UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T Z.120

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

(03/93)

CRITÈRES D'UTILISATION ET D'APPLICABILITÉ DES TECHNIQUES DE DESCRIPTION FORMELLE

DIAGRAMME DE SÉQUENCES DE MESSAGES

Recommandation UIT-T Z.120

(Antérieurement «Recommandation du CCITT»)

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 que les Commissions d'études de l'UIT-T doivent examiner et à propos desquels elles doivent émettre des Recommandations.

La Recommandation UIT-T Z.120, élaborée par la Commission d'études X (1988-1993) de l'UIT-T, a été approuvée par la CMNT (Helsinki, 1-12 mars 1993).

NOTES

1 Suite au processus de réforme entrepris au sein de l'Union internationale des télécommunications (UIT), le CCITT n'existe plus depuis le 28 février 1993. Il est remplacé par le Secteur de la normalisation des télécommunications de l'UIT (UIT-T) créé le 1er mars 1993. De même, le CCIR et l'IFRB ont été remplacés par le Secteur des radiocommunications.

Afin de ne pas retarder la publication de la présente Recommandation, aucun changement n'a été apporté aux mentions contenant les sigles CCITT, CCIR et IFRB ou aux entités qui leur sont associées, comme «Assemblée plénière», «Secrétariat», etc. Les futures éditions de la présente Recommandation adopteront la terminologie appropriée reflétant la nouvelle structure de l'UIT.

2 Dans la présente Recommandation, le terme «Administration» désigne indifféremment une administration de télécommunication ou une exploitation reconnue.

TABLE DES MATIÈRES

RÉSUMÉ

Champ d'application/Objectifs

L'objet de la recommandation diagramme de séquences de messages (MSC) (*message sequence chart*) est de fournir un langage de trace pour la spécification et la description du comportement de la communication entre les composants d'un système et leur environnement au moyen d'échanges de messages. Puisque la représentation du comportement d'une communication à l'aide des MSC est faite de manière intuitive et claire, en particulier au niveau de la représentation graphique, le langage MSC est facile à apprendre, à pratiquer et à interpréter. Il peut être utilisé en relation avec d'autres langages en tant que support méthodologique pour la spécification, la conception, la simulation, le test et la documentation de système.

Contenu

La présente Recommandation contient une définition de la syntaxe des diagrammes de séquences de messages sous forme de représentation abstraite, textuelle et graphique. Une description informelle de la sémantique est également fournie.

Champ d'application

Le champ d'application principal du MSC est la spécification d'une vue d'ensemble des comportements de la communication au sein des systèmes temps réel et plus particulièrement des systèmes de télécommunication. Tout d'abord, des traces choisies du comportement d'un système, correspondant aux cas «standard», peuvent être spécifiées au moyen de MSC. Les cas non standard correspondant aux cas de fonctionnement exceptionnels peuvent alors être construits à partir des précédents. De ce fait, les MSC peuvent être utilisés pour la spécification des besoins, la spécification des interfaces, la simulation et la validation, la spécification des cas de tests et la documentation des systèmes temps réel. Un MSC peut être utilisé en relation avec d'autres langages de spécification, en particulier le SDL. Dans cette optique, les MSC sont une base pour la conception des systèmes SDL.

Statut/Stabilité

Le statut du MSC de base est quasi-stable en ce qui concerne les constructions utilisées par une instance, la création et la terminaison d'instance, l'échange de messages, l'action, la gestion du temps et la condition. Les développements et extensions futurs concerneront principalement les concepts structurels.

Travail en rapport

Recommandation Q.65: Etape 2 de la méthode de caractérisation des services de télécommunication assurée par un RNIS.

DIAGRAMME DE SÉQUENCES DE MESSAGES

(Helsinki, 1993)

1 Introduction au MSC

Un diagramme de séquences de messages (MSC) (*message sequence chart*) représente des séquences de messages échangés entre les composants d'un système et leur environnement. En SDL, les composants d'un système sont modélisés au moyen des constructions service, processus et bloc. Les MSC sont depuis longtemps utilisés d'une part, dans des recommandations élaborées par des groupes d'études du CCITT et d'autre part, dans l'industrie, selon diverses conventions et sous différents noms tels que diagramme de séquences de signaux, diagramme de flux d'information, flux de messages et diagramme de Direction.

Normaliser les MSC permet de développer des outils de support, d'échanger des MSC entre différents outils, de rendre plus facile la correspondance avec les spécifications SDL et d'en harmoniser l'utilisation au sein du CCITT.

Une partie du travail de normalisation est de fournir une définition claire d'un MSC. Ceci est fait dans la présente Recommandation en reliant les MSC aux spécifications SDL, de la façon suivante: un MSC décrit une ou plusieurs traces d'éxécution de la spécification d'un système SDL.

Selon la définition ci-dessus, un MSC peut être dérivé de la spécification d'un système SDL et peut servir à mémoriser le résultat d'une exécution. Cependant, un MSC est généralement construit avant la spécification d'un système SDL et peut être utilisé pour:

a) décrire une vue d'ensemble du service offert par plusieurs entités;

b) formuler les besoins requis par des spécifications SDL;

c) servir de base à l'élaboration de spécifications SDL;

d) servir de base à la simulation et à la validation de système;

e) servir de base à la sélection et à la spécification de cas de tests;

f) spécifier semi-formellement une communication;

g) spécifier une interface.

Comme un MSC décrit généralement un comportement partiel, la sélection de comportements partiels est un problème crucial. Tout d'abord, on utilise les MSC pour décrire des cas «standard». Puis des cas représentant des comportements exceptionnels, causés par des erreurs diverses, sont bâtis à partir de ceux-ci.

Dans la suite du document, une syntaxe pour les diagrammes de séquences de messages est présentée sous forme abstraite, textuelle et graphique. Une description de la sémantique correspondante est fournie de manière informelle (en langage naturel).

La présente Recommandation est composée de la façon suivante: l'article 2 donne des règles générales concernant la syntaxe, les règles applicables aux dessins. A l'article 3, la définition syntaxique d'un document de diagramme de séquences de messages qui est un ensemble de diagrammes de séquences de messages est fournie. L'article 4 contient la définition syntaxique d'un diagramme de séquences de messages et les règles syntaxiques des éléments de base, c'est-à-dire l'instance, le message, la condition, le temporisateur, l'action, la création et la terminaison de processus. A l'article 5, des concepts de haut niveau concernant la structuration et la modularité sont introduits. Ces concepts permettent de construire une spécification au moyen de la méthode de conception descendante (approche top down) en raffinant les instances individuelles au moyen d'un ordre temporel généralisé (voir 5.1) et de la notion de sous-diagramme de séquences de messages (voir 5.2). L'article 6 donne des exemples d'utilisation de toutes les constructions afférentes aux MSC. L'Appendice I contient des références croisées pour les <mots-clés> et les non-terminaux.

2 Règles générales

Seules les règles spécifiques aux diagrammes de séquences de messages sont données. Les règles restantes sont identiques à celles de la Recommandation Z.100.

2.1 Règles lexicales

Contrairement à la Recommandation Z.100, <text> ne contient pas le point-virgule, parce que <text> est utilisé dans la définition de l'action et que le point-virgule sert de marque de terminaison à l'action. Le point-virgule fait partie de la définition de <end> (voir 2.3).

<lexical unit> ::=
<word>
| <character string>
| <special>
| <composite special>
| <note>
| <keyword>
| <semicolon>

<keyword> ::=
action
| **all**
| **block**
| **comment**
| **concurrent**
| **condition**
| **create**
| **decomposed**
| **endconcurrent**
| **endinstance**
| **endmsc**
| **endmscdocument**
| **endsubmsc**
| **endtext**
| **env**
| **from**
| **inst**
| **instance**
| **msc**
| **mscdocument**
| **in**
| **out**
| **process**
| **referenced**
| **related to**
| **reset**
| **service**
| **set**
| **shared**
| **stop**
| **submsc**
| **system**
| **text**
| **timeout**
| **to**

```
<text> ::=
                    { <alphanumeric>
                |   <other character>
                |   <special>
                |   <full stop>
                |   <underline>
                |   <space>
                |   <apostrophe> }*
<special> ::=
                    +
                |   -
                |   %
                |   !
                |   /
                |   >
                |   *
                |   (
                |   )
                |   "
                |   ,
                |   =
                |   :
<semicolon> ::=
                    ;
<note>::=
                /* <text> */
```

2.2 Règles de visibilité et de nommage

Les entités sont identifiées et référencées au moyen des noms correspondants. Les entités sont groupées par classes d'entités pour permettre une plus grande flexibilité des règles de nommage. Les classes d'entités sont les suivantes:

a) MSC document (document MSC);

b) MSC (MSC);

c) sub MSC (sous-MSC);

d) instance (instance);

e) condition (condition);

f) timer (temporisateur);

g) message (message).

Une entité qui contient d'autres entités au sens des règles syntaxiques est appelée unité de portée. Les unités de portée sont les suivantes:

a) MSC document (document MSC);

b) MSC (MSC);

c) sub MSC (sous-MSC);

d) instance (instance).

Deux entités au sein d'une unité de portée qui appartiennent à la même classe d'entités ne peuvent avoir le même nom. Différentes occurrences d'un nom de condition, d'un nom de temporisateur et d'un nom de message au sein d'une unité de portée désignent la même entité. Le nom d'une entité est visible par une unité de portée englobante et pas à l'extérieur. Seuls les noms visibles peuvent être utilisés dans le référençage des entités.

2.3 Commentaire

Un commentaire est une notation qui représente des commentaires associés à des symboles ou du texte.

Dans la *Concrete textual grammar* (*Grammaire textuelle concrète)*, deux formes de commentaires sont utilisées. La première forme est la <note>.

La syntaxe concrète de la deuxième forme est:

<end> ::=

[<comment>] <semicolon>

<comment> ::=

comment <character string>

Dans la *Concrete graphical grammar (Grammaire graphique concrète)*, la syntaxe suivante est utilisée:

<comment area> ::=

<comment symbol> ***contains*** <text>

is connected to <dashed association symbol>

<comment symbol> ::=

T1007630-93/d01

<dashed association symbol> ::=

T1007630-93/d02

Une des extrémités du <dashed association symbol> doit être connectée au milieu du segment vertical du <comment symbol>.

