UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

Z.100

Suplemento 1
(05/97)

SERIE Z: LENGUAJES DE PROGRAMACIÓN

Técnicas de descripción formal – Lenguaje de especificación y descripción (SDL)

Metodología SDL+: Utilización de MSC y SDL (con ASN.1)

Recomendación UIT-T Z.100 – Suplemento 1

(Anteriormente Recomendación del CCITT)

RECOMENDACIONES DE LA SERIE Z DEL UIT-T

LENGUAJES DE PROGRAMACIÓN

TÉCNICAS DE DESCRIPCIÓN FORMAL	Z.100–Z.199
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	Z.200–Z.299
CHILL: el lenguaje de alto nivel del UIT-T	Z.200–Z.209
LENGUAJE HOMBRE-MÁQUINA	Z.300–Z.499
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
METÓDOS DE VALIDACIÓN Y PRUEBAS	Z.500–Z.599

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

SUPLEMENTO 1 A LA RECOMENDACIÓN UIT-T Z.100

METODOLOGÍA SDL+: UTILIZACIÓN DE MSC Y SDL (CON ASN.1)

Resumen

Este Suplemento se publica como Suplemento a la Recomendación Z.100 (SDL), pero es asimismo relevante para la Recomendación Z.105 (SDL combinado con ASN.1) y Z.120 (MSC). Describe una metodología para la utilización combinada de estos lenguajes.

Este Suplemento abarca los temas siguientes:

1) Definiciones terminológicas y material de referencia para la utilización y aplicación del SDL.
2) Una visión general de la metodología (cláusula 4).
3) Descripciones detalladas sobre:
   a) Análisis de requisitos (cláusula 5).
   b) Diseño previo e investigación de la aplicación (cláusula 6).
   c) Formalización de la aplicación en SDL+ (cláusula 7).
   d) Temas relacionados con la implementación (cláusula 8).
   e) Validación (cláusula 9).
4) Relaciones con otros lenguajes y técnicas.
5) Una elaboración de la metodología para la especificación de servicio.

Este Suplemento no es exhaustivo. Su objetivo es que sea incluido por los usuarios de SDL+ en sus metodologías generales y adaptado a sus sistemas de aplicación y a sus necesidades específicas. En concreto, este Suplemento no cubre los aspectos propios de la obtención de una implementación a partir de la especificación o de la prueba en detalle de los sistemas. En el caso de prueba, se espera que dicho aspecto sea cubierto por un documento separado que trata de la generación de pruebas para las normas o los productos basados en SDL+.

La metodología constituye un marco que debe ser elaborado en detalle para cada contexto de utilización real. Este Suplemento consta de dos partes. En la parte I se explica el marco, ofreciéndose una visión general en la cláusula 4, con una descripción más detallada en las cláusulas 5, 6 y 7. En la parte II y comenzando en la cláusula 12, el marco general se especializa para la especificación de servicios. Parte de las cláusulas 5, 6 y 7 se elaboran en detalle en las cláusulas 13, 14 y 15. En dicha especialización se opta por determinadas posibilidades del enfoque. Muchos de los detalles de las cláusulas 13, 14 y 15 pueden utilizarse cuando el marco general se elabora con detalle para otros contextos y otras opciones.

Orígenes

El Suplemento 1 a la Recomendación UIT-T Z.100 ha sido preparado por la Comisión de Estudio 10 (1997-2000) del UIT-T y fue aprobado por el procedimiento de la Resolución N.° 1 de la CMNT el 6 de mayo de 1997.

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

Página

METODOLOGÍA SDL+: UTILIZACIÓN DE MSC Y SDL (CON ASN.1)

(Ginebra, 1997)

1 Contexto

Este Suplemento está destinado a los responsables de la metodología de aquellas áreas en las que se utilizan las Recomendaciones de la serie Z.100.

El objetivo de este Suplemento es proporcionar una metodología para la utilización efectiva de las Recomendaciones sobre lenguajes de la serie Z.100: Recomendación Z.100, Lenguaje de especificación y descripción del CCITT (SDL, *specification and description language*) [1]; Recomendación Z.105, SDL combinado con ASN.1 [2] y Recomendación Z.120, Gráfico de secuencias de mensajes (MSC, *message sequence chart*) [3]. La aplicación de esta metodología debe dar lugar a una especificación correcta de la aplicación en lenguaje SDL combinado con ASN.1 y MSC.

Cuando se hace uso de la Recomendación Z.105, el SDL se utiliza para definir el comportamiento y ASN.1 se utiliza para definir los datos. También es posible utilizar SDL para definir datos como en la Recomendación Z.100. Este Suplemento recoge tanto la utilización de SDL sin ASN.1, como la de SDL en combinación con ASN.1, tal como se hace en la Recomendación Z.105.

El MSC se utiliza para describir secuencias de sucesos y normalmente no abarca todas las posibilidades. También se utiliza para describir lo que ocurre en el caso de secuencias específicas de estímulos, mientras que SDL define el comportamiento ante cualquier estímulo cualquiera que sea el estado. La utilización de MSC sin SDL no proporciona normalmente una descripción completa del comportamiento del sistema. En la metodología descrita en este Suplemento, el SDL no se utiliza normalmente sin MSC. El Suplemento no describe una metodología general para la utilización del MSC, ya sea solo o con lenguajes distintos al SDL (con y sin ASN.1).

Debido a que la metodología asume que se utiliza SDL con ASN.1 y MSC, el término "SDL+" se utiliza en este Suplemento con el significado de "SDL (Z.100) con ASN.1 (Z.105) y MSC (Z.120)".

Es notorio que existen muchas formas de utilizar SDL+, en función de las preferencias del usuario, de la aplicación objetivo, de las reglas internas de la organización, etc. Por lo tanto, la metodología expuesta en este Suplemento es incompleta y debe ser elaborada con más detalle para generar la metodología efectiva para un contexto en particular. Este Suplemento pretende ayudar a los usuarios de SDL+ y fomentar la utilización unificada del lenguaje y de las herramientas de apoyo al mismo.

2 Definiciones

En este Suplemento se definen los términos siguientes.

2.1 **actividad**: Un procedimiento específico de un proceso de ingeniería, soportado por uno o varios métodos.

2.2 **agente**: Un objeto que modela el comportamiento de una entidad en el entorno de un sistema de aplicación.

2.3 **actor**: Una persona (u organización) en el entorno de un sistema que tiene uno o más cometidos de interacción de comportamiento con un sistema. En el contexto de este Suplemento, un actor que interactúa con las actividades de un sistema de ingeniería se denomina *ingeniero*.

2.4 **Análisis**: Actividad que se realiza paso a paso para explorar los requisitos que se han recopilado, organizando la información que contienen y registrándola en un formato que refleje dicha organización (véase la cláusula 5).

2.5 **concepto de aplicación**: Concepto que pertenece a un dominio de aplicación dado y al sistema que se especifica.

2.6 **dominio de aplicación**: Campo de actividad en el que opera el sistema de aplicación objeto de la especificación.

2.7 **sistema de aplicación**: (que se abrevia como *sistema*, donde el termino calificador se deriva del contexto) conjunto complejo de las partes utilizadas para proporcionar funcionalidad y otras características. En el contexto de este Suplemento no se hace normalmente distinción entre una especificación de sistema, una descripción de sistema y una instancia de sistema real.

2.8 **ASN.1**: Notación de Sintaxis Abstracta Uno (Recomendaciones X.208 [4] y X.680 [5]).

2.9 **asociación**: Relación de dependencia entre dos o más clases (o dos o más objetos) en la notación de modelo de objeto (véanse 13.2 y la referencia [11]) y que se expresa como una línea anotada entre símbolos de clase (u objeto).

2.10 **comportamiento**: Secuencia de acciones con estímulos y respuestas realizados por un sistema que puede cambiar de estado.

2.11 **clase**: Descripción del modelo del sistema que define un conjunto de objetos con propiedades semejantes (la misma estructura y el mismo comportamiento). Una clase representa un concepto de aplicación.

2.12 **información clasificada**: Modelo con los requisitos recopilados que se ha estructurado y definido de acuerdo con los conceptos que se han nombrado y definido.

2.13 **requisitos recopilados**: Conjunto de requisitos para la aplicación que han sido recopilados como resultado de la actividad de Recopilación de Requisitos.

2.14 **visión computacional**: Visión de un sistema y de su entorno que se centra en la descomposición del sistema para permitir la distribución (véase 4.1/X.903).

2.15 **vision de diseño**: Visión de un sistema y de su entorno que se centra en las funciones requeridas para soportar dicho sistema (véase el "punto de vista de ingeniería" en 4.1/X.903.

2.16 **Diseño Previo**: Actividad que se realiza paso a paso para realizar parcialmente la ingeniería de las especificaciones informales a partir de distintos puntos de vista y con diferentes niveles de detalle a fin de investigar las opciones de diseño de la ingeniería (véase la cláusula 6).

2.17 **diseños previos**: Diseños generales para el sistema considerado utilizando uno o más modelos que describen parcialmente el sistema.

2.18 **Documentación**: Actividad para la selección, catalogación y almacenamiento de la información sobre un producto.

2.19 **visión de empresa**: Visión de un sistema y de su entorno que se centra en el objetivo, campo de aplicación y políticas aplicables a dicho sistema (véase 4.1/X.903).

2.20 **Formalización**: Actividad que se realiza paso a paso en la que se produce la descripción formal SDL de un sistema (véase la cláusula 7).

2.21 **descripción formal SDL+**: Descripción SDL+ que es conforme a las Recomendaciones Z.100 [1], Z.105 [2] y Z.120 [3], que es consistente, inequívoca y carece de "texto informal" SDL, que describe con precisión el sistema considerado.

2.22 **validación formal**: Investigación sistemática de una especificación (o de una implementación) para determinar si ésta posee determinadas propiedades que son deseables y que incluye: la verificación de la corrección de la sintaxis y de la semántica de la especificación (o implementación) y la verificación de que ésta expresa los requisitos conocidos del sistema que se especifica (o implementación).

2.23 **directriz**: Asesoramiento al usuario de la metodología para ayudarle a identificar las decisiones que deben tomarse y que puede incluir motivos y orientaciones para la toma de decisiones.

2.24 **descripción informal SDL+**: Una descripción SDL+ resultado de un Diseño Previo y que incumple uno o más criterios de las descripciones formales; en el caso de SDL los incumplimientos incluyen ambigüedades, uso de texto informal y no conformidad con la Recomendación Z.100 [1]; en el caso de ASN.1 los incumplimientos incluyen la utilización de "any" en la descripción de un tipo de datos (siempre se asume que los MSC son formales dentro de las limitaciones de su función, cual es proporcionar una traza del mensaje).

2.25 **visión de la información**: Visión de un sistema y de su entorno que se centra en la semántica de la información y en las actividades de procesamiento de información del sistema (véase 4.1/X.903).

2.26 **instrucción**: Declaración imperativa que especifica lo que debe realizarse como parte de un paso.

2.27 **método**: Combinación de notación con instrucciones, reglas y directrices para su utilización (véanse [6] y [7]). Un *método* es una forma semántica de conseguir un objetivo específico. Un método es descriptivo.

2.28 **metodología**: Un conjunto organizado y coherente de métodos y principios utilizados en una disciplina en particular. Según un diccionario, una metodología es un sistema de métodos y principios utilizados en una disciplina concreta. Un método es una forma semántica de hacer algo, o alternativamente, las técnicas o disposiciones de trabajo para una materia o campo concreto. En el contexto de este Suplemento, las disciplinas son los desarrollos de sistemas de telecomunicación, protocolos, etc., llamados *sistemas de aplicación* (o, por brevedad, sencillamente *sistemas*, donde el término calificador puede deducirse del contexto).

2.29 **modelo**: Representación de algo que muestra algunas de las propiedades y características de comportamiento del mismo.

2.30 **objeto**: Elemento individual que es miembro de una clase y que tiene un cometido bien definido en el sistema, posee propiedades y puede ser gestionado y caracterizado por:

– su estado (todas sus propiedades);

– su comportamiento (la forma en que actúa y reacciona);

– su identidad (en qué difiere de los demás).

2.31 **orientación a objeto**: Sistema para dividir el sistema en estudio en objetos separados, agrupando éstos en clases y permitiendo que los objetos interactúen unos con otros y con el entorno a fin de realizar las funciones del sistema en su totalidad.

2.32 **producto**: La aplicación (o especificación) y la documentación asociada que debe producirse.

2.33 **requisito**: Expresión de las ideas que deben incorporarse en la aplicación que se desarrolla.

NOTA – La Directriz 2 de la ISO/CEI:1996, *Standarization and related activities – General vocabulary*, contiene la definición siguiente:

requisito: Disposición que incluye los criterios que deben satisfacerse.

2.34 **Recopilación de Requisitos**: Actividad destinada a evaluar inicialmente los requisitos capturados, respondiendo a las cuestiones que sobre ellos surgen durante el desarrollo de una descripción formal SDL.

2.35 **biblioteca de reutilización**: Almacén de conceptos de aplicación disponibles para ser utilizados en cualquier momento del proceso de desarrollo de un sistema.

NOTA – Este Suplemento no cubre la provisión y funcionamiento de una biblioteca de reutilización.

2.36 **regla**: Declaración de principios para prevenir resultados no válidos, inverificables e indeseables. Habrá conformidad con todas aquellas reglas que se expresen con el término "debe", mientras que ésta es muy recomendable cuando se utilicen términos tales como "debería" o "debiera" (aunque puedan existir excepciones).

2.37 **servicio**: Conjunto de funciones y facilidades que un proveedor ofrece a un usuario.

NOTA 1 – En esta definición el "usuario" y el "proveedor" pueden ser una pareja tal como aplicación/aplicación, persona/computador, abonado/organización (operador). Los distintos tipos de servicios incluidos en esta definición son los servicios de transmisión de datos y los servicios de telecomunicación ofrecidos a sus clientes por un operador, así como servicios ofrecidos por una capa en un protocolo de capa a otras capas.

NOTA 2 – **servicio SDL** (Z.100 [1]): forma alternativa de especificar una parte del comportamiento de un proceso (una máquina de estados finitos de comunicaciones ampliada).

2.38 **especificación**: Detalles e instrucciones que describen los comportamientos, estructuras de datos, diseños, materiales, etc. de una implementación que es conforme con una aplicación.

NOTA – **especificación SDL** (Z.100 [1]): definición de los requisitos de un sistema. Una especificación consta de los parámetros generales que requiere el sistema y la especificación funcional de su comportamiento requerido.

2.39 **paso**: Tarea con contenido propio y que se incluye en una actividad.

2.40 **sistema**: (véase *sistema de aplicación*).

2.41 **técnica**: Forma de realizar un método.

2.42 **visión de la tecnología**: Visión de un sistema y de su entorno que se centra en la elección de la tecnología para dicho sistema (véase 4.1/X.903).

2.43 **Prueba**: Combinación de dos actividades: *Especificación de Prueba* y *Ejecución de Prueba*.

2.44 **Especificación de Prueba**: Actividad que produce un conjunto de pruebas para objetivos particulares como por ejemplo, la prueba de la conformidad de una implementación con una descripción formal SDL+ o el interfuncionamiento de varios productos.

2.45 **Ejecución de Prueba**: Actividad destinada a realizar las pruebas descritas en una Especificación de Prueba a fin de obtener un conjunto de resultados de las mismas.

2.46 **herramienta**: Medios para la realización de una secuencia de pasos que tienen un objetivo dado y que normalmente (en esta metodología) se implementan como un programa de soporte lógico.

2.47 **tipo**: **tipo de sistema** SDL, **tipo de bloque**, **tipo de proceso**, definición de **procedimiento** o de **señal**.

2.48 **secuencia de uso**: Secuencia de transacciones relacionadas y que realiza un usuario de un sistema para un fin en particular.

2.49 **Validación**: Proceso con métodos asociados, procedimientos y herramientas mediante los cuales se evalúa si una aplicación (o una norma) está totalmente implementada, es conforme con las normas y criterios aplicables a dicha aplicación (o norma) y si satisface el objetivo expresado en el registro de los requisitos en los que se basa la aplicación (o norma); en el caso de una norma, indica que una implementación que sea conforme con ella goza de la funcionalidad expresada en el registro requisitos en los que se basa la norma.

2.50 **modelo de validación**: Versión detallada de una especificación o implementación, que puede incluir partes de su entorno y que se utiliza para realizar la validación formal.

3 Ventajas de la utilización de SDL con ASN.1 y MSC

Existe un amplio consenso en que el éxito de un sistema se fundamenta en el desarrollo cuidadoso de la fase de especificación y diseño del mismo. Ello requiere, por otra parte, un lenguaje de especificación adecuado que satisfaga las necesidades siguientes (en aras de la brevedad, al resultado de la especificación y diseño del sistema se le llama *especificación*):

– un *conjunto de conceptos* bien definido;

– especificaciones *inequívocas*, *claras*, *precisas* y *concisas*;

– una base para el *análisis* del grado de *compleción* y *corrección* de las especificaciones;

– una base para determinar que la *conformidad* de las implementaciones con la especificación;

– una base para determinar la *consistencia* mutua de las implementaciones;

– la utilización de *herramientas* basadas en computadora para crear, mantener, analizar y simular especificaciones.

Un sistema puede tener especificaciones a distintos niveles de abstracción. Una especificación constituye una base a partir de la cual se derivan las *implementaciones* o un modelo (por ejemplo, en una norma) con el cual se compara una *implementación*. Una especificación debiera evitar entrar en detalles propios de la implementación a fin de:

– dar una visión general de un sistema complejo;

– posponer decisiones relativas a la implementación; y

– no excluir ninguna implementación válida.

Por contraposición con un programa, una especificación formal, (es decir, una especificación escrita en un lenguaje formal de especificación) no está destinada a ser ejecutada en una computadora. Además de servir como base para obtener implementaciones, una especificación formal puede ser utilizada para la comunicación precisa e inequívoca entre las personas, particularmente para la elaboración de pedidos y de licitaciones.

La utilización de SDL+ permite analizar, fundamentar y simular soluciones de sistemas alternativos con distintos niveles de abstracción, lo cual no es viable en la práctica cuando se utiliza un lenguaje de programación debido a los costos y al tiempo que conlleva. SDL+ ofrece al usuario del lenguaje un conjunto bien definido de conceptos, mejorando su capacidad para obtener la solución a un problema y para fundamentar dicha solución.

3.1 Comprensión de una especificación SDL+

Los conceptos de SDL+ se basan en modelos matemáticos y en la teoría del significado. A continuación se expone como aplicar los modelos SDL+ a una aplicación.

El dominio de aplicación de un sistema se comprende en última instancia en términos de conceptos de un lenguaje natural. Los individuos adquieren la comprensión de estos procesos a través de un largo proceso de aprendizaje y de experiencia en la vida real. En la descripción de una aplicación en un lenguaje natural, los fenómenos se describen tal como son percibidos por un observador. La descripción en lenguaje natural del sistema utiliza los conceptos que se derivan directamente de la aplicación e implementación del mismo.

Cuando un sistema se especifica utilizando SDL+, la especificación no utiliza directamente conceptos de aplicación ni de implementación. En su lugar, el SDL+ define un *modelo* que representa las propiedades significativas (principalmente comportamiento) del sistema. A fin de comprender este modelo, debe establecerse una correspondencia entre éste y la comprensión intuitiva de la aplicación en términos de conceptos de lenguaje natural, tal como se indica en la figura 3-1. Esta correspondencia puede establecerse de varias formas, una consiste en elegir nombres para los conceptos introducidos en SDL+ que ofrecen una buena asociación con los conceptos de aplicación y otra es comentar el SDL+. Se trata de algo parecido a la comprensión de un programa o un algoritmo que soluciona un problema de la vida real.

Un modelo debiera gozar de un alto *poder analítico* y un alto *poder de expresión* a fin de facilitar la correspondencia con la aplicación. Desafortunadamente, el poder analítico y el poder de expresión entran generalmente en conflicto: a mayor poder de expresión de un modelo, más difícil resulta analizarlo. Cuando se diseña un lenguaje de especificación, debe establecerse un equilibrio entre ambas propiedades. Además, debe destacarse que un modelo representa siempre una visión simplificada de la realidad y que, en consecuencia, siempre tiene limitaciones.

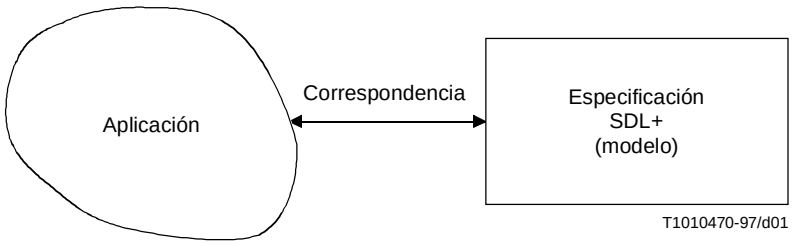


Figura 3-1/Sup. 1 a la Rec. Z.100 – Comprensión de una especificación SDL+

3.2 El área de aplicación del SDL+

Aunque el SDL es ampliamente conocido en el campo de las telecomunicaciones, tiene un campo de aplicación más amplio, utilizándose también en otros sectores industriales. El campo de aplicación del SDL+ puede caracterizarse como sigue:

– *tipo de sistema:* tiempo real, interactivo, distribuido;

– *tipo de información:* comportamiento y estructura;

– *nivel de abstracción:* de la visión general al detalle.

El SDL+ se ha desarrollado para su uso en sistemas de telecomunicación incluyendo las comunicaciones de datos, pero realmente puede utilizarse en cualquier sistema interactivo y de tiempo real. Se ha diseñado para especificar el comportamiento de dicho sistema, es decir, la cooperación existente entre el sistema y su entorno. También está concebido para la descripción de la estructura interna de un sistema, de tal forma que las distintas partes de éste puedan desarrollarse y comprenderse una a una. Esta característica es esencial en los sistemas distribuidos.

SDL+ abarca distintos niveles de abstracción, desde una visión general amplia hasta el diseño detallado. No fue concebido inicialmente como un lenguaje de implementación, pero en muchos casos es posible la traducción automática de una especificación en SDL+ a un lenguaje de programación. En algunos casos, los cambios que deben realizarse en una especificación en SDL para producir una implementación en SDL son reducidos.

3.3 Relación con la implementación

Normalmente se distingue entre lenguajes *declarativos* y *constructivos*. SDL es un lenguaje constructivo, lo cual significa que una especificación SDL+ define un *modelo* que representa propiedades significativas de un sistema (tal como se ha mencionado anteriormente). Una cuestión importante radica en cuales debieran ser estas propiedades significativas. SDL+ deja esta cuestión abierta, *siendo potestad del usuario decidir cuales debieran ser dichas propiedades.*

Si se toma una decisión para representar solamente el comportamiento del sistema (tal como éste se ve en sus límites), se habla entonces de una *especificación*, y la estructura dada del modelo es solo una ayuda a fin de estructurar la especificación en partes manejables. Si se toma una decisión para representar también la estructura interna del sistema, se habla normalmente de una descripción. Éste es el motivo por el que en la Recomendación Z.100 no se hace ninguna distinción entre especificación y descripción.

Nótese que la estructura interna del sistema se representa normalmente en SDL mediante *bloques*. Los *procesos* y la señalización interna de los *bloques* no representa necesariamente parte de la estructura interna del sistema y, por lo tanto, no necesita considerarse como requisitos de la implementación, como algunas personas suponen erróneamente. La conformidad de las implementaciones con las especificaciones se trata normalmente en los organismos de especificación mediante *sentencias explícitas de conformidad*. Ello es evidentemente necesario cuando se utiliza un lenguaje constructivo.

Debido a que SDL+ describe la estructura y el comportamiento, una *descripción* SDL+ puede representar con precisión una implementación, e igualmente, dicha *descripción* precisa puede utilizarse como la descripción de lenguaje de la que se obtiene automáticamente el soporte lógico operacional. Por lo tanto, SDL+ puede utilizarse como un lenguaje de especificación abstracto (por ejemplo, en normas para protocolos) y asimismo como un lenguaje de implementación. La utilización de un único lenguaje evita una posible fuente de error: la traducción de un lenguaje a otro. Una especificación y la correspondiente implementación en SDL+ pueden tener diferencias significativas que resultan de tener en cuenta aspectos de la implementación. Aunque se utiliza el mismo lenguaje a distintos niveles, es probable que algunas partes de una especificación SDL+ requieran modificaciones para poder ser utilizadas directamente en una implementación basada en SDL+.

En la metodología empleada en este Suplemento se supone que el trabajo de ingeniería necesario para convertir una especificación formal SDL+ detallada (ejecutable) en una descripción de implementación es bastante reducido. Este supuesto no es siempre válido. Si los cambios necesarios son significativos, debe de ampliarse la metodología a fin de abarcar el trabajo adicional para la descripción de la implementación. En todo caso, la metodología no cubre la generación de instancias reales del sistema a partir de la descripción. Los métodos que deben aplicarse para la realización varían en función del equipo involucrado, la organización, la ubicación física y muchos otros factores. La metodología produce SDL+ que puede utilizarse como una especificación o como una descripción de la implementación.

PARTE I – METODOLOGÍA MARCO

4 Visión general de las actividades y descripción general de la metodología

La metodología descrita en este Suplemento es un conjunto coherente de actividades utilizadas en la ingeniería de sistemas para producir la definición de un producto (ya sea una especificación o la descripción de una implementación). La definición del producto puede utilizarse como parte de una norma, como parte de una especificación para una compra, como una especificación anterior a la implementación, como código fuente de una implementación o como la base para el desarrollo de pruebas de un producto.

La metodología se divide en actividades. El Análisis de los requisitos y la Formalización en SDL+ constituyen *actividades*. Puede haber varios métodos alternativos para la misma actividad. Por lo tanto, la evaluación de enfoques alternativos y la selección de uno de ellos es un aspecto importante cuando se elabora la metodología. La selección debe basarse (entre otras cosas) en las características de la aplicación.

Una actividad es un método o proceso que está regido por algunos principios, acepta entradas y produce salidas o resultados (véase la figura 4-1), La persona (u organización) que realiza una actividad se llama "actor" y puede llevar a cabo distintos "cometidos" (o funciones). En el contexto de este Suplemento se utiliza el término "ingeniero" en lugar de "actor", debido a que las actividades son las propias de la ingeniería. Para la ingeniería mediante SDL+, los ingenieros pueden asumir diversos cometidos: analista de sistemas, programador, ingeniero de pruebas, diseñador de soporte físico, gerente de proyecto, operador de equipo y otras. Normalmente, estos cometidos están bien definidos. Para la ingeniería de normas, los cometidos pueden ser: autoridad para la definición del campo de aplicación de las normas, relator de normas, diseñador de normas, diseñador del conjunto de pruebas abstractas, autoridad de aprobación de normas y otras. Normalmente una actividad se corresponde con un

cometido principal (por ejemplo, especificación del sistema y especificador de sistema) pero a menudo interactúa con otros cometidos (cliente del equipo, gerente de producto, ingeniero de pruebas). Los diferentes cometidos de los ingenieros quedan fuera del ámbito de este Suplemento. Frecuentemente, un ingeniero tiene más de un cometido.

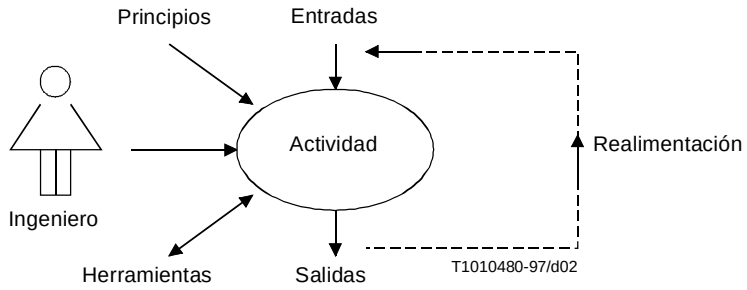


Figura 4-1/Sup. 1 a la Rec. Z.100 – Una actividad

En la ingeniería de equipos, es obvio que las actividades de mercadeo, diseño y gestión tienen lugar en paralelo y son interdependientes. Éste es normalmente el caso, incluso en la creación de normas. Existen dependencias entre las actividades que se definen en este Suplemento, pero es posible que diversas actividades se realicen en paralelo (dentro de las limitaciones impuestas por dichas dependencias). Por ejemplo, la actividad de Documentación puede comenzar tan pronto como exista una comprensión clara del campo de aplicación y los objetivos de la aplicación, pero no puede completarse hasta que no se haya producido una adecuada descripción SDL+. Aparte de las lógicas limitaciones impuestas por la necesidad de disponer de entradas procedentes de otras actividades, este Suplemento no determina la forma en que las actividades se planifican, se organizan y se gestionan. No existe implícito un modelo de "ciclo de vida".

En la ingeniería de sistemas, es en general útil tener distintas visiones del sistema. Ello permite considerar distintos hechos sobre el mismo. El enfoque del procesamiento distribuido abierto (ODP, *open distributed processing*) [8] permite cinco visiones diferentes: empresa, información, computación, diseño ("ingeniería") y tecnología (véase la figura 4-2).

NOTA – En el sentido de ODP, "ingeniería" hace referencia a los aspectos de diseño tales como mecanismos de control, calidad de funcionamiento, distribución y otros. En este Suplemento la visión se denomina "diseño" a fin de evitar confusión con "ingeniería" como término utilizado para todas las actividades.

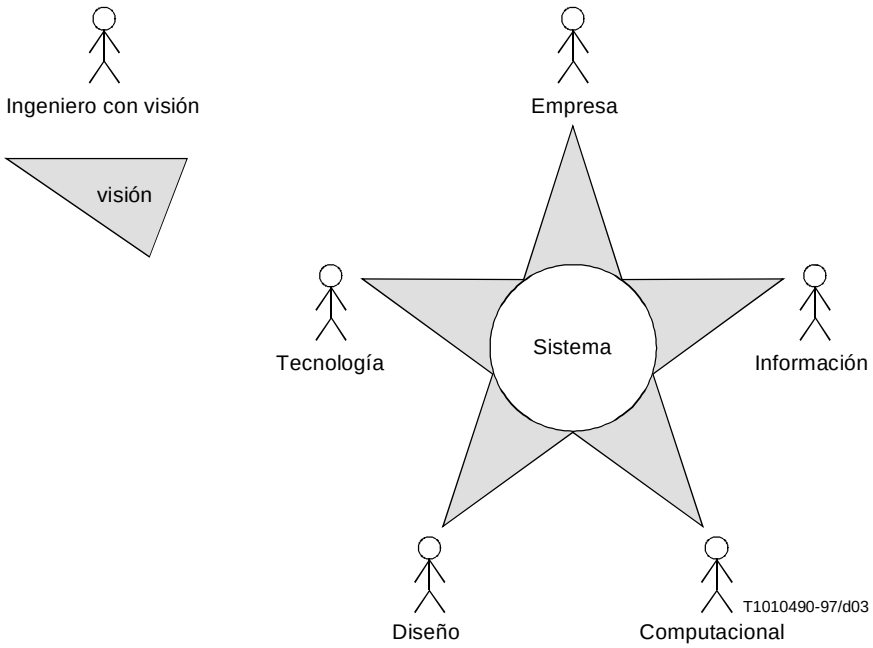


Figura 4-2/Sup. 1 a la Rec. Z.100 – Visiones ODP de un sistema

La visión de empresa revela las políticas generales, las acciones y los objetivos del sistema. Cuando se aplica en este Suplemento, la visión de empresa está relacionada con los motivos, el objetivo y la utilización de la aplicación. Esta información debiera incluirse en los requisitos que establecen la necesidad de la aplicación. La visión de empresa se expresa normalmente mediante lenguaje natural. Se consigue a través de los requisitos y debiera utilizarse en la descripción del campo de aplicación de la aplicación así como para la Validación del producto.

La visión de la información revela las estructuras de información, es decir, como se relacionan unos con otros los ítems de datos del sistema. Cuando se aplica este Suplemento, la visión de la información está relacionado con el lugar donde se almacenan los ítems de datos, los enlaces entre los ítems de datos así como la cardinalidad (es decir, la multiplicidad) de dichas relaciones (uno a uno, uno a muchos o muchos a muchos). Las salidas de las principales actividades de metodología de este Suplemento contienen visiones de la información del sistema.

La visión computacional revela como se procesan los ítems de datos en el sistema. Hace referencia a como se transfiere la información, como se consulta, se transforma y se gestiona. El estímulo para un cálculo puede provenir del propio sistema o ser exterior al mismo. Junto con la visión de la información, la visión computacional define el comportamiento del sistema. Cuando se aplica este Suplemento, la visión computacional se describe mediante procesos SDL utilizando datos que se definen en ASN.1 o SDL. El sistema se descompone de forma que pueda ser distribuido.

La visión de diseño revela como está organizado el comportamiento del sistema y como se encuentra distribuido a fin de tener en cuenta el entorno que lo soporta. Esta visión tiene en cuenta aspectos tales como las características de transmisión, la distribución y la concurrencia, pudiendo incluir también la calidad de funcionamiento y la fiabilidad. Frecuentemente, el diseño es un compromiso entre varios aspectos. Cuando se aplica este Suplemento, SDL+ puede incluir los requisitos de diseño sobre la distribución de funcionalidades y la concurrencia. Estos requisitos se reflejan en la estructura del SDL+ (bloques y procesos) y de la anotación. Si dichos aspectos son obligatorios para la aplicación, la estructura SDL debe cumplir los requisitos. En los restantes casos, el SDL debiera estar estructurado de forma tan clara como sea posible, al tiempo que facilite la prueba y la validación.

La visión de la tecnología está relacionado con los aspectos tecnológicos del sistema. Sólo es relevante si la descripción de la implementación difiere significativamente de la especificación SDL+. La metodología de este Suplemento no cubre este caso.

En resumen, la ingeniería de una aplicación implica diferentes visiones de un sistema, las cuales están contenidas en los distintos modelos del mismo. Durante el proceso de ingeniería, el sistema se construye gradualmente elaborando dichos modelos. Dado que se trata de modelos del mismo sistema, todos ellos están relacionados.

Cuando comienza la ingeniería de un sistema, el conocimiento que se tiene del mismo puede estar entre dos casos extremos:

– La comprensión del sistema es pobre y los requisitos se encuentran solo vagamente definidos. No hay disponible ni conocimiento ni experiencia sobre el sistema. No existen sistemas similares con los que los ingenieros puedan establecer analogías.

– El sistema se comprende cabalmente y está bien definido. Ya ha sido implementado y la tarea de ingeniería consiste en modificar un sistema existente. Existe experiencia, los cambios no presentan dificultades y las especificaciones del sistema existente han sido creadas en SDL+.

En el primer caso, gran parte del trabajo de ingeniería de un sistema consiste en entender cabalmente los requisitos del sistema y como funciona. En el segundo caso, el trabajo consiste en realizar cambios al SDL+ y a la documentación. Ello puede realizarse creando algunos nuevos tipos de SDL+, añadiéndolos a los tipos ya existentes o simplemente agrupando SDL+ existentes de una forma nueva a fin de crear un nuevo sistema.

