UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T Z.120

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

(03/93)

CRITERIOS PARA LA UTILIZACIÓN Y APLICABILIDAD DE TÉCNICAS DE DESCRIPCIÓN FORMAL

GRÁFICOS DE SECUENCIAS DE MENSAJES

Recomendación UIT-T Z.120

(Anteriormente «Recomendación del CCITT»)

PREFACIO

El Sector de Normalización de las Telecomunicaciones de la UIT (UIT-T) es un órgano permanente de la Unión Internacional de Telecomunicaciones. El UIT-T tiene a su cargo el estudio de las cuestiones técnicas, de explotación y de tarificación y la formulación de Recomendaciones al respecto con objeto de normalizar las telecomunicaciones sobre una base mundial.

La Conferencia Mundial de Normalización de las Telecomunicaciones (CMNT), que se reúne cada cuatro años, establece los temas que habrán de abordar las Comisiones de Estudio del UIT-T, que preparan luego Recomendaciones sobre esos temas.

La Recomendación UIT-T Z.120, preparada por la Comisión de Estudio X (1988-1993) del UIT-T, fue aprobada por la CMNT (Helsinki, 1-12 de marzo de 1993).

NOTAS

1 Como consecuencia del proceso de reforma de la Unión Internacional de Telecomunicaciones (UIT), el CCITT dejó de existir el 28 de febrero de 1993. En su lugar se creó el 1 de marzo de 1993 el Sector de Normalización de las Telecomunicaciones de la UIT (UIT-T). Igualmente en este proceso de reforma, la IFRB y el CCIR han sido sustituidos por el Sector de Radiocomunicaciones.

Para no retrasar la publicación de la presente Recomendación, no se han modificado en el texto las referencias que contienen los acrónimos «CCITT», «CCIR» o «IFRB» o el nombre de sus órganos correspondientes, como la Asamblea Plenaria, la Secretaría, etc. Las ediciones futuras en la presente Recomendación contendrán la terminología adecuada en relación con la nueva estructura de la UIT.

2 Por razones de concisión, el término «Administración» se utiliza en la presente Recomendación para designar a una administración de telecomunicaciones y a una empresa de explotación reconocida.

ÍNDICE

RESUMEN

Objetivo

El objetivo de recomendar un lenguaje de gráficos de secuencias de mensajes (MSC, *message sequence chart*) es proporcionar un lenguaje trazador para especificar y describir el comportamiento de comunicación de componentes de sistema y su entorno mediante el intercambio de mensajes. Dado que en el MSC el comportamiento de comunicación se presenta de manera muy intuitiva y transparente, especialmente en la representación gráfica, el lenguaje MSC se aprende, utiliza e interpreta con facilidad. Con respecto a otros lenguajes, puede ser empleado para sustentar metodologías de especificación, diseño, simulación, prueba y documentación de sistema.

Alcance

En esta Recomendación se presenta una definición de sintaxis para MSC en su representación abstracta, textual y gráfica. También se ofrece una descripción semántica informal.

Aplicación

El principal campo de aplicación del MSC es una especificación general del comportamiento de comunicación de los sistemas en tiempo real, en particular, los sistemas de conmutación para telecomunicaciones. Mediante los trazados de sistema seleccionados de MSC, se puede especificar primordialmente los casos «normalizados». Los casos no normalizados, que comprenden comportamientos excepcionales, se pueden describir a partir de los normalizados. De este modo, MSC puede utilizarse para la especificación de requisitos, la especificación de interfaces, la simulación y validación, la especificación de casos de prueba y la documentación para sistemas en tiempo real. El MSC se puede emplear junto con otros lenguajes de especificación, en particular el lenguaje de especificación y descripción (SDL, *specification and description languages*). En este contexto, el MSC también proporciona una base para el diseño de sistemas en SDL.

Estado/estabilidad

El estado del MSC básico es bastante estable, incluidos los constructivos de instancia, la creación y terminación de instancias, el intercambio de mensajes, las acciones, el tratamiento de temporizadores y las condiciones. Se prevén ulteriores desarrollos y ampliaciones sobre todo en lo que concierne a los conceptos estructurales.

Documentación conexa

Recomendación Q.65: Etapa 2 del método de caracterización de los servicios soportados por una RDSI.

GRÁFICOS DE SECUENCIAS DE MENSAJES

(Helsinki, 1993)

1 Introducción al MSC

El gráfico de secuencias de mensajes (MSC) muestra las secuencias de mensajes intercambiados entre componentes de sistema y su entorno. En SDL, los componentes de sistema son modelados por constructivos de servicio, proceso y bloque. Los MSC son utilizados desde hace tiempo por el CCITT en sus Recomendaciones y también en la industria, de acuerdo con diferentes convenios y bajo diversos nombres, tales como gráfico secuencial de señales (*signal sequence chart*), diagrama de flujos de información (*information flow diagram*), flujo de mensajes (*message flow*) y diagrama de flechas (*arrow diagram*).

El motivo de normalizar los MSC es suministrarles el apoyo de medios, intercambiar los MSC entre medios diferentes facilitar la correspondencia con las especificaciones SDL y armonizar su utilización dentro del CCITT.

Una parte del trabajo de normalización consiste en proporcionar una definición clara del significado de un MSC, lo que se hace en la presente Recomendación relacionando los MSC con especificaciones en SDL como sigue: un MSC describe uno o más trazados de una especificación de sistema en SDL.

Conforme a esta definición, se puede obtener un MSC de una especificación de sistema SDL existente, y luego se puede utilizar por ejemplo para anotar el resultado de una animación. Sin embargo, en general, un MSC es formulado antes de la especificación de sistema en SDL, y puede servir de:

a) visión global de un servicio ofrecido por diversas entidades;

b) declaración de los requisitos de las especificaciones SDL;

c) base para la formulación de las especificaciones SDL;

d) base para la simulación y validación de sistemas;

e) base para la selección y especificación de casos de prueba;

f) especificación semiformal de comunicación;

g) especificación de interfaz.

Puesto que un MSC suele describir solamente un comportamiento parcial, la selección de comportamientos parciales reviste suma importancia. Los candidatos a MSC son principalmente, los casos «normalizados», a partir de los cuales se construyen por lo general otros casos y se tratan comportamientos excepcionales, por ejemplo, los ocasionados por errores de diversos tipos.

A continuación se presenta una sintaxis para gráficos de secuencias de mensajes en representación abstracta, textual y gráfica. Se proporciona asimismo la descripción semántica (verbal) informal correspondiente.

Esta Recomendación está estructurada de la siguiente manera: En la cláusula 2 se exponen las reglas generales de sintaxis, dibujo y paginación. En la cláusula 3 figura la definición de sintaxis del documento de gráficos de secuencias de mensajes, que es una colección de gráficos de secuencias de mensajes. La cláusula 4 contiene la definición sintáctica de los gráficos de secuencias de mensajes y las reglas sintácticas de los constituyentes básicos, es decir, instancia, mensaje, condición, temporizador, acción, proceso, creación y terminación. En la cláusula 5 se presentan conceptos de nivel superior relacionados con la estructuración y modularización. Estos conceptos soportan una especificación en orden descendente (de arriba a abajo), y permiten refinar las instancias individuales mediante la ordenación temporal generalizada (véase 5.1) y el subgráfico de secuencias de mensajes (véase 5.2). En la cláusula 6 se ofrecen ejemplos de todos los constructivos MSC. El Apéndice I contiene referencias cruzadas para <keyword> y no terminales.

2 Reglas generales

Sólo se indican las reglas específicas del MSC. Las restantes son idénticas a las de la Recomendación Z.100.

2.1 Reglas léxicas

A diferencia de la Recomendación Z.100, <text> (<textos>) no contiene el punto y coma, puesto que <text> se utiliza dentro de la definición de acción, y el punto y coma se emplea como un terminador de acción. El punto y coma está contenido en la definición de <end> (<fin>) (véase 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 Reglas de visibilidad y denominación

Las entidades son identificadas y denominadas por medio de nombres asociados. Las entidades son agrupadas en clases de entidad para dar flexibilidad a las reglas de denominación.

Existen las siguientes clases de entidad:

a) documento MSC;

b) MSC;

c) subMSC;

d) instancia;

e) condición;

f) temporizador;

g) mensaje.