Un <comment symbol> peut être relié à tout symbole graphique au moyen d'un <dashed association symbol>. Le <comment symbol> est considéré comme un symbole fermé complété (par la pensée) par le rectangle correspondant, afin d'englober le texte à l'intérieur de celui-ci. Il contient le texte du commentaire relatif au symbole graphique.

2.4 Symbole de texte

Un <text symbol> peut être utilisé dans tout <msc diagram> et <submsc diagram> aux fins des commentaires généraux (global) (voir 4.1 et 5.2). Le texte peut être placé à l'intérieur du symbole de texte sous forme de <note>:

<text area> ::=

<text symbol> ***contains*** <note>

<text symbol> ::=

T1160120-94/d43

Dans la représentation textuelle on utilise la syntaxe suivante:

<text definition> ::=

text <note> **endtext** <end>

La <text definition> est contenue dans la définition du <msc body> (voir 4.1).

2.5 Règles applicables aux dessins

La taille des symboles graphiques peut être choisie par l'utilisateur. Les frontières des symboles ne doivent pas se chevaucher ni se croiser. Les exceptions à cette règle sont les suivantes:

a) un symbole de message peut croiser un autre symbole de message, un symbole d'expiration temporisateur, un symbole de désarmement de temporisateur, un symbole de création, un symbole d'axe d'instance et un symbole d'association tireté;

b) un symbole d'expiration temporisateur ou un symbole de désarmement de temporisateur peuvent croiser un symbole de message, un symbole d'expiration temporisateur, un symbole de désarmement de temporisateur, un symbole de création et un symbole d'association tireté;

c) un symbole de création peut croiser un symbole de message, un symbole d'expiration temporisateur, un symbole de désarmement de temporisateur, un symbole d'axe d'instance et un symbole d'association tireté;

d) un symbole de condition peut croiser un symbole d'axe d'instance;

e) un symbole d'action peut chevaucher un symbole d'axe d'instance au format colonne.

Il existe deux formats pour représenter le symbole d'axe d'instance et le symbole de corégion: le format ligne simple ou le format colonne. Il est interdit d'utiliser les deux formats au sein d'une même instance.

Si une condition partagée (voir 4.4) croise une instance non concernée par cette condition, le symbole d'axe d'instance la traverse.

Dans le cas où le symbole d'axe d'instance est au format colonne, les frontières verticales du symbole d'action doivent coïncider avec les lignes de la colonne.

Les lignes représentant les messages peuvent être horizontales ou obliques dirigées vers le bas (conformément à la direction donnée par la flèche) et peuvent être brisées.

S'il existe une flèche d'un symbole de message et une origine d'un symbole de message sur le même point d'un symbole d'axe d'instance, ceci est interprété comme si l'origine du symbole de message était dessinée en dessous de la flèche du symbole de message.

2.6 Règles d'impression des MSC

Les MSC peuvent être partitionnés verticalement sur plusieurs pages. Le partitionnement horizontal doit être géré au moyen de sous-MSC (voir 5.2).

Quand un MSC est partitionné sur plusieurs pages, le <msc heading> est répété sur chaque page, mais les symboles de fin d'instance ne doivent apparaître que sur une page (sur la «dernière page» de l'instance en question). Pour chaque instance, la <instance head area> doit apparaître sur la première page où l'instance en question commence et doit être répétée en forme tiretée sur toutes les pages suivantes où elle continue. Si des messages ou des temporisateurs doivent être représentés d'une page à la suivante, le nom du message ou du temporisateur doit apparaître sur les deux pages.

La numérotation des pages doit être indiquée sur les pages représentant un MSC afin de mentionner la séquence correcte des pages. Si la numérotation est omise, la séquence correcte des pages doit alors être exprimée au moyen des conditions globales qui agissent en tant que connecteurs (voir 4.4).

3 Document de diagramme de séquences de messages

L'en-tête d'un document de diagramme de séquences de messages contient le nom du document et optionnellement, à la suite du mot-clé **related to**, l'identificateur (nom du chemin d'accès) du document SDL auquel les MSC se réfèrent. Aucune grammaire graphique spécifique n'est introduite puisque un même document est supposé contenir les diagrammes de séquences de messages en forme textuelle et graphique.

Grammaire abstraite

MSC-document :: *MSC-document-name*
[Sdl-reference]
*[Message-sequence-chart-***set** *]*
*[Submsc-***set** *]*

MSC-document-name = *Name*

Sdl-reference :: *Sdl-document-identifier*

Sdl-document-identifier = *Identifier*

Identifier :: [*Qualifier*]
Name

Qualifier :: *Path-item+*

Path-item = *System-qualifier* |
Block-qualifier |
Process-qualifier

System-qualifier :: *System-name*

Block-qualifier :: *Block-name*

Process-qualifier :: *Process-name*

Grammaire textuelle concrète

<message sequence chart document> ::=
mscdocument <document head> <document body> **endmscdocument** <end>
<document head> ::=
<msc document name> [**related to** <sdl reference>] <end>
<sdl reference> ::=
<sdl document identifier>
<identifier> ::=
[<qualifier>] <name>
<qualifier> ::=
<path item> { **/** <path item> }*
<path item> ::=
<scope unit class> <name>
<scope unit class> ::=
system
| **block**
| **process**

<document body> ::=
{ <message sequence chart> | <submsc> | <msc diagram> | <submsc diagram> }*

Sémantique

Un document de diagramme de séquences de messages est un ensemble de diagrammes de séquences de messages, référant optionnellement un document SDL correspondant.

4 Le MSC de base

4.1 Diagramme de séquences de messages

Un diagramme de séquences de messages décrit le flux des messages entre instances, et optionnellement les actions engendrées par les messages, qui dépendent de conditions initiales, intermédiaires et finales. Un diagramme de

séquences de messages décrit un comportement partiel d'un système. Bien que le nom *message sequence chart* ait pour origine, de manière évidente, sa représentation graphique, celui-ci est utilisé pour désigner à la fois les représentations textuelle et graphique.

L'en-tête d'un diagramme de séquences de messages est constitué d'un nom de diagramme de séquences de messages et (optionnellement) d'une liste d'instances contenues dans le corps d'un diagramme de séquences de messages.

Grammaire abstraite

Message-sequence-chart	::	*MSC-name* [*MSC-interface*] *MSC-body*
MSC-name	=	*Name*
MSC-interface	::	*Instance-list*
Instance-list	=	*Instance-declaration*+
Instance-declaration	::	*Instance-name* [*Instance-kind*]
Instance-name	=	*Name*
Instance-kind	=	*System-name* | *Block-name* | *Process-name* | *Service-name* | *Name*
System-name	::	*Name*
Block-name	::	*Name*
Process-name	::	*Name*
Service-name	::	*Name*
MSC-body	::	(*Instance-definition* | *Text-definition*)*
Text-definition	::	*Informal-text*

Si la *Instance-list* dans *MSC-interface* est présente, elle doit contenir les mêmes instances que celles spécifiées dans le *MSC-body*.

Grammaire textuelle concrète

<message sequence chart> ::=
msc <msc head> <msc body> **endmsc** <end>
<msc head> ::=
<message sequence chart name> <end> [<msc interface>]
<msc interface> ::=
inst <instance list> <end>
<instance list> ::=
<instance name> [**:** <instance kind>] [**,** <instance list>]
<instance kind> ::=
[<kind denominator>] <kind name>

<kind denominator> ::=

system | block | process | service

<msc body> ::=

{ <instance definition> | <text definition> }*

Grammaire graphique concrète

<msc diagram> ::=

<msc symbol> ***contains*** { <msc heading> <msc body area> }

<msc body area> ::=

{ <instance area> | <external message area> | <text area> }*

<msc symbol> ::=

<frame symbol>

<frame symbol> ::=

T1007630-93/d03

<msc heading> ::=

msc <message sequence chart name>

Sémantique

Un MSC décrit la communication entre des composants d'un système, et entre ces composants et le reste du monde, appelé environnement. Pour chaque composant de système décrit par un MSC existe un axe d'instance. La communication entre les composants d'un système est assurée au moyen de messages. L'envoi et la consommation de messages sont deux événements différents. On suppose que l'environnement d'un MSC est capable de recevoir des messages du diagramme de séquences de messages et d'en envoyer à celui-ci; il n'y a pas d'ordre au niveau des événements liés aux messages dans l'environnement. Bien que le comportement de l'environnement soit non déterministe, on suppose qu'il obéit aux contraintes exprimées dans le diagramme de séquences de messages.

Un diagramme de séquences de messages ne comporte pas d'axe de temps global. Le long de chaque axe d'instance, le temps se déroule du haut vers le bas; cependant, il n'existe pas d'échelle des temps précise. S'il n'existe aucune corégion (voir 5.1), les événements sont totalement ordonnés dans le temps, le long de chaque axe d'instance. Les événements des différentes instances sont uniquement ordonnés à travers les messages: un message doit d'abord être envoyé avant d'être consommé (voir 4.3). Aucun autre ordre n'est imposé. En conséquence, un diagramme de séquences de messages impose un ordre partiel sur l'ensemble des événements qu'il contient. Une relation binaire qui est transitive, antisymétrique et réflexive est appelée un ordre partiel.

Par exemple, pour les messages en entrée du diagramme de séquences de messages de la Figure 1a) (labellés par ?mi) et ceux en sortie (labellés par !mi), nous avons la relation d'ordre suivante: !m2 < ?m2, !m3 < ?m3, !m4 < ?m4, ?m1 < !m2 < !m3 < ?m4, ?m2 < !m4 et sa fermeture transitive.

L'ordre partiel peut être décrit en forme minimale (sans représentation explicite de la fermeture transitive) au moyen de son graphe de connexité [voir Figure 1b)].

La sémantique d'un MSC peut être reliée à la sémantique du SDL par l'intermédiaire de la notion de graphe d'accessibilité. Chaque séquence d'un MSC décrit une trace d'exécution d'un nœud vers un autre nœud (ou un ensemble de nœuds) du graphe d'accessibilité; celui-ci décrit le comportement de la spécification d'un système SDL. Le graphe d'accessibilité est composé de nœuds et d'arcs. Les nœuds représentent les états globaux du système. Un état global est déterminé par les valeurs des variables, l'ensemble des états d'exécution de chaque processus et le contenu des files d'attente. Les arcs représentent les événements exécutés par le système, c'est-à-dire l'envoi ou la consommation de messages ou l'exécution d'une tâche. Chaque séquence MSC correspond à un ordre total des événements compatible avec l'ordre partiel défini par le MSC.

