UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T Z.109

(11/99)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

SERIE Z: LENGUAJES Y ASPECTOS GENERALES DE SOPORTE LÓGICO PARA SISTEMAS DE TELECOMUNICACIÓN

Técnicas de descripción formal – Lenguaje de especificación y descripción

---

Combinación del lenguaje de especificación y descripción con el lenguaje de modelado unificado

---

Recomendación UIT-T Z.109

(Anteriormente Recomendación del CCITT)

RECOMENDACIONES UIT-T DE LA SERIE Z

LENGUAJES Y ASPECTOS GENERALES DE SOPORTE LÓGICO PARA SISTEMAS DE TELECOMUNICACIÓN

TÉCNICAS DE DESCRIPCIÓN FORMAL	
Lenguaje de especificación y descripción (SDL)	**Z.100–Z.109**
Aplicación de técnicas de descripción formal	Z.110–Z.119
Gráficos de secuencias de mensajes	Z.120–Z.129
LENGUAJES DE PROGRAMACIÓN	
CHILL: el lenguaje de alto nivel del UIT-T	Z.200–Z.209
LENGUAJE HOMBRE-MÁQUINA	
Principios generales	Z.300–Z.309
Sintaxis básica y procedimientos de diálogo	Z.310–Z.319
LHM ampliado para terminales con pantalla de visualización	Z.320–Z.329
Especificación de la interfaz hombre-máquina	Z.330–Z.399
CALIDAD DE SOPORTES LÓGICOS DE TELECOMUNICACIONES	Z.400–Z.499
MÉTODOS PARA VALIDACIÓN Y PRUEBAS	Z.500–Z.599

Para más información, véase la Lista de Recomendaciones del UIT-T.

RECOMENDACIÓN UIT-T Z.109

COMBINACIÓN DEL LENGUAJE DE ESPECIFICACIÓN Y DESCRIPCIÓN CON EL LENGUAJE DE MODELADO UNIFICADO

Resumen

Objetivo

La presente Recomendación define el subconjunto especializado del lenguaje de modelado unificado (UML) que corresponde directamente con el lenguaje de especificación y descripción (SDL) y que puede ser utilizado en combinación con este último lenguaje.

Alcance

La presente Recomendación presenta una definición de la correspondencia del UML con el SDL y que se ha de utilizar para la combinación de estos lenguajes.

Aplicación

El principal campo de aplicación de la presente Recomendación es la especificación de sistemas de telecomunicaciones. El uso combinado de los lenguajes SDL y UML permite especificar de una manera coherente la estructura y comportamiento de los sistemas de telecomunicaciones, junto con datos.

Estado/estabilidad

La presente Recomendación es el manual de referencia completo que describe la correspondencia del UML con el SDL, que se ha de utilizar para la combinación de ambos lenguajes.

Trabajo conexo

Recomendación Z.100: Lenguaje de especificación y descripción.

Orígenes

La Recomendación UIT-T Z.109 ha sido preparada por la Comisión de Estudio 10 (1997-2000) del UIT-T y fue aprobada por el procedimiento de la Resolución N.° 1 de la CMNT el 19 de noviembre de 1999.

PREFACIO

La UIT (Unión Internacional de Telecomunicaciones) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones. El UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Conferencia Mundial de Normalización de las Telecomunicaciones (CMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución N.° 1 de la CMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT no ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB.

ÍNDICE

Recomendación Z.109

COMBINACIÓN DEL LENGUAJE DE ESPECIFICACIÓN Y DESCRIPCIÓN CON EL LENGUAJE DE MODELADO UNIFICADO

(Ginebra, 1999)

1 Introducción

La *combinación del lenguaje de especificación y descripción (SDL, specification and description language*) con el *lenguaje de modelado unificado (UML, unified modelling language*) es definida por la presente Recomendación y por la Recomendación Z.100. Esta Recomendación define un *perfil SDL UML* basado en el UML [1] y en el SDL-2000 [2]. Las partes pertinentes de la gramática gráfica del SDL y otros elementos de notación en SDL se definen en [2].

Esta Recomendación no trata de la combinación del lenguaje UML y los diagramas de secuencias de mensajes (MSC) [3].

1.1 Objetivo

El objetivo es aprovechar la base formal del SDL y la expresividad del UML, en particular el uso de los diagramas de clases UML con asociaciones. Mediante la utilización del subconjunto especializado del UML definido en la presente Recomendación es posible expresar partes de una especificación SDL en UML.

1.2 Principios de la combinación SDL-UML

La presente Recomendación define la *especialización y restricción del UML para combinarlo con el SDL* (un *perfil SDL UML*). Garantiza una correspondencia bien definida entre partes de un modelo UML y un modelo SDL. Para cada uno de los elementos de modelo (ModelElements) del UML incluidos en el *SDL UML*, existe una correspondencia *de uno a uno* con los correspondientes conceptos del SDL. La correspondencia se basa en el metamodelo UML y en la gramática abstracta del SDL. Una herramienta que utiliza el *SDL UML* debe soportar estas especializaciones y restricciones y ser capaz de proporcionar esta correspondencia de uno a uno.

Las especializaciones y restricciones UML se definen sobre la base del metamodelo UML y la gramática abstracta del SDL, es decir, independientes de la notación.

La presente Recomendación no proporciona orientaciones relativas a la notación para *SDL UML*. Para algunos de los elementos del UML, el SDL contiene elementos que tienen una notación gráfica similar al UML. Una herramienta para el uso combinado del UML y el SDL puede emplear una norma de notación gráfica UML *de facto* para el UML tratado en la presente Recomendación, pero la parte de esta herramienta específica del SDL debe proporcionar la gramática gráfica para estos elementos definidos por la Recomendación Z.100.

1.3 Restricciones relativas al SDL y al UML

No hay restricciones relativas al SDL, aunque no todo el SDL es abarcado por *SDL UML*.

Una restricción general relativa al *SDL UML* es que sólo los elementos de modelo definidos en el *perfil SDL UML* aseguran una correspondencia de uno a uno. En el uso combinado de UML y SDL, es posible utilizar más partes de UML, pero no se puede garantizar que la correspondencia de éstas funcione de la misma manera con herramientas diferentes.

1.4 Correspondencia SDL UML

Para definir la correspondencia SDL UML se utilizan mecanismos de extensión del UML: estereotipos (Stereotypes), valores rotulados (Tagged Values) y constricciones (Constraints), restringiendo el uso del UML e incorporando significados más específicos al UML.

Las *clases* (*classes*) UML representan generalmente *tipos* (*types*) de entidades SDL. En la mayoría de los casos, la entidad *género* (*kind*) está representada por *estereotipos* (*stereotypes*). Cuando existen elementos de modelo, estereotipos o palabras clave predefinidos de UML que tienen un significado similar en SDL, éstos han sido utilizados.

1.5 Reglas de buena formación

Las dos reglas generales de buena formación del *SDL UML* son:

- Sólo pueden ser utilizados elementos de modelo que tienen correspondencias con el SDL (y son definidos por la presente Recomendación).
- El modelo *SDL UML* se conformará con la semántica (estática) del SDL.

Una consecuencia de estas reglas es, por ejemplo, que el *elemento poseído* (*ownedElement)* de una clase que representa un tipo SDL debe ser una clase *SDL UML*, que representa un tipo definido en la unidad de alcance del tipo SDL.

Para cada uno de los elementos de modelo de *SDL UML*, se describen reglas de buena formación específicas.

1.6 Convenios

La definición de *SDL UML* sigue la misma organización que la definición de la semántica del UML Para cada elemento de modelo, la correspondencia con el SDL, se describe en un cuadro, seguida por las posibles restricciones y especializaciones. Cada cuadro sigue la jerarquía de generalización del metamodelo, y tiene una entrada para cada elemento de modelo, atributo y asociación considerado o que tiene una correspondencia. Los elementos de la gramática abstracta del SDL se indican en cursivas con guiones, por ejemplo: *Variable-definition* (*Definición de variable)*.

Notación

no es aplicable	no se utiliza en *SDL UML*
ningún concepto	ningún concepto correspondiente en SDL
por defecto	correspondencia por defecto
obligatorio	correspondencia obligatoria
^	las superclases del elemento de modelo para las cuales se aplica una correspondencia especial
.	atributo del elemento de modelo
=	posible valor de atributo
->	asociación del elemento de modelo

1.7 Estructura de la presente Recomendación

Esta Recomendación está estructurada según el metamodelo UML. La finalidad de esta Recomendación es saber: "qué partes de UML pueden utilizarse en combinación con el SDL y cómo se efectúa la correspondencia con el SDL". La correspondencia entre el UML y el SDL pudo también efectuarse con el SDL como punto de partida.

