes. Ainsi, il faut être capable d'utiliser des progiciels dans l'unité de portée où cela est nécessaire. Un tel changement facilitera aussi l'intégration de SDL et GDMO, comme GDMO permet l'emploi de progiciels sur des niveaux de classes d'objets gérés (ce qui pourra correspondre au niveau type de processus SDL).

12.2 Solution

Permettre l'emploi des progiciels dans d'autres unités de portée que le système, tels que blocs/types de bloc, processus/types de processus, services/types de service, etc.

12.3 Modifications

Page 18, paragraphe commençant par "Il est permis ...": après la phrase se terminant par "est le même que la partie la plus à droite du <qualifier> complet désignant cette unité de portée." ajouter la nouvelle phrase:

> Conformément à la visibilité et à l'utilisation des qualifieurs, une clause <package reference clause> associée à une unité de portée S est considérée comme représentant une définition de progiciel englobant S directement et définie dans l'unité de portée dans laquelle S est définie.
>
> NOTE – Dans la grammaire concrète, les progiciels ne peuvent pas être définis dans d'autres unités de portée. La règle ci-dessus sert seulement à définir les règles de visibilité qui s'appliquent aux progiciels. Une conséquence de cette règle est que l'on peut référencer les noms contenus dans un progiciel en utilisant différents qualifieurs, un pour chaque clause <package reference clause> du progiciel.

Page 24, 3e phrase, remplacer: "Les définitions à l'intérieur d'un progiciel sont rendues visibles à un système ou à un autre progiciel" par: "Les définitions à l'intérieur d'un progiciel sont rendues visibles à une autre unité de portée".

Page 25, dernière phrase avant "1)" et "2)" remplacer: "un autre <package>" ou la <system definition> par "une unité de portée".

Page 25, alinéa 1) remplacer: "Le <package> ou la <system definition> possède une <package reference clause> qui mentionne le <package> et" par:

> "L'unité de portée possède une <package reference clause> associée, ou toute unité de portée englobant l'unité de portée possède une <package reference clause> associée et la <package reference clause> mentionne le <package> et".

Page 26: supprimer le paragraphe commençant par "La référence aux noms ayant leur occurrence de définition dans un <package> se fait ...".

Page 26, *Modèle* ajouter à la fin:

> NOTE – Quand un type est spécialisé (voir 6.3), toute <package reference clause> associée au supertype n'est pas copiée vers le type spécialisé.

12.3.1 Modifications générales à la grammaire textuelle concrète

Ajouter "{<package reference clause>}*" au début de la partie droite des définitions de <block definition>, <process definition>, <service definition>, <procedure definition>, <operator definition>, <block substructure definition>, <channel substructure definition>, <system type definition>, <block type definition>, <process type definition>, et <service type definition>.

12.3.2 Modifications générales à la grammaire graphique concrète

Ajouter "{<package reference area>} *is associated with*" au début de la partie droite des définitions de <block diagram>, <process diagram>, <service diagram>, <procedure diagram>, <operator diagram>, <block substructure diagram>, <channel substructure diagram>, <system type diagram>, <block type diagram>, <process type diagram>, et <service type diagram>.

Ajouter la phrase "La zone <package reference area> doit être placée au sommet du symbole cadre représentant le système." dans les sous-paragraphes où les diagrammes mentionnés ci-dessus sont définis.

13 Opérateurs sans arguments dans la partie ***operators*** d'un newtype

13.1 Objectif

Le but original des opérateurs sans arguments (c'est-à-dire littéraux) était probablement de les utiliser pour contribuer à l'ensemble de valeurs d'une sorte (par exemple, pour définir des "valeurs énumérées"). Une facilité telle qu'**ordering** indique cela.