La metodología que se presenta en este Suplemento es aplicable a la ingeniería de la definición del sistema de una aplicación, pero ignorando los detalles de la implementación. A esto se denomina la "especificación" de la aplicación. La ingeniería requerida es muy similar a la que se requiere para cualquier sistema y puede encontrarse en cualquier punto entre los dos extremos arriba mencionados. La metodología no cubre con detalle los aspectos relacionados con la implementación. Para simplificar el texto en el resto de este Suplemento el término "definición del sistema de la aplicación" se denomina "sistema".

La metodología se concentra en tres actividades principales:

– Análisis, que consiste en la organización de los requisitos y la identificación de los conceptos del sistema (con nombres y definiciones).

– Diseño Previo, que consiste en la transformación de la información clasificada en forma de diseños previos que cubren parte del sistema o son parcialmente formales.

– Formalización, que consiste en la expresión de la especificación en términos de SDL formal, complementada mediante MSC y ASN.1.

La metodología incluye de forma breve otras cuatro actividades. Una es la captura de requisitos, que se ocupa de la Recopilación de Requisitos al comienzo del proyecto. La segunda actividad es la Validación y se realiza sobre las especificaciones formalizadas conforme éstas se producen. La tercera actividad es la Documentación que trata de la selección de las especificaciones del sistema para el archivo y su potencial reutilización. La cuarta es la Implementación que trata de la generación del código ejecutable para el sistema objetivo. La figura 4-3 ilustra la metodología. La actividad de captura de requisitos sólo se cubre de forma parcial en este Suplemento; este Suplemento sólo describe la Recopilación de Requisitos. La figura 4-3 no recoge todos los flujos de información.

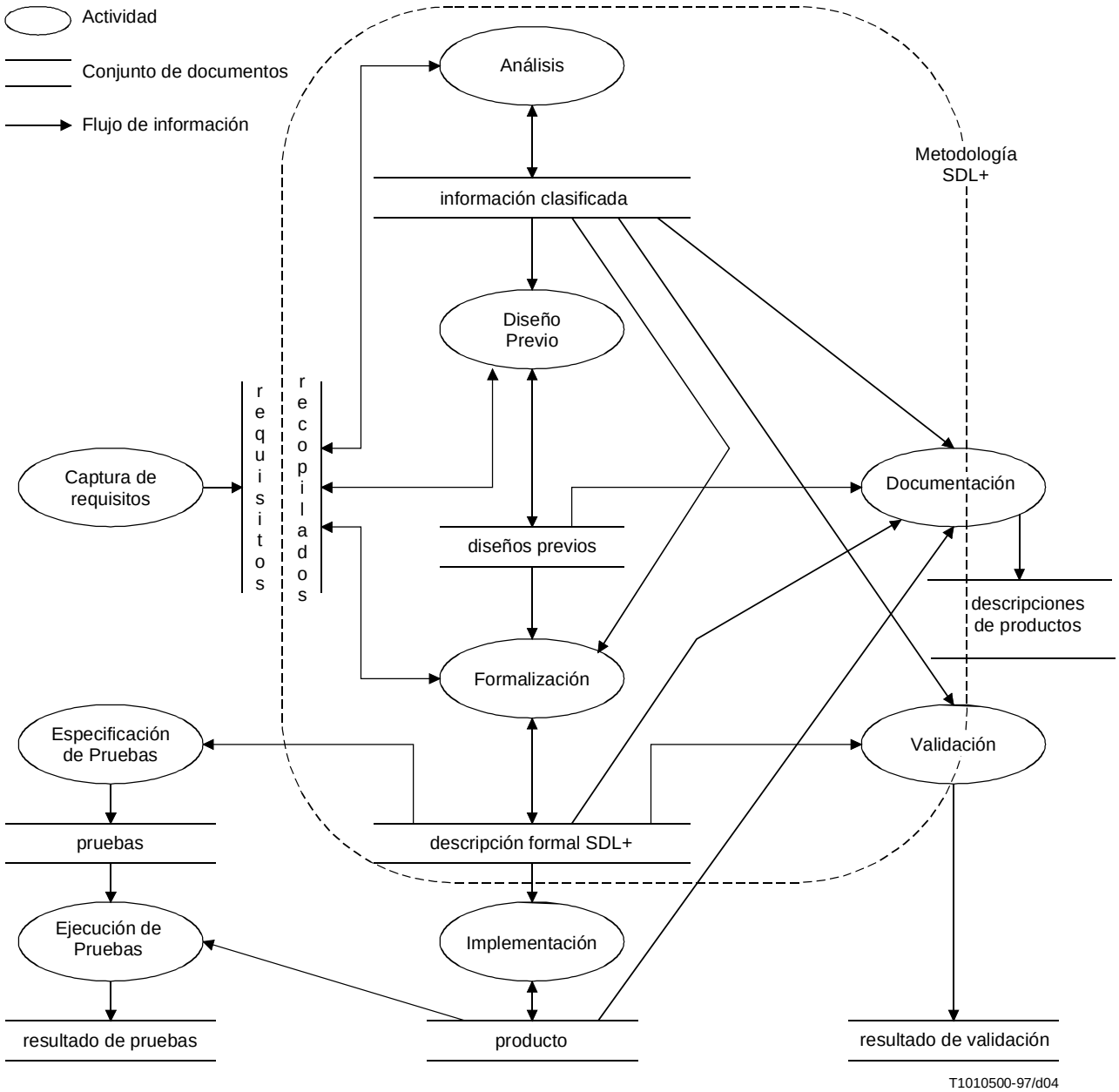


Figura 4-3/Sup. 1 a la Rec. Z.100 – Metodología SDL+

El conjunto de las siete actividades (Recopilación de Requisitos, Análisis, Diseño Previo, Formalización, Validación, Documentación e Implementación) son aplicables cuando el proyecto de ingeniería de un sistema se inicia con un conocimiento pobre del mismo. En la medida en que el sistema está bien definido, puede no ser necesario el Análisis ni el Diseño Previo. Los ingenieros que utilizan esta metodología evalúan en primer lugar el conocimiento y la definición que tienen del sistema, decidiendo entonces cual es la actividad inicial más conveniente de la metodología. Para facilitar esta elección, este Suplemento ofrece listas de criterios para iniciar y finalizar las tres actividades principales.

Las listas de criterios permiten también que los ingenieros sigan la metodología cuando los trabajos del proyecto de ingeniería de un sistema se solapan y son iterativos. Es decir, la captura de requisitos puede estar teniendo lugar al tiempo que se clasifican los requisitos existentes. A su vez, esta actividad puede no estar completada cuando se inicia el Diseño Previo. La Formalización puede comenzar en cuanto los ingenieros tengan una idea clara de las interfaces del sistema y se puede generar la documentación inicial de una especificación tan pronto como se clarifique la arquitectura general. Igualmente, a través de la formalización de una especificación en SDL+, los ingenieros pueden plantearse cuestiones que obliguen a reclasificar los requisitos o reestructurar la descripción producida durante el Diseño Previo.

Las subcláusulas siguientes describen brevemente las nueve actividades en las que se divide esta metodología. También se define en ellas la flexibilidad de que gozan los ingenieros para cambiar de una actividad a otra. El Análisis, el Diseño Previo y la Formalización se agrupan en una sola cláusula ya que están estrechamente relacionadas. La Validación, la Especificación de Prueba y la Ejecución de Pruebas también se agrupan debido a que numerosas técnicas de prueba son de utilidad para la Validación.

Las actividades de Análisis, Diseño Previo, Formalización, Implementación y Validación se desarrollan adicionalmente en las cláusulas 5, 6, 7, 8 y 9 respectivamente.

4.1 Parte Recopilación de Requisitos de la captura de requisitos

El objeto de la Recopilación de Requisitos es comprobar que los requisitos de la especificación están descritos de forma razonable. En la metodología no determina cómo deben crearse o recopilarse los requisitos; la actividad se inicia cuando se dispone de un conjunto de requisitos. La metodología supone que los requisitos recopilados contienen al menos la visión de empresa del sistema.

En esta actividad se dan unos criterios mínimos para aceptar los requisitos iniciales. También se proporcionan algunas directrices para la utilización de criterios de calidad a fin de determinar que ha finalizado la Recopilación de Requisitos. La Recopilación de Requisitos no tiene por qué detenerse por el hecho de comience otra actividad. Algunos requisitos deben ser refinados como consecuencia de una mejor comprensión del problema. La alteración de las circunstancias puede dar lugar a cambios en los requisitos, e igualmente otras actividades pueden generar cuestiones que conduzcan a la definición de nuevos requisitos o al cambio de los requisitos existentes. La figura 4-3 muestra la realimentación de otras actividades sobre los requisitos recopilados. No se muestra el flujo de información para resolver cuestiones a través de la actividad de captura de requisitos.

Aunque no se define aquí, la generación efectiva de requisitos mediante la investigación, ingeniería del conocimiento u otros métodos tiene lugar antes de que comience el desarrollo. Además, la metodología asume que siempre que los ingenieros se hagan alguna pregunta a cerca de los requisitos, la respuesta o las decisiones tendrán su origen en esta misma actividad. Las cuestiones sobre la compleción, consistencia y otros atributos de calidad de los requisitos pueden dar lugar a cambios en los requisitos existentes y a la adición de otros nuevos. Cuando mediante la metodología no se resuelven las cuestiones que surgen, es necesario utilizar la parte de la actividad de captura de requisitos externa a la metodología.

Se muestra a continuación una lista de criterios mínimos que debieran cumplir los requisitos iniciales. Si los requisitos iniciales no son adecuados, la solución debe identificarse a través de actividades de captura de requisitos ajenas al dominio de esta metodología.

1) Si hay muchos requisitos iniciales o si éstos son complejos, ¿se ha proporcionado un resumen de los requisitos?

2) ¿Contienen los requisitos iniciales una declaración del objetivo del sistema?

3) ¿Describen los requisitos iniciales la funcionalidad del sistema?

4) ¿Es factible la implementación a la luz del estado actual de la tecnología o del desarrollo probable de la misma?

La determinación de si los requisitos recopilados son satisfactorios es algo subjetivo pues éstos no están (normalmente) en una forma tal que permitan una apreciación objetiva. La lista de comprobación para juzgar si se ha completado la Recopilación de Requisitos es la siguiente:

1) debiera existir la convicción de que los requisitos recopilados son inequívocos, son completos y que su consecución es factible (estos atributos pueden ulteriormente resultar equivocados);

2) no debiera existir indicio alguno de que los requisitos son incorrectos, inverificables, internamente o externamente inconsistentes (ello no garantiza que sean correctos, verificables o consistentes, pero evita que se continúe si se conoce que alguno de ellos es falso);

3) se debe estimar que a partir de los requisitos recopilados es posible realizar una descripción formal SDL+.

4.2 Análisis, Diseño Previo y Formalización

En esta metodología, el modelado de los requisitos se realiza en tres actividades. La información clasificada incluye la visión de empresa y otras visiones de los requisitos recopilados. La información clasificada constituye un modelo en el que los requisitos recopilados se estructuran y definen según conceptos (con nombres y definiciones). Durante el Diseño Previo se realiza un refinamiento ulterior para describir partes del sistema desde la visión de la información y partes del sistema de manera informal desde la visión computacional. Las descripciones se transforman durante la Formalización en una especificación formal que proporciona una descripción precisa y completa del sistema incluyendo las visiones de la información, computacional y (si es necesario) de diseño.

Análisis

Los requisitos recopilados pueden estructurarse en base a una estructura de Análisis de los conceptos utilizada en productos anteriores o de acuerdo con una nueva estructura expresamente pensada para el sistema que se especifica. Ésta se convierte así en la información clasificada. La información clasificada se caracteriza porque los requisitos recopilados (normalmente expresados en lenguaje natural y diagramas informales) se estructuran como con un modelo de conceptos (con nombres y definiciones).

No hay un conjunto fijo de conceptos ni una estructura fija. Cuando se acomete una nueva aplicación, se busca en la biblioteca de reutilización la posible existencia de registros de otros sistemas con conceptos y estructuras similares.

El resultado es un información más estructurada y una terminología definida. Se clarifican y amplían las ideas, ya que el ingeniero debe profundizar en el proceso y hacer referencia a los requisitos recopilados.

La notación utilizada para estructurar la información se elige con el fin de proporcionar formas adecuadas de registro de los conceptos y estructura inherentes de los requisitos. Debido a que los requisitos contienen a menudo descripciones de secuencias de uso o que dichas secuencias se obtienen como resultado de las cuestiones surgidas, los gráficos de secuencias de mensajes (MSC) pueden constituir una buena herramienta para parte del comportamiento durante el Análisis.

Diseño Previo

La información clasificada informal (y/o los requisitos recopilados) se convierte en diseños previos mediante notaciones que tienen una estructura formal y normalmente una sintaxis formal. La principal característica de los diseños previos es que constituyen borradores incompletos del sistema. El examen y la determinación minuciosa del comportamiento del sistema no es esencial debido a que el objetivo es investigar posibles diseños; por lo tanto, la semántica de las notaciones utilizadas puede ser informal y los diseños previos pueden permitir diversos comportamientos. El resultado es que el significado depende de la interpretación de las palabras utilizadas y del conocimiento compartido entre los ingenieros que utilizan los documentos. Si bien esto es a menudo adecuado desde la visión de empresa y quizás desde la visión de la información, dichas interpretaciones no son suficientes desde una visión computacional completa y precisa.

Las notaciones empleadas pueden elegirse de forma que se adapten al sistema concreto, al lenguaje de especificación formal a utilizar (SDL+) y a los recursos disponibles (por ejemplo, la experiencia, el esfuerzo y las herramientas). La utilización informal de SDL+ es una notación adecuada de Diseño Previo. Siempre se utilizan gráficos de secuencias de mensajes (MSC). La ASN.1 es de utilidad bien porque los requisitos recopilados ya incluyen ASN.1 o porque es una técnica eficaz para describir la visión de la información de un sistema. Puede utilizarse una notación de modelo de clase de objeto (preferentemente la misma que se utilice en el Análisis) para modelar las entidades del sistema y las relaciones entre ellas.

Del Diseño Previo surgen a menudo mejoras y correcciones a los conceptos del sistema. Se trata de requisitos adicionales y, por lo tanto, en la figura 4-3 se muestran como una realimentación que se realiza a través de los requisitos recopilados.

Formalización

La descripción formal SDL+ se obtiene a partir de los requisitos recopilados, la información clasificada y los diseños previos. Durante la creación de la descripción formal SDL+, puede considerarse que los diagramas SDL constituyen un Diseño Previo ya que a menudo contienen partes informales o están incompletos según las reglas del lenguaje SDL. No obstante, estas descripciones intermedias proporcionan visiones útiles del sistema y ayudan a mejorar su comprensión.

En algunas ocasiones hay más de una descripción formal SDL+. En primer lugar se produce una descripción abstracta como descripción formal de máximo nivel, siendo ulteriormente refinada hasta el nivel que la aplicación requiere. Puede realizarse un refinamiento ulterior a fin de generar una descripción que pueda interpretarse con fines de Validación. Dicho refinamiento es equivalente a producir una implementación a partir de una especificación en SDL+.

La descripción formal SDL+ proporciona una visión computacional precisa del sistema y (si es necesario) una visión de diseño. Para ser una descripción formal no debiera contener "texto informal" (en el sentido SDL), de forma que el comportamiento sólo venga determinado por la semántica formal SDL+ y no dependa de la interpretación humana.

No es necesario producir la información clasificada si los conceptos involucrados se entienden cabalmente y éstos están bien documentados. Sin embargo, cuando se inicia el trabajo relativo a los conceptos, surgen a menudo las interpretaciones erróneas y es necesario producir la información clasificada. También es posible que no sea necesario producir los diseños previos si se entiende cabalmente el comportamiento del sistema. En este caso, la descripción formal SDL+ puede escribirse directamente utilizando los requisitos recopilados y siguiendo los pasos de la Formalización; se sustituye así la utilización del conocimiento y de la experiencia por la información clasificada y los diseños previos. Este enfoque presenta ventajas de economía para un ingeniero o un sistema en particular, pero por contra, no genera registros que puedan ser de utilidad ulteriormente para otro ingeniero u otro sistema. En la mayoría de los casos es preferible comenzar con un esquema elemental, de forma que los enfoques erróneos puedan ser descartados antes de que se haya desarrollado un trabajo en la Formalización significativo. En este caso, la opción de disponer de un texto informal en SDL durante el Diseño Previo es una característica muy valiosa del lenguaje SDL.

El resultado completo del enfoque contiene los tres niveles de la descripción, en contraste con la descripción formal SDL que solamente es parte de ella. El resultado completo contiene todo lo que se conoce sobre el sistema.

4.3 Validación y Prueba

A las dos actividades de prueba (Prueba de Especificación y Prueba de Ejecución) se las denomina conjuntamente como *Prueba*.

La diferencia entre Validación y Prueba estriba en cuales son los elementos que se comparan entre sí. En el caso de la Validación, la descripción SDL+ se compara con el modelo de información clasificada que produce el Análisis, mientras que en el caso de la Prueba, la principal comparación se realiza entre la descripción SDL+ y un producto implementado. Tanto la Prueba como la Validación tienen por objetivo determinar que la aplicación tiene las características deseadas, existiendo asimismo entradas a la Validación y a la Prueba que proceden de los requisitos recopilados (la figura 4-3 sólo muestra los flujos de información principales).

En esta metodología la Validación incluye todas las acciones relacionadas con la comprobación de la "validez" o el "valor" de las definiciones SDL+. La Validación tiene dos aspectos: la evaluación de la conformidad con las normas y otros criterios para la aplicación, y la evaluación de que se ha cumplido el objetivo (incluyendo la funcionalidad y la calidad de funcionamiento) expresado en los requisitos. En ambos casos, la Validación puede demostrar que una definición SDL+ es no válida, pero si ninguna de las pruebas fracasa, deberá juzgarse si el SDL+ es válido. No se utiliza el término "verificación" (que queda reservado para la prueba de la certeza de una relación entre dos modelos) no es utilizado.

En la actividad de Validación se utiliza un modelo de validación formal. Ésta puede ser la definición SDL+ que pretende ser validada o un modelo que se deriva de ésta y que incluye modelos de partes del entorno. El modelo de validación es utilizado para la validación formal: investigación sistemática de la validez, como es la comprobación de las reglas del lenguaje SDL+ y la aplicación de casos de prueba. También pueden aplicarse técnicas de validación informal, tales como la vigilancia e inspección con una lista de comprobación, y normalmente son necesarios para evaluar si se ha cumplido el objetivo deseado.

La descripción formal SDL+ se utiliza como base de un modelo de validación. Debido a que algunas construcciones SDL permiten definir sistemas que pueden ser difíciles e incluso imposibles de validar, puede ser conveniente introducir algunas limitaciones en el SDL+ utilizado en la Formalización, de manera que el sistema pueda ser validado. En función de los requisitos de la Validación, puede ser necesario hacer adiciones a la descripción formal SDL+. Por ejemplo, para poder validar el comportamiento del sistema para el caso de máximo número de procesos, el modelo de validación puede restringir el número de instancias simultáneas de un proceso SDL a un número inferior de lo que lo hace la descripción formal SDL+.

La obtención de casos de prueba para el modelo de validación es similar a la Especificación de Prueba, excepto en que los objetivos de las pruebas son diferentes. La aplicación de casos de prueba al modelo de validación es semejante a la Ejecución de Prueba. De hecho, son muchas las pruebas y las herramientas que pueden utilizarse tanto para la prueba del modelo de validación y para la prueba de un producto.

La forma más común de prueba es la *prueba de conformidad*: evaluación mediante pruebas sobre si un producto cumple su especificación. La prueba de conformidad es un elemento importante en el desarrollo del producto ya que aumenta la confiabilidad en un correcto interfuncionamiento de sistemas abiertos, en los que la conformidad con un protocolo de comunicaciones o con la especificación de un servicio constituye un requisito previo. La Recomendación Z.500 – Marco para métodos formales en pruebas de conformidad [9] – define el significado de conformidad en caso de que se utilice un método formal tal como el SDL+ para la especificación de un protocolo de comunicaciones o de un servicio y proporciona directrices para la generación de pruebas asistidas por computadora.

La Recomendación Z.500 define un marco para la utilización de métodos formales en la pruebas de conformidad. Está destinada a los responsables de las implementaciones, pruebas y especificaciones involucrados en pruebas de conformidad a fin de disponer de directrices para la definición de la conformidad y el proceso de pruebas de una implementación con respecto a una especificación que constituye la descripción formal. Es aplicable cuando se dispone de la especificación formal de un protocolo de comunicaciones o de un servicio para los que debe desarrollar un conjunto de pruebas de conformidad. Puede aportar directrices para los procesos de este Suplemento así como para el desarrollo de herramientas para la generación de casos de pruebas asistidas por computadora. La Recomendación Z.500 define un marco y no especifica la utilización de un método de generación de pruebas concreto ni ninguna relación de conformidad específica entre una especificación formal y una implementación. Complementa a las Recomendaciones X.290 – X.294, Metodología y marco para las pruebas de conformidad OSI para Recomendaciones sobre protocolos para aplicaciones CCITT [10], que se aplican a una amplia gama de productos y de especificaciones, incluyendo las especificaciones en lenguaje natural. La Recomendación Z.500 interpreta los conceptos de las pruebas de conformidad en un contexto formal.

Los métodos para otras formas de prueba, tales como las pruebas de interfuncionamiento puede utilizar como base los métodos de las pruebas de conformidad.

En este Suplemento no se hace ninguna descripción ulterior de las pruebas. Para más información véanse [9] y [10].

4.4 Documentación

Las actividades de Análisis, Diseño Previo y Formalización generan textos y diagramas, algunos de los cuales son necesarios para la especificación del producto. El objetivo de la actividad de Documentación es seleccionar las descripciones esenciales de los documentos y organizarlas de forma que la especificación de la aplicación sea fácil de entender, concisa y precisa. Si la especificación forma parte de una norma o de un documento de compra, esta actividad cobra una particular importancia.

La descripción formal SDL+ se incluye siempre en la descripción del producto. Si se ha elaborado más de una especificación formal SDL, sólo debiera incorporarse una, quedando las demás claramente identificadas como abstracciones del modelo final. Con frecuencia se necesitan otras descripciones para entender cabalmente la descripción formal SDL:

– diseños previos tales como los gráficos de secuencias de mensaje (MSC);

– información estructurada y conceptos de la información clasificada;

– texto informal y diagramas de los requisitos recopilados;

– expresiones lógicas sobre el estado del sistema;

– texto informal y diagramas adicionales que no se producen según esta metodología.

En general, la Validación se ve facilitada si las especificaciones del comportamiento se soportan mediante información estática redundante, tales como expresiones lógicas ligadas a los estados del sistema.

La descripción del producto se estructura de acuerdo a un conjunto de principios.

4.5 Paralelismo de actividades

Una actividad no puede comenzar hasta que se satisfacen los criterios para el comienzo de la misma. Una actividad se completa cuando se satisfacen los criterios necesarios a tal fin. Si no existen limitaciones (tales como disponibilidad de recursos, herramientas o experiencia), dichas actividades pueden realizarse en paralelo. Ello implica que todas las actividades pueden progresar en paralelo. En la práctica eso es lo que ocurre debido a que los requisitos o el contexto de un sistema suele cambiar conforme el sistema se desarrolla, por lo que deben analizarse nuevos requisitos y tienen que modificarse diseños previos al tiempo que se realiza la Formalización. Es también habitual dividir la ingeniería de un sistema en varias partes, las cuales pueden encontrarse en distintas etapas de desarrollo. Ello puede deberse a limitaciones en los recursos o porque determinadas partes críticas del sistema se desarrollan en primer lugar para determinar la viabilidad o coste de un diseño en particular. Por lo tanto, la metodología no requiere que se complete una actividad antes de que comience otra, sino que se cumplan los criterios que permiten el comienzo de las mismas.

5 Actividad de Análisis

El Análisis es un enfoque que se realiza paso a paso a fin de explorar los requisitos recopilados, organizando la información que contienen y registrar esta información en un formato que refleje dicha organización. El Análisis debe utilizarse cuando se tiene un escaso conocimiento del dominio de aplicación o del sistema que debe ser especificado, cuando los requisitos recopilados contienen varias descripciones del mismo concepto de aplicación o para dar a los conceptos de aplicación nombres bien definidos.

Durante el Análisis, la información de los requisitos recopilados se organiza en función de los conceptos de aplicación. Es probable que algunos de dichos conceptos de aplicación, tales como "primitiva de servicio" y "unidad de datos del protocolo" se hayan definido con anterioridad y sean bien conocidos. Es natural relacionar parte de la información de los requisitos recopilados con dichos conceptos de aplicación básicos. También se identifican y se designan nuevos conceptos de aplicación que a menudo se definen en función de otros ya existentes y que constituyen información específica del sistema. El conjunto de conceptos de aplicación y las relaciones que se establecen entre ellos permite a los ingenieros profundizar en el sistema, intercambiar ideas y alcanzar una mejor comprensión del mismo. Un método orientado a objetos está bien adaptado a este proceso de modelado de conceptos de aplicación.

La información clasificada que genera esta actividad proporciona una base para otras actividades metodológicas, en particular para la creación de la especificación formal SDL+.

La actividad de Análisis aquí descrita es un enfoque general, pero para realizar el Análisis de forma efectiva en sistemas de un cierto tamaño es preciso utilizar notaciones definidas y las correspondientes herramientas. Las técnicas adecuadas para ello se denominan normalmente técnicas de "análisis orientado a objetos" tales como (en orden alfabético) OMT [11], OOSE [12], o SOON [6]. En la parte II de la cláusula 13 de este Suplemento, se elabora el Análisis utilizando uno de dichos enfoques elegido de forma arbitraria.

5.1 Inicio del Análisis

Al inicio del Análisis se revisan los objetivos, la información disponible y las alternativas disponibles.

5.1.1 Objetivos del Análisis

Los objetivos del Análisis son los siguientes:

– resolver cualquier deficiencia que pueda haberse detectado en los requisitos recopilados;

– establecer las bases para el Diseño Previo y la Formalización;

– identificar, nombrar y definir los conceptos de aplicación del sistema;

– producir la información clasificada que constituye aún una especificación informal que estructura la información de los requisitos recopilados y que soporta el ulterior desarrollo, operación y mantenimiento del sistema.

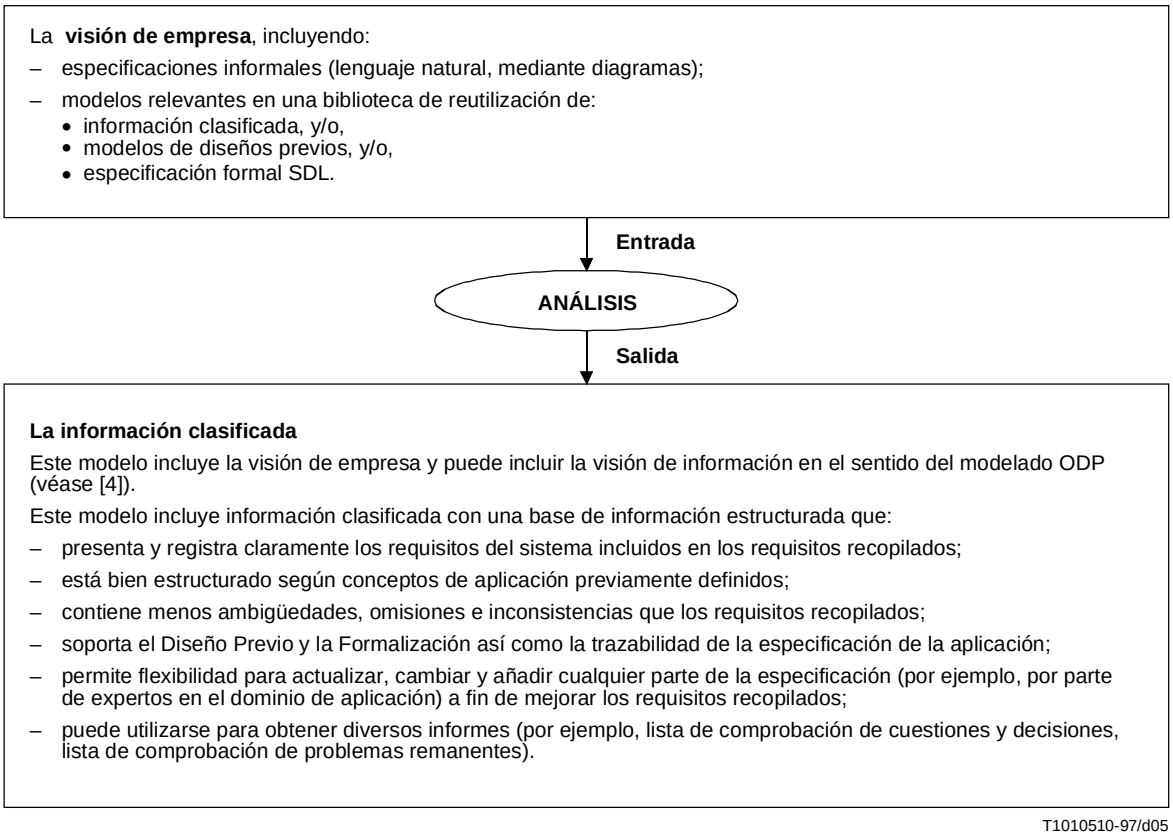


Figura 5-1/Sup. 1 a la Rec. Z.100 – Entradas y salidas del Análisis

5.1.2 Determinación de la posible omisión del Análisis

NOTA 1 – Esta subcláusula define cómo evaluar la información y el conocimiento de que dispone el ingeniero antes de que comience el Análisis. El Análisis puede no ser necesario para el desarrollo de un sistema. En ese caso, el desarrollo de la especificación pasa directamente al Diseño Previo o a la Formalización.

NOTA 2 – La metodología no impone una secuencia estricta o cuando debe realizarse la actividad de Análisis. La metodología sólo da al ingeniero una recomendación sobre cómo y cuándo hacer uso de la actividad Análisis.

Antes de iniciar el Análisis, el ingeniero debe juzgar la adecuación de los conceptos de aplicación de los requisitos recopilados y el grado de compresión que sobre los mismos tiene. Decide entonces si los requisitos recopilados están descritos al nivel de detalle adecuado para el Análisis, el Diseño Previo o la Formalización.

Cuando se satisfacen los requisitos siguientes significa que los requisitos recopilados están bien estructurados y que puede omitirse la actividad de Análisis:

1) no existe la misma funcionalidad ni los mismos datos en distintas ubicaciones;

2) es fácil extraer las características esenciales de cada concepto de aplicación para distinguirlo de los restantes y existe un término bien definido y exclusivo para cada concepto de aplicación;

3) los detalles se ocultan cuando no forman parte de las características esenciales de los conceptos de aplicación;

4) los conceptos de aplicación se dividen en un conjunto de partes coherentes que pueden refinarse por separado;

5) los conceptos de aplicación están bien ordenados. Es importante agrupar datos y funciones similares y descomponer de forma lógica los datos en partes más pequeñas.

5.2 Cuestiones durante el Análisis

La flecha que en la figura 4-3 va desde el Análisis a los requisitos recopilados representa las cuestiones que surgen durante el desarrollo de una aplicación. Las cuestiones sobre la compleción, consistencia y otros atributos de calidad de los requisitos pueden dar lugar a cambios en los requisitos existentes, así como la adición de nuevos requisitos al conjunto. Cuando dichas cuestiones no pueden resolverse en el marco de esta metodología, deben ser resueltas por las partes de la actividad de captura de requisitos ajenas a la misma.

5.3 Enfoque del modelado para el Análisis

Antes del Análisis, las ideas sobre el sistema pueden ser:

– limitadas: el ingeniero conoce un número limitado de conceptos de aplicación;

– arbitrarias: los ingenieros crean y clasifican conceptos de aplicación en base a sus diferentes conocimientos;

– específicas: cada ingeniero tiene en mente su propio modelo en base a sus propias experiencias anteriores;

– complejas: cada ingeniero considera una serie de visiones en su modelo.

El Análisis utiliza un método orientado a objetos para resolver estos problemas. Este método ayuda al ingeniero a:

– proporcionar un modelo estructurado de base modular y por lo tanto estable y que acepta con más facilidad las posibles modificaciones (que solo afectan a algunos de los objetos considerados en el modelo);

– dividir la complejidad del problema en partes más pequeñas que se consideran por separado;

– hacer que todos los ingenieros que participan en el proyecto tengan una visión común de la especificación;

– controlar las posibilidades de error, ambigüedades e inconsistencias;

– realizar un seguimiento del trabajo.

En conclusión, para definir el sistema los ingenieros tienen un modelo que consiste en:

– una visión lógica: para describir elementos claves del sistema que debe especificarse;

– una visión dinámica: para describir el comportamiento individual de un objeto, así como el comportamiento de todos los objetos.

La visión lógica se expresa mediante una notación de clase de objeto adecuada. La visión dinámica se expresa normalmente en MSC.

5.4 Pasos del Análisis

La actividad de Análisis consta de dos pasos: inspección y clasificación.

5.4.1 Inspección

Durante la inspección se supervisa de forma sistemática el contenido de los requisitos recopilados. La inspección permite al ingeniero calibrar si el conocimiento o experiencia sobre el dominio de aplicación es suficiente y, por lo tanto, si la especificación es factible. Dado que la investigación sobre un domino de aplicación poco conocido puede ser costosa, es conveniente identificar la necesidad de dicha investigación en la fase más temprana que sea posible del proceso de especificación.

Los objetivos de la inspección son los siguientes:

– obtener una idea general del dominio de aplicación;

– evaluar la relevancia de los requisitos recopilados para el dominio de aplicación;

– determinar si la cobertura que del dominio de aplicación hacen los requisitos recopilados es completa o escasa.

Los pasos en que se realiza la inspección son los siguientes:

1) Recopilar el material fuente:

   a) compilar una lista de todas las fuentes de información. Puede bastar con una bibliografía en forma de cuadro y preparada con un procesador de textos;

   b) categorizar cada documento en función de su importancia aparente para el dominio de aplicación.

2) Inspeccionar cada documento, comenzando por el más relevante.

3) Reconsiderar la relevancia de cada documento para el domino de aplicación.

4) Leer cuidadosamente el documento más relevante, pero sin detenerse para investigar pasajes del mismo que sean difíciles o de ardua comprensión.

5.4.2 Clasificación

La clasificación consiste en:

– identificar los conceptos de aplicación, su estructura y sus diversos aspectos presentes en los requisitos recopilados;

– identificar la redundancia y la pérdida de información de los requisitos recopilados;

– establecer conexiones entre las entradas al Análisis y la información clasificada.

La clasificación conduce a la elaboración de un modelo orientado a objetos lógico del sistema que describe:

– la estructura de las clases identificadas;

– la estructura de los objetos identificados de las clases;

– los aspectos asociados con cada objeto de una clase (aspectos de información, aspectos de comportamiento, aspectos de interfaces y aspectos varios).

El objetivo es identificar clases y objetos con un cierto nivel de detalle y asegurar que se designan y se definen adecuadamente.

La clasificación consta de cinco pasos:

1) Identificar y nombrar las partes del sistema que debe clasificarse.

2) Identificar y nombrar las clases.

3) Identificar las estructuras de los objetos y sus relaciones.