A noter que le graphe d'accessibilité contient plus d'événements que le MSC. De ce fait, les séquences MSC ne représentent pas des chemins complets du graphe d'accessibilité.

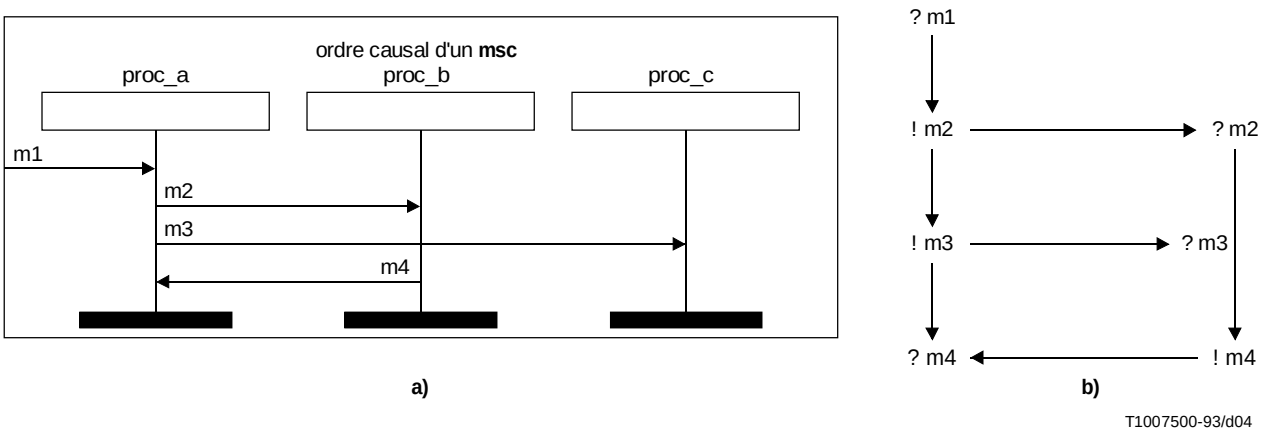


FIGURE 1/Z.120

Diagramme de séquences de messages et graphe de connexité associé

4.2 Instance

Un diagramme de séquences de messages est composé d'instances d'entités qui interagissent. L'instance d'une entité est un objet qui a les propriétés de cette entité. Si on se réfère au SDL, une entité peut être un processus SDL, un bloc ou un service. Au niveau de l'en-tête de l'instance, le nom de l'entité, c'est-à-dire le nom du processus, peut être ajouté au nom de l'instance. Au niveau du corps de l'instance, l'ordre des événements est spécifié. Un sous-diagramme de séquences de messages portant le même nom peut être associé à une instance au moyen du mot-clé **decomposed**.

Grammaire abstraite

Instance-definition :: *Instance-name*
[*Instance-kind*]
[**DECOMPOSED**]
Instance-event-list
[*Stop-node*]

Instance-event-list :: *Instance-event* *

Instance-event :: *Message-input* |
Message-output |
Create-node |
Timer-statement |
Coregion |
Action |
Condition

Pour chaque instance contenant le mot-clé **decomposed** un *Submsc* (voir 5.2) correspondant et portant le même nom doit être spécifié. Pour chaque *Message-output* d'une instance décomposée, un *Message-output* correspondant, envoyé à l'extérieur du *Submsc* doit être spécifié. Une correspondance analogue doit être faite pour les messages en entrée.

Grammaire textuelle concrète

<instance definition> ::=
instance <instance head> <instance body> **endinstance** <end>
<instance head> ::=
<instance name> [[**:**] <instance kind>] [**decomposed**] <end>

<instance body> ::=

<instance event list> [<stop>]

<instance event list> ::=

{ <message input> | <message output> | <create> | <timer statement>
| <coregion> | <action> | <condition> }*

Grammaire graphique concrète

<instance area> ::=

<instance head area> ***is followed by*** <instance body area>

<instance head area> ::=

<instance head symbol> ***is associated with*** <instance heading>

<instance heading> ::=

<instance name> [**:** <instance kind>] [**decomposed**]

<instance head symbol> ::=

T1007630-93/d05

<instance body area> ::=

<instance axis symbol>
{ ***is followed by*** <instance event area>
is followed by <instance axis symbol> }*
is followed by { <instance end symbol> | <stop symbol> }

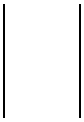
<instance axis symbol> ::=

<instance axis symbol1> | <instance axis symbol2>

<instance axis symbol1> ::=

T1007630-93/d06

<instance axis symbol2> ::=



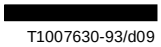
T1007630-93/d07

<instance event area> ::=

<message in area>
| <message out area>
| <create area>
| <timer area>
| <concurrent area>
| <action area>
| <condition area>

T1007630-93/d08

<instance end symbol> ::=



T1007630-93/d09

L'<instance heading> peut être placé au-dessus ou à l'intérieur du <instance head symbol>, ou bien fractionné de telle sorte que le <instance name> soit placé au-dessus du <instance head symbol> et que la <instance kind> soit à l'intérieur. Dans ce dernier cas le symbole deux points est facultatif (et sera placé au-dessus s'il est utilisé) et le mot-clé facultatif **decomposed** doit, s'il est présent, se trouver à l'intérieur. Il n'est pas autorisé d'utiliser à la fois <instance axis symbol1> et <instance axis symbol2>.

Sémantique

Les instances sont définies dans le corps du diagramme de séquences de messages. Le symbole de fin d'instance correspond à la fin de la description de l'instance au sein du MSC. Il ne correspond pas à la terminaison de l'instance (voir 4.8: Terminaison de processus). De même, le symbole de début d'instance correspond au commencement de la description de l'instance au sein du MSC. Il ne correspond pas à la création de l'instance (voir 4.7: Création de processus).

Dans le cadre du SDL, une instance peut se rapporter à un processus (mot-clé **process**), un service (mot-clé **service**) ou un bloc (mot-clé **block**). En dehors de ce cadre, une instance peut se rapporter à toute sorte d'entité. La définition d'instance fournit, en termes d'événements, une description pour la consommation et l'émission de messages, les actions, les conditions locales et partagées, les temporisations, la création de processus et la terminaison de processus. Mise à part l'utilisation de corégion (voir 5.1), on a un ordre total des événements le long de chaque axe d'instance. Au sein des corégions, on ne garantit pas un ordre temporel des événements.

Un sous-diagramme de séquences de messages avec le même nom peut être attaché à une instance au moyen du mot-clé **decomposed**.

4.3 Message

Un message dans un MSC représente un échange d'information entre deux instances ou entre une instance et son environnement.

Un message échangé entre deux instances peut être décomposé en deux événements: le message en entrée et le message en sortie; par exemple, le deuxième message de la Figure 1a) peut être décomposé en !m2 (sortie) et ?m2 (entrée). Les messages venant de l'environnement sont représentés par des messages en entrée; les messages à destination de l'environnement sont représentés par des messages en sortie. Dans la représentation textuelle, le message en entrée est représenté par le mot-clé **in**, le message en sortie par le mot-clé **out**, tous les deux suivis par le nom du message et optionnellement un nom d'instance de message. Un message peut véhiculer des paramètres; ceux-ci sont représentés entre parenthèses. La déclaration d'une liste de paramètres est optionnelle pour les messages en entrée.

La correspondance entre les messages en sortie et les messages en entrée est biunivoque. Dans la représentation textuelle, la correspondance entre les messages en entrée et les messages en sortie, se fait, en principe, par identification du nom de message et de l'adresse spécifiée. Au cas où le nom du message et l'adresse ne suffisent pas pour une correspondance biunivoque, le nom de l'instance de message doit être précisé. Dans la représentation graphique, un message est représenté par une flèche.

Grammaire abstraite

Message-input	::	*Message-identification* *Sender-address*
Message-output	::	*Message-identification* *Receiver-address*
Message-identification	::	*Message-name* [*Message-instance-name*] [*Parameter-list*]
Message-name	=	*Name*
Message-instance-name	=	*Name*
Parameter-list	=	*Parameter-name+*
Parameter-name	=	*Name*

Sender-address = *Address*

Receiver-address = *Address*

Address = *Instance-name* |
ENVIRONMENT

Il n'est pas autorisé que le *Message-output* dépende de son *Message-input* par le biais d'autres messages. Ceci est le cas lorsque le graphe de connexité (voir 4.1) contient des boucles. Si une *Parameter-list* est spécifiée pour un *Message-input*, cette liste doit également être spécifiée pour le *Message-output* correspondant. Les *Parameter-lists* doivent être identiques.

Grammaire textuelle concrète

<message input> ::=
in <msg identification> **from** <address> <end>
<message output> ::=
out <msg identification> **to** <address> <end>
<msg identification> ::=
<message name> [**,** <message instance name>] [(<parameter list>)]
<parameter list> ::=
<parameter name> [**,** <parameter list>]
<address> ::=
<instance name> | **env**

Pour les messages échangés entre instances, les règles suivantes s'appliquent: Pour chaque <message output>, un <message input> correspondant doit être spécifié et réciproquement. Au cas où le <message name> et l'<address> ne suffisent pas à déterminer une correspondance biunivoque, le <message instance name> doit être précisé.

Grammaire graphique concrète

<message out area> ::=
<flow line symbol>
<flow line symbol> ::=

T1007630-93/d10

<message in area> ::=
<message symbol> ***is associated with*** <msg identification>
is connected to { <message out area> | <msc symbol> | <submsc symbol> }
[***is followed by*** <message out area>]
<message symbol> ::=

T1007630-93/d11

L'image miroir du <message symbol> est autorisée.

<external message area> ::=
<message symbol> ***is associated with*** <msg identification>
is connected to { <message out area> { <msc symbol> | <submsc symbol> } }

NOTE – Dans la représentation graphique, le nom d'instance de message n'est pas nécessaire pour une unique description syntaxique.