Una entidad que contiene otras entidades de acuerdo con las reglas sintácticas forma una unidad de ámbito. Existen las siguientes unidades de ámbito:

a) documento MSC;

b) MSC;

c) subMSC;

d) instancia.

Dos entidades dentro de una unidad de ámbito y que pertenecen a la misma clase de entidad no pueden tener el mismo nombre. Las ocurrencias diferentes de un nombre de condición, nombre de temporizador y nombre de mensaje dentro de una unidad de ámbito denotan la misma entidad. El nombre de una entidad es visible dentro de la unidad de ámbito, pero no fuera de ella. Sólo se puede utilizar nombres visibles al referirse a entidades.

2.3 Comentario

Comentario es una notación que representa comentarios asociados con símbolos o texto.

En la *Gramática textual concreta* se emplean dos formas de comentario. La primera es <note> (<nota>). La definición de <note> aparece en la Recomendación Z.100.

La sintaxis concreta de la segunda forma es:

<end> ::=

[<comment>] <semicolon>

<comment> ::=

comment <character string>

En la *Gramática gráfica concreta* se utiliza la sintaxis siguiente:

<comment area> ::=

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

is connected to <dashed association symbol>

<comment symbol> ::=

T1007630-93/d01

<dashed association symbol> ::=

T1007630-93/d02

Un extremo de <dashed association symbol> (<símbolo de asociación con guiones>) deberá estar conectado a la mitad del segmento vertical de <comment symbol> (<símbolo de comentario>).

Un <comment symbol> puede ser conectado a cualquier símbolo gráfico mediante un <dashed association symbol>. El <comment symbol> se considera como un símbolo cerrado que completa (mentalmente) el rectángulo que encierra el texto. Contiene texto de comentario relacionado con el símbolo gráfico.

2.4 Símbolo de texto

<text symbol> puede utilizarse en cualquier <msc diagram> y <submsc diagram> con el fin de recabar comentarios generales (globales) (véanse 4.1 y 5.2). El texto puede colocarse dentro del símbolo de texto en forma de <note>:

<text area> ::=

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

<text symbol> ::=

T1160120-94/d43

En la representación textual, se utiliza la siguiente sintaxis:

<text definition> ::=

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

La <text definition> está contenida en la definición de <msc body> (véase 4.1).

2.5 Reglas de dibujo

El usuario puede elegir el tamaño de los símbolos gráficos. Las fronteras de símbolos no deben superponerse ni cruzarse. Son excepciones a esta regla:

a) el cruce de símbolo de mensaje con símbolo de mensaje, símbolo de temporización, símbolo de reiniciación, símbolo de creación, símbolo de eje de instancia y símbolo de asociación con guiones;

b) el cruce de símbolo de temporización y símbolo de reiniciación con símbolo de mensaje, símbolo de temporización, símbolo de reiniciación, símbolo de creación y símbolo de asociación con guiones;

c) el cruce de símbolo de creación con símbolo de mensaje, símbolo de temporización, símbolo de reiniciación, símbolo de eje de instancia y símbolo de asociación con guiones;

d) el cruce de símbolo de condición con símbolo de eje de instancia;

e) la superposición de símbolo de acción con símbolo de eje de instancia en la forma de columna.

Hay dos formas del símbolo de eje de instancia y del símbolo de corregión: forma de renglón y forma de columna. No se permite mezclar ambas formas dentro de una instancia.

Si una condición compartida (véase 4.4) atraviesa una instancia que no está implicada en esa condición, el eje de instancia se dibuja a través de ésta.

Cuando el símbolo de eje de instancia tiene forma de columna, las fronteras verticales del símbolo de acción tienen que coincidir con las líneas de la columna.

Las líneas de mensaje pueden ser horizontales o diagonales descendentes (con respecto al sentido de la flecha), y pueden inclinarse.

Si en el mismo punto del eje de instancia hay una punta de flecha de mensaje y un origen de mensaje, se entenderá que el origen de mensaje está dibujado por debajo de la punta de flecha de mensaje.

2.6 Formación de página

Se puede dividir los MSC verticalmente en varias páginas. La partición horizontal se puede efectuar mediante subMSC (véase 5.2).

Cuando el MSC se divide en varias páginas, se repite el <encabezamiento MSC> (<msc heading>) en cada página, pero los símbolos de fin de instancia pueden aparecer sólo en una página (en la «última» página para la instancia en cuestión). Para cada instancia, <instance head area> debe aparecer en la primera página donde comienza la instancia y se repetirán con guiones en cada una de las páginas siguientes en las que continúa. Si los mensajes o temporizadores continúan de una página a la siguiente, el nombre del mensaje o del temporizador tiene que aparecer en las dos páginas.

Se puede numerar las páginas de un MSC para indicar la secuencia correcta de páginas. Si se omite la numeración, se debe indicar la secuencia correcta de páginas mediante condiciones globales que actúan como conectores (véase también 4.4).

3 Documento de gráfico de secuencias de mensajes

El encabezamiento del documento de gráfico de secuencias de mensajes contiene el nombre del documento y, facultativamente, a continuación de la palabra clave **related to**, el identificador (nombre de trayecto, en inglés «pathname») del documento SDL al que se refiere el MSC. No se introduce ninguna gramática gráfica especial porque se supone que el mismo documento contiene gráficos de secuencias de mensajes en representación textual y gráfica.

Gramática abstracta

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*

Gramática textual concreta

<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> }*

Semántica

Un documento de gráficos de secuencias de mensajes es una colección de gráficos de secuencias de mensajes y de subgráficos de secuencias de mensajes que se refieren facultativamente a un documento SDL correspondiente.

4 MSC básico

4.1 Gráfico de secuencias de mensajes

Un gráfico de secuencias de mensajes describe el flujo de mensajes entre instancias y, facultativamente, las acciones desencadenadas por los mensajes, según las condiciones iniciales, intermedias y finales. Un gráfico de secuencias de

mensajes describe un comportamiento parcial de un sistema. Si bien es obvio que el nombre *gráfico de secuencias de mensajes* se origina en su representación gráfica, el MSC se utiliza tanto para la representación gráfica como para la textual.

El encabezamiento del gráfico de secuencias de mensajes consiste en el nombre del MSC y (facultativamente) en una lista de las instancias contenidas en el cuerpo del gráfico de secuencias de mensajes.

Gramática abstracta

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*

La *Instance-list* (*lista-de-instancias*) en el *MSC-interface* (*interfaz-MSC*), si está presente, debe contener las mismas instancias especificadas en el *MSC-body* (*cuerpo-MSC*).

Gramática textual concreta

<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> }*

Gramática gráfica concreta

<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>

Semántica

Un MSC describe la comunicación entre un número de componentes de sistema, y entre estos componentes y el resto del mundo, denominado entorno. Para cada componente de sistema tratado por un MSC hay un eje de instancia. La comunicación entre componentes de sistema se efectúa mediante mensajes. El envío y el consumo de mensajes son dos eventos distintos. Se supone que el entorno de un MSC es capaz de recibir y enviar mensajes desde y hacia el gráfico de secuencia de mensajes; no se supone ninguna ordenación de los eventos de mensaje dentro del entorno. Aunque el comportamiento del entorno es no determinístico, se supone que obedece a las restricciones impuestas por el gráfico de secuencias de mensajes.

No se supone un eje temporal global para un MSC. El tiempo transcurre desde arriba hacia abajo a lo largo de cada eje de instancia, pero no se supone una escala temporal propia. Si no se introduce la corregión (véase 5.1), se supone una ordenación temporal total de eventos a lo largo de cada eje de instancia. Los eventos de instancias diferentes se ordenan únicamente mediante mensajes: un mensaje debe ser enviado antes de ser consumido (véase 4.3). No se prescribe ningún otro tipo de ordenación. Por consiguiente, un gráfico de secuencias de mensajes impone una ordenación parcial al conjunto de eventos contenido. Una relación binaria, que es transitiva, antisimétrica y reflexiva se denomina orden parcial.

Para las entradas de mensaje (etiquetadas por ?mi) y las salidas (etiquetadas por !mi) del MSC de la Figura 1a) se obtiene la relación de ordenación siguiente: !m2 < ?m2, !m3 < ?m3, !m4 < ?m4, ?m1 < !m2 < !m3 < ?m4, ?m2 < !m4 junto con el cierre transitivo.