4) Definir clases y objetos y revisar su denominación.

5) Identificar los aspectos de cada clase.

Los aspectos del comportamiento de los objetos pueden recogerse mediante MSC. Debería haber un objeto (compuesto) que represente el sistema y uno o más objetos para el entorno. Puede haber varios casos de secuencias en MSC que muestren la utilización del sistema.

5.5 Conclusión del Análisis

Los resultados deben revisarse para determinar si se ha completado el Análisis; es decir, se aplica el criterio para determinar si debe omitirse el Análisis. El Diseño Previo y/o la Formalización pueden comenzar antes de que se decida si se ha realizado suficiente Análisis. Por lo tanto, los criterios para finalizar el Análisis son distintos a los empleados para comenzar el Diseño Previo o la Formalización.

La información clasificada se considera una base de información estructurada. Debe existir una lista de comprobación para determinar si la información clasificada está suficientemente estructurada en objetos y si éstos han sido convenientemente denominados y definidos para que se completen las actividades de Diseño Previo y Formalización. La lista de comprobación permite también la detección y tratamiento de posibles inconsistencias y falta de compleción en la investigación de los requisitos recopilados.

6 Diseño Previo

El objetivo general del Diseño Previo de un sistema es realizar la ingeniería parcial de las especificaciones informales desde distintos puntos de vista y con distintos niveles de detalle. El Diseño Previo no precisa ser completo o formal de manera estricta, sino que debe soportar la investigación de diferentes diseños para las distintas partes del sistema de forma que se faciliten las decisiones sobre las alternativas del diseño.

Se utilizan los requisitos recopilados y la información clasificada, el conocimiento adquirido durante el Análisis y, si ello es posible, los modelos encontrados en la biblioteca de reutilización.

La figura 6-1 describe las entradas y salidas de la actividad de Diseño Previo.

El ingeniero utiliza y trabaja en detalle sobre los modelos existentes para producir diseños previos que proporcionan una visión de la información más formal y esbocen visiones computacionales del sistema. También pueden incluirse esbozos de visiones de diseño si ello es necesario. Los diseños previos permiten investigar selectivamente el comportamiento del sistema antes de que a través de la actividad de Formalización se formalice sistemáticamente el mismo. En buena medida, el Diseño Previo y la Formalización pueden tener lugar en paralelo, en la medida en la que los diseños previos necesarios para la Formalización se han realizado antes de que sean necesarios en los distintos pasos de la Formalización. Por ejemplo, los diseños previos producen normalmente secuencias de uso MSC que pueden ser utilizadas cuando se elaboran los procesos esquemáticos de la Formalización.

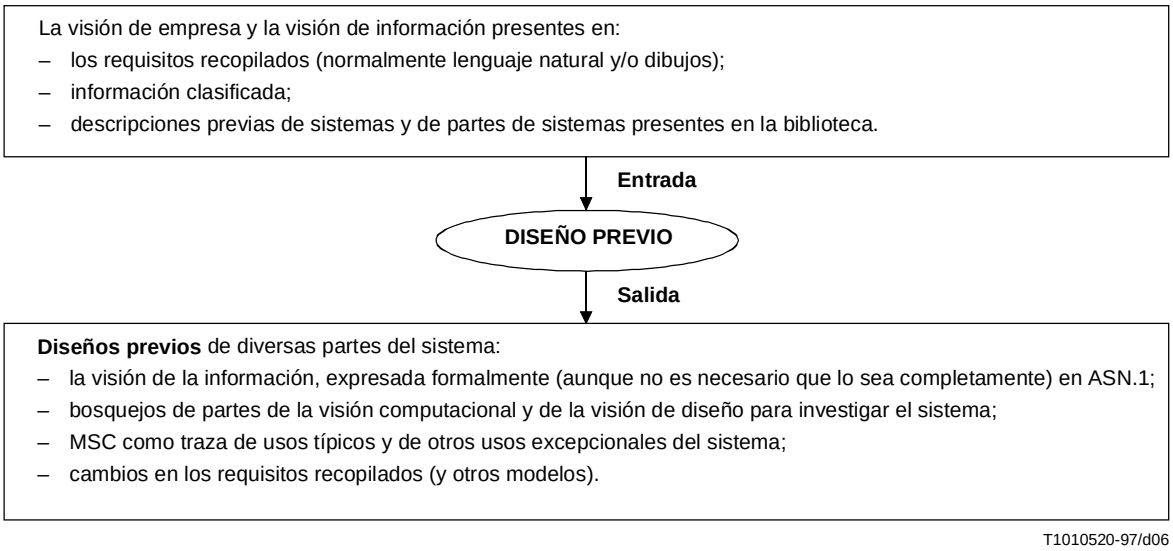


Figura 6-1/Sup. 1 a la Rec. Z.100 – Entradas y salidas del Diseño Previo

Los requisitos recopilados se utilizan de distinta forma en el Diseño Previo y en el Análisis:

– el Análisis se centra en los nombres (¿cuáles son los objetos identificados?) de los requisitos recopilados a fin de estructurar y definir conceptos para generar la visión de empresa y la visión de información de manera informal;

– el Diseño Previo se centra en los verbos (¿qué pueden hacer los objetos?) de los requisitos recopilados (o los verbos que hay en los aspectos de comportamiento de la información clasificada) para obtener la visión computacional y, si es necesario, una visión de diseño de manera formal.

La información clasificada utilizada en el Diseño Previo es el resultado del Análisis o, si éste no hubiera sido necesario, de las descripciones equivalentes presentes en los requisitos recopilados. Aunque puedan utilizarse otros lenguajes o técnicas para la estructuración del Análisis, lo más probable es que el contenido de la estructura sea lenguaje natural. El Diseño Previo desarrolla en detalle y de manera formal la visión de la información presente en la información clasificada, de forma que los datos puedan utilizarse para sustentar el comportamiento. En general, la visión de la información que produce el Diseño Previo es formal y se recomienda que ésta se defina utilizando ASN.1.

Debido a que el objetivo último de la actividad de Formalización es producir un modelo y una descripción utilizando SDL+, el Diseño Previo utiliza SDL+ siempre que sea posible. La diferencia esencial entre un Diseño Previo en SDL+ y un modelo formal es su cobertura y grado de compleción. A los efectos del Diseño Previo es correcto considerar sólo casos típicos y partes limitadas del sistema, asumir que algunos operadores de datos están bien definidos e ignorar algunas de las reglas impuestas por los lenguajes formales. Un Diseño Previo constituye un esbozo de la especificación, a diferencia de un modelo formal que debe ser preciso, inequívoco e interpretable.

Para que se pueda disponer de una traza de todos los modelos producidos, los diseños previos está ligados a la información clasificada y a los requisitos recopilados.

6.1 Inicio del Diseño Previo

Debido a que existen numerosas razones para producir diseños previos, los objetivos del Diseño Previo deben revisarse y ser registrados. Los objetivos registrados se convierten en criterios de la lista de comprobación para un adecuado Diseño Previo.

Los objetivos habituales son uno o varios de los siguientes:

1) Proporcionar una visión general mediante diagramas del sistema en forma de documentos de trabajo para los ingenieros involucrados.

2) Identificar las partes críticas del sistema.

3) Investigar la viabilidad de las partes críticas del sistema.

4) Determinar un conjunto típico de secuencias de uso.

5) Identificar las partes del comportamiento necesarias para sustentar el servicio o protocolo.

6) Determinar las partes del comportamiento necesarias (partes funcionales del comportamiento que son normativas).

7) Esbozar un modelo incluyendo las partes normativas y no normativas, donde las no normativas son normalmente añadidas de forma que el modelo pueda ser interpretado y el comportamiento pueda ser analizado.

8) Clarificar e identificar las interfaces normativas y no normativas tanto en los límites del sistema descrito como dentro del mismo.

9) Identificar los criterios de búsqueda para consultar las partes reutilizables (en SDL+) de la librería de reutilización.

Un objetivo que siempre se aplica es la identificación de ítems perdidos, las inconsistencias y las ambigüedades de la información clasificada o de los requisitos recopilados. Las cuestiones, las respuestas y los elementos que deben ser considerados ulteriormente y que están relacionados con la información clasificada se registran durante el Diseño Previo. Éste puede producir cambios en la información clasificada cuando se identifica un error de ingeniería y en los requisitos recopilados cuando éstos deban modificarse.

6.1.1 Determinación de la posible omisión del Diseño Previo

NOTA – Esta subcláusula determina cómo evaluar la información de que dispone el ingeniero y el conocimiento del sistema que éste tiene antes de comenzar el Diseño Previo. El Diseño Previo puede ser innecesario para el desarrollo de un sistema. Si así es, el desarrollo pasa directamente a la fase de Formalización, aunque en este caso se pueden invocar desde la Formalización, cuando sea necesario, los pasos necesarios para producir ASN.1 y MSC.

Antes de comenzar el Diseño Previo, se evalúa la información clasificada para determinar si es posible omitirlo y comenzar la Formalización. El Diseño Previo puede no ser necesario si se cumplen varias de las condiciones siguientes:

1) la categoría del sistema es bien conocida y entendida por los ingenieros;

2) los ingenieros tienen experiencia en SDL+;

3) la información clasificada tiene una relación clara y directa con las interfaces del sistema y una estructura de bloque en SDL;

4) la información clasificada contiene ya diseños previos en SDL+, al menos para casos típicos;

5) en la información clasificada ya existe una visión de la información expresada en ASN.1;

6) no se esperan dificultades relativas a la disposición en secuencia o la estructuración en SDL+;

7) no se espera que se tengan que modelar múltiples niveles de abstracción antes de que se produzca la descripción formal SDL+;

8) la arquitectura en la que se debe integrar el SDL+ no debe imponer limitaciones o constricciones que deban ser investigadas;

9) no existen aspectos relativos a la viabilidad que dependan del modelo de la máquina de estados finitos SDL+.

6.1.2 Criterios para iniciar el Diseño Previo

Antes de iniciar el Diseño Previo debe haber información clasificada que:

1) incluya una visión de empresa del sistema;

2) describa conceptos y nombres para algunas partes del sistema;

3) ofrezca una visión de información de las principales partes del sistema.

6.1.3 Notaciones del Diseño Previo

Las notaciones seleccionadas dependerán de la técnica de análisis orientada a objetos que se utilice para el Análisis (si éste no se ha omitido) o la forma de la información de entrada. Como notación para el Diseño Previo se propone SDL+ (utilizada informalmente) así como la notación utilizada para clases y objetos a fin de expresar con más detalle las relaciones entre entidades.

Las notaciones seleccionadas debieran proporcionar una sólida base para la Validación y especificación de las pruebas de conformidad. Las notaciones debieran elegirse para ser útiles en la Formalización. Por este motivo, se debería utilizar SDL+ siempre que sea posible y evitar introducir cualquier notación diferente a SDL+ o a la utilizada en la entrada.

6.2 Pasos del Diseño Previo

Los pasos generales del Diseño Previo son los siguientes:

1) Realizar un modelo del contexto en el que el sistema está identificado y donde se detalla el entorno del mismo y describir las interfaces de comunicación y otras relaciones que deben ser manejadas por el sistema.

2) Realizar una MSC o añadir a las MSC existentes lo necesario para describir las secuencias de uso utilizadas, es decir, las secuencias de interacción típicas (protocolos) en cada capa de las interfaces.

3) Esquematizar la estructura del sistema e identificar las partes del mismo que están sujetas a los requisitos.

4) Ampliar la MSC para que abarque situaciones menos usuales que se encuentran en los límites de los requisitos así como posibles fallos.

5) Detallar la información en las interfaces y definir el modelo de información para el sistema.

6) Definir un sistema prototipo (no necesariamente formal o ejecutable) o partes del sistema que puedan manejar las secuencias de uso que sean "interesantes" (es decir, las que deben investigarse).

Debido a que uno de los objetivos del Diseño Previo es esbozar especificaciones para investigar distintos enfoques, algunos de los modelos producidos acaban siendo inadecuados o inmanejables. Cuando los diseños previos no dan lugar directamente a un resultado formalizado, se persigue la causa del problema hasta los requisitos o el diseño, intentando posibles alternativas. Ello conduce a la repetición de uno o más pasos. Inevitablemente, se descartan algunos de los diseños previos que se han producido. No debiera suponerse que cada Diseño Previo da lugar directamente a la descripción formal SDL final. La principal ventaja de este método es que las soluciones que no son viables o que no son atractivas pueden ser investigadas y descartadas a un coste razonable.

El resultado de cada paso debe incluir "enlaces" con la información clasificada y con los requisitos recopilados y un registro de las cuestiones surgidas en dicho paso. Junto a cada cuestión existe una respuesta o bien una marca que indica que dicha cuestión está aún sin resolver. También es probable que existan algunos diseños previos que han sido descartados, estando registrada la causa por la que cada uno de ellos fue descartado.

6.3 Conclusión del Diseño Previo

Como consecuencia del Diseño Previo, se ha generado un conjunto de resultados que se indican a continuación. Éstos se revisan para determinar si se ha completado la actividad de Diseño Previo.

Los datos, las estructuras y las interfaces del sistema deben estar suficientemente bien descritas y ser cabalmente entendidas de forma que:

1) se hayan cumplido los objetivos del Diseño Previo tal como fueron identificados al comienzo del mismo;

2) puede completarse la Formalización.

7 Formalización

El objetivo general de la Formalización es producir una especificación formal del sistema que se utilice como aplicación y para la Validación. Se utiliza toda la información clasificada y los modelos de diseños previos, tal como se muestra en la figura 7-1, el conocimiento adquirido en las actividades de Análisis y de Diseño Previo y, cuando sea posible, los modelos de la biblioteca de reutilización. En la cláusula 15 sobre directrices para los pasos de la Formalización detallada se proponen algunas correspondencias entre los modelos del Análisis y del Diseño Previo y la descripción formal SDL.

El ingeniero formaliza y genera diseños refinados para identificar, entender y analizar la aplicación a un nivel de detalle. El trabajo se rige por la utilización de SDL+. La figura 7-1 describe las entradas y salidas de la actividad de Formalización.

La información clasificada a la que se hace referencia en la Formalización es el resultado del Análisis o (si el Análisis no fue necesario) de las descripciones equivalentes existentes en los requisitos recopilados. De igual forma, las referencias a diseños previos son el resultado de la actividades de Diseño Previo o (si ésta no se consideró necesaria) de las descripciones equivalentes que existían en la información clasificada (o en los requisitos recopilados).

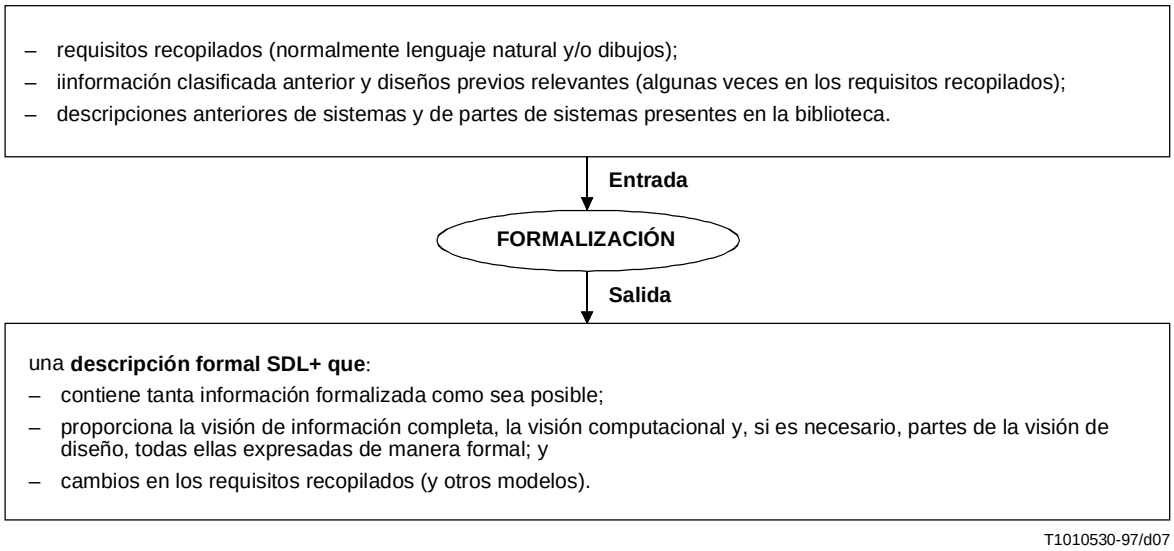


Figura 7-1/Sup. 1 a la Rec. Z.100 – Entradas y salidas de la Formalización

7.1 Inicio de la Formalización

Al inicio de la Formalización se revisan los objetivos, las entradas disponibles y la utilización de formalismos.

Los principales objetivos de la Formalización son los siguientes:

– generación efectiva de la descripción formal SDL+;

– investigación del sistema sobre la base de la utilización de SDL+:

  Conforme se genera la descripción SDL+, mejora la comprensión y avanza en la investigación. Si el Análisis o el Diseño Previo se omiten, aumenta la importancia de la investigación durante la actividad de Formalización se hace más importante. Las directrices relativas a los pasos de la Formalización incluyen la investigación que sea necesaria.

– tratamiento de las consistencias e inconsistencias.

  Durante la creación de la descripción SDL+, pueden detectarse inconsistencias y ambigüedades en los diseños previos, en la información clasificada y en los requisitos recopilados. Las cuestiones, respuestas e ítems que deben considerarse relacionados con cada una de dichas entradas se registran durante la Formalización. Ello puede dar lugar a cambios en cualquiera de las entradas, debiéndose modificar los requisitos recopilados (excepto cuando se ha cometido un error en el Diseño Previo o en el Análisis).

Antes de que se inicie la Formalización, deben cumplirse los criterios siguientes:

1) Debe existir información clasificada que proporcione la visión de empresa del sistema.
2) Debe existir información clasificada que describa los conceptos y los nombres de los principales elementos del sistema.
3) Debe existir información clasificada que incluya la descripción de las interfaces de carácter normativo.
4) Deben existir diseños previos que incluyan los principales elementos estructurales del sistema, tal como el modelo funcional utilizado en las normas de las Recomendaciones de la serie Q.1200 [13].
5) Deben existir diseños previos que incluyan los principales elementos de una visión de información del sistema.

7.2 Pasos de la Formalización

Los pasos que deben seguirse durante la Formalización dependen en cierta medida de las técnicas utilizadas en el Análisis o en el Diseño Previo, así como del contexto en el que se utiliza SDL+. La especificación formal se realizará en SDL+, por lo que los pasos no dependen de la notación utilizada. No obstante, la capacidad de las herramientas puede hacer inviable la utilización de determinadas construcciones SDL+, pudiendo existir otras limitaciones adicionales (tales como requisitos de los clientes) que hagan que varíe la utilización de SDL+. Sin embargo, en todos los casos es necesario definir la estructura, el comportamiento y los datos del sistema, aspectos estos sobre los que se pueden centrar los pasos de la Formalización. También pueden añadirse otros pasos a fin de explotar las facilidades de tipo de SDL-92. Además, si durante el Diseño Previo no se ha producido MSC, dichos pasos son necesarios como ayuda en la formalización del proceso SDL.

Debido a que existe poca dependencia con respecto al contexto, los pasos de la Formalización que se presentan en la cláusula 15 pueden utilizarse como la base para un conjunto de pasos válidos en cualquier contexto. Éstos se subdividen en pasos de Estructura, Comportamiento, Datos, Tipo y Localización. Los últimos dos grupos se aplican cuando algunas partes del sistema pueden ser reutilizadas. Cuando dichos pasos se utilizan como base para otro contexto, es necesario revisar cada paso, alguno pasos pueden suprimirse y otros pueden añadirse.

La información clasificada y los diseños previos proporcionan el nivel de comprensión y de información necesarios para la Formalización. Incluso si no se mencionan de forma explícita en cada uno de los pasos, siempre se utilizan como entrada. Los diseños previos debieran realizarse en SDL+, pudiendo ocurrir que ciertos aspectos de la Formalización sean triviales debido a que solo es preciso comprobar que algunas de las descripciones son correctas (pudiendo ser cambiadas si ello es necesario) según las reglas del lenguaje y las reglas de cada una de los pasos. Sin embargo, es normal que dichos diseños previos sean incompletos e informales debido a que sólo cubren parte del sistema. En las directrices pueden habitualmente encontrarse referencias para la utilización de diseños previos y de la información clasificada. Algunos de los diseños previos se realizan en paralelo con la actividad de Formalización (por ejemplo, los que producen un MSC detallado).

El resultado de cada paso debe incluir "enlaces" con las cuestiones, decisiones, requisitos recopilados, información clasificada y diseños previos relacionados con dicho paso. Ello permiten hacer un seguimiento de la descripción SDL hasta otras descripciones anteriores.

7.3 Conclusión de la Formalización

La Formalización produce un modelo completo del sistema en SDL-92. Este modelo proporciona, junto con el resto de la documentación procedente del Análisis y del Diseño Previo, la especificación completa del sistema. El modelo incluye la visión computacional del sistema y aquellos aspectos de la visión de diseño del mismo que son necesarios en la aplicación.

Los pasos de la Formalización tienen por objetivo producir un modelo ejecutable, con la probable excepción de algunos géneros de datos que normalmente pueden hacerse ejecutables mediante la utilización interactiva de una herramienta de apoyo o con los datos de un lenguaje de programación. La ventaja de un modelo ejecutable es que mediante la ejecución del mismo puede demostrarse la viabilidad y funcionalidad del sistema. La desventaja de este modelo es que algunos aspectos del sistema deben ser explícitamente definidos para que el sistema sea ejecutable. Por ejemplo, los datos que pasan a través del sistema sin ser modificados deben modelarse de forma explícita (por ejemplo, mediante una cadena de caracteres) de tal forma que el modelo pueda ser ejecutado, aunque para la aplicación ello constituya un concepto abstracto de una unidad de datos.

El modelo SDL+ es ejecutable de forma que dicha ejecución pueda ser utilizada en la Validación y el modelo pueda sufrir las pruebas de conformidad necesarias para garantizar que éstas son compatibles con el modelo.

El resultado de la Formalización debiera ser confrontado con los criterios siguientes:

1) el SDL+ formal debiera satisfacer los atributos de calidad del sistema;

2) el SDL+ formal debe ser conforme con las Recomendaciones Z.100 [1], Z.105 [2] y Z.120 [3];

3) la descripción formal SDL+ debe ser consistente con los diseños previos;

4) el SDL+ debe ser conforme con las reglas de las etapas de la Formalización;

5) se debe de haber demostrado (mediante una revisión o auditoría completa del diseño y mediante la ejecución con secuencias de entrada definidas) que la descripción formal SDL es un modelo satisfactorio de la funcionalidad del sistema descrito mediante los requisitos recopilados.

8 Implementación

En algunas aplicaciones, la cantidad de trabajo de ingeniería necesario para transformar un modelo formal SDL+ en una implementación utilizable puede ser muy reducido e incluso trivial (como la compilación de SDL+ para el soporte físico objetivo), pero conceptualmente el modelo SDL+ inicial y la implementación son niveles de abstracción bien diferentes. Según se deduce de la metodología expuesta en este Suplemento, la *descripción formal SDL+* de la figura 4-3 ignora los requisitos no funcionales de la implementación y las constricciones que **tienen** que (**have**) incluirse en el *producto* de la misma figura. Si la descripción formal SDL+ es muy abstracta, es posible refinarla para producir otra descripción formal SDL+ que permita incluir nuevas constricciones y requisitos de producto. Cuando las constricciones son muchas o significativas (tales como los aspectos relativos a la calidad de funcionamiento o a la distribución) el SDL+ puede refinarse de forma iterativa a fin de producir sistemas que sean funcionalmente equivalentes pero operacionalmente distintos.

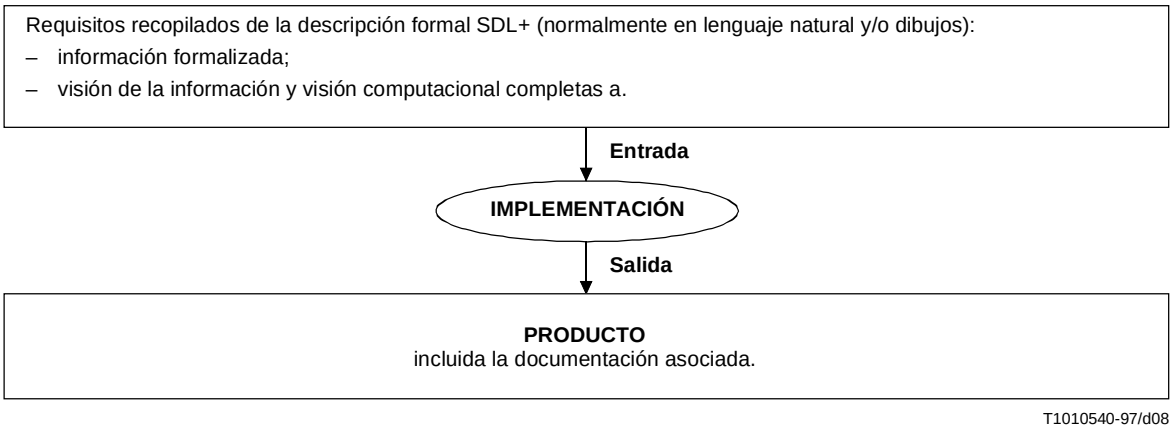


Figura 8-1/Sup. 1 a la Rec. Z.100 – Entradas y salidas de la Implementación

Pueden existir constricciones relativas a algunos elementos (tales como las interfaces con el sistema subyacente) que no pueden expresarse en SDL+ o que se expresan de forma más eficiente con otro lenguaje (por ejemplo, en C++, mediante llamadas al sistema operativo o en lenguaje ensamblador). Puede incluso ocurrir que parte del SDL no sea soporte lógico implementado (en el sentido convencional de la palabra). El hecho de que se den estas situaciones no significa que el diseño SDL+ deba escribirse de nuevo en otro lenguaje ya que existen herramientas que traducen de forma automática SDL en lenguajes de programación al tiempo que mantienen el SDL como la descripción del sistema. Estas herramientas permiten también el enlace con partes escritas de otra forma.

La implementación es más dependiente de la aplicación y de la organización que otras actividades y varía ampliamente. Se presenta aquí una descripción genérica. Los principales pasos que deben considerarse son los siguientes:

1) Realizar un análisis para ajustar razonablemente las necesidades de soporte lógico y de soporte físico (la arquitectura general).

2) Determinar la arquitectura necesaria del soporte físico para sustentar el soporte lógico (la arquitectura del soporte físico).

3) Finalizar la arquitectura del soporte lógico.

4) Reestructurar y refinar la descripción SDL+ añadiendo las descripciones que sean necesarias para definir el producto.

La descripción SDL+ puede tomarse como punto de partida junto con aspectos diversos de la información clasificada.

Los siguientes son factores clave en la arquitectura general del sistema: los requisitos para la distribución física y las anchuras de banda entre los ítems distribuidos; los requisitos de calidad de funcionamiento, incluidas las situaciones de sobrecarga, las respuestas críticas en el tiempo y el caudal medio; los requisitos de fiabilidad y la redundancia. Pueden existir varias opciones relativas a lo que es necesario, dónde y qué debe implementarse en soporte físico. En ocasiones, se requiere que el sistema se implementa totalmente mediante soporte lógico. Debe de compararse la economía del soporte físico (normalmente más caro pero más rápido) con la del soporte lógico. La utilización de diseños de soporte físico o de soporte lógico existentes o bien de equipos también existentes es un factor a tener en cuenta.

La arquitectura del soporte físico debe definirse en términos del número de procesadores y de otras unidades del mismo, del número de conexiones y del soporte físico necesario para la transmisión, de las características de calidad de funcionamiento de los elementos del soporte físico, tales como la potencia de procesamiento, y de los retardos de las conexiones. Si los requisitos definen el soporte físico, es importante determinar si dicho sistema puede efectivamente sustentar el soporte lógico y cumplir las constricciones de calidad de funcionamiento.

La arquitectura del soporte lógico es una elaboración con más nivel de detalle de la descripción SDL+. Puede requerir una cierta reestructuración de la descripción SDL a fin de establecer la correcta distribución del SDL entre las distintas unidades físicas. Ello puede exigir convertir algunos bloques del sistema SDL en sistemas SDL cooperativos. Es necesario un sistema de ejecución que soporte el SDL, así como un mecanismo (que posiblemente sea parte del sistema de tiempo de ejecución) que convierta sucesos externos en señales SDL.

En [6] pueden encontrarse directrices más detalladas.

9 Validación

La Validación tiene dos aspectos:

– comprobar que la sintaxis y la semántica de una especificación son correctas;

– comprobar que la especificación recoge todos los requisitos conocidos.

Aunque la Validación se realiza frecuentemente mediante la revisión de la aplicación por parte de expertos, una de las principales razones para realizar una especificación en SDL+ es permitir una Validación asistida por computadora. El primer aspecto de la Validación se adapta bien a la automatización. La conformidad con las reglas SDL+ garantiza que una especificación es consistente y no contiene ninguna ambigüedad no intencionada, permitiendo que las herramientas SDL+ automaticen la comprobación de la sintaxis y de la semántica estática. Para comprobar la semántica dinámica, habitualmente se requiere que se ejecute el modelo SDL+, siendo necesarias algunas pruebas para correr el modelo. La semántica dinámica puede también ser validada mediante técnicas tales como la exploración del espacio de estado.

El segundo aspecto de la Validación ha resultado más difícil de automatizar. Sólo en el caso de protocolos sencillos se ha automatizado la prueba formal de que la especificación cumple los requisitos. No obstante, es posible mejorar el nivel de confianza de que la especificación cumple los requisitos mediante técnicas automáticas, incluyendo la simulación y la realización de pruebas.

La definición de un sistema utilizando SDL+ es algo muy parecido a la escritura de soporte lógico en un lenguaje de programación. Al igual que los programas, la descripción formal SDL+ debe *probarse*, ya que es muy probable que inicialmente contenga errores de ingeniería y no proporcione los requisitos conocidos. Dichos errores suceden a pesar de la aplicación de diversas comprobaciones durante el proceso de creación del SDL+ debido a que la ingeniería no está completamente automatizada y a errores humanos. La aplicación de pruebas como parte de la Validación se asemeja a la prueba de un programa concurrente. A diferencia de un programa concurrente, que se ejecuta en una máquina real en un entorno real, los sistemas SDL+ se ejecutan sobre máquinas virtuales con recursos que son básicamente ilimitados, en lo demás, la prueba de SDL+ es similar a la prueba de soporte lógico. Dichas pruebas contribuyen a los dos aspectos de la Validación arriba mencionados.

Para que el SDL+ sea ejecutable en una máquina real y pueda probarse, puede ser necesario cambiar el modelo SDL+ (y las pruebas). El modelo modificado es un modelo de validación. Igualmente, el empleo de una técnica no ejecutiva (tal como la exploración de espacio de estado) para validar SDL+ mediante el examen automatizado puede también requerir un modelo de validación modificado obtenido del SDL+. La Validación realizada mediante pruebas o cualquier otra herramienta de validación de modelos se denomina *validación formal.*

Las diversas actividades pueden dar lugar a numerosas definiciones, donde unas son un refinamiento de otras o describen el sistema desde distintos puntos de vista. Las definiciones pueden realizarse mediante distintas técnicas de especificación, tales como lenguaje natural, MSC, ASN.1 o TTCN. El proceso de validación garantiza que las especificaciones son coherentes unas con otras. Es importante señalar que, normalmente, el SDL+ no existe de forma aislada sino en un contexto y, consecuentemente, su validación debe realizarse en dicho contexto.

La definición de un sistema no solamente hace uso de distintas notaciones, sino que puede también hacer referencia a otras especificaciones, tales como normas, sin que explícitamente las incorpore. En estos casos deben desarrollarse uno o varios modelos de validación antes de comprobar su corrección por medios formales. Si una especificación incluye opciones, debe elegirse un conjunto específico de ellas que hagan ejecutable al modelo. La figura 9-1 muestra el esquema general de la validación, en el que los círculos representan actividades y los rectángulos documentos.

La validación formal descubre errores cuya traza deben seguir los desarrolladores hasta el modelo de validación o la especificación. Una vez realizado dicho diagnóstico, se revisa el modelo de validación y, si es necesario, también se revisa la especificación, ejecutándose de nuevo el modelo de validación.

La metodología de esta subcláusula sólo se centra en la obtención y ejecución del modelo de validación, lo cual sólo es parte de la actividad de Validación que se muestra en la figura 4-3. Algunas de las comprobaciones necesarias para la Validación se incluyen en el Análisis, el Diseño Previo y la Formalización. Otras comprobaciones no cubiertas por este Suplemento pueden obtenerse a partir de la aplicación de técnicas de aseguramiento de la calidad del soporte lógico (véase la Recomendación Z.400 [14]).

9.1 Características de un modelo de validación

Un modelo de validación tiene dos características esenciales:

– el SDL+ está lo suficientemente detallado para permitir que el modelo sea ejecutable;

– es práctico utilizar el modelo para aplicar características extremas (por ejemplo, limitación en el número de circuitos, de llamadas o de otros tipos de tráfico; están limitados el tamaño de los ítems de datos).

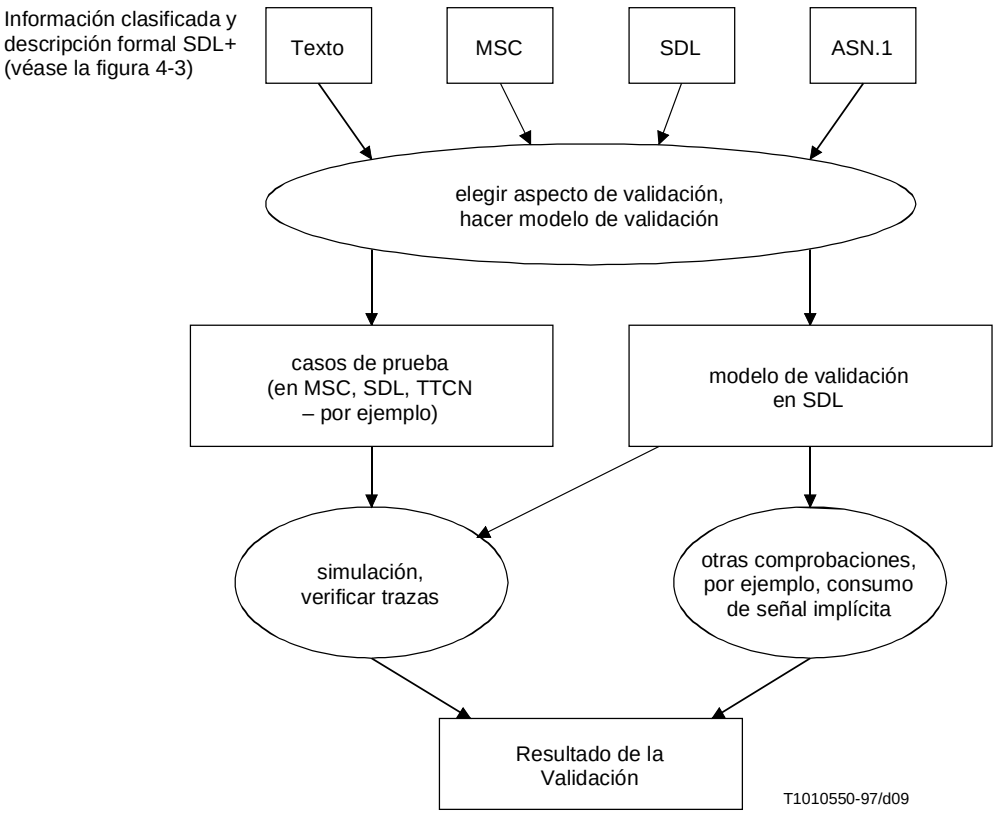


Figura 9-1/Supl. 1 a la Rec. Z.100 – Esquema general de Validación

9.2 Comparación del modelo de validación con el modelo formalizado

Cuando esta la metodología se utiliza para la descripción SDL+ de una norma, la descripción SDL+ de la misma, es decir, resulta de interés la norma formalizada.