En caso de discrepancias entre la presente Recomendación y la Recomendación Z.100 [2] en relación con la descripción del SDL, se aplica la definición de la Recomendación Z.100 [2].

2 Referencias

Las siguientes Recomendaciones del UIT-T y otras referencias contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y otras referencias son objeto de revisiones por lo que se preconiza que los usuarios de esta Recomendación investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y otras referencias citadas a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T actualmente vigentes.

[1] Unified Modelling Language (UML) (OMG UML Specification, v. 1.3: OMG document ad/99-06-08).

[2] Recomendación UIT-T Z.100 (1999), *Lenguaje de especificación y descripción*.

[3] Recomendación UIT-T Z.120 (1999), *Gráficos de secuencias de mensajes*.

3 Elementos de modelo *SDL UML*

La estructura principal del *perfil SDL UML* sigue el metamodelo del UML. La correspondencia con el SDL es definida por la correspondencia del metamodelo UML con la gramática del SDL.

3.1 Resumen de elementos de modelo *SDL UML*

El modelo *SDL UML* proporciona una especialización y restricciones de los siguientes elementos de modelo:

- Gestión de modelo:
  - Package (Lote)
  - Model (Modelo)
- Clases, con estereotipos:
  - «system» (sistema)
  - «block» (bloque)
  - «process» (proceso)
  - «procedure» (procedimiento)
  - «interface» (interfaz)
  - «object» (objeto)
  - «value» (valor)
  - «state» (estado)
- «signal» Classifier (clasificador de señales)
- Asociaciones:
  - Composición como una representación parcial de contenimiento entre agentes
  - Estereotipo de asociación con puerta que representa una puerta con constricción de punto extremo
  - Otras asociaciones que representan las asociaciones correspondientes en SDL
- Generalización

– Dependencias:
  - «import» (importar)
  - «create» (crear)
– Máquina de estados

3.1.1 Estereotipos (Stereotypes)

Se utilizan los siguientes elementos normalizados UML:

Nombre	Se aplica a	Descripción
model	Package::Model	Especificación SDL (de gestión de modelo)
interface	Classifier::Interface	Interfaz
signal	Classifier	Definición de señales
create	Dependency::Usage	Creación
import	Dependency::Permission	Cláusula de referencia de lote

La presente Recomendación define los siguientes estereotipos:

Nombre	Se aplica a	Descripción
system	Class	Sistema (es también estereotipo de Model en UML, pero aquí se utiliza Class)
block	Class	Bloque
process	Class	Proceso
procedure	Class	Procedimiento
object	Class	Tipo de objeto
value	Class	Tipo de valor
state	Class	Tipo de estado compuesto
gate	Association	Puerta con constricción de punto extremo
signature	Operación	Parámetros formales de agentes

3.1.2 Valores rotulados (TaggedValues)

La presente Recomendación define los siguientes valores rotulados:

Nombre	Se aplica a	Descripción
encloser	Class	Especifica el tipo que engloba (unidad de alcance). El valor es el identificador de tipo
• virtual • redefined • finalized	Class	Especifica la <virtuality> (virtualidad) de un tipo, de un procedimiento o de un operador • **virtual** • **redefined** • **finalized**
remote	• Attribute • Procedure	• Variable distante • Procedimiento distante Los valores son booleanos.

3.1.3 Constricciones (Constraints)

Nombre	Se aplica a	Descripción
atleast	Class	Representa una constricción de virtualidad en un tipo virtual
atleast	Parameter	Representa una constricción de virtualidad en un parámetro de contexto

3.2 Elementos de modelo de núcleo (Core Model Elements)

En esta subcláusula se indican las restricciones, especializaciones y correspondencias que se aplican a los elementos generales del metamodelo UML núcleo. Se aplican si no se indica otra cosa para elementos de modelo más específicos.

3.2.1 Abstracción (Abstraction)

No es aplicable.

3.2.2 Asociación (Association)

En esta versión de la Recomendación, se utilizan asociaciones para representar los siguientes conceptos SDL:

- Agregación compuesta para contenimiento, es decir, conjuntos de entidades contenidas en entidades.
- Posibles conexiones de conjuntos de agentes (por canales que conectan puertas con interfaces) como parte de la estructura interna de un agente.
- Puerta con constricción de punto extremo por una asociación estereotipada con puerta.

Hay otros subconjuntos de asociaciones que son válidos (y se describen a continuación) pero que son considerados como comentarios y corresponden con las asociaciones pertinentes en SDL.

Las asociaciones *no* engloban conjuntos de tipos que son conectados por canales:

1) Las asociaciones se utilizan para definir propiedades de tipos, mientras que los canales se utilizan (en los diagramas de estructura SDL) para definir enlaces entre conjuntos de casos. Diferentes canales pueden conectar diferentes conjuntos de tipos basados en el mismo tipo. Por consiguiente, la conexión de conjuntos de agentes por medio de canales es representada por colaboraciones en el UML.
2) Como base para el uso de colaboraciones, todos los tipos de agente que tienen interfaces y puertas que pueden ser conectadas por canales tendrán asociaciones implícitas en la correspondencia SDL UML. Estas asociaciones no entrañan propiedades de las clases asociadas.

3.2.2.1 Agregación compuesta (Composite aggregation)

La agregación compuesta se utiliza para representar los ejemplares de un tipo de agente que contiene conjuntos de agentes basados en tipo. Por ejemplo, los ejemplares de un tipo de sistema contienen conjuntos de bloques. Los conjuntos son representados por los puntos extremos de la agregación compuesta, y los nombres de cometido son los nombres de los conjuntos y la multiplicidad es el número de ejemplares.

La agregación de composición no tiene exactamente el mismo sentido en UML que el contenimiento en SDL. La presente Recomendación impone el significado más estricto de contenimiento del SDL a la composición de UML. Un elemento de un conjunto de agentes contenidos *no puede* cambiar los miembros del conjunto, como es el caso en el UML. Un conjunto de agentes contenidos y sus miembros no pueden existir sin el contenedor, de la misma manera que en el UML.

Obsérvese que la agregación compuesta sólo supone propiedades para el tipo en el extremo compuesto. El tipo en el otro extremo no es afectado.

3.2.2.2 Constricción de punto extremo de puerta (Gate endpoint constraint)

Una asociación estereotipada con puerta representa que el tipo de agente correspondiente de la clase de extremo fuente tiene una puerta con una constricción de punto extremo que corresponde a la clase de extremo objetivo. El nombre de la asociación es el nombre de la puerta. Sólo se representan las interfaces, señales y procedimientos distantes asociados con el sentido saliente de la puerta; las listas de señales no están representadas.

3.2.2.3 Asociaciones generales (General associations)

Aunque las asociaciones generales en *SDL UML* son comentarios en el SDL correspondiente, esta versión de la Recomendación sólo soporta un subconjunto de asociaciones UML.

UML	SDL
Association	• contenimiento: conjunto de agentes como parte de otras entidades • puerta con constricción de punto extremo • asociaciones generales: asociaciones como comentarios, pero restringidas por el *no* soporte de los siguientes elementos de asociación: – n-arias asociaciones (n>2) – Extremos de asociaciones calificadas – Clase de asociación – Agregación – Posibilidad de cambio – Alcance objetivo
->connection	
^GeneralizableElement	no es aplicable
^ModelElement	
.name	• contenimiento: no es aplicable • puerta con constricción de punto extremo: *Gate-name* • en los demás casos: <association name>

3.2.3 Clase de asociación (Association class)

No es aplicable.

3.2.4 Extremo de asociación (Association end)

En las descripciones siguientes, cuando se menciona un extremo de asociación para una asociación binaria, el extremo *source* (*fuente*) es el otro extremo; el extremo *target* (*objetivo*) es aquel cuyas propiedades son examinadas.

Un AssociationEnd puede representar una especificación parcial de dos conceptos del SDL:

– Contenimiento:
  - Agentes que contienen otros agentes: La asociación debe ser una composición y las clases participantes deben representar tipos de agente.
  - Estados compuestos que contienen estados basados en tipos: La asociación debe ser una composición y las clases participantes deben representar tipos de estados.

– La constricción de punto extremo de una puerta: La asociación debe ser estereotipada con «gate» («puerta»).