Sémantique

Pour un MSC, le message-en-sortie indique un envoi de message (correspondant à une sortie SDL) et le message-en-entrée indique une consommation de message (correspondant à une entrée SDL). Aucune construction n'est fournie pour la réception de message (entrée dans la file). Aucune définition de type n'est associée aux paramètres de la liste des paramètres.

S'il existe une flèche d'un symbole de message et une origine d'un symbole de message sur le même point d'un symbole d'axe d'instance, ceci est interprété comme si l'origine du symbole de message était dessinée en dessous de la flèche du symbole de message.

4.4 Condition

Une condition décrit soit un état global du système (condition globale) correspondant à toutes les instances contenues dans le MSC, soit un état correspondant à un sous-ensemble d'instances (condition non globale). Dans le second cas, la condition peut être locale, c'est-à-dire attachée à une seule instance. Dans la représentation textuelle, la condition doit être définie pour chaque instance à laquelle elle est attachée au moyen du mot-clé **condition** accompagné du nom de condition. Si la condition est attachée à plusieurs instances, le mot-clé **shared** ainsi que la liste des instances identifient l'ensemble par lequel la condition est partagée. Une condition se rapportant à toutes les instances peut être définie au moyen du mot-clé **shared all**.

Grammaire abstraite

Condition	::	*Condition-name* [*Shared-information*]
Condition-name	=	*Name*
Shared-information	::	*Shared-instance-list* | **ALL**
Shared-instance-list	=	*Instance-name*+

Grammaire textuelle concrète

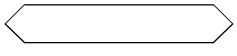
<condition> ::=
condition <condition name> [**shared** { <shared instance list> | **all** }] <end>

<shared instance list> ::=
<instance name> [**,** <shared instance list>]

Pour chaque <instance name> contenu dans une <shared instance list> d'une <condition>, une instance avec une <condition> partagée correspondante doit être spécifiée. Si l'instance *b* fait partie de la <shared instance list> d'une <condition> partagée attachée à une instance *a*, l'instance *a* doit faire partie de la <shared instance list> de la <condition> partagée correspondante attachée à l'instance *b*. Si les instances *a* et *b* partagent la même <condition>, alors pour chaque message échangé entre ces instances, le <message input> et le <message output> doivent être placés tous les deux, soit avant, soit après la <condition>.

Grammaire graphique concrète

<condition area> ::=
<local condition area> | <shared condition area>
<local condition area> ::=
<condition symbol> ***contains*** <condition name>
<condition symbol> ::=



T1007630-93/d12

<shared condition area> ::=

<condition left area> | <condition middle area> | <condition right area>

<condition left area> ::=

<condition left symbol> ***is associated with*** <condition name>

is connected to { <condition middle area> | <condition right area> }

<condition left symbol> ::=

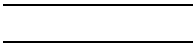
T1007630-93/d13

<condition middle area> ::=

<condition middle symbol>

is connected to { <condition middle area> | <condition right area> }

<condition middle symbol>::=



T1007630-93/d14

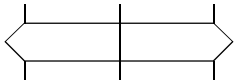
<condition right area> ::=

<condition right symbol>

<condition right symbol> ::=

T1007630-93/d15

Une <shared condition area> est fractionnée en plusieurs zones: <condition left area>, <condition middle area>, <condition right area>, qui sont alignées horizontalement pour former un seul et unique symbole associé au <condition name>. La <local condition area> est attachée à une seule instance, la <shared condition area> est connectée à d'autres instances. Si une <condition> partagée croise un <instance axis symbol> non concerné par la condition, le <instance axis symbol> la traverse.



T1007630-93/d16

Sémantique

Les conditions globales qui représentent des états globaux du système, concernent toutes les instances du MSC. Pour chaque diagramme de séquences de messages:

- une condition globale initiale (état initial global);
- une condition globale finale (état final global); et
- des conditions globales intermédiaires (états intermédiaires globaux),

doivent être spécifiées en utilisant le mot-clé **shared all** dans la représentation textuelle.

Les conditions globales initiales, intermédiaires et finales ne sont pas simplement introduites à des fins de documentation au sens de commentaires ou d'illustrations. Dans le cas d'un ensemble complet de diagrammes de séquences de messages, ces conditions ont un rôle bien défini. Les conditions globales indiquent de possibles continuations entre diagrammes de séquences de messages qui contiennent le même ensemble d'instances, par identification de la condition: lorsque la condition globale finale de MSC1 est identique à la condition initiale globale de MSC2, MSC2 peut être vu comme une continuation de MSC1.

Les conditions globales permettent aussi de définir des compositions et des décompositions de MSC. En décomposant un MSC au niveau d'une condition globale intermédiaire en MSC1 et en MSC2, la condition globale intermédiaire devient une condition globale finale pour MSC1 et une condition globale initiale pour MSC2.

Pour une utilisation intensive de la notion de composition de MSC, les conditions globales qui se rapportent aux états du système complet sont trop restrictives. Un partitionnement en conditions non globales qui se rapportent à un sous-ensemble d'instances est nécessaire. Les conditions locales attachées à des instances individuelles sont comme un cas

particulier des conditions non globales. Les conditions non globales peuvent aussi servir à définir des combinaisons de diagrammes de séquences de messages avec différents ensembles d'instances, la continuation se rapportant uniquement au sous-ensemble commun des instances.

Il faut également noter qu'un MSC se terminant par une condition globale peut être aussi continué par un MSC qui commence par une condition non globale et viceversa, si les deux conditions se rapportent au même (sous-)ensemble d'instances. Par généralisation des règles s'appliquant aux conditions globales, nous définissons la continuation de deux MSC, par un ensemble commun non vide d'instances de la manière suivante: MSC2 est la continuation de MSC1 par l'intermédiaire de conditions (non) globales si pour toute instance commune aux deux MSC, MSC1 se termine par une condition (non) globale et MSC2 commence par la condition (non) globale correspondante. «Correspondante» signifie, dans ce contexte, que les deux conditions se rapportent au même sous-ensemble d'instances et que les deux conditions concordent par identification des noms. De plus, pour chaque condition (non) globale de MSC2, on doit avoir une condition (non) globale correspondante dans MSC1. De cette façon, MSC1 et MSC2 peuvent être composés.

De même, un MSC contenant des conditions non globales intermédiaires peut être décomposé en MSC1 et MSC2. Après décomposition, les conditions intermédiaires deviennent des conditions finales pour MSC1 et des conditions initiales pour MSC2. Les MSC obtenus peuvent être combinés à nouveau selon les règles stipulées ci-dessus.

4.5 Temporisateur

Les MSC permettent de spécifier, soit l'armement d'un temporisateur avec, par la suite, son expiration, soit l'armement d'un temporisateur avec, par la suite, son désarmement (supervision du temps). Dans la représentation graphique, le symbole d'armement a la forme d'un petit rectangle. Le symbole d'expiration est représenté par un message envoyé par une instance à elle-même. Le symbole de désarmement est un symbole d'expiration modifié: la flèche en entrée est en pointillé.

La spécification d'un nom d'instance de temporisateur et la spécification de la durée du temporisateur sont optionnelles, aussi bien en représentation textuelle que graphique.

Grammaire abstraite

Timer-statement	=	*Set-node* | *Reset-node* | *Timeout*
Set-node	::	*Timer-name* [*Timer-instance-name*] [*Duration-name*]
Reset-node	::	*Timer-name* [*Timer-instance-name*]
Timeout	::	*Timer-name* [*Timer-instance-name*]
Timer-name	=	*Name*
Timer-instance-name	=	*Name*
Duration-name	=	*Name*

Grammaire textuelle concrète

<timer statement> ::=
<set> | <reset> | <timeout>

<set> ::=
set <timer name> [**,** <timer instance name>] [(<duration name>)] <end>

<reset> ::=
reset <timer name> [**,** <timer instance name>] <end>

<timeout> ::=

timeout <timer name> [**,** <timer instance name>] <end>

Pour <set> et <timeout>, les règles suivantes doivent être appliquées: pour chaque <set> une <timeout> ou un <reset> correspondant doit être spécifié et vice versa. Au cas où le <timer name> n'est pas suffisant pour une correspondance biunivoque, le <timer instance name> doit être utilisé.

Grammaire graphique concrète

<timer area> ::=

<timer set area> | <timer reset area> | <timeout area>

<timer set area> ::=

<set symbol>

<set symbol> ::=

T1007630-93/d17

<timer reset area> ::=

<reset symbol> ***is associated with*** <timer name> [(<duration name>)]

is connected to <timer set area>

<reset symbol> ::=

<reset symbol1> | <reset symbol2>

<reset symbol1> ::=

T1007630-93/d18

<reset symbol2> ::=

T1007630-93/d19

<timeout area> ::=

<timeout symbol> ***is associated with*** <timer name> [(<duration name>)]

is connected to <timer set area>

<timeout symbol> ::=

<timeout symbol1> | <timeout symbol2>

<timeout symbol1> ::=

T1007630-93/d20

<timeout symbol2> ::=

T1007630-93/d21

Un des côtés du <set symbol> doit coïncider avec le <instance axis symbol>. Dans le cas d'un <instance axis symbol2>, le <set symbol> doit être extérieur à une colonne représentant le <instance axis symbol2>.

L'extrémité supérieure du <reset symbol> et du <timeout symbol> doit être reliée au milieu du côté du <set symbol> qui est opposé au côté coïncidant avec le <instance axis symbol>.

Sémantique

Les instructions d'armement et de désarmement sont extraites du langage SDL. L'armement désigne l'armement du temporisateur et le désarmement désigne le désarmement du temporisateur. L'expiration temporisateur correspond à la consommation d'un signal de temporisation en SDL.

4.6 Action

En plus de l'échange de messages, des actions peuvent être spécifiées en MSC. Un texte informel est attaché à ces actions.

Grammaire abstraite

Action :: *Informal-Text*

Grammaire textuelle concrète

<action> ::=

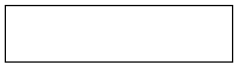
action <action text> <end>

Grammaire graphique concrète

<action area> ::=

<action symbol> ***contains*** <action text>

<action symbol> ::=



T1007630-93/d22

Lorsque l'axe d'instance est au format colonne, la largeur du <action symbol> doit coïncider avec la largeur de la colonne.