El modelo de validación y la norma formalizada se obtienen a partir del modelo formal SDL+ que se genera durante la actividad de Formalización (el modelo formalizado). La hipótesis básica es que la norma formalizada se generaliza a partir del modelo formalizado. Por ejemplo, la norma formalizada puede omitir partes que se requieren desde el punto de vista técnico, pero que no forman parte de la norma; o bien, la norma formalizada puede permitir opciones, de las que al menos una es necesario implementar para que el modelo de validación sea ejecutable. Con frecuencia es conveniente que un modelo de validación incluya partes del entorno.

La relación entre el modelo de validación y la norma formalizada es que el modelo de validación consta de SDL+ de la norma formalizada y de las modificaciones mínimas necesarias para que el modelo sea ejecutable. Estas modificaciones pueden derivarse del modelo formalizado suponiendo que éste es ejecutable, pudiendo ocurrir en algunos casos que el modelo de validación sea idéntico al modelo formalizado.

Cuando esta metodología se utiliza para producir una implementación, el modelo de validación puede tener que diferir del modelo formalizado, bien porque éste necesite alguna implementación para ser ejecutable o porque los límites de la implementación sean demasiado amplios como para la validación. En el último caso, la validación puede realizarse utilizando límites más reducidos.

Si el modelo formalizado no es ejecutable o bien lo es pero no es útil para la validación, debe desarrollarse un modelo adicional para la validación.

Instrucciones

1) Definir nuevos límites para el modelo de validación, incluyendo partes del entorno que sean necesarias para la ejecución.

2) Definir límites para circuitos, llamadas y otros tipos de tráfico.

3) Identificar casos prácticos u observadores.

Directrices

La construcción del modelo de validación puede dividirse en:

1) Identificar el campo de aplicación de la validación.

2) Elegir un conjunto concreto de opciones de la implementación.

3) Completar el sistema.

4) Optimizar el modelo para que la verificación sea factible.

En [15] pueden encontrarse directrices más detalladas.

9.3 Aspectos de la definición de la Validación de una especificación

Durante la Validación, puede generarse más de un modelo de validación, en función de las características de la aplicación y de las técnicas automatizadas seleccionadas para ejecutar el modelo. Por ejemplo, cuando en una aplicación una especificación de protocolo resulta ser demasiado compleja, es posible dividirla en partes y validar cada una por separado. Ésta y otras tácticas para simplificar las especificaciones son necesarias si una especificación compleja debe validarse mediante una amplia y completa investigación en la que durante la ejecución del modelo deben evaluarse todos los estados posibles. Por otra parte, una de las ventajas de seleccionar la validación del espacio de estado aleatoria cuando se trata de una especificación de protocolo compleja, es que el modelo de validación puede representar al protocolo completo.

10 Relación con otras metodologías y modelos

La metodología de este Suplemento puede utilizarse en combinación con otras metodologías. Tal como se aplican normalmente, las metodologías siguientes dan lugar a uno o más modelos semiformales. La mayoría de los modelos utilizan los diseños previos producidos por la actividad de Diseño Previo, pero con características adicionales que son resultado de la otra metodología. Por ejemplo, la metodología de la Recomendación I.130 [16] genera tres modelos: uno desde el punto de vista del usuario, otro que muestra la organización de las funciones de la red y un tercero que muestra las capacidades de conmutación y señalización para soportar el modelo del usuario.

10.1 Relación con las Recomendaciones I.130 y Q.65 (método de tres etapas)

Debido a que la Recomendación I.130 se utiliza normalmente como base de metodologías locales, se toma como el contexto adecuado para la elaboración de los pasos de la metodología de la parte II.

La utilización de las Recomendaciones I.130 [16] y Q.65 [17] no ha estado limitada al ámbito de la "caracterización de los servicios de telecomunicación soportados por una RDSI y las capacidades de red de una RDSI" que se indica en el título de la Recomendación I.130. Las etapas 1 y 2 se utilizan para la producción de normas relativa a la funcionalidad de los servicios, los cuales se utilizan en la etapa 3 para producir normas de protocolos que soporten el servicio. En las tres etapas del método I.130 se utiliza SDL:

– en la etapa 1, paso 1.3, para la descripción dinámica de un servicio, en el que "la información se presenta en la forma de diagrama del lenguaje de especificación y descripción (SDL)" (véase 3.1/I.130);

– en la etapa 2, paso 2.3, en el que las "funciones realizadas en una entidad funcional" se "representan en forma de diagrama del lenguaje de especificación y descripción (SDL)" (véase 3.2/I.130);

– en la etapa 3, en la que "los diagramas SDL de la etapa 2 constituyen la base " (véase 3.3/I.130) y "las Recomendaciones sobre SDL (serie Z.100)" constituyen las "herramientas de los métodos, técnicas (modelos) de descripción y la biblioteca asociada de material genérico".

En algunas ocasiones las tres descripciones SDL aparecen en las normas, mientras que en otras sólo lo hace una de ellas. A veces, las distintas descripciones aparecen en varias normas de un conjunto de normas. Las decisiones sobre las descripciones que deben aparecer en cada norma y sobre lo que debe ser normalizado (el punto de vista del usuario, la implementación funcional del servicio y/o los protocolos y procedimientos de acceso y entre nodos) son responsabilidad del grupo encargado de la elaboración de las normas en cuestión y quedan fuera del campo de aplicación de este Suplemento. Este Suplemento asume que cada descripción SDL aparece en una norma diferente.

Cuando se preparó la versión inicial de la Recomendación I.130, la Recomendación Z.120 [3] no era aún una norma, por lo cual, los "flujos de información" de la Recomendación I.130 deberían ser ahora sustituidos por MSC.

El enfoque de la Recomendación I.130 es consistente con esta metodología. En dicha Recomendación se definen las descripciones que deben proporcionarse y la metodología de este Suplemento expone en detalle como elaborar dichas descripciones. La descripción de la etapa 1 de la Recomendación I.130 puede realizarse mediante un diagrama de contexto, utilizando una descripción SDL sencilla que puede considerarse un Diseño Previo para la entidad funcional SDL de la etapa 2. El SDL de la etapa 2 puede obtenerse aplicando todas las actividades de la presente metodología. El SDL de la etapa 3 se puede obtener aplicando de nuevo esta metodología, considerando que las etapas 1 y 2 forman parte de los requisitos recopilados.

La Recomendación Q.65 ofrece más detalles para el método de la etapa 2. En el paso 2.1 se presenta un modelo funcional que muestra la relación entre las entidades funcionales. La representación de esta última debiera convertirse en un diagrama de sistema SDL en el que las entidades funcionales se representan como tipos de bloques, las instancias de entidades funcionales como bloques y las relaciones como canales. Debido a que el SDL de una norma debiera ser completo, la versión SDL del diagrama se incluye en la norma. Puede ser de utilidad incluir también en la norma el modelo de la entidad funcional, debido a que ésta no incluye toda la información en el diagrama SDL y, por lo tanto, ofrece una visión más abstracta del sistema. Los diagramas de flujo de información de la Recomendación Q.65 son MSC con una instancia de entidad funcional (bloque SDL) para cada eje de instancia.

Cuando se prepara una descripción formal SDL para la etapa 2, las acciones de la entidad funcional son normalmente redundantes debido a que el SDL describe completamente las funciones. Estas descripciones de la acción de la entidad funcional pueden ser útiles para explicar el SDL y porque algunas normas hacen un amplio uso de ellas. Si se utilizan, deben estar claramente marcadas como de carácter informativo.

10.2 Relación con el modelo de capas de OSI

Esta subcláusula actualiza material SDL-92 que se publicó inicialmente en [18], [19] y en el apéndice I/Z.100 [1].

Los conceptos OSI utilizados en esta subcláusula se definen en [20] y [21]. A fin de que las diferentes subcláusulas de la misma sean aún más autocontenidas, se ofrece una explicación de los conceptos clave.

Un *servicio de capa* es ofrecido por un *proveedor de servicio* para una capa dada. El proveedor de servicio es una máquina abstracta que ofrece una facilidad de comunicación a usuarios de la capa superior. Los usuarios acceden al servicio a través de *puntos de acceso al servicio* y mediante *primitivas de servicio* (véase la figura 10-1). Una primitiva de servicio puede utilizarse para la gestión de la conexión (conexión, desconexión, reinicio, etc.) o como un objeto de datos (datos normales o datos despachados). Sólo hay cuatro clases de primitivas de servicio:

– petición (del usuario al proveedor);

– indicación (del proveedor al usuario);

– respuesta (del usuario al proveedor);

– confirmación (del proveedor al usuario).

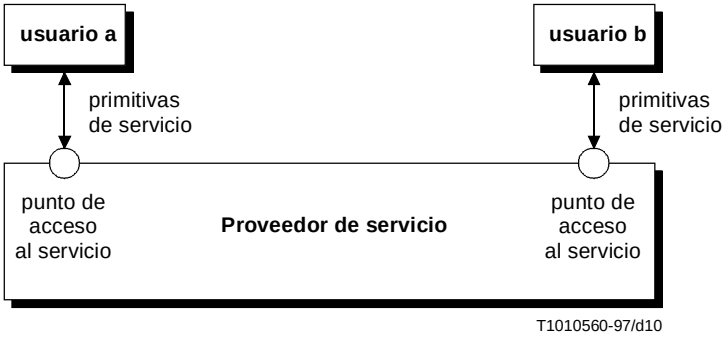


Figura 10-1/Sup. 1 a la Rec. Z.100 – Proveedor de servicio de capa

Una *especificación de servicio* es una forma de caracterizar el comportamiento del proveedor de servicio, tanto localmente, declarando las secuencias legales de las primitivas de servicio que se transfieren a un punto de acceso al servicio, como extremo a extremo, declarando la relación correcta entre primitivas de servicio que se transfieren en distintos puntos de acceso al servicio. Una especificación de servicio no se ocupa de la estructura interna del proveedor de servicio; cualquier estructura interna que se proporcione cuando se especifica el servicio es sólo un modelo abstracto para describir el comportamiento observable externamente del proveedor de servicio.

Con la excepción de la capa más alta, los usuarios de un servicio de capa son *entidades de protocolo* de la capa superior, que cooperan para mejorar las características del servicio de capa, proporcionando así un servicio a la capa superior. La cooperación se realiza acorde con un conjunto predefinido de reglas de comportamiento y formatos de mensajes que constituyen un *protocolo*. De acuerdo con esta visión, las entidades de protocolo de capa (N) y el proveedor de servicio (N-1) proporcionan conjuntamente un refinamiento del proveedor de servicio (N) (véase la figura 10-2).

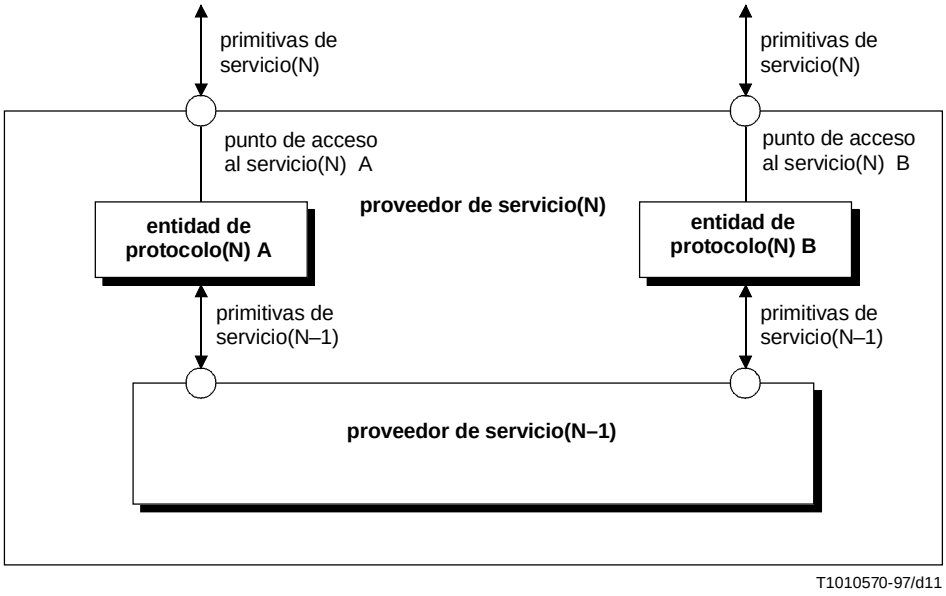


Figura 10-2/Sup. 1 a la Rec. Z.100 – Refinamiento del proveedor de servicio(N)

El refinamiento del proveedor de servicio (N) que se muestra en la figura 10-2 puede ser mucho más complicado. Por ejemplo, puede haber entidades de protocolo de retransmisión (N) que no estén conectadas a ninguna entidad de protocolo de la capa (N + 1). En aras de la brevedad, dichos casos no se consideran en este Suplemento.

Las entidades de protocolo se comunican mediante el intercambio de *unidades de datos de protocolo*. Éstas se transfieren como parámetros de las primitivas de servicio de la capa inferior. La entidad de protocolo emisora codifica las unidades de datos de protocolo en primitivas de servicio, la entidad de protocolo receptora decodifica las unidades de datos de protocolo de las primitivas de servicio recibidas. Un protocolo se basa en las propiedades del proveedor de servicio subyacente. El proveedor de servicio subyacente puede, por ejemplo, perder, corromper o reordenar mensajes para lo cual el protocolo debiera disponer de mecanismos de detección y corrección de errores, resincronización, retransmisión, etc., a fin de proveer un servicio fiable y normalmente más potente a la capa superior.

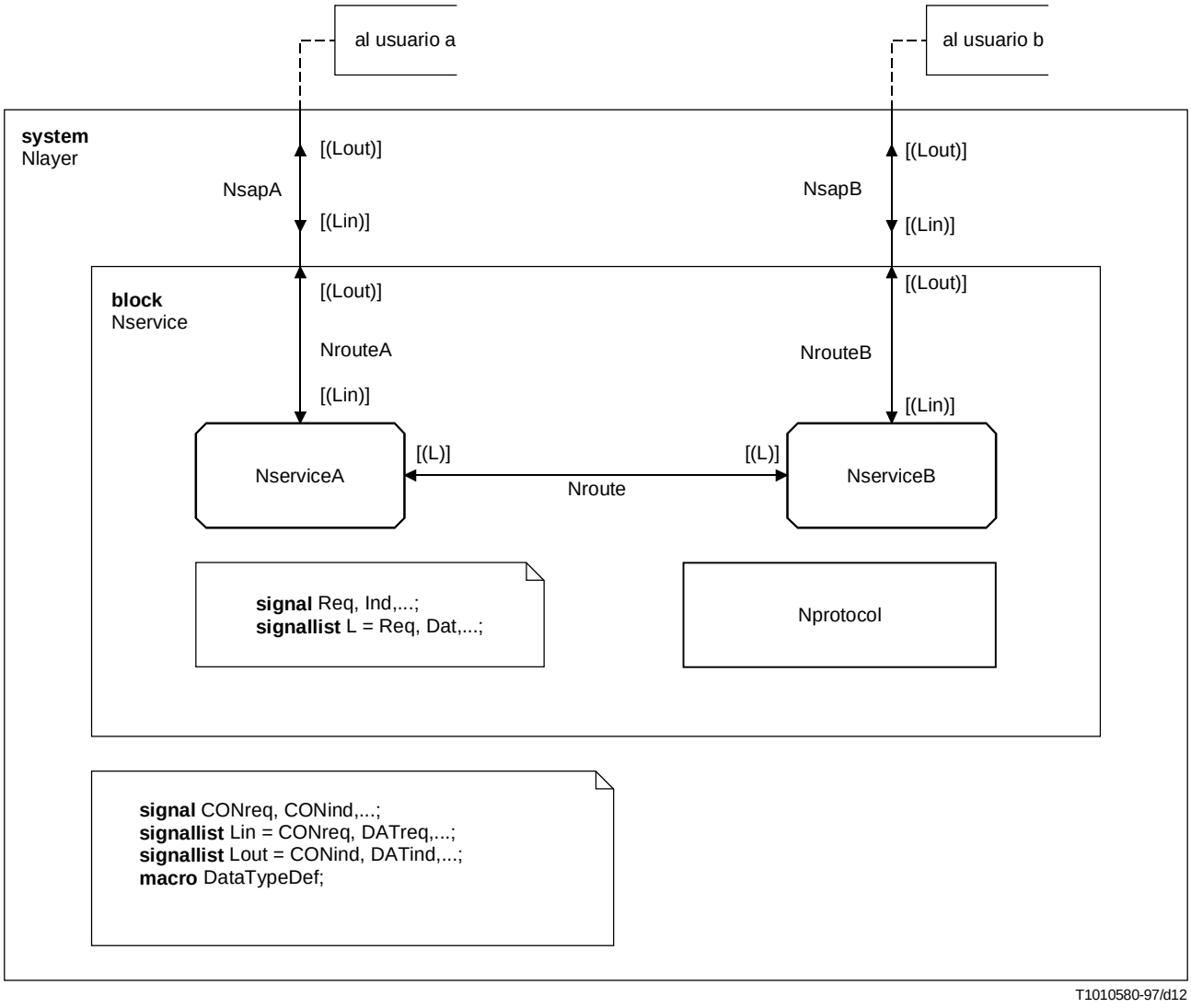
Los conceptos arquitectónicos OSI pueden modelarse en SDL de varias formas, principalmente en función de los aspectos que deben destacarse. En primer lugar se describe un enfoque básico, describiéndose a continuación otros enfoques como variantes del enfoque básico.

En los ejemplos, se utiliza en la mayor medida posible la sintaxis gráfica de SDL. Sin embargo, debe notarse que por motivos prácticos, puede omitirse parte de la información exigida por las reglas de la sintaxis o bien se representa por una serie de puntos (...) que no forman parte de la sintaxis.

10.2.1 Enfoque básico

Especificación del servicio

La especificación de servicio para la capa N puede modelarse directamente como un bloque *Nservice* con dos procesos *NserviceA* y *NserviceB* (véase el ejemplo 10-1).



Ejemplo 10-1/Sup. 1 a la Rec. Z.100 – Especificación de servicio-(N) mediante SDL

En el ejemplo 10-1, los usuarios del servicio se encuentran en el entorno del sistema y pueden considerarse procesos capaces de comunicarse con el sistema en términos de dicho sistema.

Un punto de acceso al servicio se representa mediante un canal (*NsapA* o *NsapB*) que transporta señales que representan primitivas del servicio. Una señal puede transportar valores de los géneros incluidos en la especificación de la señal. Las especificaciones de género (distintas de los géneros predefinidos) están contenidas en una especificación *macro* remota de *DataTypeDef* que aquí se omite por ser irrelevante para los objetivos de esta discusión.

En el caso más general puede, por supuesto, haber más de dos procesos involucrados en la especificación del servicio. En aras de la brevedad, sólo se considerarán aquí dos procesos, uno para cada punto de acceso al servicio. Todo lo que sigue es, sin embargo, aplicable al caso en el que haya más procesos para un punto de acceso al servicio.

En el ejemplo 10-1 se muestran algunos ejemplos de especificaciones de señales. Tal como sugieren los nombres de algunas de las señales, se trata de un servicio orientado a la conexión. En el caso de un servicio no orientado a la conexión la simplificación es importante. No obstante, y en aras de la brevedad, dicho caso no será tratado en lo que sigue.

Se tratan tanto el aspecto local como el aspecto extremo a extremo de la especificación del servicio. El aspecto local se expresa de forma independiente mediante los procesos *NserviceA* y *NserviceB*. Estos procesos se comunican entre sí mediante señales (*Req*, *Ind*,...) que son internas al bloque y que se transportan en la ruta de señal *Nroute*. El comportamiento extremo a extremo se expresa mediante la correspondencia (que realiza cada uno de los procesos) entre primitivas del servicio y las señales internas en *Nroute*. Los procesos *NserviceA* y *NserviceB* son una imagen especular el uno del otro. La razón de tener dos en lugar de uno es poder modelar de la forma más fiel posible una posible situación de colisión en el proveedor de servicio.

El comportamiento no determinístico es una característica inherente al proveedor del servicio, ya que éste puede rechazar intentos de conexión y puede interrumpir conexiones establecidas por propia iniciativa.

Nótese que la especificación del bloque *Nservice* sólo hace referencia a los procesos *NserviceA* y *NserviceB*; estos procesos se especifican mediante *especificaciones remotas* que se encuentran fuera de la especificación de bloque, y no se muestran aquí porque obligarían al lector a prestar atención a características que son específicas de un servicio determinado.

Especificación de protocolo

La especificación de protocolo para la capa N se modela mediante la subestructura *Nprotocol* del bloque *Nservice* (véase el ejemplo 10-1).

En el *diagrama de bloques* del ejemplo 10-1 se ha incluido una referencia a una subestructura de bloque (un símbolo de bloque que contiene el nombre *Nprotocol* de la subestructura de bloque). La especificación de la subestructura de bloque figura en un *diagrama de subestructura de bloque* remoto (véase el ejemplo 10-2) que contiene tres bloques: *NentityA*, *NentityB* y *N_1service*. Los primeros dos bloques representan entidades de protocolo (N), mientras que el bloque *N_1service* representa al proveedor de servicio (N-1). La especificación de *N_1service* es análoga a la de *Nservice* y no se muestra en este diagrama (es una especificación remota).

Un bloque de entidad de protocolo contiene uno o más procesos, dependiendo de las características del protocolo. En este caso se han elegido dos procesos, *Ncom* y *Ncodex*. El proceso *Ncom* maneja el envío y recepción de unidades de datos de protocolo, mientras que el proceso *Ncodex* se ocupa de la transmisión de las unidades de datos de protocolo utilizando el servicio subyacente. Conceptualmente, el proceso *Ncom* se comunica de forma directa a través de un canal implícito *Nchan* (para el transporte de las unidades de datos de protocolo), pero en realidad, se comunican indirectamente a través del proceso *Ncodex* y el proveedor de servicio subyacente.

10.2.2 Enfoque alternativo utilizando la subestructura canal

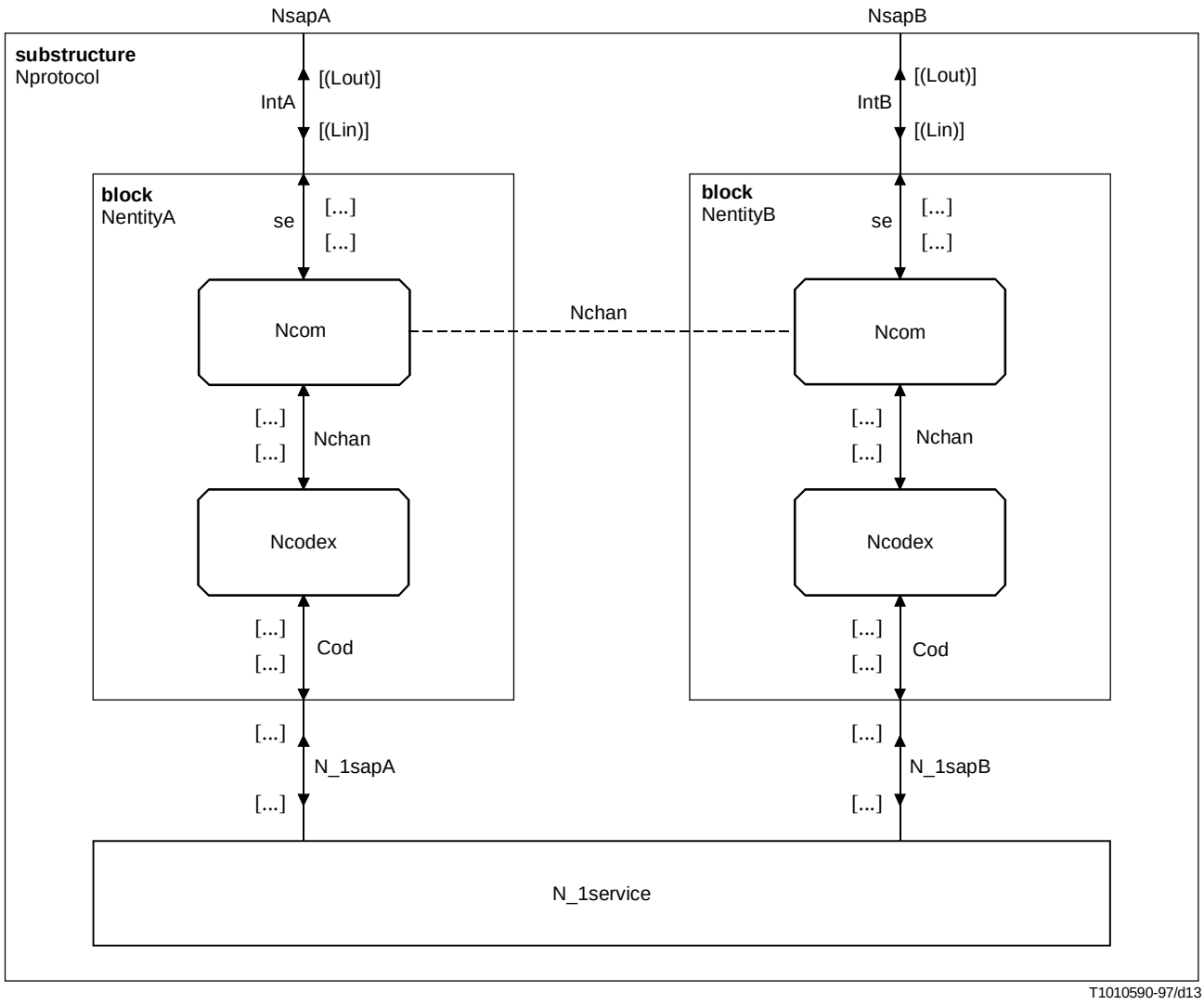
Este enfoque se deriva del enfoque básico del ejemplo 10-2 agrupando los procesos de forma diferente, introduciendo el canal real *Nchan* y utilizando la subestructura de canal (véase el ejemplo 10-3). El canal *Nchan* transporta unidades de datos de protocolo identificadas por la lista de señal *Npdu*. Este enfoque destaca sobre todo la visión del protocolo y la orientación horizontal de OSI.

Nótese que en este enfoque los bloques de una subestructura de canal no representan entidades de protocolo y se superponen a dos capas contiguas. Las primitivas de servicio quedan ocultas en dichos bloques y son transportadas en las rutas de señal *N1_sapA*, *N1_sapB*, *N2_sapA*, *N2_sapB*, etc. Sin embargo, la capa (N) más alta elegida debe ser tratada por separado, tal como se indica en el ejemplo. Nótese también que este enfoque no afecta al diagrama del sistema (ejemplo 10-1).

10.2.3 Arquitectura OSI simétrica

Cuando la arquitectura OSI es simétrica, es decir, cuando las entidades de ambos lados de la arquitectura OSI son imágenes especulares una de otra, las especificaciones de dichas entidades son idénticas salvo en el nombre de la entidad. La especificación común puede venir dada por la instanciación de un tipo, el tipo de proceso *Nservice* del bloque *Nservice*. El tipo de proceso *Nservice* se instancia en *NserviceA* y *NserviceB* (véase el ejemplo 10-4). Los valores *x* y *z* son los parámetros efectivos de las puertas del tipo de proceso *Nservice* (no se muestran en el ejemplo). En este ejemplo sólo se muestra el diagrama de bloque de *Nservice* del ejemplo 10-1. Nótese que las especificaciones de servicio son siempre simétricas y que sólo las especificaciones de protocolo pueden ser asimétricas.

Un enfoque alternativo consiste en representar solo un lado, (véase el ejemplo 10-5), lo cual constituye una modificación del diagrama del sistema del ejemplo 10-1. El canal *NsapB* ha sido sustituido por el canal *Nchannel* que transporta las señales internas *L*. Dichas señales se encuentran ahora a nivel de sistema y sus especificaciones han sido modificadas en consecuencia. Nótese que este enfoque no puede ser utilizado en combinación con la subestructuración de canal (que se muestra en el ejemplo 10-3).



Ejemplo 10-2/Sup. 1 a la Rec. Z.100 – Especificación de protocolo (N) mediante SDL

10.3 Relación con la serie de Recomendaciones Q.1200 sobre arquitectura y bloques constitutivos independientes de los servicios de la red inteligente

La Recomendación Q.1200 [13] sobre la arquitectura de la red inteligente (RI) no presenta dificultades fundamentales para adaptarse al SDL, sin embargo existen diversas maneras en las que el SDL puede utilizarse para sustentar los bloques constitutivos independientes del servicio (SIB, *service independent building blocks*). No obstante, cuando se utiliza el enfoque de la red inteligente, la relación entre el procesamiento de la llamada básica y la lógica del servicio es un aspecto significativo del modelado que debe considerarse. La mayor diferencia entre este enfoque y el de la Recomendación I.130 es la introducción de un plano funcional global por encima del plano funcional (distribuido) y la utilización de los SIB.

El plano funcional distribuido puede modelarse de igual forma que la etapa 2 de la Recomendación I.130.

Para sustentar la red inteligente utilizando SDL, debe existir un modelo SDL para el plano funcional global con las características siguientes:

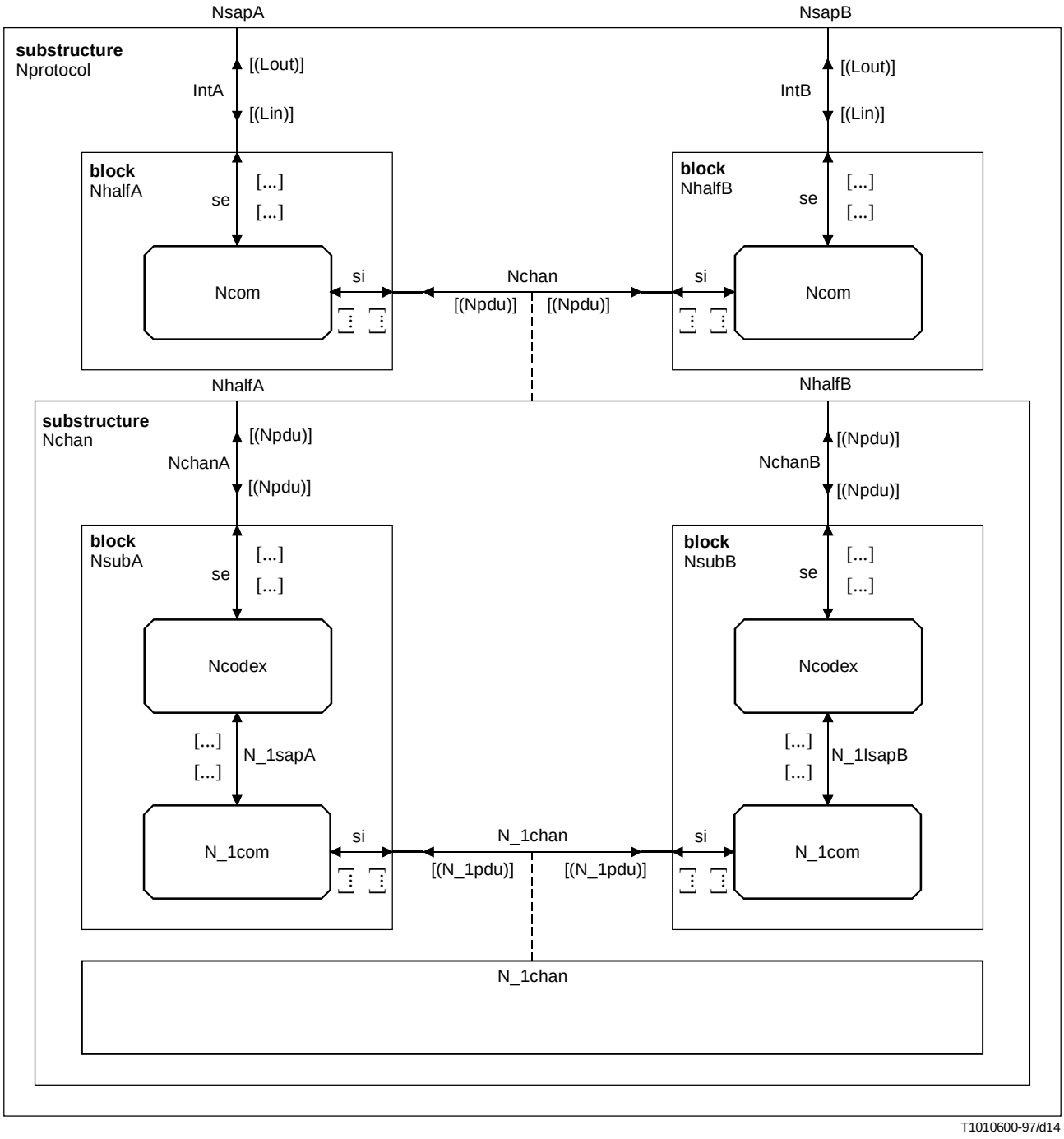
– un procesamiento de llamada básica (BCP, *basic call processing*) global bien definido, preferiblemente en SDL formal, pero que al menos debe tener interfaces claramente definidas para la llamada del primer SIB de una cadena;

– una forma de añadir al BCP global la invocación de un nuevo servicio;

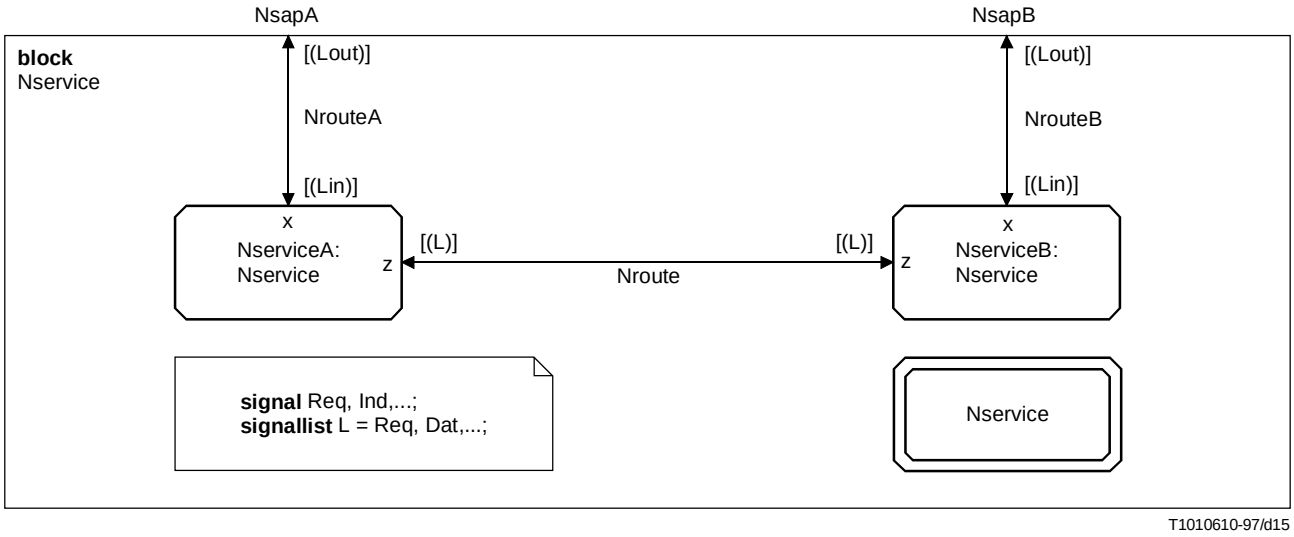
– una forma de especificar los SIB en SDL y ser encadenados conjuntamente;

– una forma de utilizar SDL de forma que el BCP global y la lógica del servicio puedan estar en normas diferentes.