Un AssociationEnd puede ser también el extremo de una asociación ordinaria. En ese caso representará el correspondiente extremo de asociación de la correspondiente asociación SDL. Esto significa que una composición que no es abarcada por los dos casos anteriores sólo representa la correspondiente asociación de composición en SDL.

UML	SDL
AssociationEnd	• Contenimiento • puerta con constricción de punto extremo en las entidades en el extremo fuente • en los demás casos: <association end>
.aggregation = composite	• contenimiento, es decir, el extremo representa un tipo de agente (sistema/bloque/proceso) que contiene conjuntos de bloques y/o procesos o el extremo representa un estado compuesto con estados basados en tipos contenidos • en los demás casos: <association kind> = – <composition not bound kind>, o – <composition part end bound kind>, o – <composition composite end bound kind>, o – <composition two ends bound kind>
= aggregate	• <association kind> = – <aggregation not bound kind>, o – < aggregation part end bound kind>, o – < aggregation aggregate end bound kind>, o – < aggregation two ends bound kind>
= none	• si el extremo fuente de la asociación tiene agregación = compuesta: el extremo objetivo representa un conjunto de agentes o un estado compuesto basado en tipos como parte de un tipo de estado compuesto • en caso de asociación estereotipada con puerta: puerta con constricción de punto extremo
.changeability = none = frozen = addOnly	No es aplicable
.ordering = unordered = ordered	no es aplicable para contenimiento y para puerta con constricción de punto extremo, en los demás casos: • *por defecto* • **ordering**
.isNavigable = true = false	no es aplicable para contenimiento y para puerta con constricción de punto extremo, en los demás casos: • navegable • no es aplicable
.multiplicity	• para contenimiento: representa *Number-of-instances* • para constricción de punto extremo de puerta: 1 • en los demás casos: <multiplicity> de <association end>

UML	SDL
.targetScope = instance = classifier	 • entidad • no es aplicable
.visibility = public = protected: default = private	• para contenimiento: no es aplicable • para puerta con constricción de punto extremo: no es aplicable • en los demás casos: – **exported** – no es aplicable – local
->qualifier	no es aplicable
->specification	• para contenimiento: no es aplicable • para puerta con constricción de punto extremo: interfaces, señales y procedimientos distantes asociados con el sentido saliente de la puerta • en los demás casos: <specifier>
->type	• para contenimiento: tipo de conjunto de agentes basados en el tipo • para constricción de punto extremo de puerta: el tipo de constricción de punto extremo • en los demás casos: <linked type>
^ModelElement	
.name	• para contenimiento: *Agent-name* en *Agent-definition* • para constricción de punto extremo de puerta: no es aplicable • en los demás casos: <role name>

El *type* (*tipo*) sólo puede indicar elementos de modelo que son clasificadores *SDL UML*, salvo interfaces.

3.2.5 Atributo (Attribute)

Los *attributes* (*atributos*) representan variables de agentes y procedimientos, o campos de objetos de datos o valores, o parámetros de señales.

UML	SDL
Attribute	variable, campo o parámetro de señal
.changeability = changeable = frozen = addOnly	 • obligatorio • no es aplicable • no es aplicable
.initialValue	<default initialization> [:= <ground expression>] en <variables of sort> en *Variable-definition*
.multiplicity	NOTA – Se puede utilizar para tipos ASN.1.
.targetScope	
->type	*Sort-reference-identifier* en *Variable-definition,* como parte de *Signal-definition,* o como parte de definición de campo
->associationEnd	no es aplicable

UML	SDL
^Feature	
.ownerScope = instance = classifier	 • obligatorio • no es aplicable
.visibility = public = protected = private	 • **exported** variable o public field • variables locales de agentes y campo protegido • no es aplicable a variables de agentes, sino a campos privados
-> owner	la entidad que define la variable o campo
^ModelElement	
.name	*Variable-name*/<field name>

El *type* de un atributo está restringido a ser una clase definida por la presente Recomendación.

Se aplican las siguientes reglas:

- *type* = clase con estereotipo «block» o «process»: Pid tipificado
- *type* = clase con estereotipo «object»: variable de referencia de objeto
- *type* = clase con estereotipo «value»: variable de tipo de valor

Los tipos predefinidos son representados por clases predefinidas correspondientes de estereotipo «process», «object» o «value».

3.2.6 Característica comportamental (BehaviouralFeature)

UML	SDL
BehaviouralFeature	Operator or procedure
.isquery *= true* *= false*	 • *Static-operator*/*Dynamic-operator* (operador) • *Static-operator*/*Dynamic-operator* (método) o *Procedure-definition*
->Parameter	• *Argument-list* en *Signature* • *Procedure-formal-parameter**

3.2.7 Vinculación (Binding)

Una *binding* (*vinculación*) es una subclase de Dependency (dependencia) que representa la provisión de parámetros de contexto reales en SDL. La vinculación como una relación explícita no existe en el SDL, pero puede ser descrita de esta manera en el UML.

Para *binding*, los argumentos representan elementos de modelo dentro del alcance (namespace) del cliente que son parámetros reales para los (templateParameters) de la fuente. Los parámetros reales deben corresponder con parámetros de contexto válidos para el tipo SDL que es representado por la fuente.

Las plantillas parcialmente creadas corresponden a tipos parametrizados donde no se proporcionan todos los parámetros de contexto.

UML	SDL
Binding	<type expression> con <actual context parameters>
->argument	<actual context parameter>
^Dependency	
->client	el tipo definido por <type expression>
->supplier	el <base type> (en <type expression>)
^ModelElement	
.name	no es aplicable

El *client* (*cliente*) y el *supplier* (*suministrador*) deben ser clases con el mismo estereotipo.

3.2.8 Clase (Class)

Las clases representan tipos y procedimientos SDL. Los diferentes géneros de tipos de entidades en SDL son representados por estereotipos.

Un tipo en SDL es definido completamente por una definición/diagrama de tipo y posiblemente por una referencia de tipo (cuando se hace referencia a la definición/diagrama de tipo). Una clase define partes del tipo *y* posiblemente una referencia de tipo. Las partes que define dependen del contenido de los compartimientos de clase y su composición. La elección de propiedades que se han de definir en *SDL UML* se deja al especificador, pero las propiedades especificadas serán coherentes con las propiedades correspondientes definidas en la definición/diagrama de tipos SDL.

Los atributos y asociaciones de una clase están restringidos/especializados y corresponden con el SDL según se describe en el siguiente cuadro. Para cada uno de los diferentes géneros de entidades SDL, en los cuadros 3.2.8.1 a 3.2.8.4 figuran más detalles. Se incluyen también los atributos definidos en Foundation Package:Auxiliary Elements (lote de fundación: elementos auxiliares).

UML	SDL
Class	tipos y procedimientos
.isActive = true = false	 • *Agent-type-definition* • *Composite-state-type-definition, Data-type-definition, Procedure-definition, Signal-definition*
^Classifier	
->feature	• depende del género de entidad
->participant – aggregation of association end: = composite = aggregate = none	 • conjuntos de agentes contenidos • no es aplicable • si el otro extremo de la asociación tiene agregación = compuesta: un conjunto de entidades
^GeneralizableElement	<specialization>
.isAbstract	**abstract**
.isLeaf	no es aplicable
.isRoot	se puede derivar de la definición de tipo
->generalization	Supertipo – el tipo identificado como parte de <type expression> de <specialization> para el tipo representado por el clasificador

UML	SDL
->specialization	se puede derivar de *generalization*
^Namespace	Unidad de alcance definida por el tipo
->ownedElement	entidades en un subconjunto del conjunto de definiciones permitidas en la unidad de alcance – depende del género de tipo
^ModelElement	
->taggedValue • (virtuality) • virtual • redefined • finalized	 • <virtuality> • **virtual** • **redefined** • **finalized**
->constraint	<virtuality constraint>
->supplierDependency • Dependencia de realización de las interfaces implementadas por la clase • Dependencia de utilización («create») de clases para las cuales los objetos son creados por objetos de esta clase • Dependencia de permiso («import») de otros lotes que utilizan la clase • Relación de generalización anotada	Dependencias donde este tipo es un suministrador • <interface gate definition> **con** <interface identifier> • <create line area> • <package reference clause>/<definition selection list> • <type expression>
->clientDependency • Dependencia de utilización («create») de clases de las cuales objetos crean objetos de esta clase • Dependencia de permiso «import» que indica el lote que esta clase utiliza • Relación de generalización anotada	Dependencias donde este tipo es un cliente • <create line area> • <package reference clause> en este tipo de definición • <type expression>
-> namespace	la unidad de alcance (que abarca) poseedora del tipo de entidad correspondiente a la clase

3.2.8.1 Agente (Agent)

Lo siguiente se aplica a tipos de agentes.