La ordenación parcial se puede describir en forma mínima (sin una representación explícita del cierre transitivo) mediante su grafo de conectividad [véase la Figura 1b)].

La semántica de un MSC se puede relacionar con la semántica del SDL mediante la noción de grafo de alcanzabilidad (*reachability graph*). Cada secuencia de un MSC describe un trazado de un nodo a otro nodo (o conjunto de nodos) del grafo de alcanzabilidad que describe el comportamiento de una especificación de sistema SDL. El grafo de alcanzabilidad consiste en nodos y bordes. Los nodos denotan estados globales del sistema. Un estado global del sistema está determinado por los valores de las variables, el estado de ejecución de cada proceso y el contenido de las colas de mensajes. Los bordes corresponden a los eventos que origina el sistema, por ejemplo, el envío y consumo de un mensaje o la ejecución de una tarea. Una secuencia de un MSC denota una ordenación total de eventos compatible con la ordenación parcial definida por el MSC.

Obsérvese que el grafo de alcanzabilidad muestra más eventos que el MSC. Por tanto, las secuencias de los MSC no son trayectos completos del grafo de alcanzabilidad.

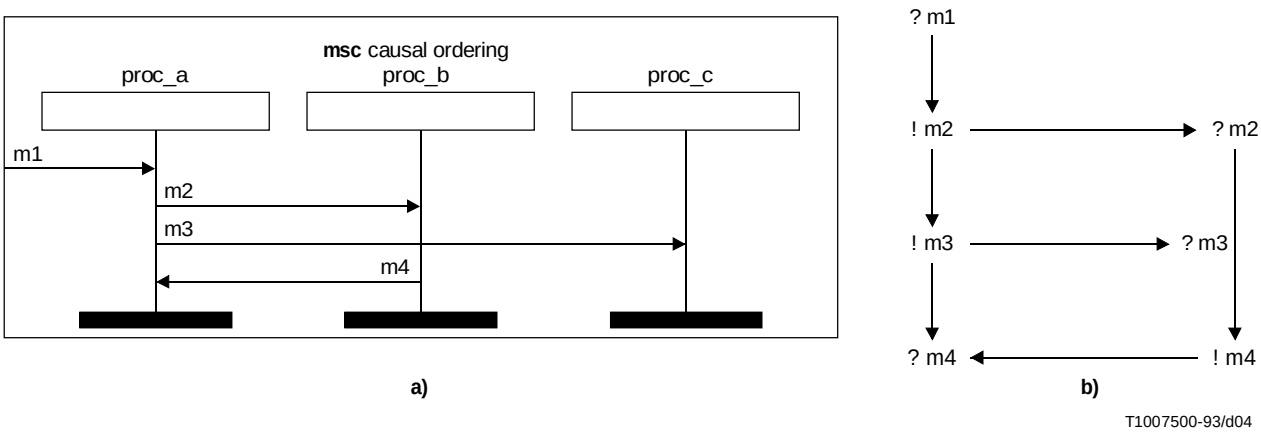


FIGURA 1/Z.120

Gráfico de secuencias de mensajes y grafo de conectividad correspondiente

4.2 Instancia

Un gráfico de secuencias de mensajes se compone de instancias de entidades que interactúan. Una instancia de una entidad es un objeto, que tiene las propiedades de esa entidad. Con relación al SDL, una entidad puede ser un proceso, bloque o servicio SDL. En el encabezamiento de la instancia se puede especificar el nombre de la entidad, por ejemplo, nombre de proceso, además del nombre de instancia. En el cuerpo de la instancia se especifica la ordenación de los eventos. Mediante la palabra clave **decomposed**, se puede unir a la instancia un subgráfico de secuencias de mensajes con el mismo nombre.

Gramática abstracta

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

Para cada instancia que contiene la palabra clave **decomposed** hay que especificar el *Submsc* correspondiente (véase 5.2), con el mismo nombre. Para cada *Message-output* (*salida de mensaje*) de una instancia descompuesta hay que especificar un *Message-output* correspondiente, enviado al exterior de *Submsc*. Los mensajes entrantes deben mantener una correspondencia análoga.

Gramática textual concreta

<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> }*

Gramática gráfica concreta

<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> }

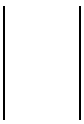
<instance axis symbol> ::=

<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

El <instance heading> (encabezamiento de instancia) puede colocarse encima o debajo del <instance head symbol> (símbolo de encabezamiento de instancia) o puede dividirse de modo que el <instance name> (nombre de instancia) esté colocado por encima del <instance head symbol> (símbolo de encabezamiento de instancia) y, la <instance kind> (clase de instancia) esté adentro. En este último caso el símbolo dos puntos es facultativo (y tiene que estar encima, si está presente) y la palabra clave facultativa **decomposed** debe estar adentro, si está presente. No se permite mezclar <instance axis symbol1> (símbolo de eje de instancia) con <instance axis symbol2>.

Semántica

En el cuerpo de gráfico de secuencias de mensajes las instancias están definidas. El símbolo de fin de instancia determina el fin de la descripción de la instancia dentro del MSC. No describe la terminación de la instancia (véase 4.8: Parada de proceso). Consiguientemente, el símbolo de encabezamiento de la instancia determina el comienzo de la descripción de la instancia dentro del MSC. No describe la creación de la instancia (véase 4.7: Creación de proceso).

En el contexto del SDL, una instancia puede referirse a un proceso (palabra clave **process**), a un servicio (palabra clave **service**) o a un bloque (palabra clave **block**). Fuera del SDL, puede referirse a cualquier clase de entidad. La definición de instancia proporciona una descripción de evento para entradas y salidas de mensajes, acciones, condiciones compartidas y locales, temporizador, creación y parada de proceso. Fuera de las corregiones (véase 5.1), se supone una ordenación total de los eventos a lo largo de cada eje de instancia. Dentro de las corregiones no se supone ninguna ordenación temporal de eventos.

Mediante la palabra clave **decomposed** se puede unir a una instancia un subgráfico de secuencias de mensajes con el mismo nombre.

4.3 Mensaje

Un mensaje dentro de un MSC representa un intercambio de información entre dos instancias, o entre una instancia y el entorno.

Un mensaje intercambiado entre dos instancias se puede dividir en dos eventos: la entrada de mensaje y la salida de mensaje; por ejemplo, el segundo mensaje de la Figura 1a) se puede dividir en !m2 (salida) y ?m2 (entrada). Los mensajes provenientes del entorno están representados por una entrada de mensaje, y los enviados hacia el entorno, por una salida de mensaje. En la representación textual, la entrada de mensaje está representada por la palabra clave **in**, y la salida de mensaje, por la palabra clave **out**, ambas seguidas por el nombre de mensaje y, facultativamente, un nombre de instancia de mensaje. A un mensaje puede asignársele parámetros entre paréntesis. En la representación gráfica, los mensajes están representados por flechas. La declaración de la lista de parámetros es facultativa para la entrada de mensajes.

La correspondencia entre salidas de mensaje y entradas de mensaje tiene que definirse de manera única. En la representación textual, normalmente la correspondencia entre entradas y salidas sigue después de la identificación de nombre de mensaje y de la especificación de dirección. Cuando el nombre de mensaje y la dirección no son suficientes para establecer una correspondencia única, hay que emplear el nombre de instancia de mensaje. En la representación gráfica, un mensaje se representa por una flecha.

Gramática abstracta

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

No se permite que el *Message-output* dependa causalmente de su *Message-input* (*entrada de mensaje*) a través de otros mensajes. Esto sucede cuando el grafo de conectividad (véase 4.1) contiene bucles. Si se especifica una *Parameter-list* (*lista de parámetros*) para un *Message-input*, habrá que especificarla también para el *Message-output* correspondiente. Las *Parameter-lists* tienen que ser idénticas.

Gramática textual concreta

<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**

Para los mensajes intercambiados entre instancias se deben respetar las reglas siguientes: para cada <message output> hay que especificar un <message input> correspondiente, y viceversa. Cuando <message name> (nombre de mensaje) y <address> (dirección) no bastan para establecer una correspondencia única hay que utilizar <message instance name>.

Gramática gráfica concreta

<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

Se autoriza la imagen especular de <message symbol> (símbolo de mensaje).

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

NOTA – En la representación gráfica, el nombre de instancia de mensaje no es necesario para una descripción sintáctica única.