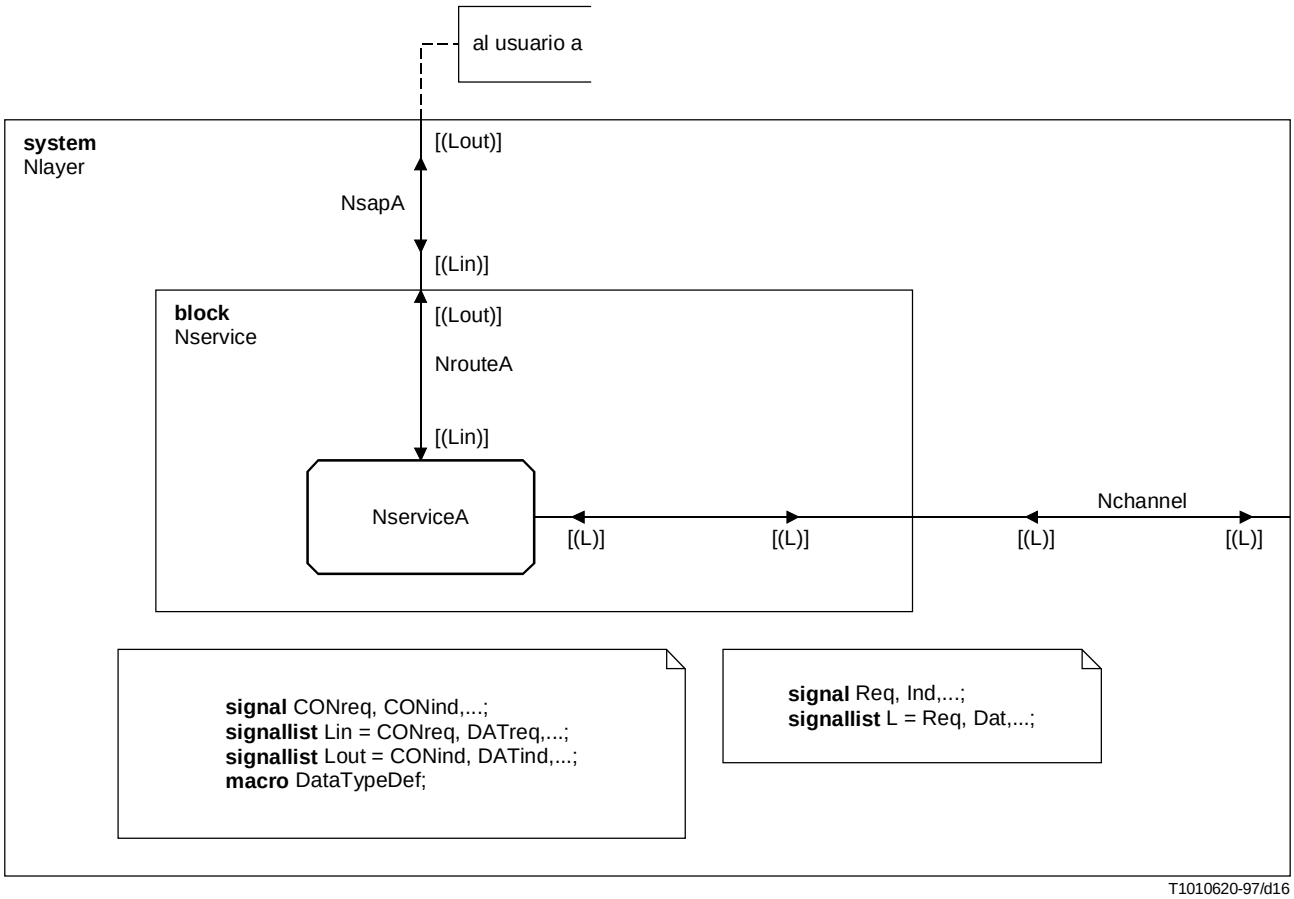
SDL-92 proporciona los mecanismos para soportar el enfoque de la RI.



Ejemplo 10-3/Sup. 1 a la Rec. Z.100 – Visión del protocolo de OSI



Ejemplo 10-4/Sup. 1 a la Rec. Z.100 – Utilización del tipo de proceso para representar una arquitectura OSI asimétrica



Ejemplo 10-5/Sup. 1 a la Rec. Z.100 – Representación de un solo lado de una arquitectura OSI simétrica

Para realizar llamadas a los SIB desde BCP, éste debe definirse como un tipo de proceso. Los puntos desde los cuales es probable que se realicen llamadas a los SIB debieran estar en procedimientos virtuales o en transiciones virtuales del BCP. La llamada al primer SIB puede añadirse al BCP modificando la parte virtual del proceso. La definición del proceso efectiva depende de las interacciones de servicio proporcionadas por el BCP.

Los SIB pueden modelarse de distintas formas. Pueden ser procedimientos del BCP, pero si así es, no pueden ejecutarse en paralelo con el BCP o de uno respecto a otro. Alternativamente, los SIB pueden ser llamadas de procedimientos remotos a uno o varios procesos de la lógica de servicio o un proceso SDL. La llamada a un SIB puede realizarse mediante llamadas de procedimientos remotos, creando un proceso SIB, o mediante el paso de señales. Debido a que pueden existir varios puntos de retorno al BCP, deben recibir una consideración especial. En consecuencia, una interfaz de procedimiento puede no ser adecuada salvo que el retorno del procedimiento venga siempre seguido de una decisión basada en un valor que devuelve el procedimiento.

Los datos globales y los datos de llamada deben estar bien definidos. Debe existir un mecanismo para el manejo de la base de datos de dichos datos.

Una vez que todos estos aspectos arquitectónicos han quedado establecidos, puede realizarse la descripción formal SDL para los BCP y los SIB, considerando la arquitectura general como parte de los requisitos recopilados y como una constricción del diseño. Puede seguirse entonces la metodología aquí descrita.

10.4 Relación con las operaciones remotas de la Recomendación X.219 (RO y ROSE)

La Recomendación X.219 [22] sobre operaciones distantes o remotas proporciona una forma de asociar una petición con sus posibles respuestas. Si se trata de una operación síncrona (clase 1), puede establecerse una correspondencia de ésta con un procedimiento remoto SDL. Si la operación nunca tiene respuesta (clase 5), puede establecerse una correspondencia con una señal SDL. Si la operación es asíncrona y proporciona alguna respuesta (éxito o fracaso), se establece una correspondencia entre ella y señales en ambos sentidos de un canal, estando los parámetros de operación y las señales de respuesta ligados informalmente mediante comentarios (e implícitamente por el comportamiento del proceso SDL). De igual forma, una asociación no se hace corresponder con una característica SDL explícita.

Debido a que ROSE utiliza ASN.1 para definir los datos, puede utilizarse directamente para los parámetros de la señal.

Ejemplo

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

se puede convertir en la versión SDL

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

utilizando estas señales en el canal hacia y desde el proceso que contiene la operación.

10.5 Relaciones con la Recomendación X.722 (GDMO)

NOTA – Aunque se trata de algo potencialmente importante para el diseño y quizás para la normalización de sistemas de telecomunicación en los que los recursos se manejan como objetos gestionados, se necesitan ulteriores estudios para acordar un método que combine GMDO con las normas relativas a los recursos.

GMDO [23] utiliza ASN.1 como base. La plantilla GMDO utiliza una palabra clave de BEHAVIOUR (COMPORTAMIENTO) con la que puede definirse el comportamiento de un objeto, pero la definición tiene la forma de una cadena que habitualmente contiene lenguaje natural. Sólo se describe el comportamiento del objeto gestionado que es relevante para la gestión. Normalmente, el objeto tiene otro comportamiento que está normalizado mediante una norma de protocolo o de servicio. Las ventajas de describir el comportamiento de gestión de un objeto en SDL son las siguientes:

– Se describe mejor el comportamiento de gestión del objeto, que puede ser bastante complicado para objetos gestionados complejos.

– Permite comprobar mejor la consistencia de la descripción de gestión y de la descripción funcional.

– La descripción de gestión y la descripción funcional pueden tener partes comunes y una base para fusionar ambas en una implementación.

La base ASN.1 del GMDO proporciona el fundamento para definir el objeto gestionado en SDL.

11 Justificación del enfoque

La metodología de este Suplemento representa la visión de expertos que han contribuido a los trabajos del UIT-T sobre la utilización del SDL+ y es fruto de varios años de investigación y experiencia. Parte del material que ha servido de base para la elaboración de este Suplemento figura en [24] y [25].

La metodología está específicamente centrada en la generación de la especificación precisa de un sistema. Esta limitación ha sido elegida por varios motivos:

1) existe un acuerdo general sobre el enfoque para la formalización de requisitos en SDL;

2) existen muchas formas diferentes de implementar un producto a partir de una especificación en SDL;

3) el tiempo de acceso al mercado es con frecuencia más importante que la eficiencia de la ejecución, pudiendo las herramientas SDL producir un código aceptable a partir de descripciones de la implementación que sean muy parecidas a las especificaciones del producto;

4) la metodología puede utilizarse tanto para producir especificaciones de productos como para especificaciones que se utilicen en normas (o en compras).

Es de esperar que conforme aumente el número de normas que se escriban utilizando SDL de una manera formal, las descripciones SDL de las normas sean utilizadas como base en las empresas, en las implementaciones de productos y en las pruebas de productos. Por lo tanto, se ha prestado especial atención a la combinación de la metodología de este Suplemento con otras técnicas empleadas en la elaboración de normas.

Esta metodología se define en términos de SDL/GR. La mayoría de los usuarios prefieren utilizar la forma SDL/GR, que consideran de más fácil comprensión. La metodología puede adaptarse a la utilización de SDL/PR.

La metodología se presenta como una serie de actividades y de pasos que deben seguirse debido a que un número importante de usuarios de SDL optó por este enfoque y porque los pasos que se describen en el apéndice I/Z.100 [1] han tenido una acogida favorable entre los usuarios. Estos pasos se desarrollan con más detalle y se amplían en la parte II de este Suplemento. Las actividades se definen en la parte I y se detallan para un caso en particular en la parte II. Por lo tanto, la parte I define el marco general y la parte II define el conjunto de pasos que deben realizarse.

La metodología no utiliza todas las características del SDL, sugiriendo en algunos casos cuales son las características que no debieran ser utilizadas. A pesar de ello, es posible que en algunos casos pudiera ser adecuado utilizar algunas de dichas características. La metodología es general y es factible que se produzcan variaciones de la misma para una aplicación u organización en particular.

PARTE II – DESARROLLO DE LA METODOLOGÍA MARCO

12 Desarrollo de la metodología para la especificación del servicio

Las actividades se describen utilizando una serie de *pasos* que algunas veces se subdividen en *instrucciones* y *directrices*. En los distintos pasos se establecen una serie de *reglas*. Las reglas se declaran mediante el término "debe", o mediante los términos "debería" o "debiera". Mediante el término "debe" se establecen reglas que han de ser seguidas para garantizar un sistema válido que sea comprensible, implementable y que pueda ser probado. El término "debería" se utiliza para reglas que pueden no cumplirse en determinadas circunstancias. La diferencia entre directriz y regla es que la primera hace referencia a aspectos que pueden ser ignorados sin que ello represente consecuencias serias o bien, hace referencia a alternativas sobre las que debe optarse.

Los términos *paso, instrucción, directriz y regla* se definen en la cláusula 2.

12.1 Metodología de tres etapas: etapa 2 (Recomendación Q.65)

En la subcláusula 10.1 se describe la relación entre la metodología de tres etapas de las Recomendaciones I.130 [16] y Q.65 [7] y la metodología de este Suplemento. En los párrafos siguientes se explica la correspondencia entre la Recomendación Q.65 y las construcciones SDL.

En teoría sería posible utilizar conceptos estructurales de SDL en el paso 1 y conceptos de comportamiento en los pasos 2-4 con una introducción paso a paso de los datos. El paso 5 queda fuera del campo de aplicación de SDL. La razón fundamental para utilizar SDL en los pasos 1-4 de la Recomendación Q.65 es que el cumplimiento de las normas mejora la legibilidad y permite la utilización de herramientas de apoyo.

En los puntos siguientes se muestra como puede establecerse una correspondencia entre los conceptos de la mayoría de los pasos de la Recomendación Q.65 y el SDL.

Estructura

El paso 1 de la Recomendación Q.65 es el siguiente: el modelo funcional, la identificación de entidades funcionales y sus relaciones se corresponde con conceptos estructurales de SDL. La figura 12-1 muestra un modelo funcional acorde con la Recomendación Q.65.

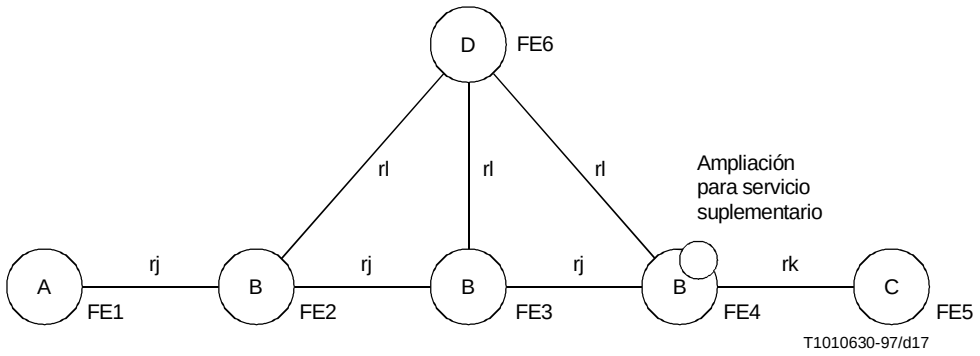


Figura 12-1/Sup. 1 a la Rec. Z.100 – Ejemplo de modelo funcional acorde con la Recomendación Q.65

Los elementos de la figura 12-1 son los siguientes:

– nombres en círculos (por ejemplo, *A*): tipos de entidades funcionales;

– nombres en mayúsculas junto a los círculos (por ejemplo, *FE1*): nombres de entidades funcionales (instancias);

– nombres en minúsculas entre círculos (por ejemplo, *rj*): relaciones entre tipos de entidades funcionales;

– círculo añadido superpuesto: ampliación para servicio suplementario.

Nótese que cuando se describe la estructura del sistema, este modelo hace claramente una distinción entre tipos de entidades funcionales (por ejemplo, *A*) y su instanciación (por ejemplo, *FE1*). La Recomendación Q.65 menciona también la conveniencia de describir dos tipos de entidades funcionales como subconjuntos del mismo tipo de entidad funcional simple, en el caso de que ambas entidades funcionales tengan bastantes aspectos en común. Estas características pueden expresarse utilizando las características de **tipo** (**type**) de SDL.

Los estudios realizados han demostrado que pueden hacerse corresponder las "entidades funcionales" de la Recomendación Q.65 con los conceptos estructurales SDL de la forma siguiente:

– Modelo → sistema;

– Entidad funcional → bloque con un proceso;

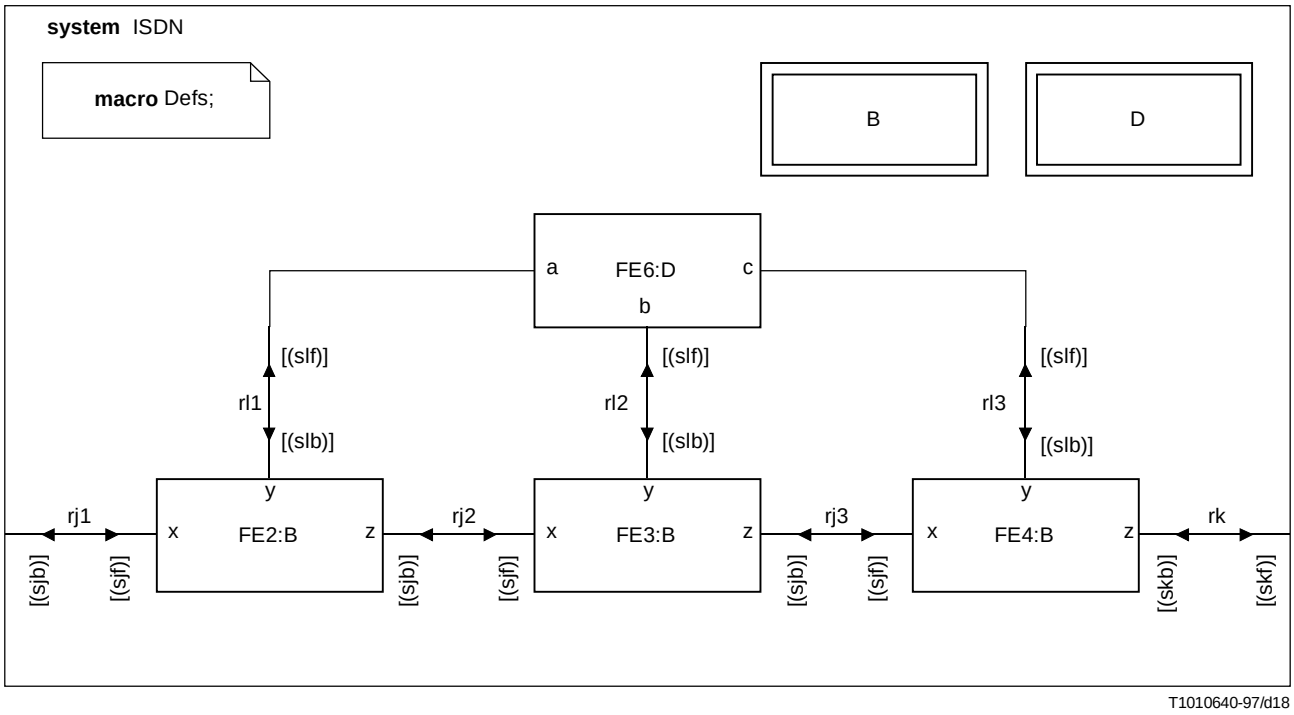
– Relación entre entidades funcionales → tipo de bloque

Además, utilizando construcciones **type**:

– Tipo de entidad funcional → tipo de bloque.

Una diferencia entre el modelo funcional de la Recomendación Q.65 y los diagramas de estructura SDL es que normalmente éstos últimos modelan sistemas abiertos.

El ejemplo 12-1 muestra el modelo funcional de la figura 12-1 en SDL. Nótese que *FE1* y *FE5* no están presentes en el sistema. La comunicación con dichas entidades funcionales se modela como comunicación con el entorno. El hecho de dejar *FE1* y *FE5* en el entorno permite apreciar que no se desea describir el comportamiento de *FE1* y *FE5*.



Ejemplo 12-1/Sup. 1 a la Rec. Z.100 – Versión SDL del ejemplo de la figura 12-1

La distinción entre relación y tipo de relación puede modelarse utilizando las definiciones de lista de señales de los diagramas de estructura SDL. *B* y *D* son tipos de bloques.

Las principales ventajas de utilizar los diagramas de estructura SDL radican en que éstos contienen las definiciones de señales, tipos de datos, etc. que son importantes para los ulteriores pasos de la Recomendación Q.65. En el ejemplo 12-1, estas definiciones se identifican mediante las macros *Defs*.

Comportamiento

El comportamiento se describe en los pasos 2-4 de la Recomendación Q.65. La interacción entre entidades funcionales se describe mediante diagramas de flujo de información. En base a ellos se prepara un diagrama SDL (es decir, un diagrama de proceso) para cada tipo de entidad funcional.

Los diagramas de flujos de información se corresponden con el MSC [3]. Los diagramas de flujos de información se producen para casos "normales", utilizándose SDL para describir el comportamiento de las entidades funcionales. Esto ocurre en las actividades de Diseño Previo y de Formalización de la metodología de este Suplemento.

El paso 4 de la Recomendación Q.65 hace referencia a la definición de una serie de acciones básicas para cada entidad funcional. La idea es reutilizar dichas acciones básicas en distintas especificaciones de servicio.

Los elementos del diagrama son los siguientes:

– nombres en la parte superior de las columnas (por ejemplo, FE1): entidades funcionales;

– nombres en minúsculas entre los círculos (por ejemplo, *rj*): relaciones entre tipos de entidades funcionales;

– nombres sobre flechas entre columnas (por ejemplo, *ESTABLISH X req.ind*): flujo de información;

– nombres entre paréntesis junto a los nombres de los flujos de información (por ejemplo, *location*): información adicional transportada por el flujo de información;

– nombres en las columnas (por ejemplo, *Action B, 100*): nombre de acciones de entidades funcionales básicas;

– nombres entre paréntesis [por ejemplo, *(5)*]: etiquetas de los diagramas SDL.

El ejemplo 12-2 muestra el MSC que corresponde al diagrama de flujo de información de la figura 12-2. La correspondencia entre los dos es la siguiente:

– nombre de entidad funcional → instancia de proceso (una instancia de proceso por bloque)

– flujo de información → mensaje;

– información adicional → parámetros del mensaje;

– acción de entidad funcional básica → acción;

– etiquetas a los diagramas SDL → comentarios.

Los diagramas de los flujos de información recomendados en la Recomendación Q.65 contienen básicamente la misma información que los MSC. Las especificaciones SDL pueden en general ser confrontadas con los MSC durante la simulación y en algunos casos simples (sin creación dinámica de procesos, sin direccionamiento dinámico de las salidas) esta comprobación puede realizarse fácilmente sin necesidad de simulación. Los MSC pueden obtenerse a partir de las especificaciones SDL, pero normalmente un MSC sólo pretende mostrar un subconjunto del comportamiento ("el comportamiento normal"), por lo que se necesita cierta interacción humana para obtener los MSC adecuados a partir de la especificación SDL. Otra posibilidad es obtener de forma automática esquemas básicos de los diagramas de proceso SDL a partir de los diagramas de flujo de información. Véase 15.2.2, **Paso B:2 – Procesos del esquema básico**.

Datos

En los diagramas SDL existentes para la RDSI las **tareas** (**tasks**) y las **decisiones** (**decisions**) son fundamentalmente informales. Por lo tanto, no es realmente necesario definir operadores para tipos de datos. Los ítems de datos aparecen principalmente en las construcciones **input**, **output** y **timer** (véase el ejemplo 12-3).

En la mayoría de los casos, son suficientes *Boolean* e *Integer*, por ejemplo, para transportar el valor de una bandera o un contador. También se utiliza ampliamente *Charstring*, debido a que permite manejar los parámetros de forma muy parecida a texto informal.

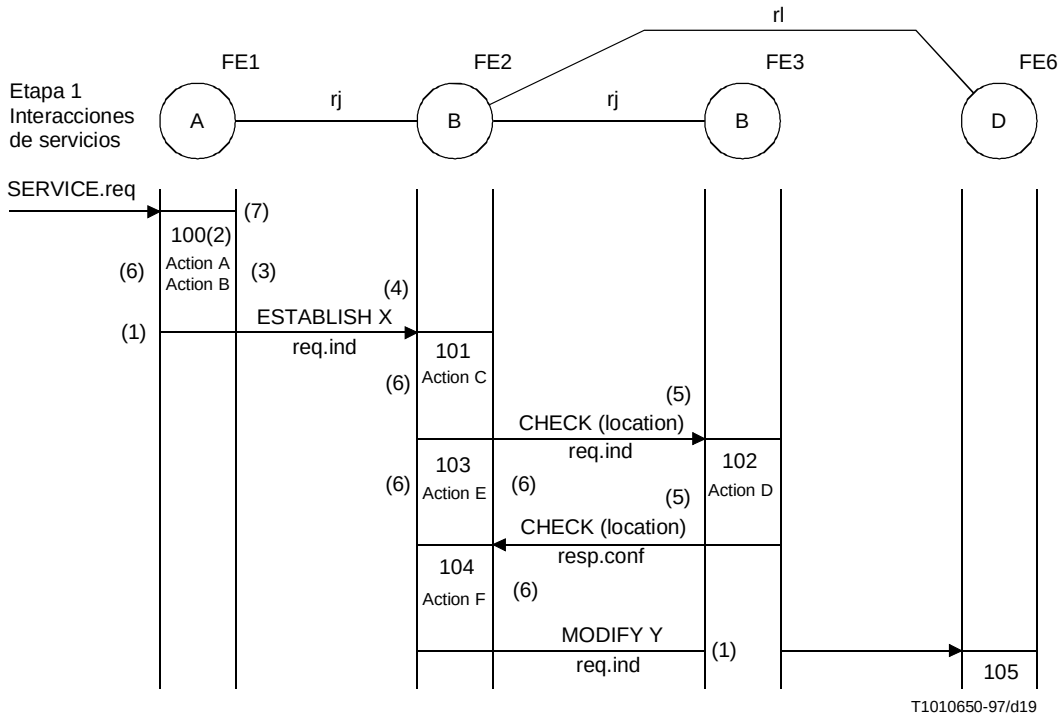
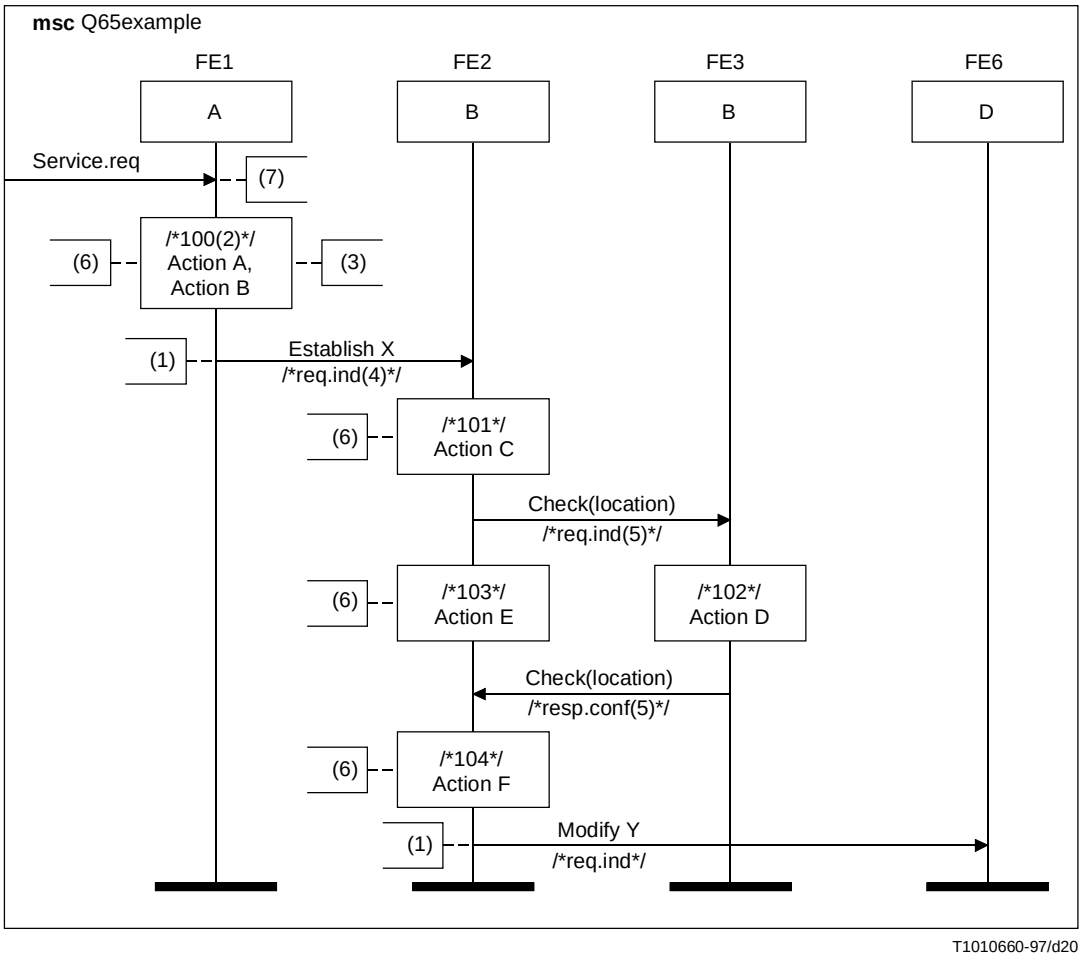
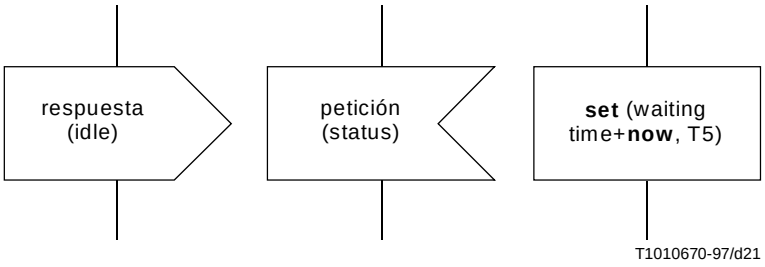


Figura 12-2/Sup. 1 a la Rec. Z.100 – Ejemplo de diagrama de flujo de información de la Recomendación Q.65



Ejemplo 12-2/Sup. 1 a la Rec. Z.100 – Versión MSC del ejemplo de la figura 12-2



Ejemplo 12-3/Sup. 1 a la Rec. Z.100 – Utilización de datos

Puede ser de utilidad usar tipos enumerados. Un **newtype** (**neotipo**) sólo con literales se corresponde con un tipo de datos enumerados (como por ejemplo en Pascal):

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

Los ítems de datos se incluyen pasos 1-4 cuando se formalizan las especificaciones, por ejemplo cuando las especificaciones de señales se aplican a los tipos de datos. Los datos pueden introducirse con varios niveles de refinamiento. La ventaja de la introducción de datos es que puede comprobarse que los parámetros de entrada y de salida, las cuestiones y las repuestas, son consistentes a lo largo de toda la especificación SDL.

13 Pasos del Análisis

La actividad de Análisis consta de dos pasos: la Inspección y la Clasificación. No es necesario inspeccionar todo el dominio de aplicación antes de comenzar el paso de clasificación. Los conceptos de aplicación o los requisitos que han sido inspeccionados pueden ser clasificados mientras que otros conceptos o requisitos están aún siendo inspeccionados.

Las actividades de Inspección y Clasificación pueden producir dos visiones de la aplicación:

– una visión lógica para describir elementos clave del sistema que debe especificarse;

– una visión dinámica para describir el comportamiento de todos los objetos.

La visión lógica puede realizarse mediante un modelado componente-relación (llamado *modelado de objeto*). Ello se corresponde con la visión de información del sistema. Con el fin de facilitar la reutilización – incluir componentes de dominio externos así como crear nuevos componentes de dominio reutilizables – se deben utilizar preferentemente técnicas de Análisis orientado a objetos (OOA, object oriented Analysis) tales como OMT [11], OOSE [12] o SOON [6]. El concepto de encapsulación de datos y de servicios y el concepto de herencia facilitan especialmente la reutilización. Estos dos conceptos permiten la reutilización de componentes más generales o de componentes con un comportamiento parecido al esperado sin tener que modificarlos. La especialización de componentes generales o la adaptación de componentes parecidos se realiza incluyendo nuevos componentes adecuados. Como consecuencia de ello, las técnicas OOA son muy útiles para la creación y reutilización de componentes de dominio (es decir, componentes adecuados a un dominio). Los pasos del análisis se explican en detalle utilizando el enfoque OMT, pero podría utilizarse cualquiera de las restantes técnicas OOA.

El MSC es perfectamente adecuado para modelar una visión dinámica (o comportamiento) que se corresponda con una visión computacional inicial del sistema. Esta tarea consiste en definir secuencias de uso esperadas por el usuario del sistema. Dichas secuencias de uso se componen generalmente de los escenarios típicos que conforman el comportamiento habitual del sistema y de escenarios excepcionales que describen la respuesta del sistema a entradas fallidas, fallos en general, etc. Pueden añadirse otras secuencias de uso a fin de especificar aspectos particulares del comportamiento, por ejemplo, cuando el sistema arranca o cuando se detiene. Este modelado es semejante al modelado de uso definido en [12].

En esta cláusula también se describen en detalle algunas normas de consistencia que deben cumplirse cuando se realizan en paralelo el modelado de objeto y el modelado de secuencias de uso.

La actividad de Análisis debe realizarse sin hacer ninguna hipótesis sobre la arquitectura de la aplicación o sobre detalles de la implementación. En particular, los datos deben modelarse de acuerdo con los requisitos establecidos desde el punto de vista del usuario y no desde el punto de vista del ingeniero de sistemas que trata de optimizar la implementación del mismo. En este sentido resulta útil un modelo de empresa que identifique claramente quienes son los usuarios y que cometido tienen.

13.1 Pasos de la inspección

En 5.4.1 se presentan las distintas tareas que constituye el paso de inspección. Dichas tareas no se soportan específicamente mediante técnicas de ingeniería de sistemas. Los ingenieros pueden utilizar preferentemente un procesador de texto para preparar la bibliografía y redactar las notas para el lector.

13.2 Pasos de la clasificación para el modelado de objetos

El modelado de objetos se ocupa de dos categorías de objetos:

– los objetos físicos: modelos de entidades físicas que componen el entorno del sistema o que son utilizadas por el sistema para proporcionar los servicios previstos;

– objetos lógicos: modelos de la información consumida, producida o almacenada en el sistema, o bien, modelos de conceptos de aplicación.

Los objetos que modelan entidades del entorno (tanto físicas como lógicas) se denominan *agentes*. Muchos de los objetos aplicables a las telecomunicaciones (tales como llamadas, rutas, cuenta de abonado) son más bien entidades lógicas que físicas. Esto es particularmente cierto para el sistema de especificación de una norma.

Las *asociaciones* son relaciones de dependencia entre objetos, tales como:

– conexiones físicas entre objetos físicos;

– relaciones productor/consumidor entre objetos (físicos o lógicos);

– relación de uso entre objetos (físicos o lógicos);

– enlaces de *agregación* entre objetos (físicos o lógicos);

– enlaces de herencia para permitir la reutilización de objetos más generales o parecidas.

Una *agregación* es una asociación entre un objeto compuesto y los objetos componentes, que en ocasiones se denomina refinamiento.