Se introduce la característica de operación del signature para especificar los parámetros formales, dado que el UML no proporciona parámetros para clases. Tiene el formato de una operación con el nombre de proceso como el nombre de operación y con el estereotipo «signature».

UML	SDL
{ «system» | «block» | «process» } Class	*Agent-type-definition* • si se define un *templateParameter*, es un tipo de agente parametrizado, • si se define una *generalization,* es un subtipo
.isActive = true = false	 • obligatorio • no es aplicable
^Classifier	
->feature • attributes • operations	 • *Variable-definition* • *Procedure-definition, o* • *Procedure-definition* (signature) que representa <formal parameters> del tipo de agente
->participant	Contenimiento u otro extremo de asociación donde puede participar el agente
^GeneralizableElement	
->generalization	*Agent-type-definition* identificada por *Parent-type-identifier* como parte de este tipo de agente
^Namespace	unidad de alcance definida por el tipo de agente
->ownedElement	entidades en el subconjunto de <entity in agent> definido más adelante
^ModelElement	
->taggedValue • virtual • redefined • finalized	<virtuality> del tipo de agente, excepto para tipo de sistema
->constraint	<virtuality constraint> impuesta al tipo de agente, excepto para tipo de sistema
-> namespace	• *Package-definition* con *Agent-type-definition* • *Agent-type-definition* que define el tipo de agente

La generalization (*generalización*) sólo puede ser una clase, y debe ser del mismo estereotipo que la clase considerada.

El *ownedElement* (*elemento poseído*) está restringido a elementos de modelo que representan entidades en el siguiente subconjunto de <entity in agent>:

- *Signal-definition*,
- *Variable-definition*,
- *Procedure-definition*,
- <remote procedure definition>,
- <remote variable definition>,
- *Data-type-definition*,
- *Composite-state-type-definition*,
- *Interface-definition*,
- *Agent-type-definition*, *o*
- *Agent-definition*.

El *namespace* (*espacio de nombre*) debe ser un lote o una clase estereotipada con «system», «block» o «process». El *namespace* de una clase «system» o «block» no puede ser una clase «process» y el namespace de una clase «system» no puede ser una clase «block» o «process».

3.2.8.2 Procedimiento (Procedure)

Procedure es un estereotipo de clase que corresponde con un procedimiento SDL. Esto se hace para englobar la especialización de procedimientos, que no forman parte del UML.

UML	SDL
«procedure» Class	*Procedure-definition* o <remote procedure definition> • si se define un *templateParameter,* es un procedimiento parametrizado • si se define una *generalization,* es un subprocedimiento
.isActive = false = true	 • obligatorio • no es aplicable
->taggedValue remote	 definición <remote procedure definition>
^Classifier	
->feature • attributes • operations	 • *Variable-definition* • *Procedure-definition*
->participant	no es aplicable
^GeneralizableElement	
->generalization	*Procedure-definition* identificada por el *Procedure-identifier* facultativo como parte de la *Procedure-definition* vigente
^Namespace	unidad de alcance definida por el tipo de servicio
-> ownedElement	entidades en el subconjunto de <entity in procedure> definido más adelante
^ModelElement	
->taggedValue • virtual • redefined • finalized	<virtuality> de procedimiento • **virtual** • **redefined** • **finalized**
-> namespace	• *Package-definition* con *Procedure-definition* • *Agent-type-definition* con *Procedure-definition* • *Procedure-definition* con *Procedure-definition*

Los *ownedElements* son elementos de modelo que representan entidades en el siguiente conjunto de <entity in procedure>:

- *Data-type-definition,*
- *Variable-definition, o*
- *Procedure-definition.*

El *namespace* debe ser un lote o una clase con estereotipo «system», «block», «process», o «procedure».

3.2.8.3 Tipos de datos (Data types)

Las clases estereotipadas con *value* y *object* representan tipos de datos: *value* es una clase que corresponde con un tipo de valor SDL y *object* es una clase que corresponde con un tipo de objeto SDL.

UML	SDL
• «value» Class • «object» Class	• *Value-type* • *Object-type* Para ambos: • si se define un *templateParameter,* es un tipo parametrizado • si se define una *generalization,* es un subtipo
.isActive = false = true	 • obligatorio • no es aplicable
^Classifier	
->feature • attributes • operations	 • field> en la <structure definition> • *Static-operator/Dynamic-operator* (operadores y métodos)
->participant	contenimiento del tipo de datos o cualquier punto extremo de asociación en que puede participar el tipo
^GeneralizableElement	
->generalization	*Data-type-definition* identificada por *Data-type-identifier* como parte de *Data-type-definition* vigente
^Namespace	unidad de alcance definida por *Data-type-definition*
-> ownedElement	*Data-type-definitions* en la unidad de alcance del tipo de entidad representada por esta clase
^ModelElement	
-> namespace	• *Package-defnition, Agent-type-defnition, Procedure-definition, Data-type-definition* con *Data-type-definition*

La *generalization* debe ser clases «value» u «object».

El *ownedElement* debe ser clases que corresponden con *Data-type-definition*s.

El *namespace* debe ser un lote, o una clase con estereotipo «system», «block», «process», «service», «procedure», «state», «object», o «value».

3.2.8.4 Estado (State)

State es un estereotipo de clase que representa un tipo de estado compuesto SDL. El SDL tiene la noción de tipo de estado compuesto (además del estado compuesto), y como las máquinas de estados del UML no lo tienen, en *SDL UML* es representado por una clase con un estereotipo state.

Los puntos de conexión (entrada y salida) del estado son representados como operaciones, porque son las propiedades visibles de un tipo de estado compuesto.

UML	SDL
«state» Class	*Composite-state-type-definition* • si se define un *templateParameter,* es un tipo de estado compuesto parametrizado • si se define una *generalization,* es un subtipo
.isActive = false = true	 • obligatorio • no es aplicable
^Classifier	
->feature • attributes • operations	 • *Variable-definition*s • *Procedure-definition*s • puntos de conexión de estado definidos en *State-entry-point-definition*
->participant	no es aplicable
^GeneralizableElement	
->generalization	*Composite-state-type-definition* identificada por [*Parent-type-identifier*] como parte de esta *Composite-state-type-definition*
^Namespace	unidad de alcance definida por esta *Composite-state-type-definition*
->ownedElement	entidades del subconjunto de <entity in composite state type> definido más adelante
^ModelElement	
.name	El *State-type-name*
->taggedValue • virtual • redefined • finalized	<virtuality> del tipo de estado compuesto
->constraint	<virtuality constraint>
-> namespace	*Package-definition, Agent-type-definition, Procedure-definition*, *Composite-state-type-definition* con la *Composite-state-type-definition*

La *generalization* debe ser una clase "state".

El *ownedElement* deber ser elementos de modelo que representan entidades en el subconjunto de <entity in composite state>, es decir:

- *Variable-definition,*
- *Data-type-definition,*
- *Procedure-definition,*
- <textual procedure definition>, o
- *Composite-state-type-definition*.

El *namespace* debe ser un lote o una clase con estereotipo «system», «block», «process», «procedure» o «state».

3.2.9 Clasificador (Classifier)

Un clasificador representa las características comunes del SDL: tipos de entidades, tipos de interfaz, señales y procedimientos. Si no es especificado concretamente para los distintos géneros (kind), se aplica lo siguiente.

UML	SDL
Classifier	tipos de entidades, interfaces, señales y procedimientos
->feature	variables, procedimientos, métodos, operadores, parámetros de señales – según el género de entidad
->participant	depende del género de clasificador
->powertypeRange	no es aplicable
^GeneralizableElement	depende del género
^Namespace	unidad de alcance
-> ownedElement	entidades definidas en la unidad de alcance de esta entidad
^ModelElement	
.template	el tipo parametrizado si este tipo es un <formal context parameter>
.templateParameter	<formal context parameters> para un tipo parametrizado
->constraint	<virtuality constraint>, si está presente
-> supplierDependency	depende del género
-> clientDependency	depende del género
->taggedValue • virtual • redefined • finalized	<virtuality>, si está presente • **virtual** • **redefined** • **finalized**

3.2.10 Comentario (Comment)

Un comentario es una notación anexa a un elemento de modelo que representa una nota o un comentario anexado a la entidad SDL que corresponde con el elemento de modelo.