Semántica

En un MSC, la salida de mensaje denota el envío de mensaje (correspondiente a salida SDL), la entrada de mensaje denota el consumo del mensaje (correspondiente a entrada SDL). No se proporciona ningún constructivo especial para la recepción del mensaje (entrada en la memoria tampón). Tampoco se unen definiciones de tipo a los parámetros de la lista de parámetros.

Si en el mismo punto del eje de instancia hay una punta de flecha de mensaje y un origen de mensaje, se entiende que el origen del mensaje se dibuja por debajo de la punta de flecha de mensaje.

4.4 Condición

Una condición describe o bien un estado de sistema global (condición global) que se refiere a todas las instancias contenidas en el MSC o bien a un estado que se refiere a un subconjunto de instancias (condición no global). En el segundo caso, la condición puede ser local, es decir, estar unida a una sola instancia. En la representación textual la condición tiene que definirse para cada instancia a la que está unida utilizando la palabra clave **condition** junto con el nombre de la condición. Si la condición se refiere a varias instancias, entonces la palabra clave **share**, junto con la lista de instancias, designa el conjunto de instancias por la que es compartida la condición. Puede definirse una condición global que se refiere a todas las instancias por medio de la palabra clave **shared all**.

Gramática abstracta

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

Gramática textual concreta

<condition> ::=

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

<shared instance list> ::=

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

Para cada <instance name> (nombre de instancia) contenido en una <shared instance list> (lista de instancias compartidas) de una <condition> hay que especificar una instancia con la <condition> compartida correspondiente. Si la instancia *b* está contenida en la <shared instance list> de una <condition> compartida unida a la instancia *a*, entonces la instancia *a* debe estar contenida en la <shared instance list> de la <condition> compartida correspondiente, unida a la instancia *b*. Si la instancia *a* y la instancia *b* comparten la misma <condition>, en cada mensaje intercambiado entre estas instancias, <message input> y <message output> se deben colocar antes o después de <condition>.

Gramática gráfica concreta

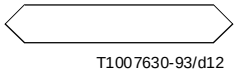
<condition area> ::=

<local condition area> | <shared condition area>

<local condition area> ::=

<condition symbol> ***contains*** <condition name>

<condition symbol> ::=



T1007630-93/d12

<shared condition area> ::=

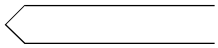
<condition left area> | <condition middle area> | <condition right area>

<condition left area> ::=

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

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

<condition left symbol> ::=



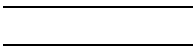
T1007630-93/d13

<condition middle area> ::=

<condition middle symbol>

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

<condition middle symbol>::=



T1007630-93/d14

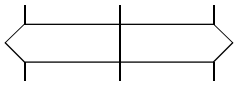
<condition right area> ::=

<condition right symbol>

<condition right symbol> ::=

T1007630-93/d15

Una <shared condition area> (zona de condición compartida) se divide en varias zonas: <condition left area> (zona izquierda de condición), <condition middle area> (zona central de condición) y <condition right area> (zona derecha de condición) que están alineadas horizontalmente, a fin de formar un solo símbolo asociado con el <condition name>. <local condition area> (zona de condición local) se refiere a una instancia, y <shared condition area> (zona de condición compartida) tiene una conexión con otras instancias. Si una <condition> compartida atraviesa un <instance axis symbol> que no está implicado en esta condición, se traza la línea de <instance axis symbol> a través del símbolo de condición:



T1007630-93/d16

Semántica

Las condiciones globales, que representan estados globales de sistema, se refieren a todas las instancias involucradas en el MSC. Mediante la palabra clave compartida en toda la representación textual se puede especificar en cada MSC:

- una condición global inicial (estado inicial global);
- una condición global final (estado final global); y
- condiciones globales intermedias (estados intermedios globales).

Las condiciones globales inicial, intermedia y final no se introducen meramente a efectos de documentación, en el sentido de comentarios o ilustraciones. En el caso de un conjunto completo de gráficos de secuencias de mensajes, estas condiciones tienen una función bien definida. Las condiciones globales determinan posibles continuaciones de gráficos de secuencias de mensajes que contienen el mismo conjunto de instancias mediante la identificación de condición: cuando la condición global final de MSC1 es idéntica a la condición global inicial de MSC2, puede considerarse que MSC2 es una continuación de MSC1.

Las condiciones globales definen también posibles composiciones y descomposiciones de MSC. Tras descomponer un MSC en una condición global intermedia en MSC1 y MSC2, la condición global intermedia global se convierte en condición final global para MSC1 y condición inicial global para MSC2.

Para utilizar ampliamente la composición de MSC, las condiciones globales que se refieren al estado del sistema completo resultan demasiado restrictivas. Es necesario dividirlas en condiciones no globales, que se refieren a un subconjunto de instancias. Las condiciones locales unidas a instancias individuales son un caso especial de las

condiciones no globales. Mediante las condiciones no globales también se pueden definir combinaciones de gráficos de secuencias de mensajes con diferentes conjuntos de instancias, mientras que la continuación sólo se refiere a un subconjunto común de instancias.

Hay que señalar que un MSC que termina con una condición global puede continuar en un MSC que empieza con una condición no global y viceversa, si ambas condiciones se refieren al mismo (sub)conjunto de instancias. Como generalización de las reglas para las condiciones globales, se define la continuación de dos MSC con un conjunto común no vacío de instancias de la siguiente manera: MSC2 es una continuación de MSC1 mediante condiciones (no) globales si para cada instancia que ambos MSC tienen en común, MSC1 finaliza con una condición (no) global y MSC2 comienza con una condición (no) global correspondiente. En este contexto, «correspondiente» significa que ambas condiciones se refieren al mismo subconjunto de instancias, y que ambas condiciones concuerdan con respecto a la identificación de nombre. Además, cada condición (no) global de MSC2 debe tener una condición (no) global correspondiente en MSC1. Por consiguiente, MSC1 y MSC2 pueden componerse.

Un MSC que contiene condiciones no globales intermedias se puede descomponer en MSC1 y MSC2. Después de la descomposición, las condiciones intermedias se convierten en condiciones finales para MSC1, y condición inicial para MSC2. Asimismo, los MSC obtenidos se pueden combinar de acuerdo con las leyes estipuladas más arriba.

4.5 Temporizador

En los MSC se puede especificar la fijación de un temporizador y una temporización subsiguiente debida a la expiración del temporizador o la fijación de un temporizador y una reiniciación subsiguiente (supervisión del tiempo). En la representación gráfica, el símbolo de fijación es un rectángulo pequeño. El símbolo de temporización está representado por un mensaje enviado por una instancia a sí misma. El símbolo de reiniciación es un símbolo de temporización modificado con una flecha de entrada de trazo interrumpido.

La especificación del nombre de instancia de temporizador y la duración del temporizador es facultativa en las representaciones textual y gráfica.

Gramática abstracta

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*

Gramática textual concreta

<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>

Para <set> (fijación) y <timeout> (temporización) deben cumplirse las reglas siguientes: para cada <set> hay que especificar un correspondiente <timeout> o <reset> (reiniciación) y viceversa. Cuando <timer name> no basta para establecer una correspondencia única, se empleará <timer instance name> (nombre de instancia de temporizador).

Gramática gráfica concreta

<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> ::=

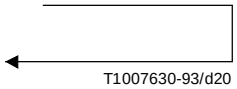
<timeout symbol> ***is associated with*** <timer name> [(<duration name>)]

is connected to <timer set area>

<timeout symbol> ::=

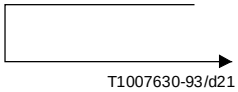
<timeout symbol1> | <timeout symbol2>

<timeout symbol1> ::=



T1007630-93/d20

<timeout symbol2> ::=



T1007630-93/d21

Un lado de <set symbol> (símbolo de fijación) debe coincidir con <instance axis symbol>. En el caso de <instance axis symbol2>, <set symbol> debe quedar fuera de la columna formada por <instance axis symbol2>.

El extremo de comienzo de <reset symbol> y <timeout symbol> debe conectarse a <set symbol> en el medio del lado opuesto al lado que coincide con <instance axis symbol>.

Semántica

Fijar y reiniciar son constructivos de temporizador tomados del SDL. Fijar indica la puesta en marcha del temporizador, y reiniciar, su reiniciación. La temporización corresponde al consumo de la señal de temporizador en el SDL.