Sémantique

Une action décrit une activité interne à une instance.

4.7 Création de processus

De même qu'en SDL, la création et la terminaison de processus peuvent être spécifiées en MSC. Une instance de processus peut être créée par une autre instance de processus. Aucun message avant création ne doit se rapporter à l'instance créée.

Grammaire abstraite

Create-node :: *Instance-name*
*Parameter-name**

Grammaire textuelle concrète

<create> ::=

create <instance name> [(<parameter list>)] <end>

Pour chaque <create> doit exister une instance correspondante portant le nom spécifié. Le <instance name> doit se rapporter à l'instance d'un type de processus, si ce type est spécifié. Une instance ne peut être créée qu'une seule fois, c'est-à-dire qu'au sein d'un MSC, deux <create>s ou plus ne doivent pas apparaître. Aucun événement lié aux messages avant création ne doit se rapporter à l'instance créée.

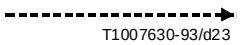
Grammaire graphique concrète

<create area> ::=

<createline symbol> [***is associated with*** <parameter list>]

is connected to <instance head symbol>

<createline symbol> ::=



T1007630-93/d23

L'image miroir du <createline symbol> est autorisée.

Sémantique

La création correspond à la création dynamique d'une instance de processus par une autre. Une instruction de création est immédiatement exécutée.

4.8 Terminaison de processus

La terminaison de processus est le pendant, dans un certain sens, de la création de processus. Cependant, une instance de processus décide elle-même de sa terminaison alors qu'une instance de processus est créée par une autre instance de processus.

Grammaire abstraite

Stop-node :: *( )*

Le *Stop-node* à la fin de *Instance-definition-d'instance* est autorisé uniquement pour des instances de type processus.

Grammaire textuelle concrète

<stop> ::=

stop <end>

Grammaire graphique concrète

<stop symbol> ::=

T1007630-93/d24

Sémantique

L'arrêt à la fin d'un corps d'instance provoque la terminaison de cette instance de processus.

5 Concepts structurels

Dans cet article, des concepts de haut niveau sont introduits. Ils portent sur une généralisation de l'ordre temporel (corégion) et la composition et la décomposition d'instances.

Les instances d'un MSC se rapportent à des entités ayant différents niveaux d'abstraction comme cela est déjà indiqué par les mots-clés **block**, **service**, **process**. Les opérations de décomposition d'instances correspondantes peuvent être définies en déterminant la transition entre différents niveaux d'abstraction. Quand des instances sont composées pour former une instance, l'ordre total des événements pour cette instance ne doit pas être contraint, afin de préserver le comportement externe observable.

5.1 Corégion

L'ordre total des événements le long de chaque instance (voir 4.1) n'est en général pas approprié pour les entités de plus haut niveau que les processus SDL.

De ce fait, la notion de corégion est introduite pour spécifier des événements non ordonnés sur une instance. Une telle notion de corégion couvre en particulier le cas pratique important de deux messages entrants (ou plus) pour lesquels l'ordre de consommation n'a pas d'importance.

Grammaire abstraite

Coregion :: *Coevent**

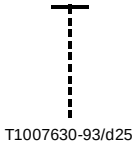
Coevent = *Message-input* |
Message-output

Grammaire textuelle concrète

<coregion> ::=
concurrent { <coevent> }* **endconcurrent** <end>
<coevent> ::=
<message input> | <message output>

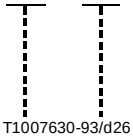
Grammaire graphique concrète

<concurrent area> ::=
<coregion start symbol>
is followed by <coevent area>
is followed by <coregion end symbol>
<coregion start symbol> ::=
<coregion start symbol1> | <coregion start symbol2>
<coregion start symbol1> ::=



T1007630-93/d25

<coregion start symbol2> ::



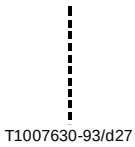
T1007630-93/d26

<coevent area> ::=
{ { <message in area> | <message out area> } ***is followed by*** <coregion symbol> }*

<coregion symbol> ::=

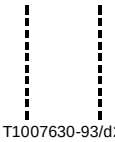
<coregion symbol1> | <coregion symbol2>

<coregion symbol1> ::=



T1007630-93/d27

<coregion symbol2> ::=



T1007630-93/d28

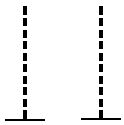
<coregion end symbol> ::=

<coregion end symbol1> | <coregion end symbol2>

<coregion end symbol1> ::=

T1007630-93/d29

<coregion end symbol2> ::=



T1007630-93/d30

Les symboles <coregion start symbol1>, <coregion symbol1>, <coregion end symbol1> et <instance axis symbol1> ne doivent pas être utilisés avec les symboles <coregion start symbol2>, <coregion symbol2>, <coregion end symbol2> et <instance axis symbol2> au sein d'une même instance.

Sémantique

Pour les MSC, l'ordre temporel total des événements est assuré au sein de chaque instance. En utilisant une corégion, une exception peut être faite à cette règle: les événements contenus dans une corégion ne sont pas ordonnés dans le temps.

5.2 Sous-diagramme de séquences de messages

Une instance d'un MSC peut être décomposée sous forme d'un sous-diagramme de séquences de messages (sous-MSC), afin de permettre de spécifier selon la méthode de conception descendante (approche top-down).

Un sous-MSC a une structure quasi-analogue à celle d'un MSC. Il se distingue du MSC par le mot-clé **submsc**. La particularité d'un sous-MSC est sa relation avec une instance décomposée: celle-ci contient le mot-clé **decomposed** et possède le même nom que celui du sous-MSC. La relation est donnée par les messages connectés à l'extérieur du sous-MSC et par les envois de messages et les consommations de messages correspondants effectués par l'instance décomposée.

Grammaire abstraite

Submsc :: *Message-sequence-chart*

Le nom du *Submsc* doit être le même que celui de la *Instance-definition* correspondante qui contient le mot-clé **decomposed** et qui se trouve dans un autre *Message-sequence-chart* ou un autre *Submsc*. Pour chaque

Message-output, envoyé à l'extérieur du *Submsc*, un *Message-output* correspondant doit être spécifié au niveau de l'instance décomposée. On a une correspondance analogue pour les messages en entrée.

Grammaire textuelle concrète

<submsc> ::=

submsc <msc head> <msc body> **endsubmsc** <end>

Grammaire graphique concrète

<submsc diagram> ::=

<submsc symbol> ***contains*** { <submsc heading> <msc body area> }

<submsc symbol> ::=

<frame symbol>

<submsc heading> ::=

submsc <submsc name>

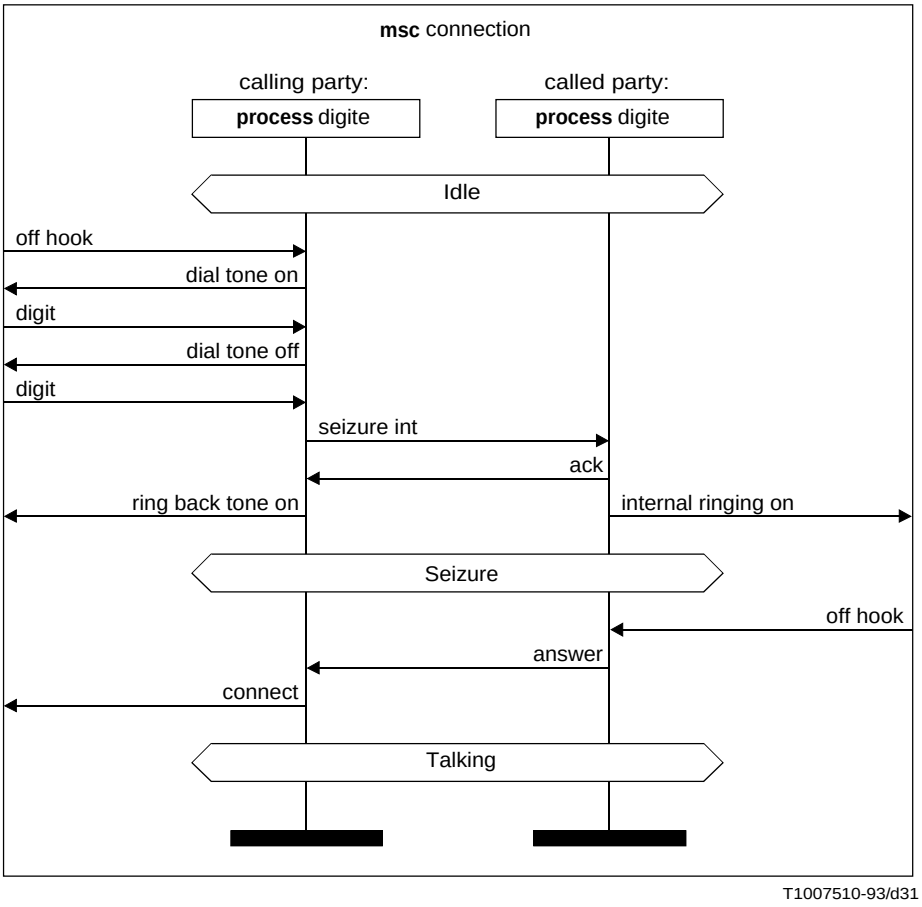
Sémantique

Un sous-diagramme de séquences de messages peut être attaché à une instance au moyen du mot-clé **decomposed**. Le sous-MSC représente une décomposition de cette instance qui n'affecte pas son comportement observable. Dans la représentation textuelle, les messages adressés vers ou depuis l'extérieur du sous-MSC possèdent l'adresse **env**; dans la représentation graphique, ils sont caractérisés par la connexion à la frontière du sous-MSC (symbole de cadre). Leur connexion avec les instances externes est faite par les messages émis et consommés par l'instance décomposée, au moyen de l'identification du nom du message. Il est possible d'établir une correspondance entre le comportement externe d'un sous-MSC et les messages de l'instance décomposée. L'ordre des événements liés aux messages dans une instance décomposée doit être préservé dans le sous-MSC. Les actions et les conditions au sein du sous-MSC peuvent être vues comme un raffinement des actions et des conditions de l'instance décomposée. Cependant, contrairement à ce qui est fait au niveau des messages, on n'a aucune correspondance formelle vers l'instance décomposée, c'est-à-dire que le raffinement des actions et des conditions n'obéit à aucune règle formelle.