3.2.11 Componente (Component)

No es aplicable.

3.2.12 Constricción (Constraint)

No se soportan constricciones generales, porque no hay conceptos correspondientes en el SDL. Se soportan constricciones de virtualidad. El texto UML es <virtuality constraint> como texto.

UML	SDL
Constraint	
.body = atleast – Class name – Parameter name	 • <virtuality constraint> en un tipo virtual o un procedimiento/método • <virtuality constraint> en un parámetro <formal context parameter>
->constrainedElement	• un tipo virtual si la constricción es una constricción de virtualidad en un tipo virtual • un parámetro <formal context parameter> si la constricción es una constricción de virtualidad en un parámetro de contexto
^ModelElement	
.name	no es aplicable

3.2.13 Tipo de datos (Data type)

No es aplicable.

3.2.14 Dependencia (Dependency)

Las dependencias se utilizan para representar:

- la dependencia «create» para crear líneas;
- la dependencia «import» para representar que un lote utiliza las definiciones de otros lotes, y
- la dependencia «realizes» para interfaces soportadas por la clase.

3.2.15 Elemento (Element)

No es aplicable en sí mismo, pero se utilizan atributos; véase la correspondencia de los elementos más específicos del metamodelo.

3.2.16 Propiedad de elemento (ElementOwnership)

No es aplicable. Todos los elementos contenidos son partes de la especificación de la unidad de alcance contenedora, y la visibilidad de elementos contenidos es proporcionada por las propiedades de éstos.

3.2.17 Residencia de elementos (ElementResidence)

No es aplicable.

3.2.18 Característica (Feature)

Clase abstracta. Los atributos y asociaciones son detallados en relación con behaviouralFeature (característica comportamental) y structuralFeature (característica estructural).

3.2.19 Flujo (Flow)

No es aplicable.

3.2.20 Elemento generalizable (GeneralizableElement)

Clase abstracta. Los atributos y asociaciones se detallan en relación con sus subclases.

3.2.21 Generalización (Generalization)

La especialización del SDL es representada por la generalización en el UML. Esto se hace también para los procedimientos. El metamodelo UML tiene la noción de generalización, mientras que el SDL tiene la noción opuesta de especialización.

Generalization tiene la contrapartida <specialization> opuesta en el SDL, pero refleja la propiedad de herencia correspondiente del subtipo.

UML	SDL
Generalization	<specialization>
.discriminator	no es aplicable
->child	un subtipo
->parent	*Parent-type-identifier* (<base type> (en <type expression>)) de subtipo,
->powertype	no es aplicable
^ModelElement	
.name	no es aplicable

La *Generalization* se aplica a clases estereotipadas para los siguientes géneros de tipos: sistemas, bloques, procesos, estados, objetos, valores, para procedimientos y para interfaces y señales (clasificadores con estereotipo «interface» o «signal»).

El *parent* (*progenitor*) debe ser una clase del mismo estereotipo que el subtipo, salvo para interfaces que pueden tener múltiples progenitores.

No se aplican Standard Constraints (Constricciones normalizadas).

3.2.22 Interfaz (Interface)

Las interfaces son clasificadores que corresponden con interfaces en el SDL. Las interfaces del UML sólo pueden tener operaciones (correspondientes a procedimientos exportados en el SDL), mientras que las interfaces SDL pueden tener también variables y señales. En *SDL UML*, las interfaces tienen por tanto una lista de operaciones que representan señales y variables exportadas. El estereotipo «signal» se utiliza para indicar señales, mientras que las variables y procedimientos son representados como operaciones.

Las interfaces sólo pueden ser aplicadas a clases que representan tipos de agentes, no a clases que representan procedimientos, tipo de estados o tipo de datos.

UML	SDL
Interface	*Interface-definition*
^Classifier	
->feature • *attribute (not supported)* • *operation*	 • no es aplicable • *Signal-definition*, <remote variable definition> y <remote procedure definition>
->participant	extremos de asociación que son interfaces, en las cuales agregación de caso = ninguna
^GeneralizableElement	
.isAbstract	**Abstract**
.isLeaf	no es aplicable
.isRoot	se puede derivar de *Interface-definition*
->generalization	superinterfaces, que son las interfaces identificadas por <interface specialization> para esta *Interface-definition*
->specialization	puede ser derivada de *generalization*
^Namespace	unidad de alcance de esta *Interface-definition*
-> ownedElement	<entity in interface>: las entidades definidas en la unidad de alcance de esta *Interface-definition*
^ModelElement	
->supplierDependency	dependencia de realización que representa **implements**
->clientDependency	lo opuesto de *supplierDependency*

El *ownedElement* debe ser elementos de modelo que representan entidades en <entity in interface>, es decir:

- *Signal-definition*;
- <remote variable definition>; o
- <remote procedure definition>.

3.2.23 Método (Method)

UML	SDL
Method	cuerpo de *Procedure-definition*, *Static-operator*/*Dynamic-operator* (operador o método)
.body	no es aplicable
->specificication	*Procedure-formal-parameter*, Signature*
^ModelElement	
->constraint	<virtuality constraint>

La *constraint* describe que el cuerpo de un procedimiento u operador virtual hereda la especificación de comportamiento de la constricción. Se trata de un uso especializado de la asociación de constricción del método UML.

3.2.24 Elemento de modelo (ModelElement)

Un *ModelElement* representa propiedades comunes de definiciones (de tipo) de entidad SDL. Si no se especifica concretamente para las diversas especializaciones, se aplica lo siguiente.

UML	SDL
ModelElement	definición de entidad
.name	<name> (nombre) de entidad
-> clientDependency	depende del género de entidad
-> constraint	depende del género de entidad
->implementationLocation	no es aplicable
-> namespace	la unidad de alcance (que abarca) poseedora donde se define la entidad
-> presentation	no es aplicable
-> supplierDependency	depende del género de entidad
->templateParameter	depende del género de entidad

3.2.25 Espacio de nombre (Namespace)

Un namespace representa una unidad de alcance SDL.

UML	SDL
Namespace	unidad de alcance
-> ownedElements	tipos de entidad definidos en la unidad de alcance representada por este elemento de modelo

El *ownedElement* debe ser los elementos de modelo que representan entidades en SDL de conformidad con *SDL UML*.

3.2.26 Nodo (Node)

No es aplicable, porque la especificación de realización está fuera del alcance del SDL, y por tanto del *SDL UML*.

3.2.27 Operación (Operation)

Una operación representa un procedimiento o un operador/método. La operación se utiliza simplemente para representar el hecho de que un tipo define un procedimiento o un operador/método, junto con su signatura. Una clase anidada de estereotipo de procedimiento representa el propio procedimiento con el mismo nombre que la operación. Los operadores y métodos no están representados por clases con estereotipo, sino por métodos UML. Las operaciones estereotipadas con «signature» representan los parámetros formales del tipo de agente contenedor.

UML	SDL
Operation	*Procedure-definition*, *Static-operator*/*Dynamic-operator* (operador o método) o parámetros formales del tipo de agente
.concurrency = sequential = guarded = concurrent	aplicable solamente para operador y método • depende de la semántica de datos • no es aplicable • depende de la semántica de datos
.isAbstract = true = false	 • **abstract** • no **abstract**
.isLeaf = true = false	 • **finalized** • **virtual** o **redefined**
.isRoot = true = false	no es aplicable
^Feature	
.ownerScope = instance = classifier	 • obligatorio • no es aplicable
.visibility = public = protected: default = private	 • **exported procedure**, **public** (<visibility>) para operador/método • local **procedures**, **protected** (<visibility>) para operador/método • no es aplicable para procedimientos, **private** (<visibility>) para operador/método

Obsérvese que los parámetros formales de agentes son dados por una operación estereotipada con «signature» y con el mismo nombre que el agente definido por la clase real. Otras operaciones de agentes y procedimientos representan procedimientos definidos localmente.

La virtualidad de un proceso se especifica también como parte de la clase que representa el procedimiento. La virtualidad de operadores y métodos se especifica como parte de los métodos UML.

Obsérvese que el tipo de un valor que devuelve el procedimiento o el tipo de resultado de un operador no puede ser representado directamente por una propiedad de la operación *SDL UML* correspondiente, sino por un parámetro de devolución de género.

3.2.28 Parámetro (Parameter)

Un parámetro representa un parámetro formal para un procedimiento, operador o método. Los parámetros de tipo de agente son representados por parámetros (con *kind* = in) de un procedimiento con el mismo nombre que el agente y estereotipados con «signature».