4.6 Acciones

Además del intercambio de mensajes, se puede especificar acciones en los MSC. Se une un texto informal a las acciones.

Gramática abstracta

Action :: *Informal-Text*

Gramática textual concreta

<action><action> ::=

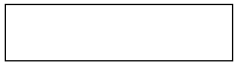
action <action text> <end>

Gramática gráfica concreta

<action area><action area> ::=

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

<action symbol><action symbol> ::=



T1007630-93/d22

Cuando el eje de instancia es una columna, la anchura de <action symbol> debe coincidir con la anchura de la columna.

Semántica

Una acción describe una actividad interna de una instancia.

4.7 Creación de proceso

Al igual que en el SDL, en los MSC se puede especificar la creación y terminación de instancias de proceso. Una instancia de proceso puede ser creada por otra instancia de proceso. Ningún evento de mensaje antes de la creación debe referirse a la instancia creada.

Gramática abstracta

Create-node :: *Instance-name*

*Parameter-name**

Gramática textual concreta

<create> ::=

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

Para cada <create> (crear) debe existir una instancia correspondiente con el nombre especificado. <instance name> debe referirse a una instancia con proceso de tipo, si su tipo está especificado. Una instancia sólo puede ser creada una vez, es decir, dentro de un MSC no deben aparecer dos o más <create> con el mismo nombre. Ningún evento de mensaje antes de la creación debe referirse a la instancia creada.

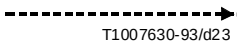
Gramática gráfica concreta

<create area> ::=

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

is connected to <instance head symbol>

<createline symbol> ::=



T1007630-93/d23

Semántica

Crear define la creación dinámica de una instancia de proceso por otra. Crear se ejecuta inmediatamente.

4.8 Parada de proceso

En cierto sentido, la detención o parada de proceso es la contrapartida de la creación de proceso. Sin embargo, una instancia de proceso sólo puede detenerse por sí misma, mientras que una instancia de proceso es creada por otra instancia de proceso.

Gramática abstracta

Stop-node :: *( )*

Gramática textual concreta

<stop> ::=

stop <end>

Gramática gráfica concreta

<stop symbol> ::=

T1007630-93/d24

Semántica

La parada al final de un cuerpo de instancia causa la terminación de esa instancia de proceso.

5 Conceptos estructurales

En esta cláusula se presentan conceptos de nivel superior, que se refieren a la ordenación temporal generalizada (corregión) y a la composición y descomposición de instancias.

En los MSC, las instancias pueden referirse a niveles de abstracción diferentes, como lo indican las palabras clave (**block, service, process**). Las correspondientes operaciones de descomposición en las instancias se pueden definir determinando la transición entre diferentes niveles de abstracción. Si las instancias están compuestas de una sola instancia, es preciso relajar la ordenación total de eventos en esa instancia para preservar el comportamiento observable externamente.

5.1 Corregión

En general, la ordenación total de eventos en cada instancia (véase 4.1) puede no ser apropiada para las entidades que se refieren a un nivel más alto que los procesos SDL.

En consecuencia, se introduce la corregión para especificar los eventos no ordenados en una instancia. La corregión trata, en particular, el caso prácticamente importante de dos o más mensajes entrantes cuya ordenación de consumo se puede intercambiar.

Gramática abstracta

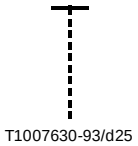
Coregion	::	*Coevent**
Coevent	=	*Message-input* |
		Message-output

Gramática textual concreta

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

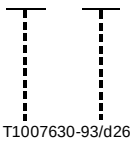
Gramática gráfica concreta

<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> ::



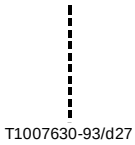
T1007630-93/d26

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

<coregion symbol> ::=

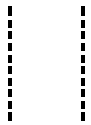
<coregion symbol1> | <coregion symbol2>

<coregion symbol1> ::=



T1007630-93/d27

<coregion symbol2> ::=

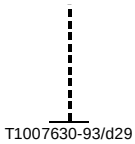


T1007630-93/d28

<coregion end symbol> ::=

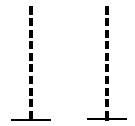
<coregion end symbol1> | <coregion end symbol2>

<coregion end symbol1> ::=



T1007630-93/d29

<coregion end symbol2> ::=



T1007630-93/d30

Dentro de una instancia, no se debe mezclar <coregion start symbol1> (símbolo de comienzo de corregión) <coregion symbol1>, <coregion end symbol1> (símbolo de fin de corregión) e <instance axis symbol1> con <coregion start symbol2>, <coregion symbol2>, <coregion end symbol2> e <instance axis symbol2>.

Semántica

Para los MSC, se supone una ordenación temporal total de eventos dentro de cada instancia. La corregión permite una excepción: los eventos contenidos en la corregión no están ordenados temporalmente.

5.2 Subgráfico de secuencias de mensajes

Una instancia de un MSC se puede descomponer en forma de subgráfico de secuencias de mensajes (subMSC), lo que permite formular una especificación de arriba a abajo.

Un subMSC tiene esencialmente una estructura análoga a la de un MSC. Se distingue del MSC por la palabra clave **submsc**. Un subMSC se caracteriza por su relación con una instancia descompuesta, que contiene la palabra clave **decomposed** y que tiene el mismo nombre que el subMSC. La relación es suministrada por los mensajes conectados al exterior del subMSC y los correspondientes mensajes enviados y consumidos por la instancia descompuesta.

Gramática abstracta

Submsc :: *Message-sequence-chart*

El nombre de un *Submsc* debe ser idéntico al nombre de una *Instance-definition* correspondiente que contiene la palabra clave **decomposed** dentro de otro *Message-sequence-chart* o *Submsc*. Para cada *Message-output* enviado al exterior de

un *Submsc* hay que especificar un *Message-output* correspondiente en la instancia descompuesta. Se mantendrá una correspondencia análoga para los mensajes entrantes.