6 Exemples de diagrammes de séquences de messages

6.1 Diagramme standard de flux de messages

L'exemple 6.1 illustre une connexion simplifiée réalisée dans un système de commutation. L'exemple contient les constructions MSC les plus basiques: instances (de processus), environnement, messages, conditions globales.



msc connection; **inst** calling party: **process** digite, called party: **process** digite;

instance calling party: **process** digite;

condition Idle **shared all**;

in off hook **from env**;

out dial tone on **to env**;

in digit **from env**;

out dial tone off **to env**;

in digit **from env**;

out seizure int **to** called party;

in ack **from** called party;

out ring back tone on **to env**;

condition Seizure **shared all**;

in answer **from** called party;

out connect **to env**;

condition Talking **shared all**;

endinstance;

instance called party: **process** digite;

condition Idle **shared all;**

in seizure int **from** calling party;

out ack **to** calling party;

out internal ringing on **to env**;

condition Seizure **shared all**;

in off hook **from env**;

out answer **to** calling party;

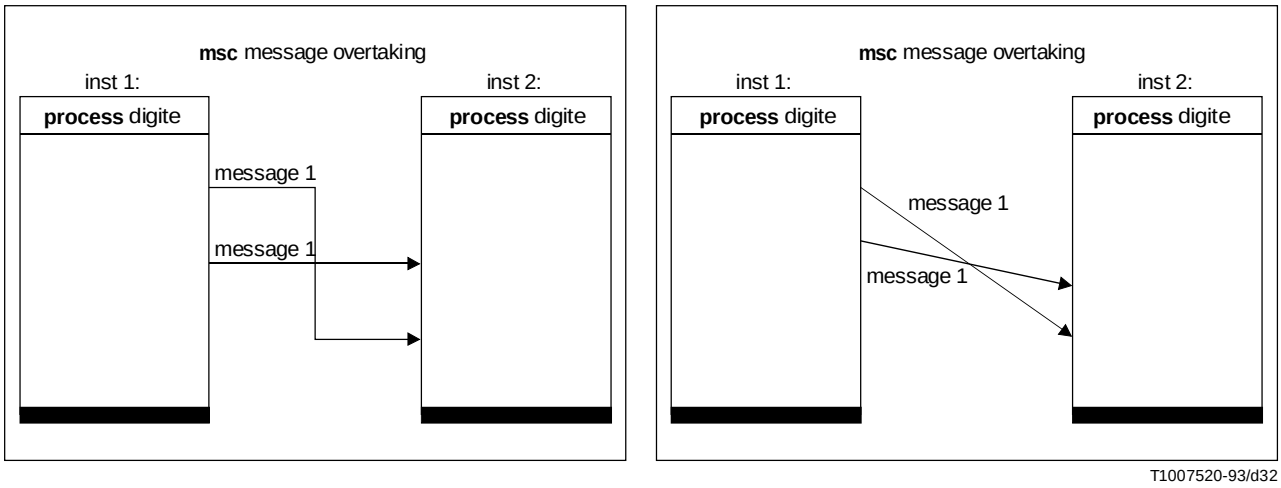
condition Talking **shared all**;

endinstance;

endmsc;

6.2 Dépassement de message

L'exemple 6.2 illustre le dépassement d'un message par un autre, les deux messages ayant le même nom: “message 1”. Dans la représentation textuelle, les noms des instances de messages (a, b) sont utilisés afin d'avoir une correspondance biunivoque entre le message en entrée et le message en sortie. Dans la représentation graphique, les messages sont soit représentés par des flèches horizontales, l'une avec une portion de ligne brisée afin d'illustrer le dépassement, soit par des croisements de flèches obliques.



msc message overtaking; **inst** inst 1, inst 2;

instance inst 1**: process** digite;

out message 1, a **to** inst 2;

out message 1, b **to** inst 2;

endinstance;

instance inst 2: **process** digite;

in message 1, b **from** inst 1;

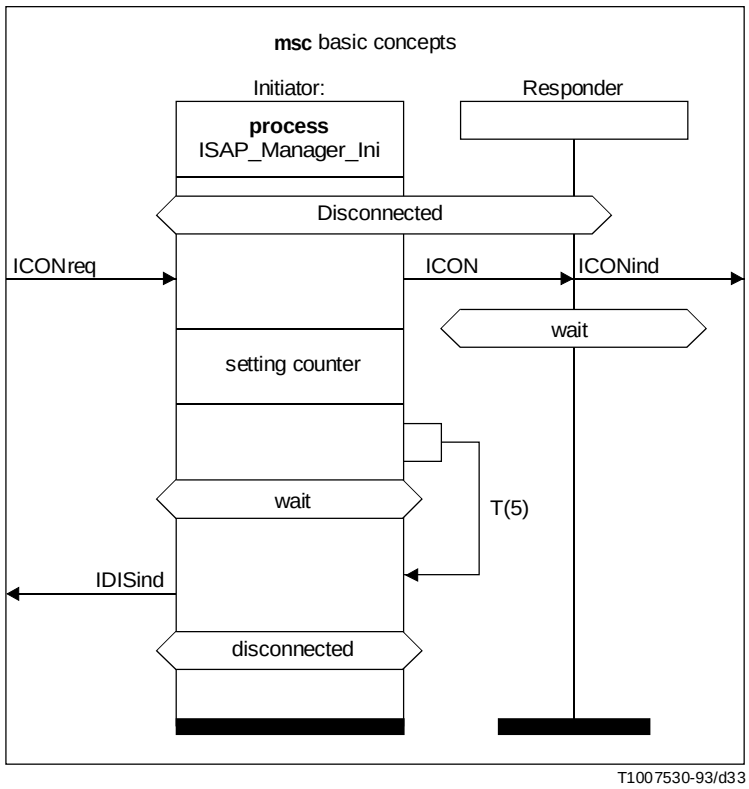
in message 1, a **from** inst 1;

endinstance;

endmsc;

6.3 Concepts de base du MSC

L'exemple 6.3 donne les constructions MSC de base: instances, environnement, messages, conditions, actions et expiration temporisateur. Dans la représentation graphique, les deux types de symboles d'axe d'instance sont utilisés: la forme ligne simple et la forme colonne.



T1007530-93/d33

msc basic concepts; **inst** Initiator: **process** ISAP_Manager_Ini, Responder;

instance Initiator: **process** ISAP_Manager_Ini;

condition Disconnected **shared all**;

in ICONreq **from env**;

out ICON **to** Responder;

action setting counter;

set T (5);

condition wait;

timeout T;

out IDISind **to env**;

condition disconnected;

endinstance;

instance Responder;

condition Disconnected **shared all**;

in ICON **from** Initiator;

out ICONind **to env**;

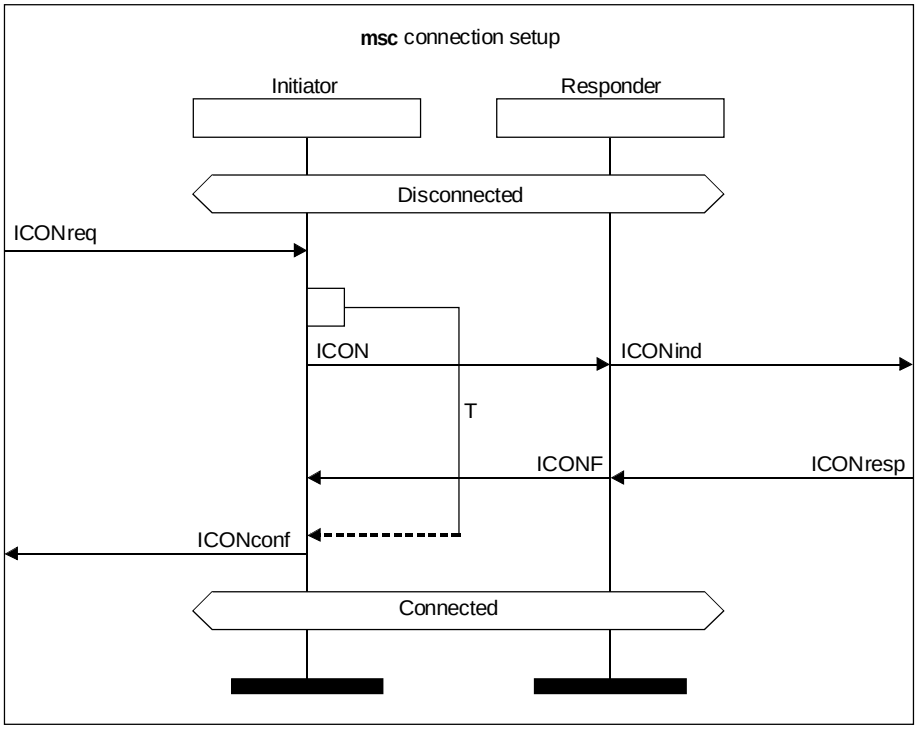
condition wait;

endinstance;

endmsc;

6.4 MSC avec supervision du temps

L'exemple 6.4 MSC connection set-up contient un désarmement temporisateur.



msc connection setup; **inst** Initiator, Responder;

instance Initiator;

condition Disconnected **shared all**;

in ICONreq **from env**;

set T;

out ICON **to** Responder;

in ICONF **from** Responder;

reset T;

out ICONconf **to env**;

condition Connected **shared all**;

endinstance;

instance Responder;

condition Disconnected **shared all**;

in ICON **from** Initiator;

out ICONind **to env**;

in ICONresp **from env**;

out ICONF **to** Initiator;