Une possibilité en SDL-92 est d'utiliser les synonymes à la place, mais ceci présente les inconvénients suivants:

- les synonymes ne peuvent pas être surchargés;
- les "signatures" des synonymes sont définies dans d'autres endroits de la spécification que les signatures des opérateurs non-nullaires correspondants ("hors" et "à l'intérieur" du **newtype**, respectivement). Si le **newtype** a des paramètres de contexte, cela rend impossible la définition du synonyme hors du **newtype**;
- le "comportement" des synonymes est défini à un autre endroit de la spécification que le comportement des opérateurs non-nullaires correspondants. Premièrement, les synonymes sont définis "hors" du **newtype**, alors que les axiomes ou les définitions algorithmiques d'opérateurs sont donnés "à l'intérieur" du **newtype**. Deuxièmement (dans le cas d'opérateurs algorithmiques), le comportement des opérateurs non-nullaires peut être défini dans des diagrammes d'opérateurs **referenced** – c'est-à-dire encore séparés du "comportement" des synonymes correspondants;
- les synonymes ne peuvent pas être utilisés dans des axiomes.

13.2 Solution

En SDL-96, les opérateurs sans arguments sont donc permis dans les parties **operators** d'un **newtype**. Contrairement aux opérateurs nullaires de la partie **literals**, les opérateurs nullaires de la partie **operators** ne sont pas influencés par des fonctionnalités telles que **ordering**. Au contraire, les mêmes fonctionnalités sont disponibles pour les opérateurs nullaires de la partie **operators** comme pour tous les autres opérateurs – par exemple, la possibilité de les définir algorithmiquement ou de manière externe et la possibilité de définir de "nouveaux" opérateurs nullaires de "vieux" types (par exemple, Pi du type Real).

13.3 Exemple

```
newtype Color

    literals Yellow, Blue, Green, Red;

    operators

        ordering;

        First: -> Color; /* not allowed  SDL-92  */

        Last: -> Color; /* not allowed  SDL-92 */

        /* other operators */

    axioms

        First == Yellow;

        Last  == Red;

        /* other axioms */

endnewtype;
```

les littéraux Yellow, Blue, Green, et Red constituent l'ensemble de valeurs de Color, et les quatre littéraux sont ceux ordonnés par **ordering**. Aujourd'hui, First et Last doivent aussi être définis dans **literals**, et l'ordonnancement doit donc être défini de façon beaucoup plus complexe (pour l'utilisateur) que de seulement indiquer **ordering** sous **operators**.

13.4 Impact sur la Recommandation Z.105

Cette extension permet un accès plus aisé aux opérateurs First et Last définis pour le type prédéfini Enumération dans la Recommandation Z.105. Comme les opérateurs en SDL-92 ne peuvent pas avoir zéro argument, il faut donner un argument factice à ces deux opérateurs pour les utiliser.

Pour la version de Z.105 qui correspond à SDL-96, les deux opérateurs sont surchargés:

```
First: -> Enumeration

Last: -> Enumeration.
```

13.5 Modifications

Sous-paragraphe 5.2.2

Titre du sous-paragraphe: le titre du sous-paragraphe est modifié en "**Littéraux et opérateurs**".

Sous-paragraphe 5.2.2

Grammaire abstraite: les règles de grammaire de *Signature, Literal-signature, Operator-signature* et *Argument-list* sont remplacées par:

Signature :: Operator-name

Argument-list

Result

*Argument-list = Sort-reference-identifier**

La règle de grammaire de *Literal-operator-name* est effacée.

Sous-paragraphe 5.2.2

Grammaire textuelle concrète: la règle de grammaire de <literal signature> est remplacée par:

<literal signature> ::=

<literal name> |

<name class literal>

<literal name> ::=

<literal operator name> |

<extended literal name>

La règle de grammaire de <argument list> est modifiée en:

<argument list> ::=

[<argument sort> { , <argument sort> }*]

L'ensemble du texte après les règles de grammaire est remplacé par:

"Les alternatives <name class literal>, <extended literal name>, <ordering>, <noequality>, <extended operator name>, <generator sort> et <syntype> ne font pas partie du noyau de données.