Gramática textual concreta

<submsc> ::=

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

Gramática gráfica concreta

<submsc diagram> ::=

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

<submsc symbol> ::=

<frame symbol>

<submsc heading> ::=

submsc <submsc name>

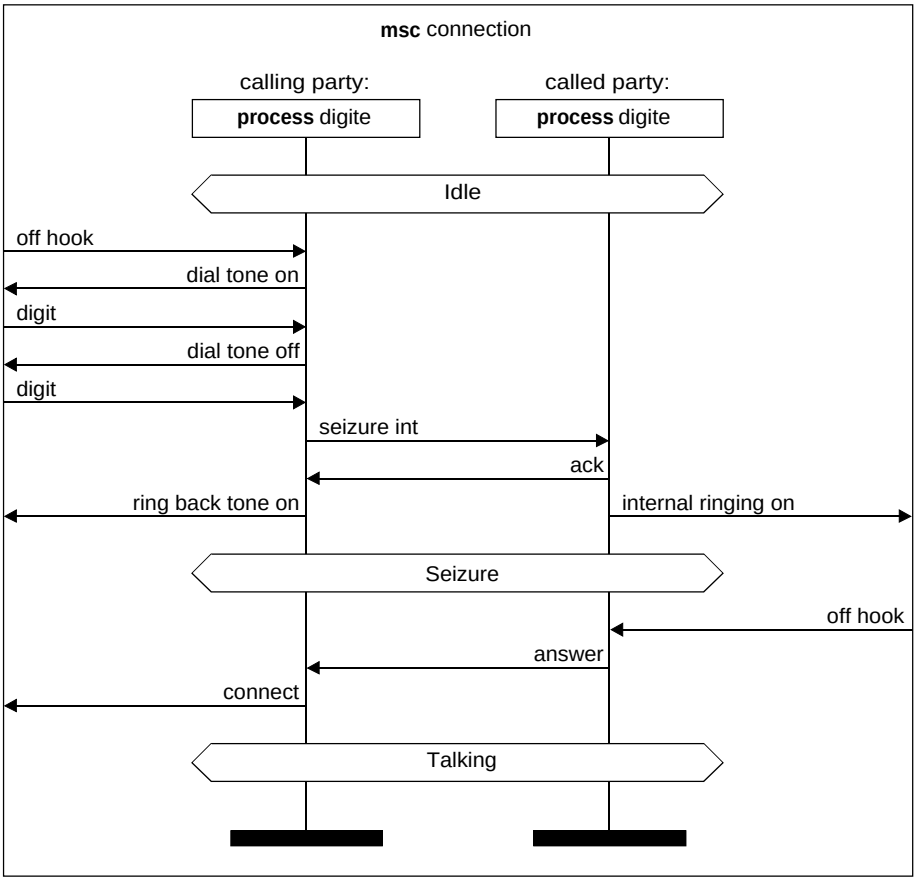
Semántica

Un subMSC puede estar unido a una instancia por medio de la palabra clave **decomposed**. El subMSC representa una descomposición de esa instancia que no afecta su comportamiento observable. En la representación textual, los mensajes dirigidos hacia y desde el exterior del subMSC se caracterizan por la dirección **env**, y en la representación gráfica, por la conexión con la frontera del subMSC (símbolo de casilla). Su conexión con las instancias externas es suministrada por los mensajes enviados y consumidos por la instancia descompuesta, mediante la identificación de nombre de mensaje. Debe ser posible hacer corresponder el comportamiento externo del subMSC con los mensajes de la instancia descompuesta. La ordenación de los eventos de mensaje especificada en una instancia descompuesta debe ser preservada en el subMSC. Las acciones y condiciones dentro del subMSC podrán ser consideradas como un refinamiento de las acciones y condiciones de la instancia descompuesta. Sin embargo, y a diferencia de lo que sucede con los mensajes, no se supone una correspondencia formal con la instancia descompuesta, es decir, el refinamiento de las acciones y condiciones no tiene que obedecer a reglas formales.

6 Ejemplos de gráficos de secuencias de mensajes

6.1 Diagrama de flujo de mensajes normalizado

El ejemplo 6.1 ilustra el establecimiento simplificado de la conexión en un sistema de conmutación. Muestra los constructivos de MSC más básicos: instancias (de proceso), entorno, mensajes y condiciones 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;

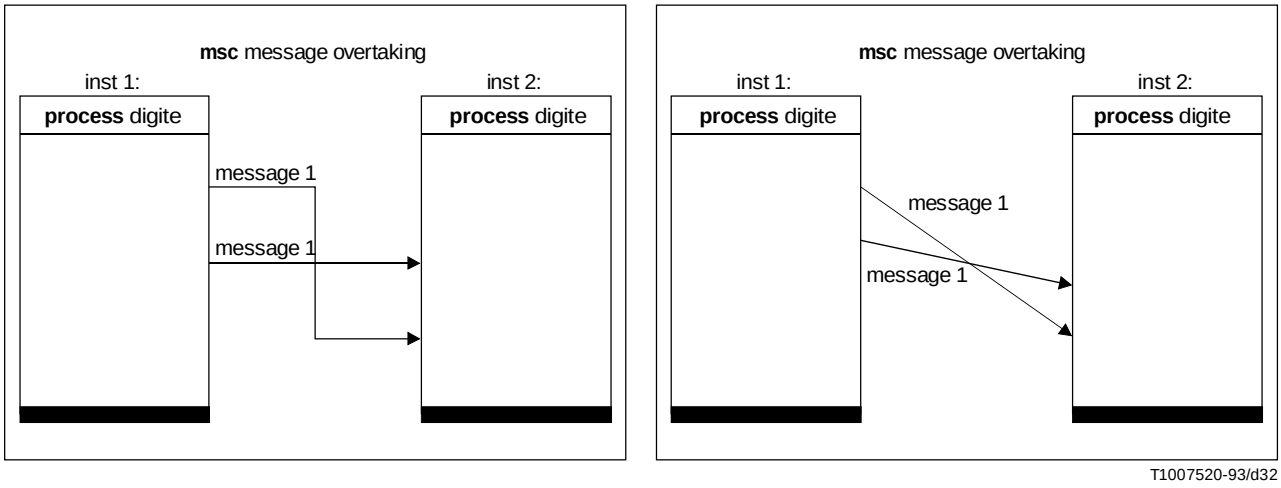
condition Talking **shared all**;

endinstance;

endmsc;

6.2 Adelantamiento de mensajes

En el ejemplo 6.2 se muestra el adelantamiento de dos mensajes con el mismo nombre: "message 1". En la representación textual, se emplea los nombres de instancia de mensaje (a, b) para una correspondencia única entre entrada y salida de mensaje. En la representación gráfica, los mensajes se representan mediante flechas horizontales, una doblada para indicar el adelantamiento o mediante flechas inclinadas que se cruzan.



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;

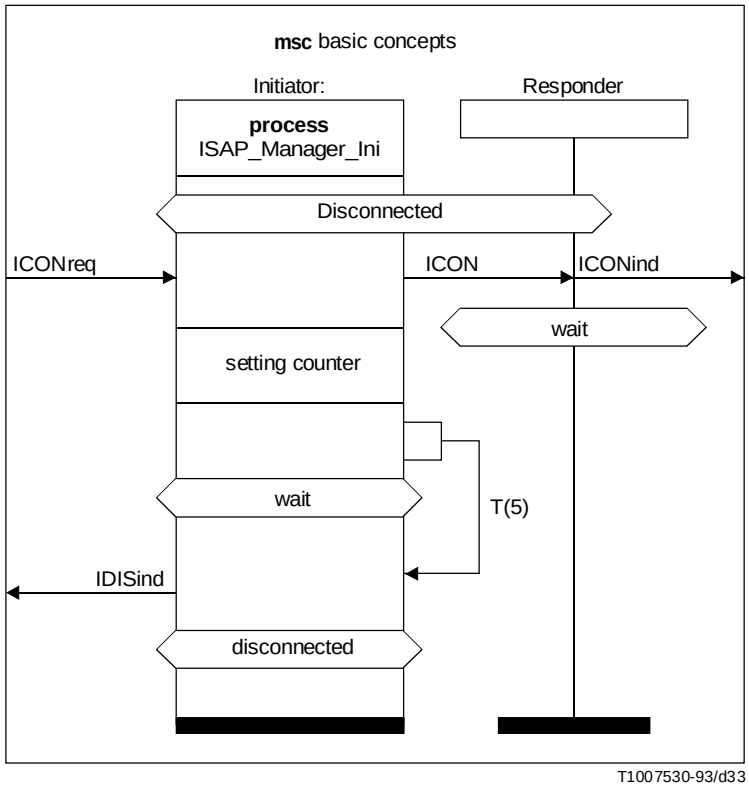
in message 1, a **from** inst 1;

endinstance;

endmsc;

6.3 Conceptos básicos de MSC

En el ejemplo 6.3 se ofrece los constructivos MSC básicos: instancias, entorno, mensajes, condiciones, acciones y temporización. En la representación gráfica se utiliza dos tipos de símbolo de instancia: el renglón y la columna.



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**;

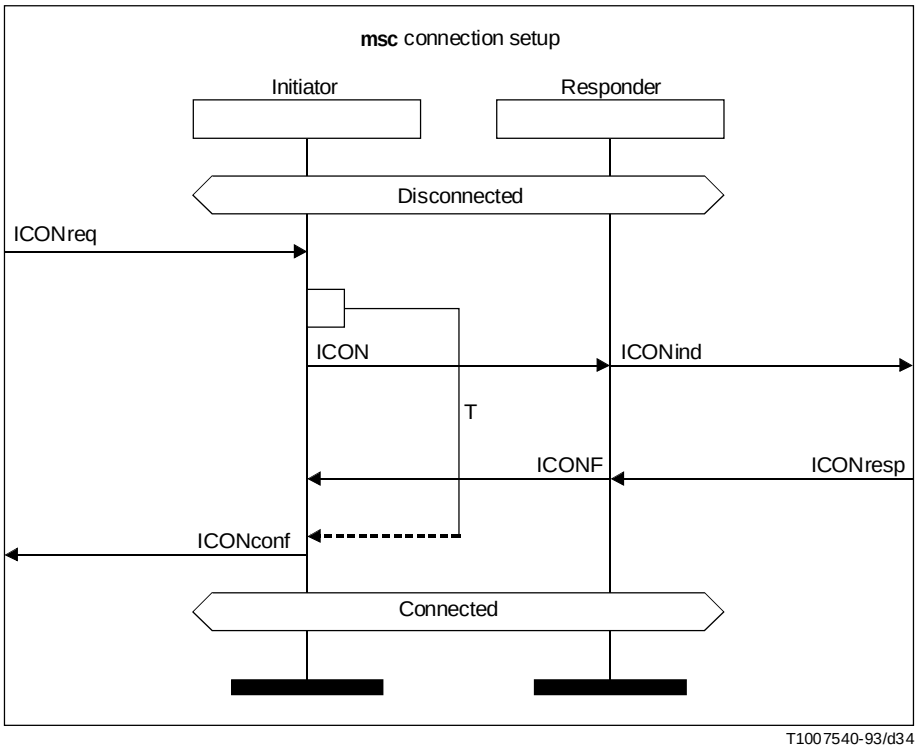
condition wait;

endinstance;

endmsc;

6.4 MSC con supervisión temporal

El MSC de establecimiento de conexión del ejemplo 6.4 contiene una reiniciación de temporizador.