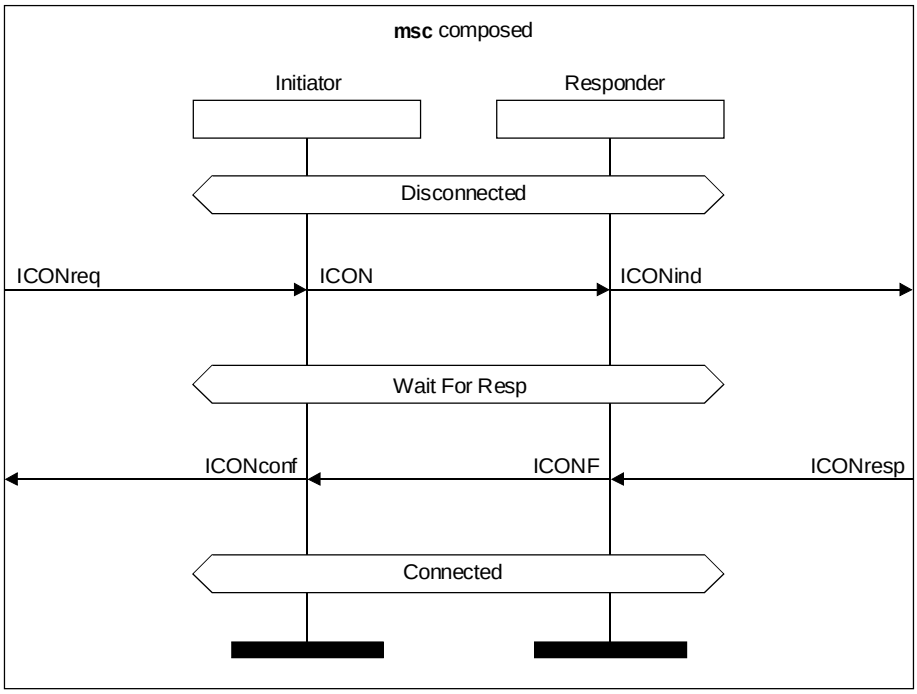
condition Connected **shared all**;

endinstance;

endmsc;

6.5 Composition de MSC/Décomposition de MSC

L'exemple 6.5 illustre la composition de MSC au moyen de conditions globales. La condition globale finale “Wait For Resp” du MSC connection request est identique à la condition globale initiale du MSC connection confirm. En conséquence, les deux MSC peuvent être composés pour donner le MSC résultant composed.



msc composed; **inst** Initiator, Responder;

instance Initiator;

condition Disconnected **shared all**;

in ICONreq **from env;**

out ICON **to** Responder;

condition Wait For Resp **shared all**;

in ICONF **from** Responder;

out ICONconf **to env**;

condition Connected **shared all**;

endinstance;

instance Responder;

condition Disconnected **shared all**;

in ICON **from** Initiator;

out ICONind **to env;**

condition Wait For Resp **shared all**;

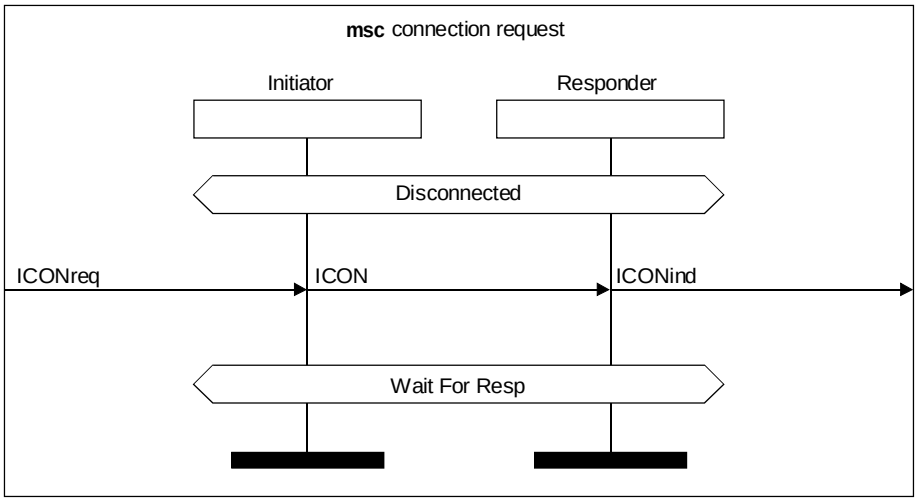
in ICONresp **from env**;

out ICONF **to** Initiator;

condition Connected **shared all**;

endinstance;

endmsc;

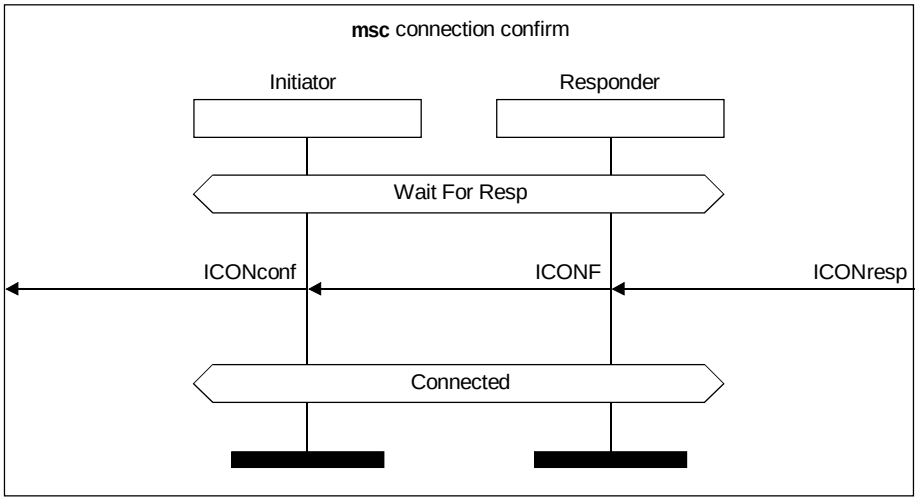


T1007560-93/d36

msc connection request; **inst** Initiator, Responder;

instance Initiator;
condition Disconnected **shared all**;
in ICONreq **from env**;
out ICON **to** Responder;
condition Wait For Resp **shared all**;
endinstance;
instance Responder;
condition Disconnected **shared all**;
in ICON **from** Initiator;
out ICONind **to env**;
condition Wait For Resp **shared all**;
endinstance;

endmsc;



T1007570-93/d37

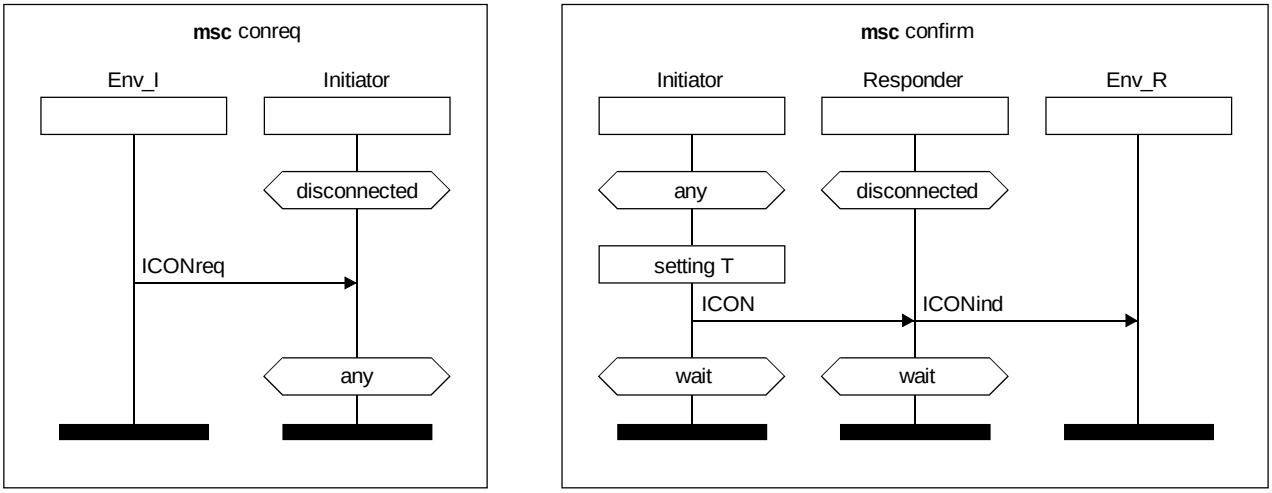
msc connection confirm; **inst** Initiator, Responder;

instance Inititator;
condition Wait For Resp **shared all**;
in ICONF **from** Responder;

out ICONconf **to env**;
condition Connected **shared all**;
endinstance;
instance Responder;
condition Wait For Resp **shared all**;
in ICONresp **from env**;
out ICONF **to** Initiator ;
condition Connected **shared all**;
endinstance;
endmsc;

6.6 Conditions locales

Dans l'exemple 6.6, les conditions locales d'une instance sont utilisées pour indiquer de possibles continuations de cette instance. A noter que les deux conditions avec le même nom “wait” du MSC “confirm” sont différenciées par les instances auxquelles elles sont attachées.



T1007580-93/d38

msc conreq; **inst** Env_I, Initiator;

instance Env_I;
out ICONreq **to** Initiator;
endinstance;
instance Initiator;
condition disconnected;
in ICONreq **from** Env_I;
condition any;
endinstance;
endmsc;

msc confirm; i**nst** Initiator, Responder, Env_R;

instance Initiator;
condition any;
action setting T;
out ICON **to** Responder;
condition wait;
endinstance;
instance Responder;
condition disconnected;
in ICON **from** Initiator;
out ICONind **to** Env_R;
condition wait;
endinstance;

instance Env_R;

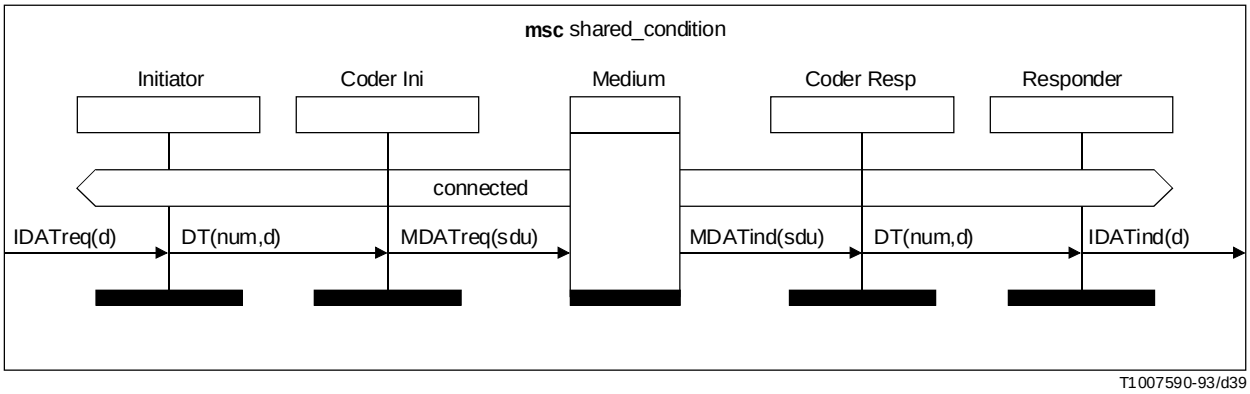
in ICONind **from** Responder;

endinstance;

endmsc;

6.7 Condition partagée et messages avec paramètres

L'exemple 6.7 contient la condition partagée “connected”. Cette condition est partagée par les instances “Initiator” et “Responder”. Les instances “Coder Ini”, “Medium”, “Coder Resp” ne sont pas concernées par cette condition locale. Dans la représentation textuelle, le mot-clé **shared** et la liste d'instances indiquent les instances auxquelles la condition est attachée.