Une *Signature* est représentée par une signature <literal signature> qui contient un nom <literal name> qui n'est pas un nom <generator formal name>, ou par une signature <operator signature> qui contient un nom <operator name> qui n'est pas un nom <generator formal name>.

Si la *Signature* est représentée par une signature <literal signature>, l'*Argument-list* est vide, et le résultat *result* est la sorte introduite par la définition <partial type definition> définissant la signature <literal signature>. Si la *Signature* est représentée par une signature <operator signature>, chaque *Sort-reference-identifier* dans l'*Argument-list* est représenté par une sorte <argument sort> dans la liste <argument list>, et le résultat *result* est représenté par <result>.

Les autres formes possibles de signature <literal signature> et de signature <operator signature> sont des raccourcis. Chacun des ces raccourcis correspond en général à plusieurs *Signatures*.

Un nom <literal name> ou un nom <operator name> qui n'est pas un nom <generator formal name> correspond à un *Operator-name*. Cet *Operator-name* est unique dans la définition de l'unité de portée même si le nom <literal name> ou le nom <operator name> correspondant peut ne pas être unique.

L'unique *Operator-name* est obtenu à partir:

1) du nom <literal name> ou du nom <operator name>, et
2) de la liste (pouvant être vide) d'identificateurs d'arguments de sorte, et
3) de l'identificateur de sorte résultant, et
4) de l'identificateur de sorte de la définition partielle de type dans laquelle le nom <literal name> ou le nom <operator name> est défini.

Chaque fois qu'un identificateur <literal identifier> ou un identificateur <operator identifier> est spécifié, on obtient l'unique *Operator-name* dans *Operator-identifier* de la même façon avec la liste des sortes d'arguments et de la sorte résultante obtenue à partir du contexte. Deux opérateurs (chacun d'entre eux pouvant être un littéral ou un opérateur "normal") ayant le même nom, mais différant par un ou plus des sortes des arguments ou du résultat, ont des *Operator-names* différents.

Chaque fois qu'un <qualifier> ou un identificateur <literal identifier> ou un identificateur <operator identifier> contient un élément <path item> avec le mot clé **type**, le nom <sort name> après ce mot clé ne fait pas partie du *Qualifier* de l'*Operator-identifier*, mais est utilisé pour obtenir le *Nom* unique de cet *Identifier*. Dans ce cas, le *Qualifier* est formé à partir de la liste des éléments <path item> précédant le mot clé **type**."

Sous-paragraphe 5.2.2

Sémantique: les 3e et 4e paragraphes sont modifiés en:

"Un opérateur (sans arguments) défini dans une signature <literal signature> est appelé littéral. Un opérateur sans arguments et défini dans une signature <operator signature> est appelé un opérateur nullaire.

Un littéral ou un opérateur nullaire représente une valeur fixe appartenant à la sorte résultante de l'opérateur."

Le nouveau paragraphe suivant est inséré immédiatement avant le dernier:

"NOTE 1 – Du point de vue de la grammaire abstraite, il n'y a pas de différence entre les littéraux et les opérateurs nullaires. La différence repose dans leur utilisation en SDL: certaines possibilités, par exemple **ordering**, s'appliquent uniquement aux littéraux, pas aux opérateurs nullaires. Les opérateurs nullaires sont traités comme des opérateurs ayant un ou davantage d'arguments – par exemple, leur comportement peut être défini par le biais des définitions <operator definition>."

La Note existant dans la version 1993 est modifiée en:

"NOTE 2 – A titre de directive: une signature <operator signature> doit *normalement* mentionner la sorte introduite par la définition <partial type definition> englobante comme étant soit une sorte <argument sort> soit un <result>. L'exception à cette directive est le cas où une définition <partial type definition> est seulement utilisée comme "conteneur" ("mini-progiciel") pour grouper des "nouveaux" opérateurs sur des "vieilles" sortes quand ces opérateurs sont d'une certaine façon liés. Le nom <sort name> défini par une telle définition <partial type definition> doit seulement être utilisé dans des <qualifier> des opérateurs définis."