El modelado no está relacionado con cómo ni cuando se genera la información o éste se pasa de un lugar a otro, ni tampoco con cómo ni cuando interaccionan las entidades físicas. Es un modelo estático (o estructurado).

Los modelos de clase se producen para que el modelo sea general y pueda aplicarse a distintas instancias de la aplicación. Una clase X (por ejemplo, un teléfono) define las características que son aplicables a todos los objetos X (es decir, a todos los teléfonos). Los modelos de clases muestran las asociaciones (relaciones) entre clases. En una aplicación real, los objetos (instancias de clases) deben ser conformes con el diagrama de la clase. Por ejemplo, si la clase "central" está "conectada a tres o más" (una asociación) clases "teléfono", una central real siempre tendrá conectados al menos tres teléfonos. Aunque es posible dibujar modelos de instancias de objetos, no es algo que se realice habitualmente.

La notación utilizada es la notación del modelo de objeto de OMT [11], cuyos elementos básicos se muestran en la figura 13-1. La figura 13-2 muestra un diagrama parcial en el que el dominio de aplicación es el de las operaciones bancarias con tarjetas monedero (cash cards).

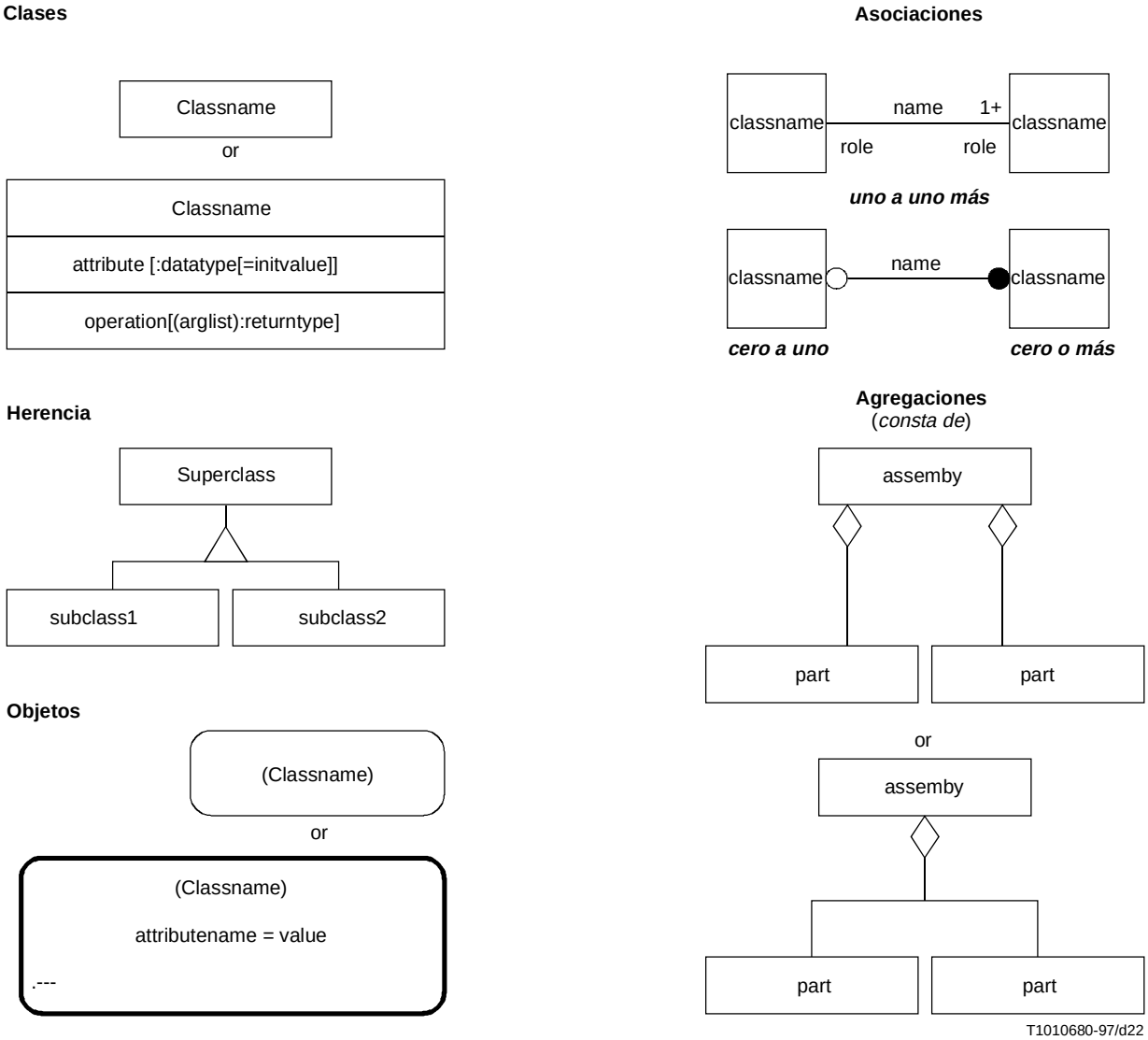


Figura 13-1/Sup. 1 a la Rec. Z.100 – Elementos básicos de la notación del modelo de objeto

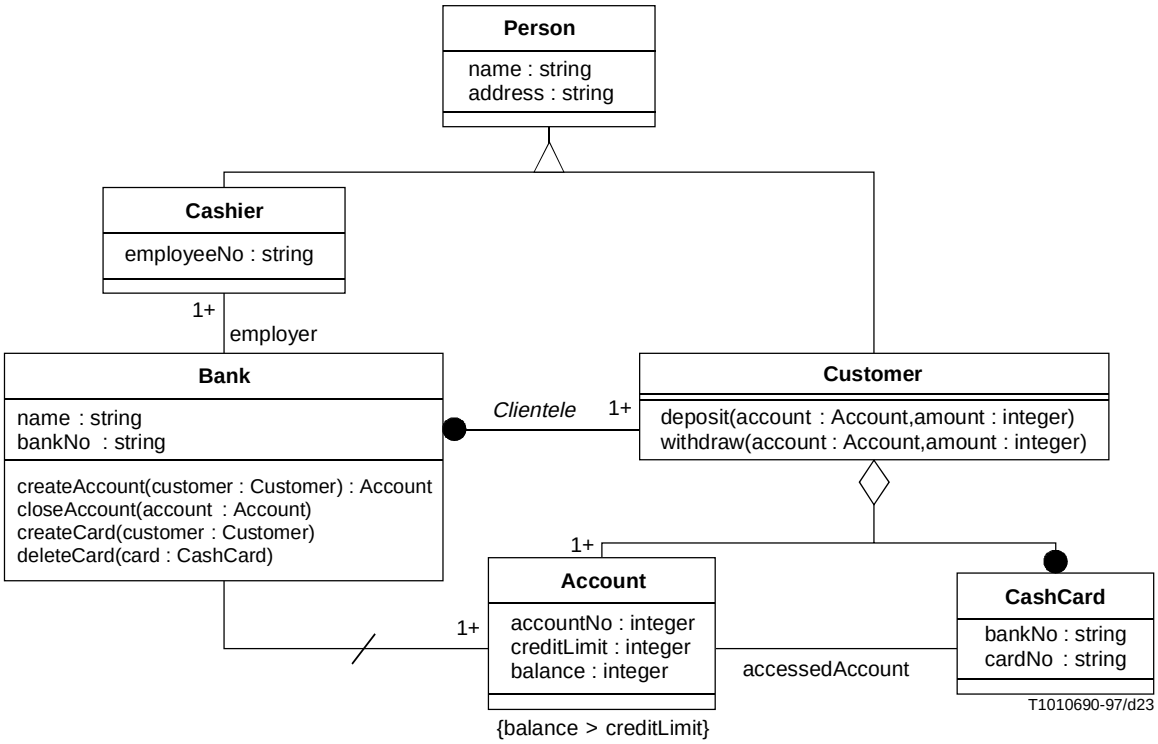


Figura 13-2/Sup. 1 a la Rec. Z.100 – Ejemplo de notación de modelo de objeto

– Clases (por ejemplo, Bank): Contienen atributos (por ejemplo, name y bankNo) y operaciones (por ejemplo, createAccount y closeAccount); es facultativo tipificar los atributos y se pueden declarar las signaturas de las operaciones.

– Asociaciones (por ejemplo, Clientele entre Bank y Customer): Una asociación no tiene un sentido preferente; puede nombrarse mediante un nombre o mediante sus dos cometidos para nombrarla (los cometidos de las dos clases ligadas mediante esta asociación); las multiplicidades se indican para cada cometido de la asociación.

– Agregaciones (por ejemplo, entre Customer y Account y entre Customer y CashCard): Una agregación es una asociación especial que modela un enlace de refinamiento; es la forma de describir la estructura jerárquica.

– Enlaces de herencia (por ejemplo, entre Person y Cashier y entre Person y Customer): Es el soporte para la reutilización de las propiedades de clase.

Los aspectos relativos a la información de una clase están constituidos por los atributos de la misma, la multiplicidad de asociaciones que tiene con otras clases y en las asociaciones de agregación en las que está presente la clase.

Los aspectos relativos a las interfaces de una clase están constituidos por las asociaciones que tiene con otras clases que no constituyen asociaciones de agregación, junto con las signaturas de las operaciones de la clase que pueden ser invocadas por otras clases.

En general, el modelo de clase que se produce durante el Análisis no detalla aspectos de comportamiento, los cuales permanecen como texto en los requisitos recopilados, posiblemente enlazados (de alguna forma) con las descripciones de clase cuando ello sea conveniente. Los aspectos misceláneos de una clase (de tipo general: flexibilidad, compatibilidad, usabilidad, fiabilidad, seguridad, protección; Constricciones del diseño; Modos de fallo; Calidad de funcionamiento y otros) también permanecen como texto. Sin embargo, en la medida de lo posible, los textos sobre el comportamiento y los aspectos misceláneos debieran estar enlazados con las clases producidas, de forma que sea posible determinar cual es el aspecto que maneja cada objeto.

Con el fin de gestionar la complejidad de un diagrama de clase y preservar su legibilidad, éste se estructura en módulos. Los módulos normalmente representan diferentes vistas del sistema, tales como el punto de vista del usuario, el del entorno físico y el punto de vista de un servicio que debe proporcionar el sistema. El conjunto de todos estos módulos constituye el diagrama de clase completo. Las clases se pueden representar, y así se hace a menudo, en varios módulos. Por ejemplo, en el dominio de las operaciones de la tarjeta monedero, los módulos significativos son: la descripción de un cliente desde el punto de vista del banco (cuenta, tarjeta monedero, etc.), la red de cajeros automáticos (punto de vista del consorcio de bancos), la composición de un cajero automático desde el punto de vista del equipo y un módulo que describa cuales son las entidades físicas y los conceptos involucrados cuando se realizan las transacciones bancarias.

Instrucciones

1) Identificar y denominar las clases y crearlas en el modelo.

2) Identificar y denominar las asociaciones y las agregaciones entre clases.

3) Identificar y denominar los atributos de clases.

4) Identificar y denominar las operaciones de clases.

5) Organizar y simplificar las clases mediante la herencia.

6) Agrupar las clases en módulos.

7) Verificar que existen trayectos de acceso para las preguntas habituales así como todos los datos requeridos.

8) Iterar y refinar el modelo.

Directrices

Identificación de las clases

El análisis de los requisitos de la aplicación permite identificar objetos físicos y lógicos. En general, las clases se identifican inicialmente mediante el examen de los nombres utilizados en los documentos de texto que se recopilan los requisitos. Puede ser útil crear un diccionario de datos que liste todas las clases, junto con una descripción tomada u obtenida a partir de los requisitos capturados. Alternativamente, pueden utilizarse enlaces de hipertexto hacia los requisitos recopilados. Es importante determinar los límites entre las clases que se encuentran en el sistema y las clases exteriores al mismo. Esta distinción puede reflejarse en los nombres dados a los agentes exteriores al sistema. Es conveniente dar un nombre al sistema.

Denominación de clases (asociaciones, atributos y operaciones)

La elección de nombres es una decisión importante: con frecuencia en los requisitos recopilados existen varios términos para una clase de objetos (asociaciones, atributos u operaciones) y una mala elección del nombre puede, posteriormente, causar confusión. No obstante, los nombres elegidos deberían revisarse ulteriormente y quizás ser modificados debido a que la comprensión del sistema habrá mejorado después de que se hayan realizado algunos trabajos de ingeniería.

Mantenimiento de las clases correctas

Las clases sólo deberían generarse a partir de los requisitos recopilados y no deberían introducir detalles arquitectónicos que no formen parte de los requisitos recopilados, es decir, la única constricción arquitectónica que debe considerarse se refiere a las condiciones bajo las cuales debe ejecutarse el sistema. En consecuencia, las clases que representen entidades físicas deberían limitarse a las entidades relacionadas con el dominio de aplicación y no con la implementación de la aplicación. Las entidades lógicas deberían representar conceptos de aplicación comúnmente aceptados, siendo de ayuda para decidir si una clase es la adecuada el disponer de una biblioteca de conceptos así como la experiencia en general. Por ejemplo, una base de datos constituida por una lista de clientes con sus nombres, direcciones y la lista de productos que han comprado puede modelarse mediante la clase "Cliente" con dos atributos, "nombre" y "dirección" y la clase "producto", sin crear una tercera clase que represente la base de datos. Deben eliminarse las clases redundantes (las que representan el mismo concepto, la misma entidad física o que contienen los mismos atributos). Igualmente deben eliminarse las clases vacías, es decir, clases sin atributos y sólo ligeramente relacionadas con otras clases del modelo.

Identificación de asociaciones

Cualquier dependencia entre clases se modela como una asociación. Una asociación puede representar un enlace físico, una relación productor/consumidor o una relación de uso (las agregaciones se detallan más adelante). Las asociaciones se identifican inicialmente examinando los verbos (los cuales "enlazan" nombres, como por ejemplo "empleos" o "accesos"). La mayoría de las asociaciones son binarias, es decir, sólo involucran dos clases (existe una notación para asociaciones múltiples si fuera necesario). No debe haber una preocupación excesiva con respecto a la denominación de las asociaciones debido a que puede haber muchas que se correspondan a la propiedad ["tiene" ("has") "ostenta la propiedad de" ("owns"), ...] y a las conexiones ["unido a" ("joined to"), "enlaces con" ("links to"), "direcciones" ("addresses")...] y que no necesitan nombres ya que la asociación es obvia a partir del modelo de clase.

Identificación de agregaciones

Las asociaciones que representen un enlace de propiedad o un enlace de composición normalmente se modelan mejor como agregaciones. Una agregación añade una nueva constricción en comparación con una asociación simple: el comportamiento del objeto que se agrega tiene una influencia importante en el comportamiento de objetos agregados, en particular la creación y la supresión de objetos que se agregan conduce en general a la creación y a la supresión de sus objetos agregados. Las agregaciones se crean transformando asociaciones que se denominan "parte de" ("part of"), "compuesta por" ("composed by"), "elemento de " ("element of"), etc.

Mantenimiento de las asociaciones y agregaciones correctas

Las asociaciones redundantes pueden ser eliminadas o explícitamente marcadas como tales (la notación de asociación "derivada" ("derived"); véase en la figura 13-2, la asociación entre "Account" y "Bank"). Las interacciones entre clases se modelan como asociaciones sólo si representan propiedades estructurales del dominio de aplicación y no sucesos transitorios. La multiplicidad de las asociaciones (número involucrado en cada parte) debería definirse adecuadamente.

Identificación de los atributos correctos

Los atributos son propiedades de las clases. Se utilizan para capturar los datos definidos en los requisitos recopilados, tales como nombre, dirección, número de tarjeta, etc. Deben eliminarse los detalles del diseño o de la implementación, tales como el Pid del proceso. Los atributos complejos (datos estructurados, datos de tamaño ilimitado, etc.) pueden transformarse en clases (de datos). Los atributos pueden clasificarse en tipos, pero debieran evitarse las descripciones de tipos sofisticados si con ello se hace referencia a detalles de la implementación. Los atributos no pueden ser punteros hacia otras clases, debiéndose utilizar para ello las asociaciones.

Identificación de las operaciones correctas

Las clases se enriquecen mediante la adición de operaciones. Las operaciones no se identifican fácilmente cuando realizan exclusivamente el modelado de objetos. El modelado de la visión dinámica de los requisitos ayuda notablemente a identificar las operaciones que deben proporcionar las clases. En consecuencia, los ingenieros deben modelar la visión dinámica en paralelo con la visión lógica. Las operaciones están ligadas a los servicios (incluyendo el acceso o la generación de datos) que la clase proporciona al entorno o a las clases con las que está conectada por medio de asociaciones. También pueden definirse las signaturas de las operaciones (parámetros formales y valores de retorno).

Utilización de la herencia

Las clases que tienen una estructura parcialmente en común (atributos o asociaciones) pueden ser reorganizadas mediante una nueva clase que encapsule dicha subestructura común. Las clases iniciales se convierten entonces en subclases especializadas de la superclase. La introducción de superclases puede también estar originadas por la reutilización de clases ya definidas que están disponibles en los dominios de aplicación. Sin embargo, debiera evitarse la existencia de muchos niveles de herencia.

Agrupación de las clases en módulos

Los módulos se crean para representar puntos de vista significativos del dominio de aplicación. Las clases se agrupan en módulos. Pueden estar presentes en más de un módulo y normalmente éste es el caso porque los puntos de vista no constituyen una partición del sistema. Para mejorar la legibilidad de los diagramas, el ingeniero no debiera crear más de doce clases en un módulo. El número ideal de clases de un módulo está comprendido entre 4 y 8 en función de la complejidad de cada clase, del número de atributos, del número de operaciones y del número de asociaciones.

Verificación de los trayectos de acceso y la cobertura de los datos

El modelo de clase puede ser contrastado con los requisitos. Deben verificarse dos aspectos: la posibilidad de navegar a través del modelo para acceder a la información y la cobertura de los datos. Si no se soporta una pregunta sobre la navegación aunque sea indirectamente a través de un conjunto de asociaciones, significa que deben crearse nuevas asociaciones. El modelo dinámico es muy útil para comprobar si se han completado los trayectos de acceso porque muestra cuales son las entidades que interaccionan entre sí y la información a la que accede cada entidad (información transmitida mediante mensajes). Si un conjunto de datos requeridos no está presente en el modelo como atributo, como conjunto de atributos o como resultado de una operación, deben aplicarse nuevos atributos u operaciones.

Refinamiento del modelo

El modelado de un objeto es un proceso iterativo. El diagrama de clase inicial es muy parecido a los requisitos recopilados (los nombres como clases, los verbos como asociaciones, etc.). Durante sucesivas iteraciones el modelo se reorganiza para ser inequívoco, carezca de redundancias, y sea fácilmente comprensible y completo respecto a los requisitos recopilados. La identificación de los atributos y las operaciones de cada clase se refinan en ulteriores iteraciones. Se deberá evaluar con criterios de ingeniería como completar el modelo. El modelado dinámico enriquece el modelo del objeto, en particular en lo que se refiere a las operaciones y asociaciones de clase. Por lo tanto, las iteraciones sobre el modelo de objeto deberían realizarse en paralelo con las iteraciones del modelo dinámico. Dicho refinamiento puede considerarse más adecuado de realizar durante la actividad de Diseño Previo, pero al ser ésta facultativa, puede considerarse como parte de la Clasificación.

Regla 1 – Para nombrar clases no deben utilizarse nombres en plural (existe una clase y varias instancias).

Regla 2 – El modelo de objeto debe contener, mediante clases, todos los agentes relevantes al dominio del entorno del sistema.

Regla 3 – El modelo de objeto debe contener, mediante atributos u operaciones, todos los datos expresados en los requisitos recopilados.

Regla 4 – El modelo de objeto debe cumplir, mediante asociaciones y agregaciones, todos los requisitos en materia de navegación a través de objetos.

Resultado: Un modelo de objeto coherente y completo que recoge mediante diagramas la visión lógica de los requisitos recopilados.

13.3 Pasos de la clasificación para el modelado de la secuencia de uso

Los objetivos del modelado de la secuencia de uso son capturar y clasificar el comportamiento que se espera tenga el sistema en respuesta a estímulos procedentes de su entorno, sin hipótesis previas sobre como se consideran los estímulos dentro del sistema y sobre como se preparan las respuestas. El comportamiento se clasifica mediante un conjunto de MSC constituido normalmente por secuencias de uso típicas (de hecho, la misión del sistema) y por secuencias de uso excepcionales que describen la robustez esperada. Se pueden añadir otras secuencias para clarificar estados específicos del sistema (arranque, parada, etc.).

Debido a que el modelado de la secuencia de uso tiene por objeto clasificar el comportamiento esperado del sistema durante el Análisis, no es necesario producir diagramas de transición de estado. Dichos diagramas se crean en los pasos de Diseño Previo o de Formalización.

Las secuencias de uso descritas en este paso ignoran la arquitectura interna del sistema. Los únicos elementos involucrados son: el sistema, representado como una instancia, y todos los agentes implicados en la comunicación con el sistema. Estos agentes se corresponden con entidades dinámicas tales como un usuario o un dispositivo que interactúan con el sistema. Por otro lado, cuando la arquitectura del sistema se representa de forma simplificada en el Diseño Previo, las secuencias de uso producidas en el Análisis pueden refinarse de acuerdo con ello para producir secuencias de uso que reflejen los componentes internos del sistema.

Una de las mayores dificultades identificadas es la gestión con éxito del gran número de MSC que se crean. El conjunto de MSC creados cubre todos los escenarios de los requisitos recopilados, debiéndose evitar la redundancia para disponer de un conjunto mínimo. A este fin, los ingenieros deberían prestar una atención especial a la estructura del documento de los MSC.

Instrucciones

1) Identificar las secuencias de uso nominales y excepcionales que han de crearse.
2) Descomponer las secuencias de uso complejas en otras más pequeñas e indicar cómo deben agruparse las secuencias de uso de nivel inferior en las MSC de alto nivel (HMSC, *high-level MSC*).
3) Para cada secuencia de uso de bajo nivel, dibujar el MSC con el sistema y todos los agentes relevantes de la secuencia en forma de instancias (barras verticales).
4) Describir la secuencia prevista en forma de mensajes del MSC (en función de los requisitos) de los flujos de datos y de control.
5) Verificar que todas las secuencias de uso típicas y excepcionales recopiladas en los requisitos quedan cubiertas mediante el documento MSC.
6) Verificar la consistencia del modelo dinámico tal como se describe en el documento MSC, con el modelo del objeto.

Directrices

Identificación de las secuencias de uso requeridas

Las secuencias de uso que deben producirse se identifican a partir de los requisitos relacionados con los aspectos de comportamiento del sistema. A menudo son de interés secuencias particulares ["¿Qué ocurre si...?" ("What if...?")], que no fueron consideradas en los requisitos recopilados. En este caso los ingenieros pueden elegir una secuencia adecuada y confirmarlo formulando una pregunta sobre los requisitos recopilados.

Organización de las secuencias de uso

Normalmente el comportamiento esperado de un sistema está constituido por un conjunto de variantes de determinadas secuencias nominales. Dichas variantes representan todas las excepciones que pueden tener lugar y deberían considerarse cuando se están ejecutando las secuencias nominales.

A fin de evitar la redundancia entre secuencias descritas y para facilitar la reutilización de partes de las secuencias, el comportamiento esperado debe dividirse en MSC pequeños y reutilizables. En MSC-96 existen diversas formas de combinar las MSC.

– Las referencias a MSC se utilizan para hacer referencia a otros MSC desde un MSC.

– Las secuencias de uso complejas deberían describirse mediante MSC de alto nivel – un grafo dirigido de referencias MSC.

– También pueden describirse combinaciones de MSC como expresiones MSC dentro de referencias MSC; por ejemplo "MSC1 **seq** (MSC2 **alt** MSC3)".

– Ligeras variaciones en las que, por ejemplo, se elige entre mensajes alternativos, y que se conocen mejor mediante expresiones en línea.

Los MSC simples deben nombrarse con cuidado para que proporcionen una idea sobre como pueden combinarse. Por ejemplo, "no_answer_form_third_party" o "caller_hangs_up".

Identificación de instancias de un MSC

Las instancias de MSC que figuran en una secuencia de uso deben ser instancias de agentes definidos como clases en el modelo de clase y en el sistema de aplicación. Normalmente las instancias del MSC corresponden a entidades físicas. Los objetos lógicos del entorno se corresponden normalmente a elementos pasivos que se transmiten como parámetros a los mensajes del MSC.

En un MSC, el sistema tiene el aspecto de barras verticales. Puede haber tantas instancias del sistema en el MSC como instancias concurrentes comunicándose entre si en el mundo real, aunque es más habitual considerar sólo un sistema. Por ejemplo, en el caso de una red de sistemas interconectados, el MSC contiene varias instancias del sistema. Igualmente, un agente puede presentar múltiples instancias en un MSC si dicha circunstancia ocurre en el mundo real. Las distintas instancias del mismo agente deben ser cuidadosamente denominadas para que el MSC sea comprensible; por ejemplo, "Caller" (llamante) y "Called" (llamado) en un MSC que incluye dos abonados que están dialogando. Es habitual limitar el número de instancias del MSC al número mínimo requerido para representar cual es el comportamiento en el mundo real.

En el Análisis no es necesario describir formalmente los medios (canales de comunicación e identificación de instancias) utilizados por los agentes y el sistema para interaccionar. Los ingenieros deben considerar que puede accederse a las instancias independientemente, sin problemas de identificación y de conexión.

Identificación y dibujo de los mensajes

Los mensajes se corresponden normalmente a operaciones de clases en el modelo de objeto, correspondiendo habitualmente sus parámetros a atributos de clase o a resultados de operaciones de clase.

Para cumplir los criterios de accesibilidad de la información, debieran existir en el modelo de objeto trayectos (directos o indirectos) entre las instancias del MSC que están dialogando. Sin embargo, las asociaciones del modelo de objeto no dan lugar necesariamente a mensajes de los MSC.

Verificación de la compleción del modelo dinámico

Cada posible escenario debiera estar cubierto por un MSC. El modelo dinámico se completa normalmente cuando se abarcan todos los posibles escenarios. No obstante, normalmente es imposible alcanzar un estado de compleción total debido a que el número de posibles secuencias de uso es muy alto. Las iteraciones del modelo dinámico finalizarán cuando los ingenieros consideren que se ha producido un número razonable de secuencias de uso. Dicha consideración es subjetiva y es función del sistema.

Verificación de la consistencia entre modelo dinámico y el modelo de objeto

Debe realizarse una comprobación del modelo dinámico en relación con el modelo de objeto, debiendo añadir a este último la información que no está presente a fin de que se cumplan las reglas siguientes:

1) Las instancias de MSC deben corresponder al sistema o a los objetos de agentes (instancias de clases).

2) Los mensajes deben corresponder a operaciones de clases. Normalmente un mensaje que se envía desde una instancia de "a" a una instancia de "b" se declara en la clase "b" indicando que "b" proporciona este servicio a "a". En ocasiones, una operación relacionada puede también ser declarada en "a" indicando que "a" proporciona este servicio a otras clases que no desean acceder directamente a "b" (por ejemplo, porque "b" proporciona un servicio de nivel demasiado bajo).

3) Los parámetros de los mensajes deben corresponder a atributos de clase o a resultados de operaciones de clase. Los atributos debieran declararse en las clases que se corresponden al mensaje o en otras clases a las que se accede mediante asociaciones.

4) Para dos instancias de MSC que estén dialogando, sus clases correspondientes deben conectarse en el modelo de objeto, directamente o indirectamente a través de asociaciones.

La consistencia entre los dos modelos se comprueba durante las últimas iteraciones de los modelos. Es bastante habitual enriquecer el modelo de objeto mediante operaciones que se derivan de los mensajes de los correspondientes MSC.

Regla 5 – Se debe utilizar la notación MSC para mostrar mediante diagramas las secuencias de mensajes intercambiados entre el sistema y su entorno a lo largo del tiempo.

Regla 6 – Las secuencias de uso deben mostrar al sistema como un todo que interactúa con los agentes del entorno, sin considerar la arquitectura interna del sistema.

Resultado: Un modelo de secuencia de uso (documento MSC) consistente con el modelo de objeto que recoge mediante diagramas la visión dinámica de los requisitos recopilados.

14 Pasos del Diseño Previo

Los pasos que se detallan en 14.1 a 14.6 se presentan en un orden que resulta conveniente para iniciar su ejecución, no obstante pueden realizarse en cualquier orden excepto cuando la información de un modelo es necesaria en otro modelo. Los pasos pueden asimismo realizarse en paralelo debido a que cada uno se centra en una forma distinta de modelado. Cada modelo puede abarcar todo el sistema o parte del mismo.

Debido a que el número de diseños previos necesarios varía de forma significativa, es posible indicar solamente los diseños previos que son más útiles para la Formalización. Ello se resume en el cuadro 1.

Cuadro 1/Sup. 1 a la Rec. Z.100 – Utilidad de los diseños previos en la Formalización

Modelo de Diseño Previo	Utilización
Modelado de relaciones de componentes	A veces es útil establecer cuantos bloques, procesos o ítems de datos hay
Diagramas de flujo de control y de datos	Diagrama de contexto a menudo útil. Jerarquía de diagramas útil para ASN.1
Estructura de la información	Necesita hacerse, bien como Diseño Previo o como parte de la Formalización
Modelado de la secuencia de uso	Las secuencias de uso se utilizan normalmente para diseño de procesos. Es útil para partes informativas de la aplicación
Modelado del comportamiento del proceso	Sólo se realiza de forma ocasional. Se utiliza como base de diagramas formales
Modelado de la visión general del estado	Útil para la comprobación de procesos, pero no precisa estar en la aplicación

La utilización de cuadros de bits para modelar la visión de la información puede resultar inicialmente atractiva, pero tiene desventajas notables. Los cuadros de bits pueden confundir la definición de la estructura de información con la codificación de información. Asimismo tienden a desenfatizar el objetivo de cada elemento de datos. En ocasiones, en los cuadros de bits hay elementos de datos importantes que no reciben nombres con un significado comprensible debido a que se representan mediante un solo bit o mediante unos pocos bits. Los cuadros de bits no son flexibles, en parte porque mezclan significado y estructura con codificación: un cambio en el medio de transporte puede requerir un cambio de codificación pero sin ningún otro cambio semántico. Los cuadros de bits pueden ser mal interpretados cuando no está claro si el bit de mayor numeración (o más a la izquierda) es el más o el menos significativo, se alinea a izquierda o a derecha o se transmite primero o último. La presentación de los cuadros de bits no está normalizada. Los cuadros de bits no soportan la composición, la investigación analítica ni la comprobación semántica, cosa que por el contrario soportan bien las herramientas para ASN.1 y SDL.

Regla 7 – Los cuadros de bits no deben utilizarse para modelar la visión de la información.

NOTA – Los cuadros de bits pueden utilizarse para describir la codificación.

14.1 Modelado de la relación de componentes

El objetivo de este paso es completar el modelo de clase producido en el Análisis para describir la arquitectura interna del sistema. Este modelo se utiliza para el modelado del flujo de control y de datos, el modelado de la información, el modelado de la secuencia de uso y la Formalización. El detalle que se añade al modelo de clase se corresponde con la arquitectura interna del sistema: los aspectos de la información y el diseño de asociaciones. No obstante, este modelado se considera parte del Análisis si el Diseño Previo no se incluye como actividad. Este modelado puede realizarse de dos formas: como un refinamiento del modelo de clase del Análisis o sustituyendo clases con clases alternativas más detalladas.

Instrucciones

1) Definir los géneros de datos de los atributos de clase y de las operaciones de clase declaradas en el modelo de objeto del Análisis.

2) Diseñar las asociaciones de clases.

Estas actividades enriquecen el modelo de clase introduciendo nuevos atributos, nuevas asociaciones, nuevos enlaces de herencia o nuevas clases.

Directrices

Tipificación de atributos y de operaciones

En el Análisis, los géneros de los datos utilizados en atributos y operaciones deberían ser simples o bien nombres no refinados. No debieran utilizarse listas, cuadros o estructuras complejas. Si son relevantes para el Análisis, es decir si capturan conceptos de aplicación, deberían modelarse como clases y asociaciones, en otro caso, se ignoran los ítems de información de datos complejos. En el Diseño Previo, los géneros de los datos deben definirse con mayor precisión, aunque la descripción de todos los géneros de datos del modelo de clase puede ser un esfuerzo redundante si se compara con otros pasos, tales como el modelado del flujo de datos o de la estructura de la información. Por lo tanto, el modelado de los géneros de los datos debiera limitarse a los atributos y operaciones más importantes y que resultan de ayuda en el modelado de flujos de datos sin tener que realizar por ello un trabajo superfluo.

En OMT, los géneros de los datos sólo se declaran al nivel del modelo y son accesibles por todas las clases; los géneros de los datos no pueden declararse localmente a una clase. Los atributos se tipifican mediante la creación de nuevos géneros globales de datos o añadiendo agregaciones o asociaciones en caso de que las correspondientes estructuras de datos complejas contengan clases o accesos a clases. También se pueden añadir nuevas clases al modelo si las estructuras de datos complejas contienen más información que la disponible en el modelo de objeto del Análisis (a través de los atributos).

La adición de géneros de datos para parámetros y resultados de operaciones puede conducir a una reorganización de las jerarquías de la herencia a fin de facilitar la redefinición de operaciones. Las signaturas de las operaciones deben cumplir las jerarquías de la herencia.

Designación de asociaciones

Las asociaciones definidas en el modelo de objeto del Análisis son bidireccionales y permiten el acceso a instancias de clases sin conocer como se realiza. El Diseño Previo debiera explicar como se accede a las instancias. En general para cada asociación debiera elegirse una dirección preferente así como, para la clase destinataria de esta asociación se debería elegir el "atributo clave" de entre los existentes (o bien debiera crearse en caso de que no existiera); éste identifica una instancia de esta clase de entre las restantes instancias. Para solucionar el problema de asociaciones con cardinalidades múltiples a ambos lados pueden añadirse nuevas clases.

Regla 8 – El modelo de objeto se completa utilizando la misma notación que la utilizada para el Análisis.

14.2 Modelado del flujo de control y de datos

Instrucciones

1) Producir un diagrama de contexto.

2) Descomponer el bloque SDL del diagrama de contexto en subbloques utilizando la agregación de nivel superior y la estructura de descomposición a partir de la información clasificada (es decir, el modelo de clase).

3) Repetir la descomposición de los subbloques hasta que no pueda dividirse más el comportamiento de cada bloque o hasta que sea evidente que el bloque corresponde a un solo proceso SDL.

Directrices

El diagrama de contexto es un diagrama de sistema SDL que contiene un bloque sencillo que representa el comportamiento del sistema. Los canales hacia y desde el bloque representan el flujo de información entre el sistema y los agentes del entorno. Estos pueden identificarse en la información clasificada a partir de las asociaciones entre el sistema y los agentes del entorno y a partir de los eventos MSC entre las instancias del sistema y la instancia de agente. Todos ellos constituyen los aspectos de la **interfaz** del sistema. Existe un canal para cada flujo de información identificado de forma separada (normalmente uno por cada instancia de agente en el MSC). Si se requiere que en el entorno se fusionen dos o más flujos de información, éstos se representan conectados al mismo punto en los límites del sistema SDL que se conecta al entorno. Las señales no se adjuntan a los canales, pero el flujo de información se infiere a partir de descripciones realizadas en lenguaje natural.