UML	SDL
Parameter	*Procedure-formal-parameter*
.default value	ningún concepto
.kind = in = inout = out = return	 • *In-parameter* (palabra clave **in** en <parameter kind>) • *Inout-parameter* (palabra clave **in/out** en <parameter kind>) • no es aplicable • <sort> en *Result* u <operation result>
.name	*Variable-name*
-> type	*Data-type-reference-identifier* o <sort> (en *Result* como parte de *Procedure-definition*

3.2.29 Permiso (Permission)

El permiso "import" se utiliza para representar cláusulas de referencia de lotes SDL, que indican los lotes que utiliza esta clase.

UML	SDL
Permission «import»	<package reference clause>
^Dependency	
->client	el cliente es una *Package-definition* o definición de tipo que utiliza tipos definidos en el lote representado por el suministrador
->supplier	el suministrador es un lote utilizado por el cliente
^ModelElement	
.name	no es aplicable

3.2.30 Elemento de presentación (PresentationElement)

No es aplicable.

3.2.31 Relación (Relationship)

Éste es un concepto abstracto y no es aplicable directamente.

3.2.32 Característica estructural (StructuralFeature)

Éste es un concepto abstracto, utilizado solamente para Attribute. Para más detalles véase este elemento.

3.2.33 Parámetro de plantilla (TemplateParameter)

Representa un parámetro de contexto en SDL.

3.2.34 Utilización (Usage)

La utilización del género *create* representa la creación de línea.

3.3 Mecanismos de extensión (Extension mechanisms)

Los mecanismos de extensión del UML pueden ser utilizados también en SDL UML, pero como para el UML, corresponde al usuario de estos mecanismos definir su significado.

3.4 Tipos de datos (Data types)

No se utilizan. Se supone que los tipos predefinidos de SDL son predefinidos como clases con estereotipos y propiedades apropiados.

3.5 Elementos comportamentales (Behavioural elements)

Aunque sería posible, esta versión de la Recomendación no proporciona una correspondencia detallada de los elementos comportamentales del UML con el SDL, salvo para las señales.

3.5.1 Comportamiento común (Common Behaviour)

No se prevé utilizar el *SDL UML* para especificar acciones en detalle, porque el SDL proporciona mecanismos completos de especificación de acciones. Por tanto, no se especifica correspondencia para el lote Common Behaviour. No obstante, en el apéndice I figura una visión general de una posible correspondencia.

3.5.2 Señal (Signal)

Signal se define en el UML como una subclase de Classifier (clasificador). Corresponde con una señal SDL con las siguientes restricciones.

UML	SDL
«signal»	*Signal-definition* • si se define un *templateParameter,* es una señal parametrizada • si se define una *generalization,* es un subtipo
->context	se puede derivar de las transiciones que envían señales de este tipo
->reception	se puede derivar: los tipos de agente que tienen esta señal en su conjunto de señales de entrada válidas
^Classifier	
->feature	la lista de *Data-type-reference-identifiers* como parte de *Signal-definition*, es decir, parámetros de señal
->participant	no es aplicable
^GeneralizableElement	
->generalization	*Signal-definition* identificada como parte de <type expression> de <specialization> para esta *Signal-definition*
^Namespace	unidad de alcance definida por *Signal-definition*
->ownedElement	no es aplicable
^ModelElement	
->taggedValue	ningún concepto para señales virtuales
->constraint	no es aplicable
->supplierDependency	no es aplicable
->clientDependency	no es aplicable
-> namespace	*Package-definition* o <agent type definition> que define la *Signal-definition*

La *generalization* debe ser un clasificador de «signal».

El *namespace* debe ser un lote o una clase de agentes («system», «block» o «process»).

3.6 Colaboraciones (Collaborations)

Se utiliza un subconjunto de Collaborations para representar la estructura interna de la máquina de agentes/estados y sus conexiones. Los conjuntos de agentes contenidos son representados por los ownedElements de una Collaboration:

- AssociationEnds para los conjuntos de agentes, con *collaborationMultiplicity* que representa el <number of instances> y los ClassifierRoles que representan los tipos de los conjuntos de agentes.
- AssociationRoles para los canales que conectan los conjuntos de agentes.

No se utiliza la noción de interacción. Las interacciones, como partes de Collaborations, sólo tienen un significado como modelos de propiedad de comportamiento, y corresponden con los modelos descritos por MSC [3].

El UML es poco claro con respecto a las repercusiones de Collaborations cuando el Classifier está representado por Collaboration (*representedClassifier*). La presente Recomendación determina que una Collaboration *define* el contenido de ejemplares del Classifier representado.

3.6.1 Cometido de extremo de asociación (AssociationEndRole)

Un AssociationEndRole puede representar uno de los siguientes elementos de {<interaction area> | <agent body area> } ***set***:

- un conjunto de agentes contenidos,
- la máquina de estados del agente contenedor,
- una puerta del agente contenedor,

UML	SDL
AssociationEndRole	El clasificador de interfaz asociado con una puerta de un conjunto de agentes al cual está conectado el canal
.collaborationMultiplicity	• Si es un conjunto de agentes contenidos: *Number-of-instances* • Si es la máquina de estados del agente: no es aplicable • Si es una puerta del agente contenedor: no es aplicable
->availableQualifier	El conjunto de señales y procedimientos distantes en el canal
->base	El calificador de interfaz representado por la interfaz de la puerta a la cual el canal está conectado
^AssociationEnd	
^GeneralizableElement	no es aplicable
^ModelElement	
.name	*Gate-name*

3.6.2 Cometido de asociación (AssociationRole)

Los AssociationRoles representan canales que conectan conjuntos de agentes, la máquina de estados del agente contenedor y las puertas del agente contenedor.

UML	SDL
AssociationRole	Una *Channel-definition*
.multiplicity	El número de conexiones implicadas por la multiplicidad de los AssociationEndRoles en ambos extremos
->base	• a asociación implícita entre los tipos de los conjuntos de agentes, • las asociaciones compuestas entre el tipo del agente contenedor y los tipos de los conjuntos de agentes, o • la asociación implícita entre el tipo de estado compuesto de la máquina de estados y el tipo de agente contenedor y el tipo de agente contenedor y los tipos de los conjuntos de agentes.
^Association	
^GeneralizableElement	Los canales siguen la generalización del tipo de agente contenedor, es decir, las conexiones de un tipo de agente son heredadas
^ModelElement	
.name	*Channel-names*

3.6.3 Cometido de clasificador (ClassifierRole)

Un ClassifierRole representa una interfaz de una puerta de un conjunto de agentes.

UML	SDL
ClassifierRole	el tipo de un conjunto de agentes contenido en un agente
.multiplicity	igual que la multiplicidad del correspondiente AssociationEndRole, es decir, *Number-of-instances*
->availableContents	las señales y procedimientos exportados que están disponibles a través de la puerta de interfaz
->availableFeature	las señales y procedimientos exportados que están disponibles a través de la puerta de interfaz
->base	el tipo de conjunto de agentes
^Classifier	
^GeneralizableElement	la interfaz sigue la generalización del tipo de agente contenedor, es decir, es heredada
^ModelElement	
.name	nombre de conjunto de agentes

3.6.4 Colaboración (Collaboration)

Una Collaboration representa la estructura interna del agente/máquina de estados de un agente y sus conexiones. Las propiedades definidas por una Collaboration son propiedades de todos los casos de la clase representada por *representedClassifier*.

La asociación *representedOperation* no se utiliza, es decir, Collaboration no representa la estructura interna de las entidades SDL que serían representadas por Operations.

UML	SDL
Collaboration	{ <interaction area> | <agent body area> } }***set*** como parte de <agent diagram content>
->constrainingElement	no es aplicable
->interaction	no es aplicable
->ownedElement	los elementos de { <interaction area> | <agent body area> } }***set***
->representedClassifier	el tipo de agente con { <interaction area> | <agent body area> } }***set*** representado por esta colaboración
->representedOperation	no es aplicable
^GeneralizableElement	sigue la correspondiente propiedad del tipo de agente contenedor, es decir, todas las propiedades de un tipo de agente especificado por una Collaboration son heredadas
^Namespace	la unidad de alcance del tipo de agente con el { { <interaction area> | <agent body area> } }***set*** representado por esta colaboración. Mientras el ownedElement de una colaboración sólo representa el { { <interaction area> | <agent body area> } }***set***, el ownedElement del Namespace del tipo de agente representa todas las entidades en el tipo de agente.
^ModelElement	los atributos y asociaciones del ModelElement son los mismos que para la clase que representa el tipo de agente cuyo {<interaction area> | <agent body area>}}***s*** es representado por esta Collaboration

3.6.5 Interacción (Interaction)

No es aplicable.