Le nouveau paragraphe suivant est ajouté à la fin:

"NOTE 3 – A titre de directive: un opérateur (sans arguments) qui est utilisé pour contribuer à l'ensemble de valeurs d'une sorte, doit être défini comme un littéral. Un opérateur (sans arguments) qui est utilisé pour définir un nom pour une valeur "existante", doit être défini comme un opérateur nullaire. Par exemple, cette directive a été utilisée pour définir la sorte énumérée suivante:

```
newtype Color
    literals
        Red, Green, Blue, Yellow;
    operators
        ordering;
        First: -> Color;
        Last: -> Color;
        /* other operators */
    axioms
        First == Red;
        Last  == Yellow;
        /* other axioms */
endnewtype Color;
```

"

Sous-paragraphe 5.2.3

Grammaire abstraite: la règle de grammaire de *Ground-term* est modifiée en:

*Ground-term :: Operator-identifier Ground-term** |

Conditional-ground-term

La règle de grammaire de *Literal-operator-identifier* est supprimée.

Sous-paragraphe 5.2.3

Grammaire textuelle concrète: la règle de grammaire de <ground term> est modifiée en:

<ground term> ::=

<literal operator identifier> |

<operator identifier> ([<ground term>] { , <ground term> }*) |

(<ground term>) |

<extended ground term>

Dans le paragraphe commençant par "c)", "*literal operator identifier*" est modifié en "*Operator-identifier*", et "littéral" est remplacé par "littéral ou opérateur".

Dans le 4e paragraphe en partant de la fin, "<operator name>" est modifié en "<literal name> ou <operator name>" (deux occurrences), et "opérateur" est modifié en "littéral ou opérateur" (trois occurrences).

Dans l'avant-dernier paragraphe, "*operator identifier* ou *literal operator identifier*" est modifié en "*Operator-identifier*".

Le dernier paragraphe est modifié en:

"NOTE – A titre de directive: un axiome doit appartenir aux littéraux et aux opérateurs de la définition partielle de type englobante en mentionnant un littéral ou un opérateur dont la signature est définie dans la définition partielle de type. Un axiome ne doit être défini qu'une seule fois."

Sous-paragraphe 5.3.1

Grammaire textuelle concrète: la règle de grammaire de <extended ground term> est modifiée en:

<extended ground term> ::=

<extended literal identifier> |

<extended operator identifier> ([<ground term>] { , <ground term> }*) |

<ground term> <infix operator> <ground term> |

<monadic operator> <ground term> |

<conditional ground term>

L'alternative <name class literal> est supprimée de la règle de grammaire de <extended literal name>.

Sous-paragraphe 5.3.1.2

Grammaire textuelle concrète: dans le 2e paragraphe après les règles de grammaire, "*Literal-operator-identifier*" est modifié en "*Operator-identifier*". Dans le 3e paragraphe, "*Literal-operator-name*" est modifié en "*Operator-name*".

Sous-paragraphe 5.3.1.7

Grammaire abstraite: dans le dernier paragraphe, "*literal operator identifier*" est modifié en "*Operator-identifier* correspondant à un littéral".

Sous-paragraphe 5.3.1.7

Modèle: le paragraphe suivant est ajouté à la fin:

"NOTE – Le mot clé **ordering** n'affecte pas les opérateurs nullaires définis dans les signatures <operator signature>."

Sous-paragraphe 5.3.1.11

Modèle: le paragraphe suivant est ajouté à la fin:

"NOTE 2 – Les littéraux hérités sont renommés au moyen de <literal renaming>. Les opérateurs nullaires hérités sont manipulés par le mot clé **all** et sont renommés par le biais de la liste <inheritance list>, tout comme les autres opérateurs "normaux" hérités."