Si no existe clase en la información clasificada del sistema, puede asumirse que son las clases al nivel superior (no las partes de otras clases, que no se agregan) y no los agentes (clases para objetos del entorno) las que se agregan para constituir el sistema.

Los subbloques que tienen un importante componente de **información** y poco o ningún comportamiento específico (distinto al almacenamiento de información) se consideran como "datos". Nunca debería establecerse una conexión directa entre dos de dichos "bloques de datos". Los canales se identifican a partir de información de la interfaz. Los canales hacia o desde "bloques de datos" constituyen un flujo de datos. Los canales entre otros bloques (comportamiento) constituyen flujo de control. Debería existir una descripción del comportamiento de dichos bloques. Si dicha descripción no existe, se escribe en lenguaje natural.

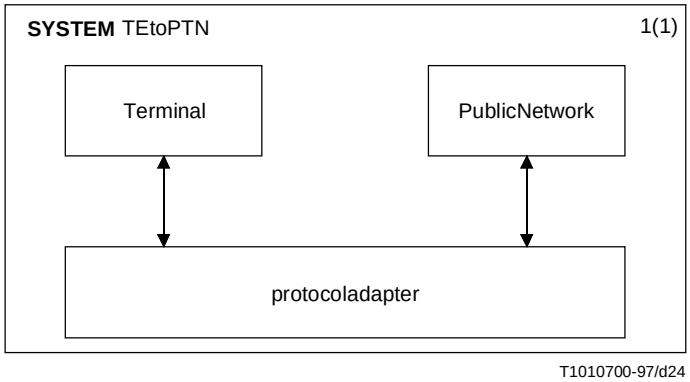
Estas descripciones utilizan SDL para mostrar el flujo de control y de datos pero no muestran el detalle de la señal ni el comportamiento del proceso. Este enfoque permite investigar varias descomposiciones y enfocarse en el flujo de datos y en los almacenes de datos. Cuando un "bloque de datos" tiene conexiones con más de un bloque, debiera reconsiderarse la descomposición ya que los datos deberían estar incluidos en un solo proceso del modelo formalizado.

Se debería tener en cuenta la arquitectura aceptada que se utiliza en el área de aplicación. Por ejemplo, una arquitectura que separe el flujo de información puede constituir un requisito tal que existan varios planos de protocolo: "plano de usuario", "plano de control" y "plano de gestión".

A menudo no es posible especificar todo el contexto utilizando un enfoque arriba-abajo (top-down). En este caso el diagrama del contexto inicial puede necesitar una compilación abajo-arriba (bottom-up) a partir de las especificaciones del proceso.

Regla 9 – Todos los diagramas de contexto deben realizarse en SDL.

Ejemplo



NOTA – Para el diagrama de contexto no se necesita el SDL completo: en este ejemplo los canales no tienen nombre y las señales no están definidas.

Figura 14-1/Sup. 1 a la Rec. Z.100 – Diagrama de contexto para el adaptador de protocolo

14.3 Modelado de la estructura de la información

Las estructuras de información del sistema se modelan utilizando ASN.1 a fin de desarrollar en detalle la visión de información del modelo de clase de la información clasificada. Incluso si la codificación y la estructura de la información se mezclan en los requisitos recopilados, la codificación debe considerarse separadamente de la estructura de la información. Es posible producir un modelo lógico del sistema sin codificación, siendo ésta necesaria sólo cuando se tiene en cuenta la constricción de la correspondencia entre la estructura de información y los bits. Si las reglas de codificación por defecto son satisfactorias, puede omitirse la codificación.

Instrucciones

1) Identificar los flujos de información en los diagramas de flujo de control y de datos menos abstractos (más detallados) o, si la Formalización ha comenzado, los canales procedentes de los bloques que incluyen procesos y rutas de señales.

2) Identificar la estructura a partir de los aspectos de información asociados con los flujos de información y definir un nombre de tipo ASN.1 con significado para cada género de datos que no sea parte de una agregación (mensajes de datos de nivel superior).

3) Identificar y nombrar el siguiente nivel de la estructura de los mensajes de datos de nivel superior, eligiendo el mismo nombre de tipo ASN.1 con sentido si se utiliza la misma información en varios sitios.

4) Definir el tipo de ASN.1 de nivel superior expresado mediante los nombres de tipos de nivel siguiente, repitiendo los dos últimos pasos hasta que cada parte sea un tipo ASN.1 simple.

5) Identificar la estructura a partir de los aspectos de información de los bloques más internos (comportamiento y datos) en los diagramas de flujo de control y de datos menos abstractos (más detallado).

6) Identificar y nombrar los géneros de los datos de dichos bloques de forma similar a los datos para las interfaces, pero reutilizando los tipos de ASN.1 definidos para las mismas.

7) Identificar cualquier valor de los tipos de ASN.1 a los que se haga referencia mediante el nombre, definiendo los nombres ASN.1 para dichos valores comenzando con un tipo ASN.1 simple y reutilizando los nombres de valores cuando sea adecuado en tipos ASN.1 compuestos.

8) Identificar a partir del comportamiento aspectos de los bloques más interiores, identificar los operadores necesarios para los géneros de los datos y registrar éstos como un comentario a la ASN.1.

Directrices

Normalmente existen géneros agregados de datos para distintos mensajes, llamados unidades de datos de protocolo (PDU, *protocol data unit*) para protocolos. Su estructura puede estar descrita de forma general o definida en los requisitos recopilados, y el modelo de clase en la información clasificada.

Los tipos ASN.1 intermedios se eligen de forma que puedan ser utilizados en distintos mensajes. En particular, cuando la información se transforma de un protocolo a otro, a menudo hay algunos elementos de datos comunes que se transfieren desde mensajes de una interfaz a mensajes de otra interfaz.

La agregación se corresponde a tipos compuestos en ASN.1, de forma que SEQUENCE, SEQUENCE OF, SET, SET OF y CHOICE se utilizan cuando se descomponen los tipos.

En este paso, además de buscar en los tipos definidos, se deberían consultar la biblioteca de "tipos útiles" ("useful types") ASN.1 y las definiciones ASN.1 utilizadas para las interfaces de normas relacionadas con la aplicación.

Durante el desarrollo puede ser conveniente utilizar el "any" de ASN.1. Cuando se completan las definiciones de tipo ASN.1, "any" debería reemplazarse por un nombre específico, posiblemente definido externamente.

Si los cuadros de bits se han proporcionado en los requisitos recopilados, estos deberían escribirse de nuevo en ASN.1. La codificación básica en ASN.1 (Recomendación X.209 [10]) no es siempre adecuada para un protocolo, por ejemplo, puede ser demasiado ineficiente. En este caso se utiliza la codificación explícita. A fin de describir la codificación, puede ser necesario disponer en la documentación de una forma simplificada de cuadros de bits.

Los datos SDL pueden utilizarse en lugar del ASN.1:

– si no se requiere ASN.1 para las definiciones de la interfaz;

– si no se utiliza la codificación básica ASN.1;

– si SDL proporciona un buen soporte para los géneros de datos necesarios.

Cuadro 2/Sup. 1 a la Rec. Z.100 – ASN.1 para el establecimiento del control de llamada de radio troncal

```
CC-SETUP::=     SEQUENCE {
        pd      ProtocolDiscriminator;
        ti      TransactionIdentifier;
        mt      MessageType;
        pi      PortableIdentity OPTIONAL;          -- only present if "incoming call" is implemented
        fdi     FixedIdentity OPTIONAL;             -- only present if "incoming call" is implemented
        bs      BasicService OPTIONAL;              -- only present if "incoming call" is implemented
        ia      IWU_Attributes OPTIONAL;            -- mandatory if basic service indicates "other"
                                                    -- not allowed if basic service indicates "default
                                                    -- attributes"
        ri      RepeatIndicator OPTIONAL;           -- if ri appears before call attributes, not after,
                                                    -- this indicates a prioritized list of call attributes for
                                                    -- negotiation
        cla     SEQUENCE SIZE(0..3) OF
                        CallAttribute OPTIONAL;
        ri      RepeatIndicator OPTIONAL;           -- if ri appears after call attributes and not before

        cnal    SEQUENCE OF ConnectionAttribute OPTIONAL;
        cri     CipherInfo OPTIONAL;
        cni     ConnectionIdentity OPTIONAL;
        fy      Facility OPTIONAL;
        pi      ProgressIndicator OPTIONAL;         -- not allowed if signal is sent from P to F
        dy      Display OPTIONAL;                   -- not allowed if signal is sent from P to F
        kp      KeyPad OPTIONAL;                    -- not allowed if signal is sent from F to P
        sl      Signal OPTIONAL;                    -- not allowed if signal is sent from P to F
        fea     FeatureActivate OPTIONAL;           -- not allowed if signal is sent from F to P
        fei     FeatureIndicate OPTIONAL;           -- not allowed if signal is sent from P to F
        np      NetworkParameter OPTIONAL;          -- not allowed if signal is sent from F to P
        tc      TerminalCapability OPTIONAL;        -- not allowed if signal is sent from F to P
        eec     EndToEndCompatibility OPTIONAL;     -- mandatory if system uses LU6 (V.110/I.30 rate)
        rp      RateParameters OPTIONAL;            -- mandatory for call set-up of a rate adaptation service
        td      TransitDelay OPTIONAL;
        ws      WindowSize OPTIONAL;
        cgpn    CallingPartyNumber OPTIONAL;
        cdpn    CalledPartyNumber OPTIONAL;
        cds     CalledPartySubaddress OPTIONAL;
        sc      SendingComplete OPTIONAL;           -- mandatory if all necessary information for call set-up
        iti     IWU_to_IWU OPTIONAL;
        ip      IWU_Packet OPTIONAL
        }
```

Si no existieran motivos para preferir los datos SDL, debería utilizarse ASN.1.

La codificación explícita se realiza tan tarde como sea posible. En ocasiones puede hacerse después de la Formalización. La codificación de mensajes no cambia el comportamiento de la descripción formal SDL, pero codifica las interfaces de forma que los patrones de los bits se adapten a la utilización en el entorno.

Regla 10 – Los "tipos útiles" ("useful types") de ASN.1 así como otros tipos ya normalizados deberían utilizarse siempre que fuera posible.

Regla 11 – La codificación debe hacerse separadamente de la estructuración y denominación de tipos.

Ejemplo

Véase el cuadro 2.

14.4 Modelado de la secuencia de uso

El modelado de la secuencia de uso se inicia en la fase de Análisis mediante la captura del punto de vista del usuario del sistema. En esta etapa está disponible un modelo de flujo de control y de datos SDL y el modelo de secuencia de uso se detalla de acuerdo con esta descripción SDL.

Las tareas adicionales se refieren principalmente a lo siguiente: la utilización de entidades SDL en lugar de clases del modelo de objeto, la utilización de señales SDL en lugar de eventos relacionados con el modelo de objeto, la sustitución de los intercambios abstractos modelados en el Análisis por los protocolos reales Por ejemplo, en el modelo de Análisis, la información generada por una entidad externa se modela como un mensaje abstracto único, mientras que en el Diseño Previo esta información se modela mediante el conjunto de interacciones que tienen lugar en la realidad. Estas tareas adicionales conducen asimismo a la introducción de un nuevo MSC que detalle los intercambios abstractos.

Debido a que los MSC se utilizan a menudo para sustentar los pasos de comportamiento de la Formalización, puede ser necesario producir el MSC como parte de la Formalización. En tal caso, se utiliza este paso en combinación con el paso de 13.3 a fin de generar el MSC.

Instrucciones

1) Seguir las instrucciones 1 a 5 de 13.3 para producir el MSC pero modelando las instancias MSC dentro del sistema que corresponde a las entidades SDL (bloques, procesos o servicios) internas al mismo.

2) Comprobar la consistencia entre el MSC a dicho nivel y el MSC en la información clasificada.

3) Producir (con carácter facultativo) el MSC para interacciones de nivel inferior en el SDL y comprobar la consistencia con el MSC a niveles superiores.

Directrices

Las interfaces con el entorno (o con niveles superiores) deben utilizar los lados del gráfico. Sin embargo, si hay tres o más interfaces con el entorno, es más conveniente que éstas aparezcan como ejes de instancias.

Puede haber más de una instancia en la misma clase de un gráfico de secuencias de mensajes. Por ejemplo, pueden existir dos ejes de instancias que representen instancias diferentes del control de una llamada o interfaces con el entorno.

Si sólo hay una instancia de cada clase o ítem SDL, los nombres de las instancias en el gráfico pueden ser los mismos que los nombres de las correspondientes clases o ítems SDL. Si hay varias instancias, éstas deben tener nombres exclusivos, y la clase (o ítem SDL) se indica mediante su tipo. Por ejemplo, en la figura 14-2 PTNX es un tipo de instancia y PTNX_A es un nombre de instancia.

Los mensajes han sido normalmente identificados antes de que los MSC se dibujen ya sea como partes del comportamiento y de aspectos de las interfaces de las clases correspondientes a las instancias en la información clasificada, o como señales utilizadas en el SDL. La secuencia de mensajes puede identificarse a partir de los aspectos de comportamiento y de interfaz de las clases de la información clasificada.

No es esencial dividir los gráficos de secuencias de mensajes en MSC sencillos a los que se haga referencia desde un MSC principal, aunque ello es útil cuando muchas secuencias tienen el mismo "set up" (establecimiento) o "clear down" (desactivación). Cuando el MSC hace referencia a MSC simples y se ponen condiciones, deben utilizarse nombres con sentido pleno tales como "call_in_progress" ("llamada en curso").

La elección de un conjunto de secuencias de uso razonables es algo subjetivo y función del sistema. No es posible normalmente dibujar todas las secuencias de mensajes posibles para un sistema en particular ya que su número es muy elevado.

Regla 12 – Para representar mediante diagramas la secuencia de mensajes intercambiados entre partes del sistema (con el entorno) a lo largo del tiempo, se debe utilizar la notación MSC.

Ejemplo

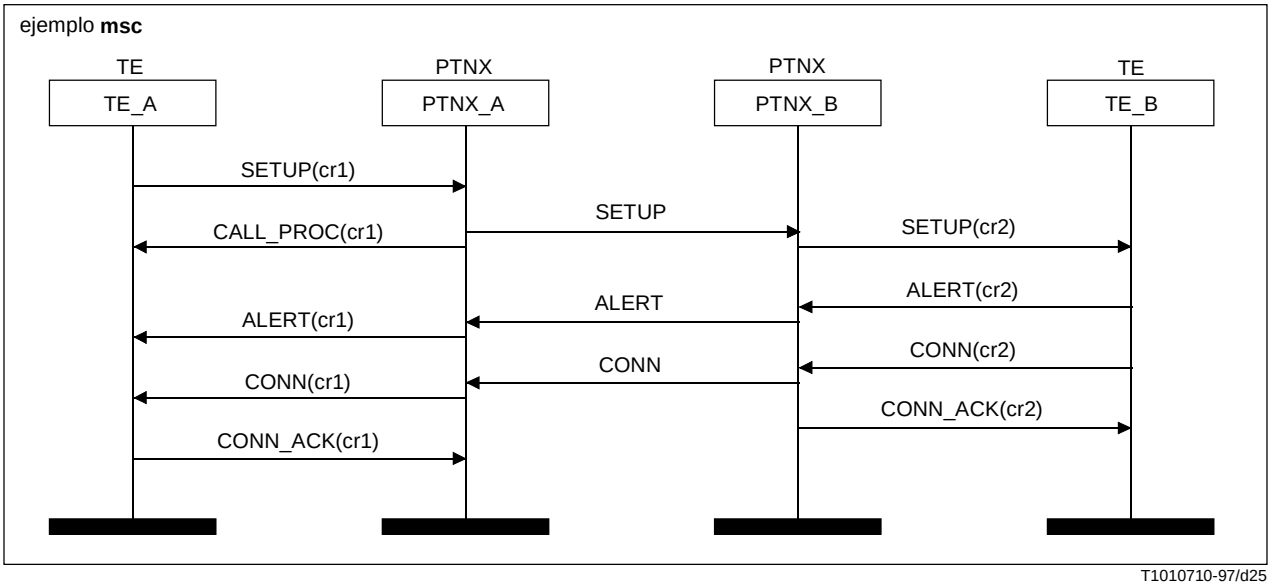


Figura 14-2/Sup. 1 a la Rec. Z.100 – Envío en bloque, llamada exitosa

14.5 Modelado del comportamiento del proceso

El modelado se realiza utilizando informalmente diagramas de procesos SDL para esquematizar el comportamiento de los procesos. Durante el Diseño Previo este paso sólo se utiliza normalmente en procesos críticos. El comportamiento de otros procesos se infiere de las secuencias de uso y de otras descripciones. Puede ocurrir que no se defina formalmente el conjunto de señales, utilizándose "texto informal" SDL en lugar de expresiones formales en tareas, decisiones y parámetros.

Debido a que este modelado puede realizarse de forma similar a la Formalización, no se describe con detalle en este paso. La diferencia esencial entre la Formalización y el modelado del proceso en el Diseño Previo es el nivel de formalidad. Los diagramas SDL del Diseño Previo pueden ser SDL informales, incompletos, inconsistentes y no válidos. No es preciso aplicar las reglas de la Formalización. Ello no impide que los diagramas proporcionen visiones útiles de la aplicación y que permitan que se consigan ahorros.

14.6 Modelado de la visión general del estado

Este modelado utiliza los diagramas SDL auxiliares para proporcionar visiones generales del comportamiento del sistema. Se trata de diagramas de árbol, diagramas de la visión general de estado y diagramas de la matriz de la señal de estado (véanse I.9.2/Z.100 a I.9.4/Z.100 [1]).

También puede ser de utilidad tomar nota de los estados de la visión general (o de los esquemas SDL indicados en 14.5) que presentan propiedades de estado que se expresan mediante expresiones lógicas. Estas expresiones pueden ser útiles cuando se definen pruebas.

15 Pasos de la Formalización

NOTA – En todos los pasos en los que se define un sistema, bloque, proceso o procedimiento, debe siempre buscarse un tipo existente que pueda ser utilizado o modificado. Esta directriz básica se ubica ahí en lugar de repetirla en todos los pasos en los que se aplica.

Si en el Análisis se ha utilizado la notación de modelo del objeto OMT, el cuadro 3 proporcionan directrices generales para obtener el SDL para la notación del modelo de objeto.

Cuadro 3/Sup. 1 a la Rec.Z.100 – Conversión de la notación del modelo de objeto OMT en SDL

Notación de Modelo de Objeto	Posible transformación SDL
clase	bloque (tipo), proceso (tipo), servicio (tipo), declaración de datos (o neotipo o sintipo)
clase abstracta	tipo de bloque, tipo de proceso, tipo de servicio, (datos) neotipo (o sintipo o generador)
relación de instanciación	línea de creación de proceso
atributo (de clase de no datos)	declaración de datos de variable local
atributo (de clase de datos)	campo de estructura o dos operadores (get, set)
tipo de atributo, tipo de parámetro	género de datos de la construcción SDL derivada del atributo
operación	procedimiento (local, exportado, importado), definición de señal, operador
parámetro de una operación	parámetro de (procedimiento o señal u operador)
herencia simple	herencia
asociación, enlace	ruta de señal, canal
nombre de asociación	nombre de (ruta de señal o canal)
multiplicidad de agregaciones	número de instancias de (bloques o procesos), matriz (de datos)
nombre del cometido	puerta, nombre de variable
agregación	estructura para bloques o procesos, variables locales en un proceso, dos operadores (o un struct) en los datos

Las construcciones de la notación del modelo de objeto OMT que no se mencionan en el cuadro 3 no son normalmente utilizadas.

En [26] y [27] pueden encontrarse directrices adicionales.

Los pasos para la Formalización se dividen en grupos para Estructura (pasos S), Comportamiento (pasos B), Datos (pasos D), Tipos (pasos T) y Localización (pasos L). No es necesario finalizar un grupo de pasos antes de comenzar con otro, siendo normal que el trabajo progrese de forma simultánea en más de un grupo. Por ejemplo, los pasos D pueden utilizarse para definir los parámetros de las señales en paralelo con los pasos S y los pasos B. Igualmente, conforme el sistema se refina es posible aplicar pasos S a un bloque al tiempo que otro bloque se detalla mediante los pasos B, D o T. Para aprovechar todas las ventajas de los tipos del SDL-92, los pasos T y L deberían aplicarse simultáneamente a los pasos S y B.

15.1 Pasos de estructura (pasos-S) (S-steps, *structure steps*)

Los pasos de estructura se utilizan para definir en SDL las interfaces externas e internas del sistema así como la partición estática del sistema en bloques SDL.

15.1.1 Paso S:1 – Límites y entorno del sistema

Instrucciones

1) Identificar los límites entre el sistema a describir y su entorno.

2) Encontrar un nombre adecuado para el sistema.

3) Dibujar un diagrama de sistema SDL asignándole un nombre y explicando informalmente el sistema y sus relaciones con el entorno por medio de un comentario en el diagrama de sistema SDL.

Directrices

Aunque este paso pueda parecer trivial no debe subestimarse la importancia de la preparación del mismo. El Análisis o el Diseño Previo realizado antes del mismo ayudará a elegir tanto el límite correcto como un nombre adecuado. Este paso sólo se realizará cuando se tenga un conocimiento razonable de los requisitos del sistema.

El nombre del sistema y el comentario descriptivo pueden probablemente obtenerse directamente a partir de información clasificada. El nombre del sistema y el comentario descriptivo deben aparecer en la descripción formal SDL y por lo tanto deberían ser adecuados para todo el sistema.

La información clasificada puede contener descripciones de clases internas y externas al mismo. Puede haber una clase que sea fácilmente identificable como adecuada para el SDL o puede ser necesario identificar un conjunto de clases con las que se dispone de interfaces a fin de constituir el sistema.

Cuando la información clasificada contiene un aspecto de comportamiento para cada clase y no se han identificado interfaces ligadas al entorno, puede tratarse de un sistema SDL cerrado sin interfaces externas. En este caso, el sistema SDL describe el comportamiento de interés de la aplicación y de las clases que comunican con ella. Dichas clases son informativas.

Un "diagrama de contexto" que se produce utilizando informalmente SDL como Diseño Previo (véase 14.2) y MSC, puede ser útil para determinar el límite entre el sistema SDL y el entorno. Al considerar la adecuación de un "diagrama de contexto" de Diseño Previo en SDL, se debe prestar especial atención al nombre del sistema y al comentario descriptivo. También se debe considerar qué parte del entorno figura en la descripción formal SDL. Puede ser necesario que las interfaces normativas de la aplicación sean interfaces internas del sistema SDL.

Los MSC del sistema se producen a menudo como diseños previos después de que se ha realizado este paso. Si ya se han producido los MSC, debe distinguirse entre los ejes de entidades que pertenecen al sistema y los ejes de entidades que pertenecen al entorno exterior. Estos últimos no tienen una descripción formal en el sistema SDL (o sólo se proporcionan como partes informativas del sistema). Es posible considerar que el sistema se compone de sistemas SDL que se comunican, pero en este caso el sistema debe describirse como un sistema SDL sencillo con bloques que representan las partes que se comunican. Los sistemas SDL comunicantes pueden considerarse cuando los requisitos recopilados son relevantes para un conjunto de aplicaciones relacionadas.

Debe considerarse si existen tipos útiles en la biblioteca que estén relacionados con las interfaces del entorno o del sistema y que puedan incluirse en un lote SDL.

Si el sistema tiene temporizadores, debe definirse la unidad de tiempo. Normalmente el milisegundo o el segundo pueden resultar valores adecuados. Asimismo es útil definir sinónimos SDL para múltiplos útiles como el "segundo" o la "hora".

Regla 13 – El límite del sistema sólo se comunica mediante el intercambio de mensajes discretos.

Regla 14 – Los datos de la unidad de tiempo deben registrarse en un comentario en el diagrama del sistema.

Resultado: El resultado es un nombre con pleno sentido para el sistema y una descripción recogida en un comentario.

15.1.2 Paso S: 2 – Comunicaciones discretas del sistema

Instrucciones

1) Identificar el flujo de información en términos de mensajes discretos que se establecen entre el sistema y su entorno.

2) Modelar estos mensajes mediante señales definidas en el sistema.

3) Establecer la relación entre cada señal y los objetos externos al sistema.

4) Establecer el objetivo de cada señal en un comentario que acompañe la definición de la misma.

5) Agrupar en una lista de señales aquellas que están relacionadas (a menudo todas) para un objeto en una dirección.

6) Incluir en el diagrama del sistema las definiciones de señales y las listas de señales.

Directrices

Las señales hacia y desde el entorno pueden identificarse normalmente en la información clasificada como aspectos de las interfaces de la clase o clases más externas. Los aspectos de la información relacionada describen el contenido de las señales.

El "diagrama de contexto" o el MSC a nivel del sistema permiten identificar las señales. El número de señales puede reducirse mediante señales calificadoras con parámetros (véase el paso D:1). Las señales para las comunicaciones externas se definen a nivel del sistema y se corresponden con eventos discretos entre el sistema y su entorno. Éstas pueden estar definidas en la visión de la información como unidades de descripción de protocolo o como flujo de información.

Si se utiliza como marco la metodología de tres etapas (véase 13.1) y se ha realizado la descripción de la etapa 1, las señales necesarias para la comunicación con el usuario son las mismas que son necesarias para comunicarse con el entorno. A veces se requiere un comportamiento particular del usuario, en cuyo caso, parte (o todo) el comportamiento puede modelarse en el propio sistema.

Un sistema que incluya el asunto de la aplicación y todos los objetos comunicantes, es un sistema cerrado y no dispone de señales para comunicarse con el entorno.

Cuando se añade una señal con un parámetro, identifica o denomina al correspondiente género de datos. El género de datos debería corresponder con un tipo de ASN.1 designado de la visión de la información.

Las definiciones de señales y las definiciones de listas de señales son normalmente demasiado grandes como para ser incluidas en un símbolo de texto de la primera página SDL del diagrama del sistema de la documentación. Debieran añadirse al sistema páginas adicionales para que contengan dichas descripciones textuales. Los diagramas SDL no debieran incluir amplias declaraciones sobre la relación entre señales y clases externas, ni descripciones con dibujos informales. A dichas declaraciones se debe hacer referencia desde el SDL. En las pertinentes definiciones SDL se deberían incluir declaraciones más breves.

NOTA – Es recomendable situar las partes de texto del SDL en los diagramas SDL. La ubicación del texto SDL en un símbolo de texto de un diagrama identifica claramente que el texto es SDL e igualmente identifica el diagrama SDL al que pertenece el texto.

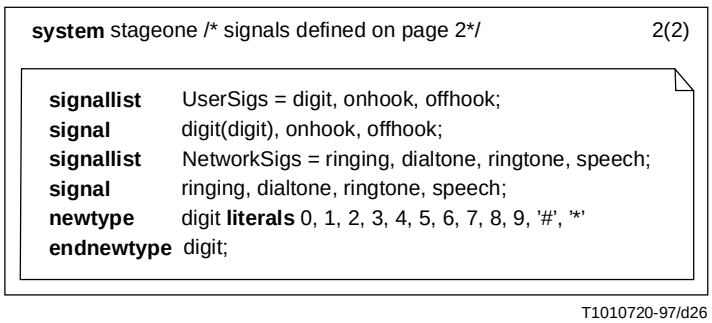
Para ayudar en la documentación, las definiciones de las listas de señales se ubican antes de la definición de las señales utilizadas en aquéllas. Las definiciones de datos se ubican después de las definiciones de señales. Las definiciones se agrupan en grupos lógicos situando en un solo símbolo de texto las definiciones que están relacionadas entre sí.

Regla 15 – Las definiciones textuales de un diagrama debieran ubicarse en símbolos de texto de diagramas.

Regla 16 – Si las definiciones textuales ocupan más del 50% de un diagrama, éste debe dividirse en dos o más páginas con las definiciones textuales en grupos lógicos de símbolos de texto separados.

Resultado: El resultado es el "alfabeto" del sistema: las señales definidas a nivel del sistema para la comunicación con el entorno, aunque dichas señales puedan necesitar un ulterior refinamiento.

Ejemplo



NOTA – ASN.1 no se ha utilizado para "digit" porque se pretendía que los nombres de los dígitos fueran 0, 1, 2 y así sucesivamente.

Figura 15-1/Sup. 1 a la Rec. Z.100 – Lista de señales del paso S:2

15.1.3 Paso S:3 – Partes identificables externamente

Instrucciones

1) Identificar las partes principales del sistema y dibujarlas como bloques del mismo.

2) Asignar un nombre adecuado a cada bloque y describirlo, así como sus relaciones con el entorno (su estructura circundante) de una manera informal en un comentario dentro del propio bloque.

NOTA – Los pasos S:3 a S:6 se utilizan de forma recursiva cuando un bloque se descompone en subbloques. Cuando se aplican de nuevo los pasos, el "sistema" que se considera es realmente un bloque SDL y la palabra "sistema" debería ser sustituida por la frase "bloque circundante". La diferencia entre un bloque y el sistema es que las señales (lista de señales y datos) que se utilizan para la comunicación con la estructura circundante se definen **fuera (outside)** del bloque en el caso de un bloque, y **dentro (inside)** del sistema en el caso de un sistema.

Directrices

La información clasificada contiene clases de componentes que se corresponden con los bloques (conectados mediante canales) y con las definiciones tipo realizadas en SDL. Las clases de componentes del sistema (o bloques cuando este paso se utiliza de forma recursiva) corresponden a bloques y a canales. Algunas clases que carecen de aspecto de comportamiento pueden representar objetos del entorno.

Los requisitos arquitectónicos, tales como la división del sistema en planos de protocolos separados, se utilizan para ayudar a identificar las partes del sistema.

En este paso se manejan los bloques internos, mientras que los canales se manejan en el paso S:4. Los bloques son objetos que contienen un aspecto de comportamiento con estados internos. Los estados se pueden corresponder con datos descritos en el aspecto de información. El ingeniero debe decidir si una clase debería describirse en SDL como un tipo de bloque, tipo de proceso, procedimiento, señal, temporizador o género de datos. En la mayoría de los casos la elección del tipo de SDL correspondiente a una determinada información clasificada es obvia, pero en otros casos puede ser función de la presentación abstracta objeto de la especificación formal. Por ejemplo, el comportamiento del objeto puede describirse en base al estado (proceso, procedimiento) o utilizando axiomas (operador y axiomas en definiciones de neotipo (newtype). En este caso se recomienda una descripción basada en los estados, salvo que el comportamiento se corresponda claramente con funciones de datos abstractas.

Las estructuras de los diseños previos pueden también utilizarse para determinar la estructura del bloque.

Los bloques delimitan la visibilidad. Por este motivo las señales, los géneros y los tipos que sólo se utilizan en un bloque deben definirse en dicho bloque, de tal forma que la información quede oculta a capas superiores, haciendo que estos niveles sean más fáciles de entender. Este principio también se aplica a los pasos S:6 y S:7 para dichos ítems, y se expresa como un principio general para la ocultación de la información en la Regla 18.

Un bloque es "informativo" si no tiene un comportamiento normativo. El objetivo de un bloque informativo es conseguir que el sistema sea completo, de tal forma que la función del sistema pueda ser:

– entendida por la utilización que hace de dichas partes informales y por su interacción con las mismas;

– ejecutada y analizada dando lugar a un producto y a una Validación soporte de mejor calidad.

Un bloque es informativo si todos los bloques, procesos y procedimientos (incluidos los procedimientos remotos) que circunda son informativos. Si un bloque (proceso o procedimiento) se marca como informativo implica que todos los bloques, procesos y procedimientos que circunda son "informativos", no siendo necesario marcarlos como tales.

Cuando el sistema (o bloque) circunda directamente un gran número de bloques algunos de ellos se agrupan y se encapsulan en un bloque tal como ocurre en el paso S:6.

NOTA – "informativo" no es lo mismo que "informal". En la descripción formal SDL las partes informativas y normativas debieran ser formales (es decir, expresadas formalmente).

Regla 17 – No deberían existir más de cinco bloques a nivel de sistema (o directamente circundados por un bloque).

Regla 18 – Una definición debe tener el campo de aplicación menor posible que incluya todos los usos del ítem definido.

Regla 19 – Si un bloque, proceso o procedimiento, es informativo y no está a su vez circundado por un bloque informativo (proceso o procedimiento), debe tener la anotación "informativo" en el diagrama que lo referencia, en su diagrama referenciado o en ambos.

Resultado: El resultado es un diagrama que contiene un conjunto de bloques y posiblemente la identificación de tipos para dichos bloques.

Ejemplo

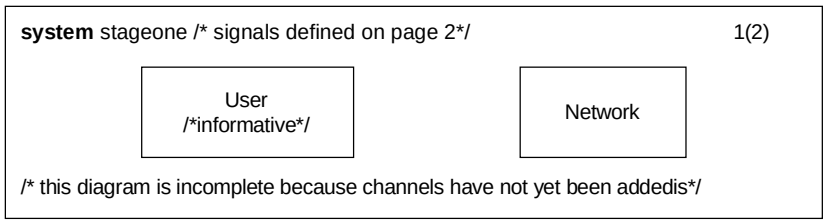


Figura 15-2/Sup. 1 a la Rec. Z.100 – El usuario y la red para una descripción de la etapa 1 de I.130 [7]

15.1.4 Paso S:4 – Trayectos de comunicación entre las partes

Instrucciones

1) Identificar los canales necesarios entre los bloques y el límite del diagrama y entre bloques dentro de un diagrama.

2) Para cada canal, identificar el sentido o los sentidos de la comunicación.

3) Asociar una lista de señales a cada sentido del canal.

4) Elegir un nombre de lista de señales relacionado con la función y uso de la comunicación.

Directrices

Las descripciones relevantes sobre como se conectan los bloques internos en un diagrama se incluye en los aspectos de la interfaz de la clase que corresponde al diagrama en la información clasificada. Los canales pueden estar representados explícitamente como clases en la información clasificada, particularmente si existen requisitos específicos sobre los canales. Un canal y su lista o listas de señales asociadas definen la signatura de una interfaz.

Las listas de señales identificadas en el paso S:2 corresponden con los extremos de un canal en el límite del sistema. Cada punto del límite representa la comunicación con un objeto externo diferente (un canal o bloque externo al sistema SDL). Pueden existir uno o más canales que conecten dicho punto a los bloques del diagrama del sistema SDL. Cada canal agrupa algunos flujos del comportamiento del sistema en el "diagrama de contexto" de los diseños previos. Se tienen en cuenta requisitos del modelado tales como las interfaces independientes.

Si el sistema sólo contiene un bloque, éste se conecta al entorno por medio de canales. En cualquier otro caso, los bloques a nivel de diagrama se conectan también mediante canales de acuerdo con los flujos de información entre ellos. No hay motivos suficientemente concluyentes como para que un diagrama de bloque sólo contenga un bloque.

Los casos de comunicación típicos representados en el MSC ayudan a identificar los canales.