T1007590-93/d39

msc shared_condition; **inst** Initiator, Coder Ini, Medium, Coder Resp, Responder;

instance Initiator;

condition connected **shared** Responder;

in IDATreq(d) **from env**;

out DT(num,d) **to** Coder Ini;

endinstance;

instance Coder Ini;

in DT(num,d) **from** Initiator;

out MDATreq(sdu) **to** Medium;

endinstance;

instance Medium;

in MDATreq(sdu) **from** Coder Ini;

out MDATind(sdu) **to** Coder Resp;

endinstance;

instance Coder Resp;

in MDATind(sdu) **from** Medium;

out DT(num,d) **to** Responder;

endinstance;

instance Responder;

condition connected **shared** Initiator;

in DT(num,d) **from** Coder Resp;

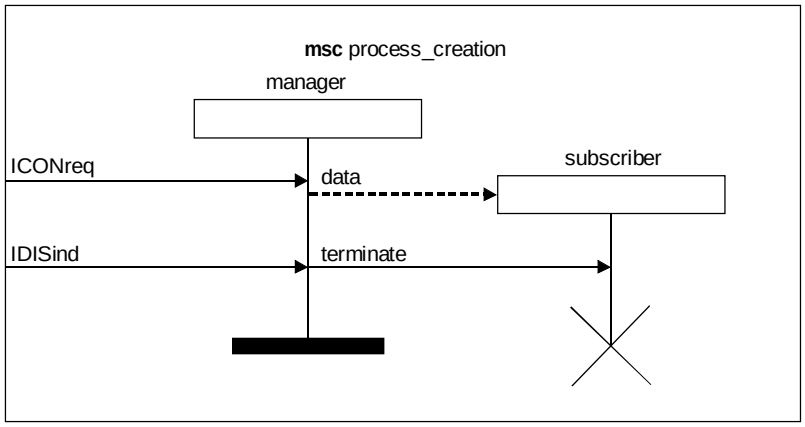
out IDATind(d) **to env**;

endinstance;

endmsc;

6.8 Création et terminaison de processus

L'exemple 6.8 illustre la création dynamique de l'instance “subscriber” engendrée par une demande de connexion (connection request) et sa terminaison engendrée par une demande de déconnexion (disconnection request).



T1007600-93/d40

msc process_creation; **inst** manager, subscriber;

instance manager;

in ICONreq **from env**;

create subscriber(data);

in IDISind **from env**;

out terminate **to** subscriber;

endinstance;

instance subscriber;

in terminate **from** manager;

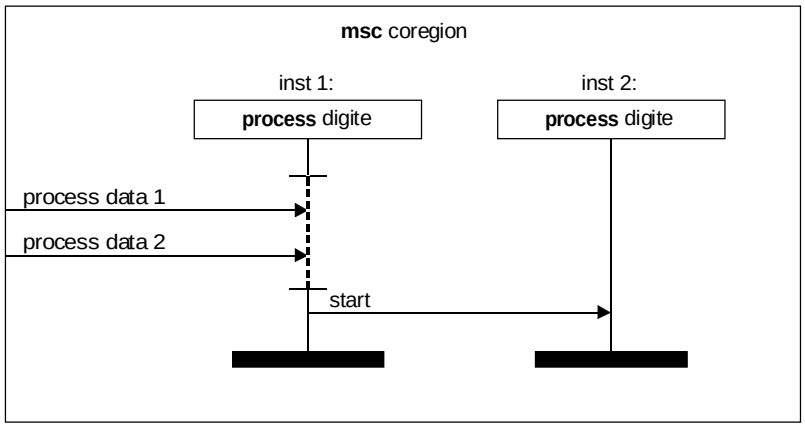
stop;

endinstance;

endmsc;

6.9 Corégion

L'exemple 6.9 montre une région de concurrence qui indique que la consommation de “process data 1” et celle de “process data 2” ne sont pas ordonnées dans le temps: “process data 1” peut être consommé avant “process data 2” ou inversement.



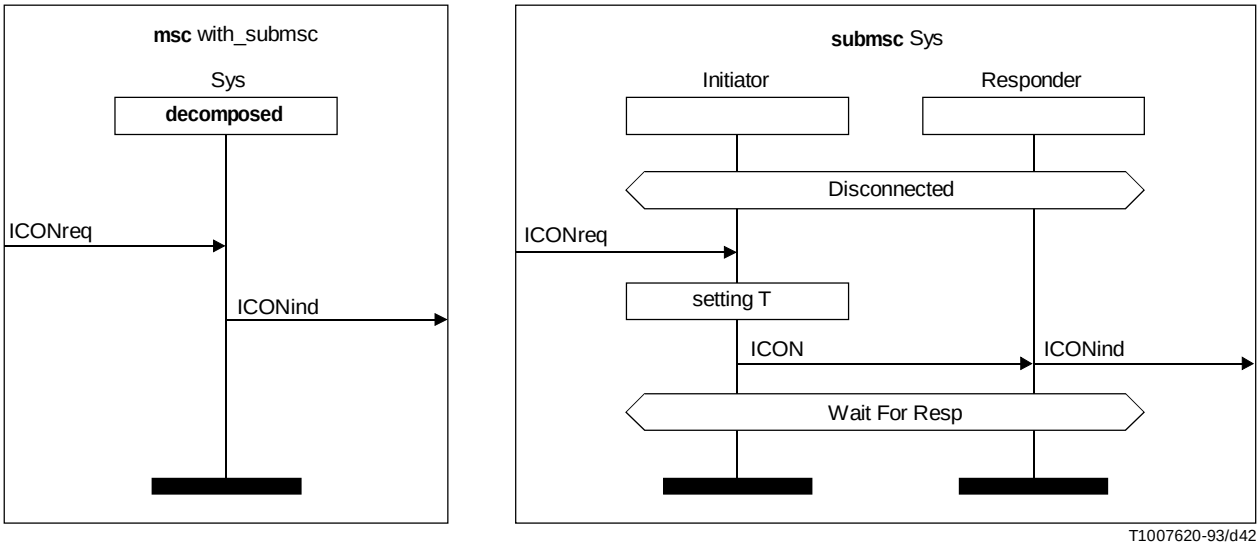
T1007610-93/d41

msc coregion; **inst** inst1, inst2;

instance inst1**: process** digite;
concurrent
in process data 1 **from env**;
in process data 2 **from env**;
endconcurrent;
out start **to** inst2;
endinstance;
instance inst2**: process** digite;
in start **from** inst1;
endinstance;
endmsc;

6.10 Sous-diagramme de séquences de messages

L'exemple 6.10 contient le sous-MSC "Sys". Ce sous-MSC est attaché à l'instance "Sys" et représente une décomposition de cette instance.



msc with_submsc; **inst** Sys;

instance Sys **decomposed**;
in ICONreq **from env**;
out ICONind **to env**;
endinstance;
endmsc;

submsc Sys; **inst** Initiator, Responder;

instance Initiator;
condition Disconnected **shared all**;
in ICONreq **from env**;
action setting T;
out ICON **to** Responder;
condition Wait For Resp **shared all**;
endinstance;
instance Responder;
condition Disconnected **shared all**;
in ICON **from** Initiator;
out ICONind **to env**;
condition Wait For Resp **shared all**;
endinstance;
endsubmsc;

Annexe A

Index

(Cette annexe fait partie intégrante de la présente Recommandation)

Les entrées de l'index sont les <**keyword**> et les symboles non-terminaux de la *Abstract grammar (Grammaire abstraite)*, de la *Concrete textual grammar (Grammaire textuelle concrète)* et de la Concrete graphical grammar (*Grammaire graphique concrète)*. Les numéros de page en **gras** correspondent aux définitions de non-terminaux.

<action area> 10; **17**
<action symbol> **17**
<action text> 17
<action> 10; **17**
<address> **12**
<alphanumeric> 3
<apostrophe> 3
<character string> 2; 4
<coevent area> **19**
<coevent> **19**
<comment area> **4**
<comment symbol> **4**
<comment> **4**
<composite special> 2
<concurrent area> 10; **19**
<condition area> 10; **13**
<condition left area> **14**
<condition left symbol> **14**
<condition middle area> **14**
<condition middle symbol> **14**
<condition name> 13; 14
<condition right area> **14**
<condition right symbol> **14**
<condition symbol> **13**
<condition> 10; **13**; 14
<coregion end symbol1> **20**
<coregion end symbol2> **20**
<coregion end symbol> 19; **20**
<coregion start symbol1> **19**; 20
<coregion start symbol2> **19**; 20
<coregion start symbol> **19**
<coregion symbol1> **20**
<coregion symbol2> **20**
<coregion symbol> **20**
<coregion> 10; **19**
<create area> 10; **18**
<create> 10; **17**; 18
<createline symbol> **18**
<dashed association symbol> **4**
<document body> **6**
<document head> **6**
<duration name> 15; 16
<end> **4**; 6; 7; 9; 12; 13; 15; 16; 17; 19; 21
<external message area> 8; **12**
<flow line symbol> **12**
<frame symbol> **8**; 21
<full stop> 3

Imprimé en Suisse

Genève, 1994