(La note existant dans la version 1993 devient NOTE 1.)

Sous-paragraphe 5.3.1.15

Modèle: le paragraphe suivant est ajouté à la fin:

"NOTE – Les mappings de littéraux n'affectent pas les opérateurs nullaires définis dans les signatures <operator signature>."

Sous-paragraphe 5.3.2

Grammaire textuelle concrète: dans la règle de grammaire d'<operator definition>, "<formal parameters> <end>" est remplacé par "[<formal parameters> <end>]". Dans la règle de grammaire de <textual operator reference>, "<formal parameters>" est remplacé par "[<formal parameters>]".

Dans le 3e paragraphe après les règles de grammaire, le texte "(s'ils sont présents)" est ajouté après "<formal parameters>".

Dans le 5e paragraphe, "<formal parameters>" est remplacé par "[<formal parameters>]" (deux occurrences).

Sous-paragraphe 5.3.2

Grammaire graphique concrète: dans la règle de grammaire de <operator heading>, "<formal parameters>" est remplacé par "[<formal parameters>]".

Sous-paragraphe 5.3.2

Modèle: le paragraphe suivant est ajouté à la fin:

"NOTE – Il n'est pas possible de spécifier une définition <operator definition> pour une signature <literal signature>."

Sous-paragraphe 5.3.3.2

Grammaire textuelle concrète: la règle de grammaire de <ground primary> est modifiée en:

<ground primary> ::=

<literal identifier> |

<operator identifier> ([<ground expression list>]) |

(<ground expression>) |

<conditional ground expression>

14 Reformulation du sous-paragraphe 6.3.2 concernant les virtuels

Un type de bloc, type de processus, type de service ou une procédure peuvent être spécifiés comme types virtuels quand ils sont définis localement à un autre type (dénoté comme le type *englobant* dans ce sous-paragraphe). Un type virtuel peut être redéfini dans des spécialisations du type englobant.

Grammaire textuelle concrète

<virtuality> ::=

virtual | **redefined** | **finalized**

<virtuality constraint> ::=

atleast <identifier>

<virtuality> et <virtuality constraint> font partie de la définition de type. Voir sous-paragraphes 6.1.1.2 (type de bloc), 6.1.1.3 (type de processus), 6.1.1.4 (type de service) et 2.4.6 (procédure).

Un type virtuel est un type possédant **virtual** ou **redefined** comme <virtuality>. Un type redéfini est un type possédant **redefined** ou **finalized** comme <virtuality>.

Chaque type virtuel possède une contrainte de virtualité associée qui est un <identifier> de la même sorte d'entité que le type virtuel. Si la contrainte <virtuality constraint> est spécifiée, la contrainte de virtualité est l'<identifier> contenu, sinon la contrainte de virtualité est obtenue comme décrit plus bas.

Un type virtuel et sa contrainte ne peuvent pas posséder de paramètres formels de contexte.

Seuls les types virtuels peuvent avoir une contrainte <virtuality constraint> spécifiée.

Si <virtuality> est présente à la fois dans la référence et dans la définition référencée, les deux doivent être égales. Si le préambule <procedure preamble> est présent à la fois dans la référence de procédure et dans la définition de procédure référencée, les deux doivent être égaux.

Un type virtuel doit avoir exactement les mêmes paramètres formels, accès et signaux sur les accès que sa contrainte.

Sémantique

Un type virtuel peut être redéfini ...

L'accès à un type virtuel au moyen d'un ...