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;

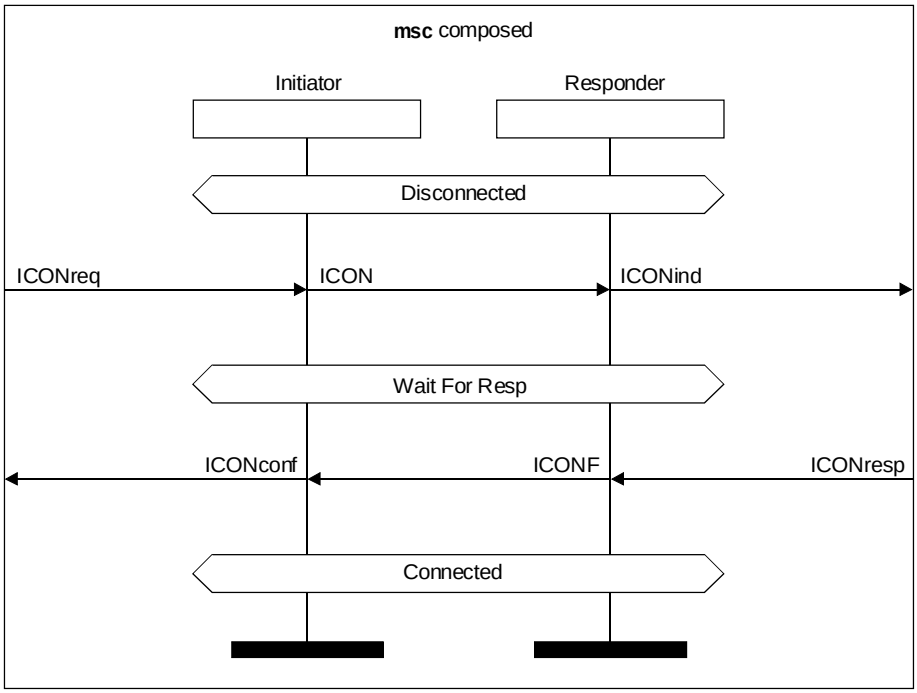
condition Connected **shared all**;

endinstance;

endmsc;

6.5 Composición/descomposición de MSC

En el ejemplo 6.5 se muestra la composición de los MSC mediante condiciones globales. La condición global final "Wait For Resp" («Espera de respuesta») del MSC de petición de conexión es idéntica a la condición global inicial del MSC de confirmación de conexión. Por tanto, los dos MSC se pueden componer para obtener el MSC compuesto resultante.



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**;

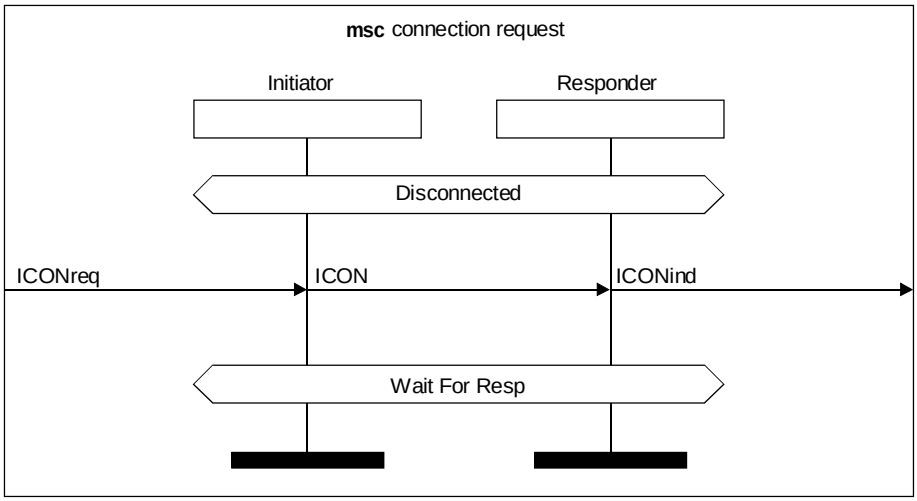
in ICONresp **from env**;

out ICONF **to** Initiator;

condition Connected **shared all**;

endinstance;

endmsc;

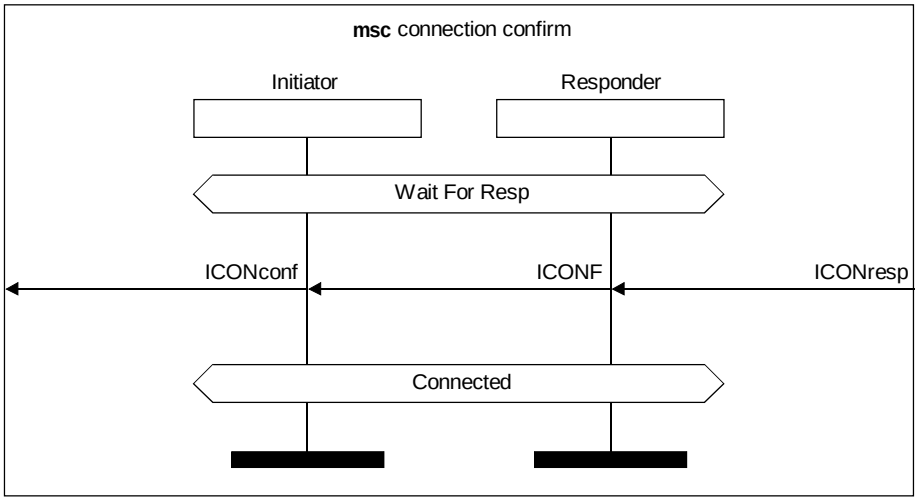


T1007560-93/d36

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

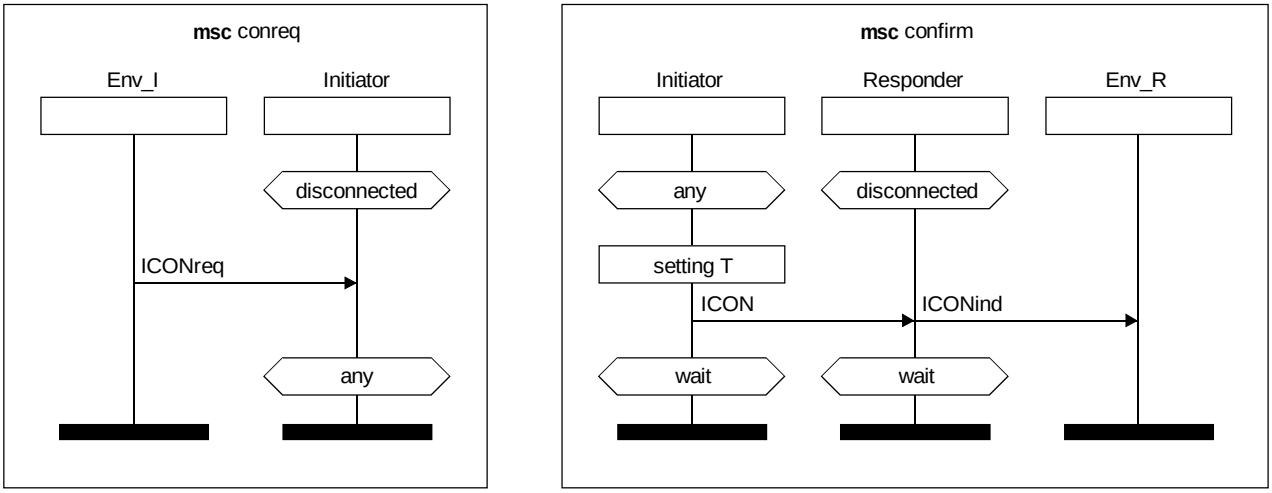
msc connection confirm; **inst** Initiator, Responder;

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 Condiciones locales

En el ejemplo 6.6 se emplean condiciones locales que hacen referencia a una instancia para indicar posibles continuaciones de esa instancia. Nótese que las dos condiciones con el mismo nombre "wait" en el MSC "confirm" se diferencian por las instancias a las que están unidas.