3.6.6 Mensaje (Message)

No es aplicable.

3.7 Casos de utilización (Use Cases)

En la presente Recomendación no se emplean casos de utilización, que describen un sistema en una forma que está fuera del ámbito del SDL. Sin embargo, se pueden utilizar en combinación con *SDL UML*.

3.8 Máquinas de estados (State machines)

3.8.1 Evento de llamada (CallEvent)

CallEvent representa la entrada de un procedimiento distante en SDL.

UML	SDL
CallEvent	entrada de llamada de procedimiento distante
->operation	<remote procedure identifier>
^Event	
->parameters	<procedure formal parameters>

Obsérvese que en el SDL una entrada de un procedimiento distante no puede ser descartada, sólo puede ser diferida si no es ejecutada.

3.8.2 Evento de cambio (ChangeEvent)

ChangeEvent representa la entrada de una señal continua en SDL.

UML	SDL
ChangeEvent	*Continuous-signal*
.changeExpression	*Continous-expression*
^Event	
->parameters	no es aplicable

En el SDL, la *Continuous-signal* sólo es evaluada cuando no se encuentran otros estímulos en el puerto de entrada.

3.8.3 Estado compuesto (CompositeState)

Un estado compuesto representa un estado compuesto SDL.

El hecho de que un tipo de estado compuesto pueda ser un tipo virtual o un subtipo es representado en la clase con el estereotipo «state».

UML	SDL
CompositeState	*Composite-state-part*
->subvertex	*State-start-nodes, State-nodes* y *Return-nodes* de *Composite-state-graph*
.isConcurrent = false = true	 Estado compuesto que no es un *Multi-state* *Multi-state*
.isRegion = false = true	 Estado compuesto como parte de una *State-machine-definition* *State-partition*

3.8.4 Evento (Event)

Event no tiene contrapartida en SDL, aunque las subclases de Event tienen una correspondencia en SDL.

3.8.5 Estado final (FinalState)

FinalState representa un retorno desde un estado compuesto SDL.

UML	SDL
FinalState	*Return-node* no etiquetado desde un estado compuesto

3.8.6 Guarda (Guard)

Guard representa una condición de habilitación en SDL.

UML	SDL
Guard	*Provided-expression*
.expression	*Boolean-expression*

Obsérvese que en el UML, si Guard evalúa falso, el evento de entrada es descartado, mientras que en el SDL el evento de entrada es aplazado (guardado).

3.8.7 Pseudoestado (PseudoState)

PseudoState representa una abstracción que abarca diferentes tipos de nodos transitorios en un gráfico de máquina de estados.

UML	SDL
PseudoState	Nodos en *State-machine-definition*
.kind = initial = deepHistory = shallowHistory = join = fork = junction = choice	 *Start-state-node* no etiquetado de un estado compuesto ningún concepto ningún concepto ningún concepto ningún concepto *Join-node* para fusión, *Decision-node* para derivación ningún concepto

3.8.8 Evento de señal (SignalEvent)

SignalEvent representa la entrada de una señal en SDL.

UML	SDL
SignalEvent	entrada de señal
->signal	caso de la clase «signal»
^Event	
->parameters	parámetros de señal
.kind = in = inout = return	 obligatorio no es aplicable no es aplicable

3.8.9 Estado simple (SimpleState)

SimpleState representa un estado básico en SDL.

UML	SDL
SimpleState	*State-node* sin *Composite-state-part*

3.8.10 Estado (State)

State representa una combinación de un estado SDL y el contenido de un estado compuesto SDL. Los procedimientos de entrada y salida y las transiciones internas forman parte del contenido del estado compuesto SDL, mientras que *deferrableEvent* (*evento aplazable*) corresponde a mantener el estado como tal.

UML	SDL
State	*State-node* y contenido de *Composite-state-part*
->deferrableEvent	<save part> asociada con el *State-node*
->entry	*Entry-procedure-definition* (de *Composite-state-part*)
->exit	*Exit procedure-definition* (de *Composite-state-part*)
->doActivity	ningún concepto
->internalTransition	*Transitions* internas (de *Composite state-part)*

3.8.11 Máquina de estados (StateMachine)

StateMachine representa una máquina de estados de un agente o estado compuesto.

UML	SDL
StateMachine	*State-machine-definition*
->context	*Agent-type-definition* con *State-machine-definition* o *Procedure-definition* con *Procedure-graph*
->top	el estado compuesto contenedor de la *State-machine-definition*
->transition	el *State-transition-graph* de la definición *State-machine-definition*

3.8.12 Vértice de estado (StateVertex)

StateVertex representa una abstracción de un nodo en un gráfico de máquina de estados. Los posibles nodos en SDL son (en este contexto): *State-node*, *Nextstate-node*, *Start-node*, *Return-node*, *Stop-node*, *Decision-node* y <join>.

UML	SDL
StateVertex	*Transition*
->outgoing	*Transition*s que van de un nodo a *Terminators* (*terminadores*) de transición
->incoming	*Transition*s que tienen el nodo como *Terminator* (*terminador*) de transición
->container	*State-transition-graph* contenedor

3.8.13 Estado matriz (StubState)

StubState no tiene contrapartida directa en el SDL. El concepto similar en el SDL es el punto de conexión de entrada/salida del estado. Se considera como esencial en el SDL para definir los posibles puntos de entrada/salida como parte de la interfaz de un estado compuesto, y no para permitir la entrada a cualquier subestado.

3.8.14 Estado de submáquina (SubmachineState)

En la semántica del UML no está claro si un SubmachineState se basa o no en clase, aunque se menciona la reutilización. La correspondencia más próxima en SDL es con el estado compuesto basado en tipo. Alternativamente, corresponde con una referencia de estado compuesto en el SDL, y

es soportado por el hecho de que el mismo estado de submáquina puede ser referenciado más de una vez en la máquina de estados contenedora, pero la expansión de tipo macro descrita en el UML indica que no se trata de una referencia.

3.8.15 Estado síncrono (SynchState)

SynchState no tiene contrapartida en el SDL.

3.8.16 Evento de temporización (TimeEvent)

TimeEvent representa la entrada de un temporizador en el SDL.

UML	SDL
TimeEvent	entrada de temporizador
.when	*Time-expression*
^Event	
->parameters *.kind* = in = inout = return	*Variable-identifier**s en *Input-node* obligatorio no es aplicable no es aplicable

3.8.17 Transición (Transition)

Transition representa una transición en el SDL. La única diferencia es que una transición en el SDL está asociada con la *source* (*fuente*), mientras que ésta es una asociación de *transition*.

UML	SDL
Transition	*Transition*
->trigger	*Input-node* o *Continuous-signal*
->guard	*Provided-expression*
->effect	*Graph-node** de *Transition*
->source	*State-node* al cual está asociada la transición
->target	*Terminator*

Cuando *source* (*fuente*) y *target* (*objetivo*) están situados en diferentes máquinas de estado, *effect* (*efecto*) será ejecutado en la máquina de estados que posee *source*. Cuando se hace corresponder con el SDL, un *effect* que atraviesa una frontera de estado para ir de una máquina de estados interna a una máquina de estados externa introducirá un punto <state exit point> (<punto de salida de estado>) en la máquina de estados interna. En cambio, un *effect* que atraviesa una frontera de estado para ir de una máquina de estados externa a una máquina de estados interna introducirá un <state entry point> (<punto de entrada a estado>) en la máquina de estados interna.

3.9 Gráficos de actividad (Activity Graphs)

En la presente Recomendación no se utilizan gráficos de actividad.

3.10 Gestión de modelo (Model Management)

En el lote Model Management, sólo se utilizan las metaclases Package (lote) y Model (modelo), no Subsystem (subsistema). Model y Package son clases básicas de metamodelo en UML. No se define ningún estereotipo particular, pero el uso estará restringido como se describe en esta subcláusula.

3.10.1 Importación de elemento (ElementImport)

No es aplicable.

3.10.2 Modelo (Model)

Model representa una *SDL-specification*. Esto entraña que un modelo puede contener solamente un conjunto de lotes y una clase de sistema.

NOTA – Posiblemente también clases en el entorno, tipos jerarquizados, interfaces y asociaciones.