Un type virtuel ou redéfini qui est spécifié sans <specialization> peut posséder une <specialization> implicite. La contrainte de virtualité et la <specialization> implicite possible sont obtenues comme suit:

Etant donné un type virtuel V avec une contrainte <virtuality constraint> VA et une spécialisation VS et étant donné un type R redéfini de V avec une contrainte <virtuality constraint> RA et une <specialization> RS, ce qui suit s'applique (toutes les règles sont appliquées, et ce dans l'ordre donné):

- si VA est omise, alors VA est identique à V
- si VS est omise et VA ne dénote pas V, alors VS est identique à VA
- si VS est présente, alors VS doit être identique ou être un sous-type de VA
- si RA est omise alors RA est identique à R
- si RS est omise alors RS est identique à VA
- RA doit être identique ou être un sous-type de VA
- RS doit être identique ou être un sous-type de RA
- si R est un type virtuel, les mêmes règles s'appliquent pour R et pour V.

Un sous-type d'un type virtuel est un sous-type du type virtuel original et non pas d'une possible redéfinition.

15 Maintenance du SDL

Le présent paragraphe décrit la terminologie et les règles pour la maintenance du SDL agréées à la réunion de novembre 1993 de la Commission d'études 10, et la "procédure de demande de modification associée".

15.1 Terminologie

15.1.1 **erreur**: incohérence interne au sein de la Recommandation Z.100.

15.1.2 **correction textuelle**: modification apportée à des textes ou à des diagrammes de la Recommandation Z.100 qui corrige des erreurs d'écriture ou de typographie.

15.1.3 **question ouverte**: problème identifié mais pas résolu. Une question ouverte peut être identifiée soit par une demande de modification , soit par acceptation de la Commission d'études ou du Groupe de travail.

15.1.4 **déficience**: problème identifié où la sémantique de SDL n'est pas (clairement) définie par la Recommandation Z.100.

15.1.5 **clarification**: modification apportée au texte ou aux diagrammes de la Recommandation Z.100 qui clarifie du texte ou des diagrammes existants qui pourraient être compris de manière ambiguë sans la clarification. La clarification doit viser à faire correspondre la Recommandation Z.100 à la sémantique de SDL telle que comprise par la Commission d'études ou le Groupe de travail.

15.1.6 **modification**: changement apporté au texte ou aux diagrammes de la Recommandation Z.100 qui change la sémantique de la SDL.

15.1.7 **caractéristique déclassée**: caractéristique de langage SDL qui devra être supprimée dans la prochaine révision de la Recommandation Z.100.

15.1.8 **extension**: nouvelle caractéristique qui ne doit pas changer la sémantique des caractéristiques définies dans la Recommandation Z.100.

15.2 Règles de maintenance

Dans le texte qui suit, les références à la Recommandation Z.100 doivent être considérées pour inclure ce supplément et la Recommandation Z.105.

1) Quand une erreur ou une déficience est détectée dans la Recommandation Z.100, elle doit être corrigée ou clarifiée. La correction d'une erreur doit impliquer le moins de changements possible. Les corrections d'erreurs et les clarifications seront placées dans la liste principale des corrections à la Recommandation Z.100 et prennent effet immédiatement.

2) Excepté pour les corrections d'erreurs et la résolution de questions ouvertes de la période d'études précédente, les modifications et les extensions à SDL doivent être seulement considérées comme le résultat d'une demande de changement par une importante communauté d'utilisateurs. Une demande de modification doit être suivie d'une investigation par la Commission d'études ou le Groupe de travail en collaboration avec les représentants du groupe des utilisateurs, de façon que la nécessité et le bénéfice soient clairement établis et qu'il soit certain qu'une caractéristique existante du SDL ne conviendrait pas.

3) Les modifications et extensions ne résultant pas d'une correction d'erreur doivent être largement diffusées et les points de vue d'utilisateurs et de fabricants d'outils minutieusement examinés avant que la modification ne soit adoptée. A moins qu'il y ait des circonstances particulières nécessitant d'implémenter ces changements dès que possible, de tels changements ne seront pas recommandés jusqu'à la Recommandation Z.100.