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;

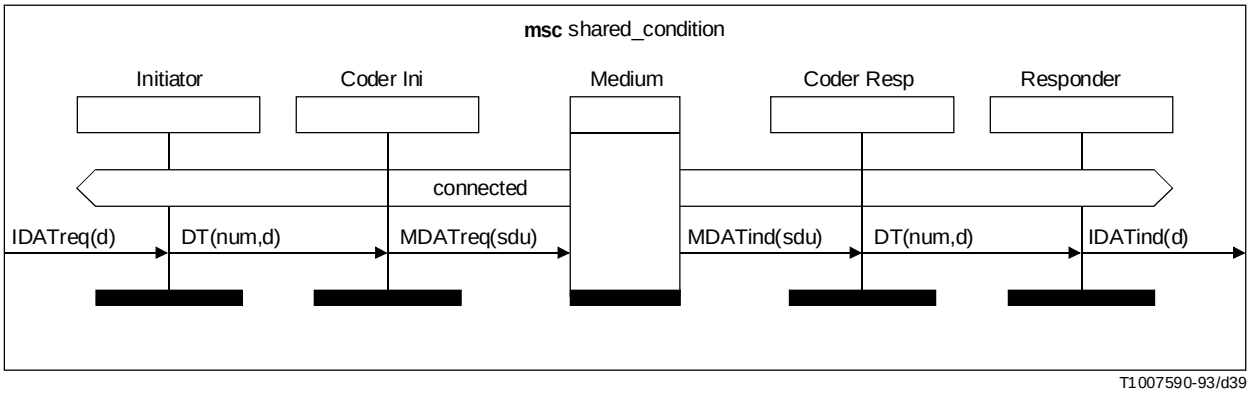
msc confirm; i**nst** Initiator, Responder, Env_R;

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 Condición compartida y mensajes con parámetros

El ejemplo 6.7 contiene la condición compartida "connected". Esta condición es compartida por las instancias "Initiator" (iniciador) y "Responder" (respondedor). Las instancias "Coder Ini", "Medium", "Coder Resp" no están implicadas. En la representación textual, la palabra clave **shared**, junto con una lista de instancias, indica las instancias a las que está unida la condición.



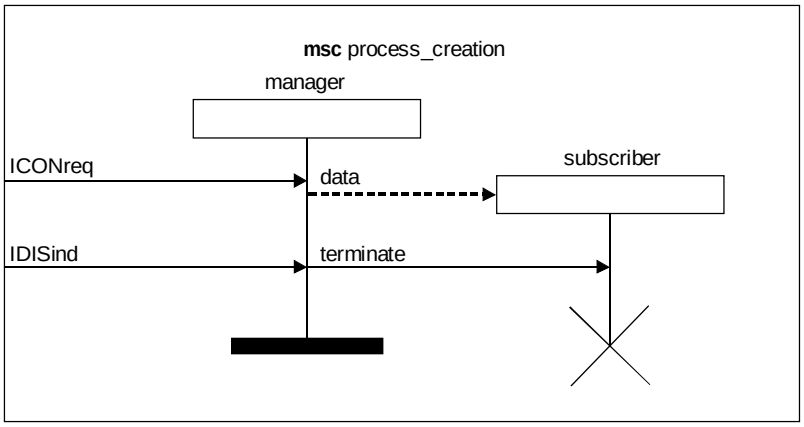
T1007590-93/d39

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 Creación y terminación de procesos

En el ejemplo 6.8 se muestra la creación dinámica de la instancia "subscriber" (abonado) ocasionada por una petición de conexión, y su correspondiente terminación, ocasionada por una petición de desconexión.



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;

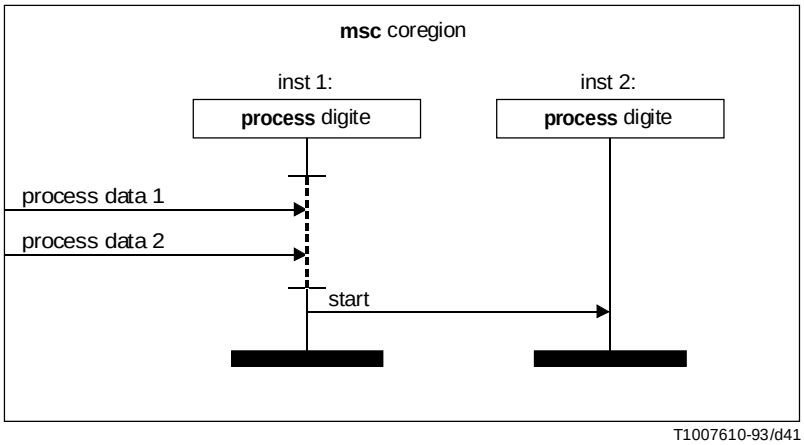
stop;

endinstance;

endmsc;

6.9 Corregión

En el ejemplo 6.9 se muestra una región concurrente, que indicará que el consumo de "process data 1" y el consumo de "process data 2" no están ordenados temporalmente, es decir, "process data 1" se puede consumir antes de "process data 2" o al revés.



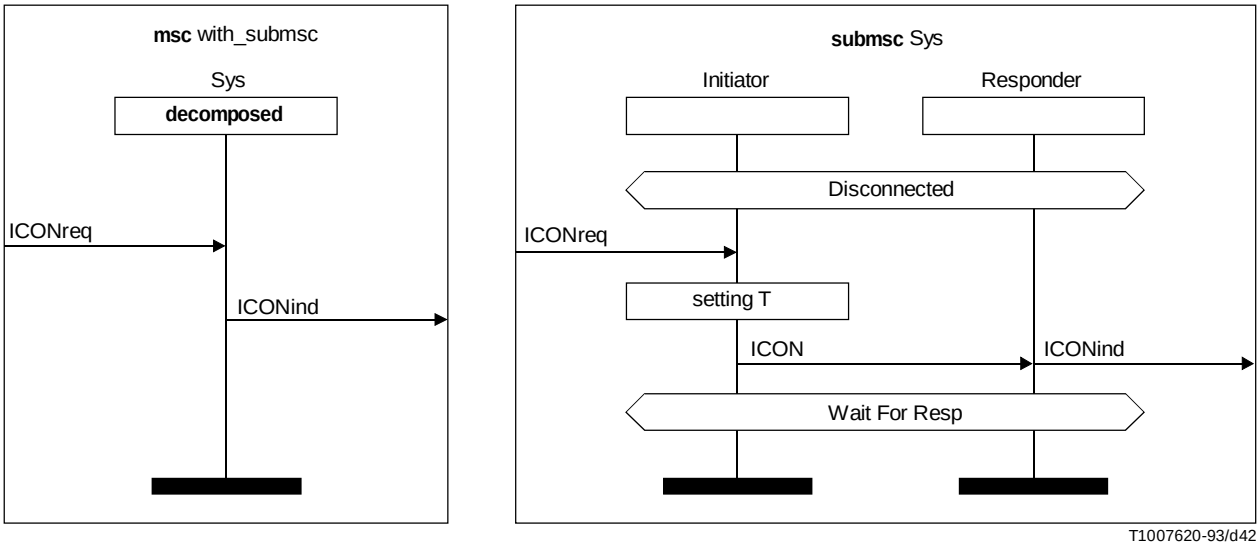
T1007610-93/d41

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 Subgráfico de secuencias de mensajes

El ejemplo 6.10 contiene el subMSC "Sys". Este subMSC está unido a la instancia "Sys", por lo que representa una descomposición de esa instancia.



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;

Anexo A

Índice

(Este anexo es parte integrante de la presente Recomendación)

Las entradas son las <**keyword**> y los no terminales de la *Gramática abstracta*, la *Gramática textual concreta* y la *Gramática gráfica concreta*. Los números de página en **negritas** corresponden a las definiciones de no terminales.

Impreso en Suiza

Ginebra, 1994