UML	SDL
Model	*SDL-specification*
^Package	
-> *importedElement*	ningún concepto
^GeneralizableElement	no es aplicable
^Namespace	Unidad de alcance
-> *ownedElement*	*Package-definition*-**set** [*Agent-definition*]
^ModelElement	
.name	ningún concepto (pero queda en estudio).
-> *constraint*	ningún concepto
-> *namespace*	no es aplicable – una especificación SDL-no se define en una unidad de alcance
.template	no es aplicable
.templateParameter	no es aplicable
-> *supplierDependency*	no es aplicable
->*clientDependency*	no es aplicable

Sólo los ModelElements que corresponden con entidades contenidas en *SDL-specification* pueden ser *ownedElements* de un modelo.

3.10.3 Lote (Package)

Un Package representa un lote *Package-definition* de SDL.

UML	SDL
Package	*Package-definition*
-> *importedElement*	Definiciones hechas visibles a este lote por medio de <package use clause> y <definition selection list>
^GeneralizableElement	no es aplicable
^Namespace	Unidad de alcance de lote
-> *ownedElement*	Entidades en el subconjunto de <entity in package> que se define más adelante
^ModelElement	
.name	
-> *constraint*	ningún concepto
-> *namespace*	la unidad de alcance que engloba y contiene el lote, por ejemplo, la especificación *SDL-specification* o *Package-definition*

UML	SDL
.template	no es aplicable
.templateParameter	no es aplicable
-> supplierDependency	Para un lote en un <package use clause> de tal forma que este lote se utilice para un cierto número de lotes
-> clientDependency	Para un lote con un <package use clause> de tal forma que este lote utilice otros lotes

El *namespace* debe ser Model o Package.

El *ownedElement* debe ser elementos de modelo que representan los siguientes elementos de <entity in package>:

- *Package-definition*;
- *Agent-type-definition*;
- *Signal-definition*;
- <remote variable definition>;
- *Data-type-definition*;
- *Procedure-definition*;
- *Composite-state-type-definition*; *o*
- *Interface-definition.*

Un lote SDL es definido completamente por la *Package-definition*. Un Package define partes del lote SDL. Las partes que define dependen del contenido del lote. La elección de este contenido se deja al usuario, pero concordará con las propiedades correspondientes definidas en la *Package-definition* de SDL.

La <interface> de un <package heading> no es abarcada directamente por esta extensión. Indirectamente puede ser representada por el atributo de visibilidad de cada Feature (característica) en los *ownedElement*s del lote.

Las Dependencies (dependencias) entre lotes representan <package reference clause>s (en <package reference area>). La <definition selection list> no está representada. Para un lote en un <package use clause> *supplierDependency* indica las Dependencies correspondientes a la utilización del lote por un cierto número de lotes. Para un lote con un <package use clause> *clientDependency* indica las Dependencies correspondientes a la utilización de otros lotes.

3.10.4 Subsistema (Subsystem)

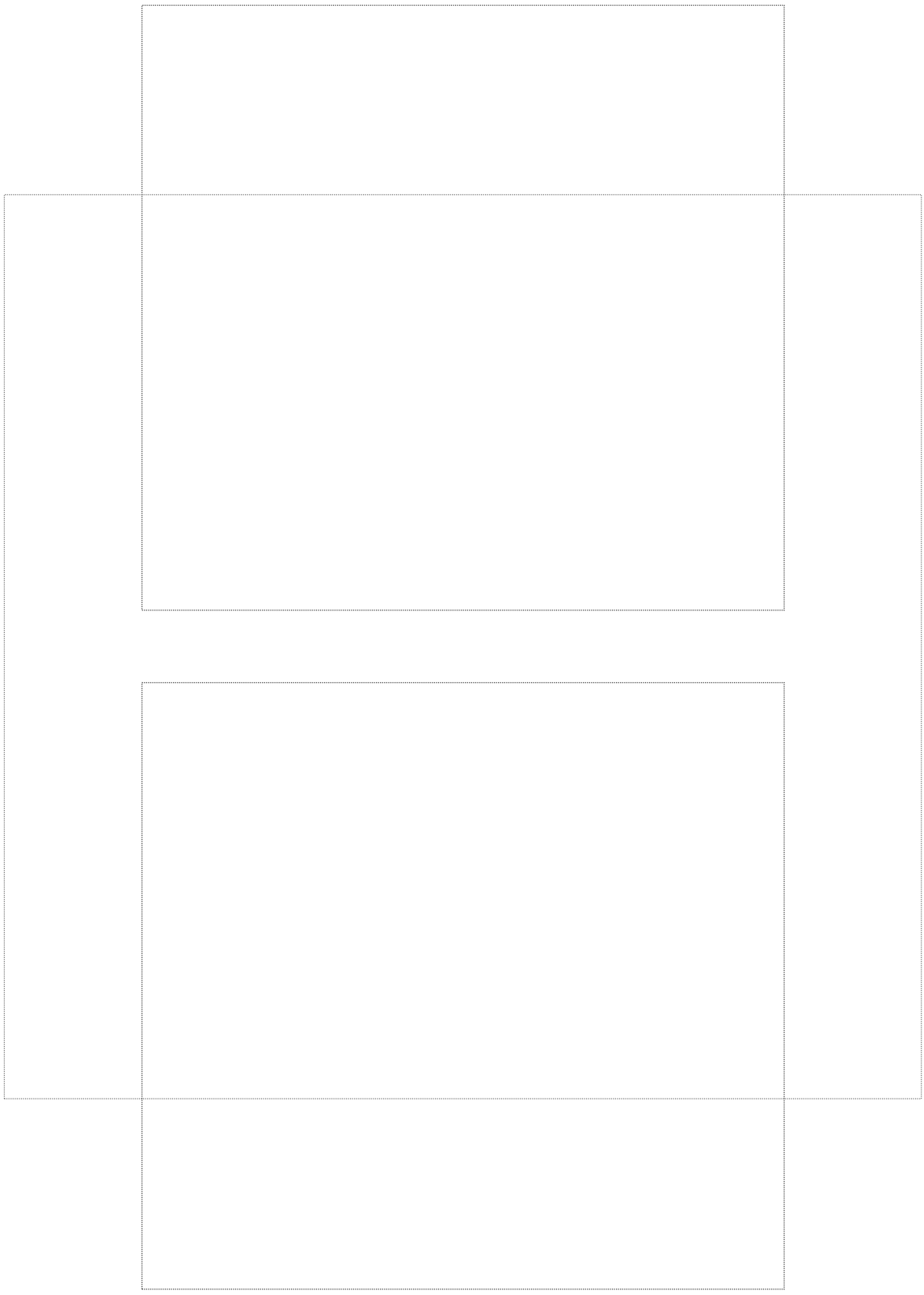
No es aplicable.

APÉNDICE I

Comportamiento común

A continuación figura una visión general de una posible correspondencia del lote Common Behaviour (Comportamiento común) de UML.

UML	SDL
Action	Acción
ActionSequence	Cadena de transición
Argument	Parámetros de señal de salida
AssignmentAction	Asignación
AttributeLink	Vinculación de un identificador con la definición de la variable o campo
CallAction	Llamada de procedimiento distante o invocación de método
ComponentInstance	No aplicable
CreateAction	Petición de creación
DestroyAction	Ningún concepto
DataValue	Valor
Exception	Ejemplar de excepción
Instance	Ejemplar/entidad
Link	Ningún concepto
LinkEnd	Ningún concepto
LinkObject	Ningún concepto
NodeInstance	Ningún concepto
Object	Entidad basada en tipo
Reception	Entrada
ReturnAction	Retorno de un procedimiento que retorna un valor
SendAction	Salida
Signal	Señal – Véase cuadro aparte en 3.5.2
Stimulus	Consumo de una señal o ejecución de un procedimiento distante
TerminateAction	Parada
UninterpretedAction	Acción informal



SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie B	Medios de expresión: definiciones, símbolos, clasificación
Serie C	Estadísticas generales de telecomunicaciones
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedios
Serie I	Red digital de servicios integrados
Serie J	Transmisiones de señales radiofónicas, de televisión y de otras señales multimedios
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	RGT y mantenimiento de redes: sistemas de transmisión, circuitos telefónicos, telegrafía, facsímil y circuitos arrendados internacionales
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	Redes de datos y comunicación entre sistemas abiertos
Serie Y	Infraestructura mundial de la información y aspectos protocolo Internet
Serie Z	**Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación**

Impreso en Suiza
Ginebra, 2000