4) Jusqu'à ce qu'une Recommandation Z.100 révisée soit publiée, une liste principale des corrections de la Recommandation Z.100 sera maintenue couvrant la Recommandation Z.100 et toutes les annexes excepté la définition formelle. Des appendices, addenda ou suppléments seront publiés sur décision de la Commission d'études. Pour s'assurer de la distribution effective de la liste principale des corrections à la Recommandation Z.100, elle sera publiée comme Rapports COM et par les moyens électroniques appropriés.

5) Pour les déficiences de la Recommandation Z.100, la définition formelle sera consultée. Ceci pourra mener soit à une clarification soit à une correction, qui est notée dans la liste principale des corrections de la Recommandation Z.100.

15.3 Procédure de demande de modification

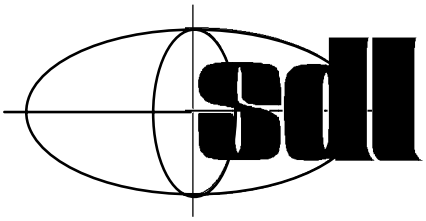
La procédure de demande de modification est conçue pour permettre aux utilisateurs du SDL dans et au dehors de l'UIT-T de poser des questions sur la signification précise de la Recommandation Z.100, de faire des suggestions pour des changements au SDL ou à la Recommandation Z.100, et pour fournir un retour sur les modifications proposées au SDL. Les modifications proposées au SDL seront publiées par le groupe d'experts SDL avant qu'elles soient implémentées.

Les demandes de modifications peuvent soit utiliser le formulaire de demande de modification (voir plus bas) soit fournir les informations énumérées dans le formulaire. Le type de requête doit être clairement indiqué (correction d'erreur, clarification, simplification, extension, modification ou caractéristique désengagée). Il est aussi important que pour tout changement autre qu'une correction d'erreur, le nombre d'utilisateurs représentés pour cette requête soit indiqué.

Toutes les demandes de modification seront traitées par des réunions du Groupe de travail de l'UIT-T responsable de la Recommandation Z.100. Pour des corrections ou clarifications les changements peuvent être insérés dans la liste de corrections sans consulter les utilisateurs. Sinon une liste de points ouverts est compilée. Les informations doivent être distribuées aux utilisateurs:

- comme rapports blancs de contribution de l'UIT-T;
- via la SDL Newsletter (ISSN 1023-7151);
- courrier électronique à des listes d'envoi SDL (comme "SDL_News");
- autres moyens agréés par les experts de la Commission d'études 10.

Les réactions des utilisateurs seront évaluées par les experts de la Commission d'études 10 pour déterminer le niveau de support et d'opposition à chaque changement. Un changement sera mis sur la liste des changements acceptés uniquement s'il y a un soutien de la part de beaucoup d'utilisateurs et pas d'objections sérieuses à la proposition de la part de plus de seulement quelques utilisateurs. Finalement tous les changements acceptés seront incorporés dans une Recommandation Z.100 révisée. Les utilisateurs doivent être conscients que, jusqu'à ce que les changements aient été incorporés et approuvés par la Commission d'études responsable de la Recommandation Z.100, ils ne sont pas recommandés par l'UIT-T.



Formulaire de demande de modification

Prière de fournir les informations suivantes		
Type de modification:	q correction d'erreur	q clarification
	q simplification	q extension
	q modification	q désengagement
Brève description de la demande de modification		
Brève justification de la demande de modification		
Ce point de vue est-il partagé dans votre organisme?	q oui	q non
Avez-vous consulté d'autres utilisateurs?	q oui	q non
Combien d'utilisateurs représentez-vous?	q 1-5	q 6-10
	q 11-100	q plus de 100
Votre nom et votre adresse		

prière de joindre d'autres pages avec des détails si nécessaire

SDL (Z.100) Rapporteur, c/o UIT-T, Place des Nations, CH-1211, Genève 20, Switzerland.
Fax: +41 22 730 5853, e-mail: SDL.rapporteur@itu.ch

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	Maintenance: 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**