No deberían utilizarse normalmente los efectos especiales que dependen del retardo o de la ausencia de retardo de cada canal. Si sólo se utiliza un canal entre dos bloques, las señales enviadas a un bloque llegan al otro bloque en el mismo orden. Se supone que los canales tienen un retardo, salvo que exista el requisito de comunicación sin retardo.

Si la comunicación sobre un canal necesita disponer de mensajes específicos con un formato específico y una codificación acorde con la información clasificada y los requisitos recopilados, el canal es "normativo". Los canales que son internos al sistema y no identifican una interfaz en particular para la prueba de productos u otros objetivos son normalmente informativos. La comunicación sobre canales informativos afecta al comportamiento del sistema, pero puede haber un sistema alternativo con distintos canales o comunicaciones que tenga el mismo comportamiento.

Regla 20 – Entre dos bloques sólo debería haber un canal.

Regla 21 – Cada canal normativo debe tener adjunto el comentario "normativo".

Resultado: El diagrama se define con un conjunto de trayectos de comunicación que corresponden a las interfaces externas y a las interfaces internas entre los bloques directamente circundados.

Ejemplo

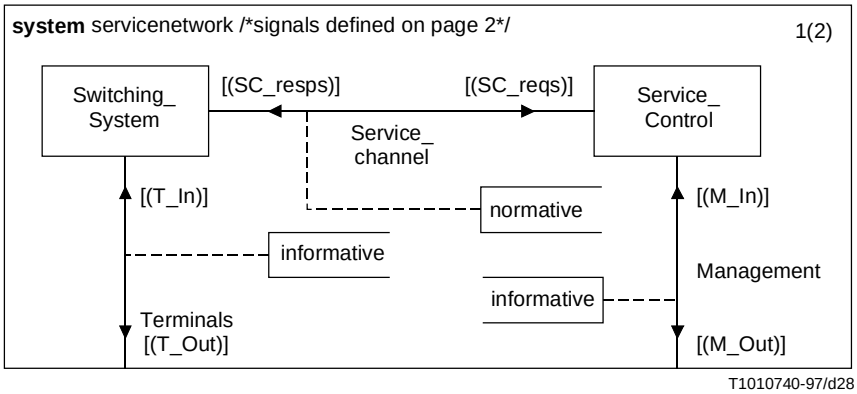


Figura 15-3/Sup. 1 a la Rec. Z.100 – Modelo de una red para el manejo de servicios

15.1.5 Paso S:5 – Asociación de señales a trayectos de comunicación

Instrucciones

1) Para cada nombre de la lista de señales identificar las señales adecuadas.

2) Nombrar y definir cada nueva señal.

Directrices

La asociación entre señales y listas de señales utilizadas entre bloques puede exigir la definición de nuevas señales en el diagrama que contiene los bloques.

La descripción relevante se encuentra en los aspectos de las interfaces internas del sistema que figura en la información clasificada (si los canales se describen como objetos parte) en la información sobre la señal del correspondiente canal en la información clasificada.

Si se ha elaborado un Diseño Previo para el diagrama SDL, debe reconsiderarse cada lista de señales asociada con un sentido de un canal. También deben verificarse los mensajes entre los ejes relevantes del MSC.

Considerar si se redefine la lista de señales asociada con un canal a niveles superiores, utilizando la nueva lista de señales. Ello puede mejorar la estructura y claridad del SDL. La definición de la lista de señales debe estar en el diagrama de nivel más alto en el que se utiliza.

Aunque el SDL permite que el símbolo de lista de señal (es decir [...]) de un canal en un diagrama contenga directamente señales, no es recomendable listar las señales en el símbolo. Normalmente, la lista de señales se utiliza en más de un lugar, resultando más sencillo modificarla si se define una sola vez en una lista de señales.

Regla 22 – En un símbolo de lista de señal no debieran listarse más de tres señales (o listas de señales), en su lugar, debe utilizarse una lista de señales adjunta al canal.

Regla 23 – Las listas de señales, las señales y los datos utilizados en las comunicaciones externas de un sistema debieran definirse en uno (o más) símbolos de texto separados de cualquier otra definición.

Resultado: Las señales se asocian a nombres de listas de señales (es decir, indirectamente a canales).

15.1.6 Paso S:6 – Ocultación y subestructuración de información

Instrucciones

1) Considerar uno a uno cada bloque del diagrama actual y decidir si el bloque debe contener bloques o procesos.

2) Si el bloque debe contener bloques, aplicar de forma recursiva al bloque la secuencia de pasos S:3 a S:6, considerándolo como un (sub)sistema e introduciendo las nuevas señales, bloques, canales y listas de señales requeridas.

3) En otro caso, aplicar el paso S:7 para dividir el bloque en procesos.

Directrices

Si el sistema es grande, algunos bloques pueden ser considerados como sistemas, y divididos según las reglas del sistema. Ello da lugar al anidamiento de bloques. Cada bloque puede entonces ser elaborado con más detalle, tal como se ha descrito anteriormente. Este paso puede ser producto del anidamiento de clases en la información clasificada. Una clase cuyo principal aspecto es el comportamiento y no sufre una ulterior partición es una buena candidata para constituir un proceso; por lo tanto, el objeto que la circunda directamente es un bloque. Una clase que contiene clases que han sido ulteriormente particionadas, es normalmente un bloque que debe ser particionado.

Si no está claro si el contenido de un bloque deben ser bloques o procesos, se realiza inicialmente una división en bloques. Si ésta resulta difícil realizar, probablemente se trata de un proceso.

NOTA – Un **bloque** (**block**) interior directamente circundado por un **bloque** exterior es una notación abreviada de un **bloque** interior directamente circundado por una **subestructura** (**substructure**) de **bloque** exterior a su vez está circundada por un **bloque** exterior que en cualquier otra circunstancia estaría vacío. Véase la figura 15-4.

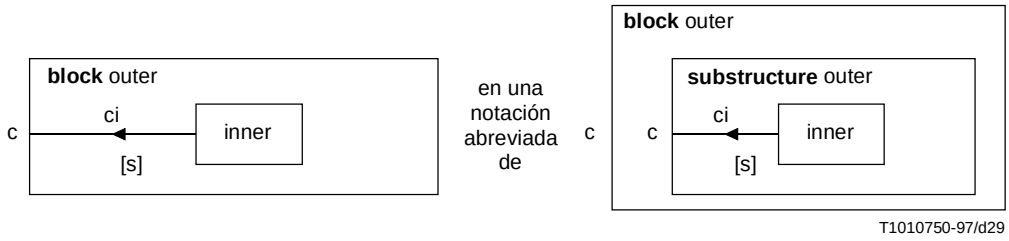


Figura 15-4/Sup. 1 a la Rec. Z.100 – Notación abreviada de bloques circundados

En algunos casos, un proceso debe encapsularse en un bloque para que se ajuste a los diagramas.

El SDL también permite que un bloque contenga la descripción de procesos y una subestructura con más bloques. Puede ser útil hacer uso de esta característica durante el Diseño Previo, pero no es recomendable usarla en la Formalización.

La aplicación recursiva de este paso da lugar a una serie de niveles. Incluso si el sistema es complejo, no deberían de existir más de tres niveles de bloques. Si cada nivel tiene de tres a cuatro bloques con un número de dos a cinco procesos en cada bloque hoja , el sistema SDL contará con más de 100 procesos.

Los bloques circundados debieran dibujarse como referencias de bloques más que como diagramas de bloques debido a que los diagramas directamente anidados se hacen demasiado grandes y complejos para ser manejados o entendidos. El mismo principio se aplica a todos los diagramas (bloque, proceso, procedimiento y diagramas tipo).

Si en un bloque existen más de cinco definiciones obvias de procesos, el bloque debiera dividirse en dos o más bloques con un menor número de procesos cada uno. Las interacciones en el MSC entre ejes de procesos pueden ayudar a identificar los "grupos" ("clusters") naturales de procesos para cada bloque no particionado.

Debería dibujarse un diagrama de árbol de bloques.

Regla 24 – Los diagramas se deberían anidar mediante referencias, en lugar de utilizar la circundación directa.

Regla 25 – El número de canales de cada bloque no debiera ser superior a cinco.

Resultado: El resultado es un conjunto de diagramas de bloque y un árbol de bloques cuya raíz es el sistema y en el que los bloques que no sufren una ulterior partición en bloques constituyen las hojas del mismo (bloques hoja).

15.1.7 Paso S:7 – Constituyentes del bloque

Instrucciones

1) Identificar los objetos con comportamiento para el bloque que se ha decidido dividir en procesos en el paso S:6 y definir dichos objetos como los procesos de dicho bloque.

2) Encontrar un nombre adecuado para cada proceso y describirlo, así como sus relaciones con su entorno (el bloque circundante) de manera informal mediante un comentario incluido en la referencia del proceso.

3) Para cada proceso definir su número de instancias inicial y máximo.

4) Utilizar rutas de señales para conectar los conjuntos de procesos a canales en el límite del bloque.

Directrices

Las rutas de señales del bloque pueden obtenerse tal como se obtuvieron los canales en los pasos S:3 y S:4.

En el caso de que un proceso haya sido encapsulado en un bloque "simulado", el bloque no particionado contiene una descripción de proceso de un solo tipo cuya relación con el entorno se deriva de la interfaz externa del bloque.

Considerar modelos de objetos con asociaciones entre clases correspondientes a procesos. La multiplicidad respectiva (1:1, 1:muchos; muchos:muchos) de la asociación permite definir el número inicial y máximo de instancias para dichos procesos.

Regla 26 – En cada bloque debe haber al menos un proceso cuyo número de instancias iniciales sea superior a cero, de forma que pueda crear otras instancias en el bloque.

Regla 27 – El número de definiciones de procesos en cada bloque no debe ser superior a cinco.

Resultado: Identificación de los procesos en los bloques hoja del árbol de bloques.

15.1.8 Paso S:8 – Señales locales de un bloque

Instrucciones

1) Identificar las señales entre los procesos locales del bloque.

2) Definir a nivel de bloque todas aquellas señales que sean adicionales (que no se han definido como externas al bloque).

3) Identificar los procedimientos importados y exportados pertenecientes a procesos del bloque así como la correspondiente definición de procedimiento remoto, definiendo a nivel de bloque (o tipo de bloque).

Directrices

La descripción relevante se encuentra en los aspectos de la interfaz interna de la correspondiente clase con el bloque circundante. Si las rutas de señales se describen como objetos parte, la información sobre la señal en la ruta de señal viene dada por la correspondiente ruta de señal local de la información clasificada. En el caso especial de un bloque "simulado" (un bloque que contiene un único proceso), no habrá señales adicionales a nivel de bloque.

El SDL permite tres formas (además del paso de señal) para utilizar en otro proceso el valor de las variables de datos de un proceso: visión y revelación, importación y exportación, y mediante una llamada a un procedimiento remoto del otro proceso. Todos estos mecanismos pueden dar lugar a que el diseño del sistema tenga errores inherentes. Deben utilizarse con precaución y se recomienda que se realice una simulación de la interacción existente. Se permiten los procedimientos remotos. La visión y el revelado pueden producir un comportamiento no determinístico. La importación y la exportación pueden ser sustituidas por llamadas a un procedimiento remoto.

Regla 28 – No deben utilizarse la visión ni la revelación.

Resultado: El resultado es la definición de cada señal y de cada procedimiento remoto a nivel de bloque (o tipo de bloque).

15.2 Pasos de comportamiento (pasos B) (B-steps, *behaviour steps*)

Estos pasos describen el comportamiento de los componentes en lo que respeta a la comunicación pero sin proporcionar una definición de los datos de los pasos de datos. Esta subcláusula describe dichos pasos para un **proceso** SDL, pero estos pasos como los pasos de datos son en general adecuados para definir el comportamiento de un **procedimiento** SDL.

15.2.1 Paso B:1 – Conjunto de señales dirigidas a un proceso

Instrucciones

Identificar el alfabeto de entrada del proceso, denominado conjunto de señales (signalset) del mismo mediante la identificación de:

– cualquier señal que pueda recibir el proceso (un 'anuncio' en terminología ODP), a la que el proceso puede o no responder;

– cualquier procedimiento exportado por el proceso. Tal es el caso cuando el proceso actúa como servidor en un modelo cliente-servidor (una 'interrogación' en terminología ODP), en el que la correspondiente señal está implícita pero el nombre del procedimiento remoto puede considerarse parte del alfabeto.

Directrices

Aunque el conjunto de señales (signalset) puede normalmente obtenerse del diagrama del bloque circundante, el objetivo de este paso es revisar el conjunto de señales considerando exclusivamente el proceso actual. Si se decide cambiar el conjunto de señales de otros procesos, deben también cambiarse los correspondientes diagramas de bloque. Si los procesos que envían los datos han sido formalizados, deben de ser actualizados (probablemente en otro momento). Una fuente de señales que no se muestra en el diagrama de bloque está constituido por el proceso que envía señales hacia sí mismo o hacia otras instancias de la misma definición de proceso.

El conjunto de señales se utiliza cuando se decide cual es el comportamiento del proceso en cada uno de los estados del paso B:4. Si el conjunto de señales no puede obtenerse automáticamente a partir de las herramientas utilizadas en el SDL, se recomienda mantener un registro del mismo.

En un procedimiento, se utilizan las señales del proceso circundante, pero pueden identificarse nuevas señales que deben añadirse al conjunto de señales del proceso (identificados en este paso pero para el proceso circundante).

Resultado: El resultado es la definición del conjunto de señales y el conjunto de procedimientos exportados de cada proceso.

15.2.2 Paso B:2 – Procesos esquemáticos

Instrucciones

1) Producir MSC por lo menos para los usos "típicos" si éstos no se han generado como diseños previos.

2) Producir un proceso esquemático estableciendo una correspondencia con el MSC considerando exclusivamente usos "típicos":

   2.1 A partir del símbolo de inicio del proceso, construir un árbol de estados considerando los cambios de estado "normales" del proceso.

   2.2 Construir el árbol del proceso estableciendo ramificaciones desde cada estado sobre la base de cada una de las entradas del MSC que se consumen pero no se ignoran, continuando cada una de ellas con una transición (incluyendo salidas y creaciones) a distintos estados.

   2.3 Identificar que un estado es distinto a otros en función de que tenga un conjunto de señales consumidas o conservadas sea diferente, o que su respuesta a una señal sea diferente.

   2.4 Incluir la supervisión temporal (**set** y **reset**) y la correspondiente entrada de temporizador.

   2.5 Conforme se dibuja el árbol, identificar los puntos en los que el proceso retorna el mismo estado y convierte el árbol en un grafo.

3) Dibujar dicho grafo como un diagrama de proceso.

4) Determinar si el proceso tiene dos o más conjuntos disjuntos de interfaces para las distintas partes de comportamiento del proceso que pueden ser entrelazadas, dividiendo entonces el proceso en múltiples procesos (si los comportamientos son independientes).

NOTA – Si los comportamientos se emparejan utilizando datos comunes, la división del proceso en varios procesos exige uno o más procedimientos remotos para acceder a los datos. También es posible dividir dichos procesos en servicios SDL.

Directrices

Los MSC se producen como parte del Diseño Previo.

Los MSC pueden conducir de forma semiautomática a un proceso esquemático. El orden de los eventos en el eje vertical del MSC indica cual es el orden de los mismos en el proceso para el que se produce la descripción esquemática. En la práctica esto se hace mediante casos de utilización típicos en el MSC y escribiendo la definición del proceso a partir de dichos casos (los "casos simples y habituales"). A partir del MSC puede también determinarse la creación del proceso dinámico.

La supervisión temporal puede también mostrarse en el MSC. Ésta se utiliza para modelar un lapso de tiempo, supervisar la liberación de un recurso y supervisar las respuestas de recursos poco fiables.

En ocasiones, los MSC muestra un mensaje enviado a un proceso que requiere respuesta basada en información que no es accesible al proceso o no que está en el mensaje. Si el proceso solicitante tiene la información, la solución consiste normalmente en garantizar que ésta se envía en un parámetro de señal (actualizando las definiciones según convenga). Ésta

puede estar en el mensaje de petición o en un mensaje anterior relacionado. Si la información se encuentra en otro proceso o es externa al sistema, el proceso debe comunicarse con el propietario de la misma mediante señales o con un procedimiento remoto. Se deben actualizar los MSC o hacer una anotación indicando que son incompletos y simplificados.

Un proceso esquemático exhibe normalmente el comportamiento esencial de un proceso, pero no el comportamiento en toda su amplitud. A fin de comprobar la consistencia del proceso esquemático con respecto a otros diseños previos, éstos (y los MSC) deben compararse con el proceso esquemático.

Cuando dos procesos con N y M estados respectivamente se combinan en un solo proceso, éste tiene N × M estados. Por lo tanto, la división de dicho proceso produce dos procesos mucho más sencillos y de más fácil comprensión.

Cuando un proceso (o un procedimiento) es informativo, pueden haber transiciones espontáneas que comienzan con **none** que modelan el comportamiento del usuario u otros eventos no predecibles (véase por ejemplo la figura 15-5).

Regla 29 – En las partes normativas de una norma no deben utilizarse las transiciones espontáneas.

Regla 30 – Los MSC deben constituir trazas correctas del tratamiento de los mensajes por parte del sistema SDL o bien deben anotarse claramente cómo y porqué difieren del comportamiento SDL.

Resultado: El resultado es un proceso esquemático en forma de un diagrama de visión general de estado que incluye temporización y MSC típicos consistentes.

Ejemplo

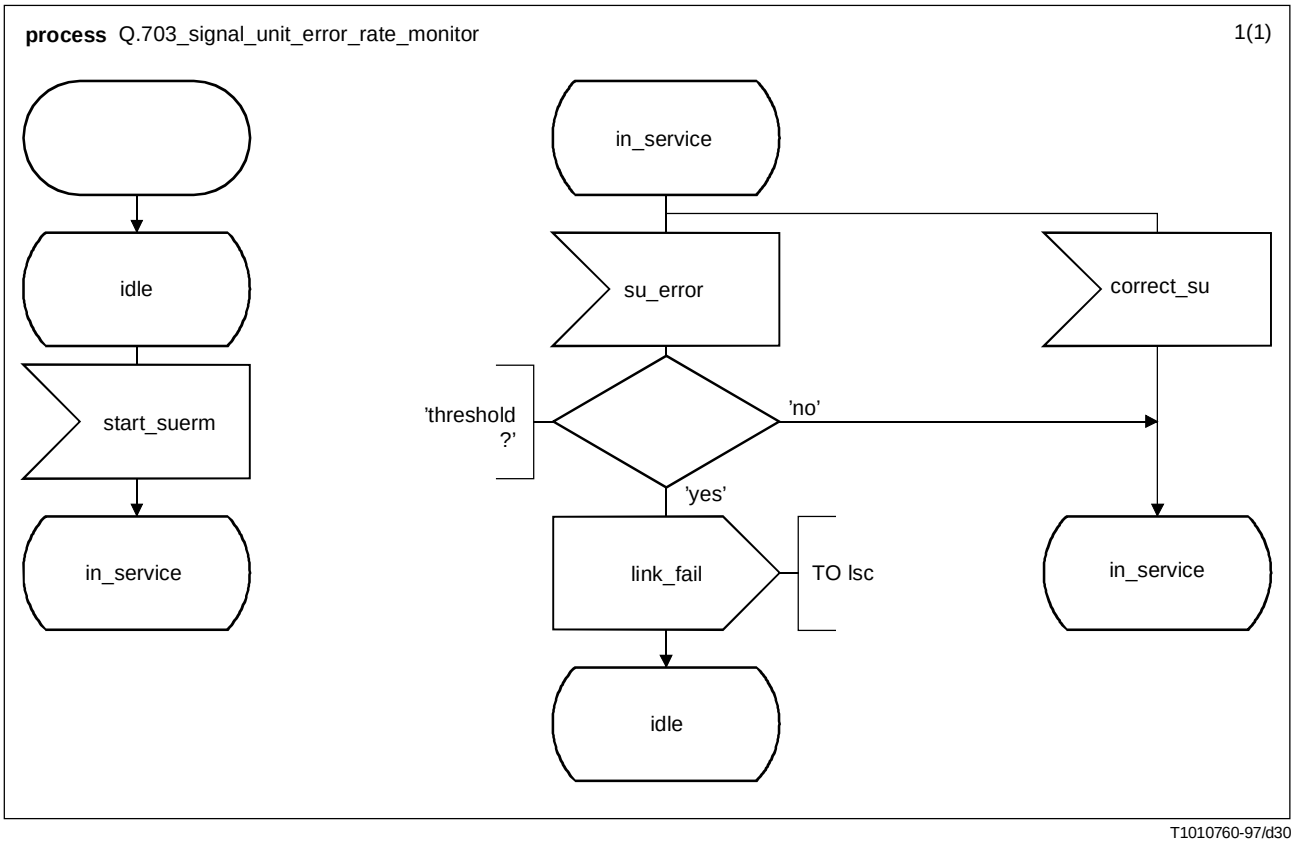


Figura 15-5/Sup. 1 a la Rec. Z.100 – Proceso esquemático

15.2.3 Paso B:3 – Procesos informales

Instrucciones

1) Identificar combinaciones de usos.

2) Identificar la información que almacena el proceso y analizar si está implícita en los estados del proceso o si se necesitan datos internos.

3) Utilizar esta información para definir las acciones internas de cada proceso.

4) Añadir tareas o procedimientos y posiblemente más decisiones en las transiciones, pero sólo debe utilizarse texto informal en tareas y decisiones.

5) Cuando se utilice un procedimiento, darle un nombre con sentido y realizar (posteriormente) los pasos B:1 a D:9 para definirlo.

Directrices

Utilizar los MSC existentes y generar nuevos MSC para identificar combinaciones de usos.

Decidir si se hace uso de una tarea o de un procedimiento para describir el procesamiento de la información en una transición, aunque esto pueda cambiarse ulteriormente. Utilizar un procedimiento si se prevé que otro proceso necesite la información, si es probable que la tarea sea compleja o para que exista dependencia de varios valores de datos almacenados; en cualquier otro caso se elige una tarea. Si se elige un procedimiento, puede ser adecuado considerar los parámetros del mismo (número y género).

Algunas veces es obvio que las mismas acciones se realizan en varios lugares del proceso. Dichas acciones pueden recopilarse y configurar un procedimiento.

Resultado: El resultado es un proceso informal que abarca todos los casos del MSC.

Ejemplo

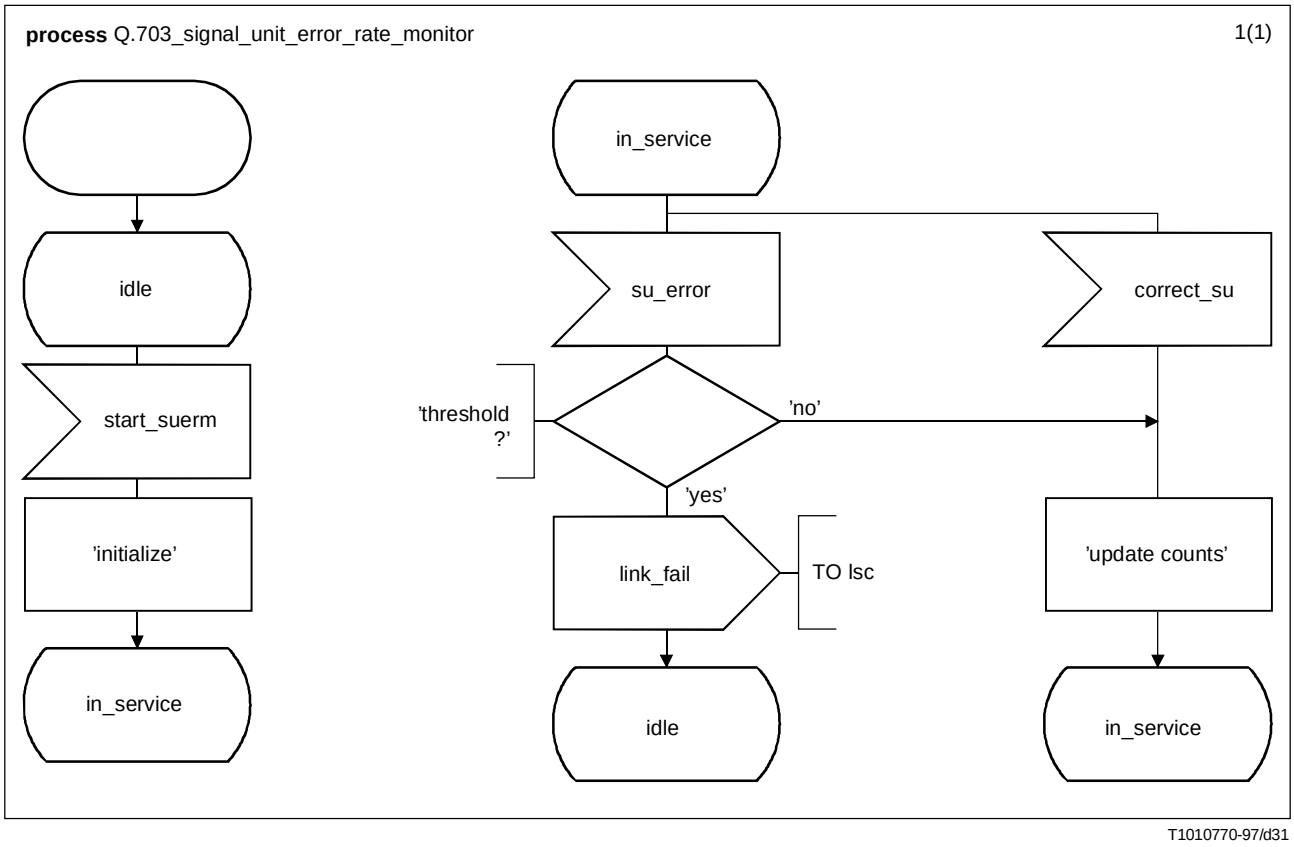


Figura 15-6/Sup. 1 a la Rec. Z.100 – Versión informal del mismo ejemplo del paso B:2

15.2.4 Paso B:4 – Compleción de los procesos

Instrucciones

1) En cada estado y para cada ítem del conjunto de señales (señal o procedimiento remoto) determinar si la señal (o la llamada al procedimiento remoto) es una entrada que realiza el proceso o es conservada por el mismo.

2) Si el ítem es una entrada, determinar cual es la transición (puede ser una transición implícita de vuelta al mismo estado).

3) Continuar con las instrucciones 1 y 2 hasta que no haya estados al final de una transición que no se hayan considerado, de forma que para cada estado se hayan considerado todos los ítems del conjunto de señales.

4) Analizar en primer lugar cada proceso y a continuación las combinaciones hasta abarcar todo el sistema SDL a fin de comprobar la posible existencia de propiedades indeseadas y, si es preciso, rediseñarlo para evitarlas.

Directrices

Mediante una matriz de señales de estado (véase I.9.4/Z.100 [1]) puede comprobarse la acción de cada señal en cada estado. Esta matriz puede ayudar a identificar cuando dos de los estados definidos dan lugar al mismo comportamiento y pueden combinarse, o cuando dos señales tienen los mismos estados adyacentes. En dichos casos es posible reducir el número de señales o de estados.

Puede dibujarse un diagrama de la visión general de estado (véase I.9.3/Z.100 [1]) a fin de disponer de una visión general del comportamiento del proceso. Si el proceso no finaliza, cada estado puede alcanzarse normalmente desde cualquier otro estado; no obstante, esto no tiene que ser necesariamente así pues pueden existir estados iniciales para el arranque del proceso y estados finales para la terminación del mismo.

Existe un límite sobre cuanto es conveniente investigar en torno a un proceso. Los resultados más interesantes de la investigación se logran al aumentar el número de procesos "completos", y pueden analizarse combinados en un bloque o conformando el sistema en su totalidad. Las propiedades indeseadas que puedan detectarse (tales como estados inalcanzables, punto muerto o punto de rearranque) dependen de la complejidad del sistema y de las herramientas disponibles.

Con fines de investigación puede suponerse que cada decisión contiene un constructivo **any** SDL.

La investigación de todo lo que no sea un proceso simple es difícil sin herramientas que generen el espacio de estado y que puedan comprobarlo. Incluso las herramientas tienen dificultades cuando hay un gran número de procesos o un proceso muy complejo. No se trata, pues, de un paso trivial, pero en esta etapa la investigación puede ahorrar mucho tiempo y esfuerzo si se determina que es necesario realizar de nuevo el diseño, lo que constituye una de las principales ventajas de la utilización de SDL.

Resultado: En esta etapa se ha completado, aunque de manera informal, la definición del proceso (procedimiento o servicio).

Regla 31 – Todos los estados de un proceso deben ser alcanzables desde el inicio del mismo.

Regla 32 – Un procedimiento que sea exportado por un proceso (para ser utilizado como un procedimiento remoto) no debe ser conservado en cada estado de dicho proceso.

Regla 33 – Cada señal recibida por el proceso debiera tener al menos una entrada que conduzca a una transición no vacía.

15.3 Pasos de datos (pasos D)

El objetivo de los pasos de datos es proporcionar una definición formal de los datos utilizados en los procesos. Si datos formales, cualquier decisión sobre el comportamiento y los operadores empleados en las expresiones tienen resultados inciertos.

Los primeros dos pasos (D:1 a D:2) se refieren a los valores de la interfaz. Los tres pasos siguientes (D:3 a D:5) se refieren a las variables de los procesos. Los restantes pasos (D:6 a D:9) se refieren a la definición formal de nuevos géneros de datos. Los últimos pasos son necesarios si los nuevos géneros de datos tienen nuevos operadores. Por ello, los pasos D:6 a D:9 sólo son necesarios ocasionalmente y resultan innecesarios si se ha utilizado ASN.1 para definir los datos.

El paso D:1 se aplica al sistema en su conjunto, mientras que los pasos D:2 a D:9 se pueden aplicar a cada proceso o procedimiento.

Regla 34 – Los géneros de los datos empleados se definen utilizando SDL combinado con ASN.1 tal como se define en la Recomendación Z.105 [2].

15.3.1 Paso D:1 – Parámetros de la señal

Instrucciones

1) Identificar los valores que deben transportar las señales, comenzando por las señales a nivel de sistema.

2) Buscar géneros de datos predefinidos para representar los valores identificados.

3) Ampliar las definiciones de las señales con el género de los datos.

4) Identificar y definir, si ello es necesario, nuevos géneros de datos.

Ejemplo

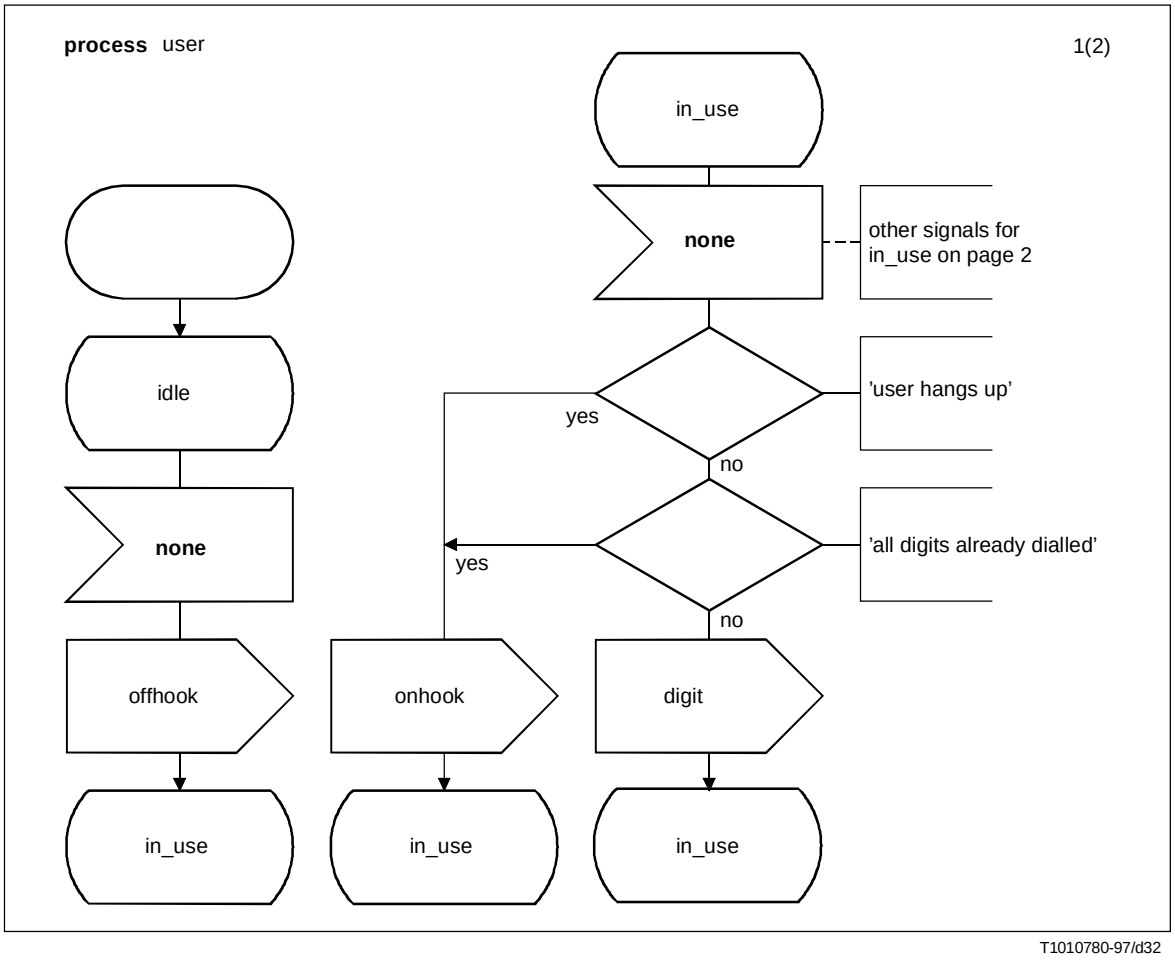


Figura 15-7/Sup. 1 a la Rec. Z.100 – Parte de un proceso completo pero informal

Directrices

La descripción de la información clasificada de la comunicación proporciona una fuente de nombres de géneros de datos.

ASN.1 proporciona una descripción formal de los datos utilizados en las señales y, por lo tanto, puede utilizarse directamente el correspondiente género de datos. Si no se ha utilizado ASN.1, debe decidirse si se definen los géneros de datos utilizando ASN.1 o SDL. Si la codificación de los datos es importante o si los datos están en la interfaz con el entorno debería utilizarse ASN.1. Puede utilizarse SDL si no se necesita precisa ninguna codificación en particular y si el género de los datos sólo se utiliza en el sistema SDL.

Regla 35 – Si se han utilizado cuadros de bits para definir los datos, éstos deben convertirse a ASN.1 o SDL.

Regla 36 – Los nombres de los géneros de datos existentes no debieran utilizarse para nuevos géneros de datos, incluso si éstos tienen campos de aplicación diferentes.

NOTA – Si los géneros de los datos tienen el mismo campo de aplicación, las reglas del lenguaje SDL no permiten que los nombres sean los mismos.

Resultado: Las señales tienen parámetros con los géneros de datos definidos correspondientes.

Ejemplo

En la figura 15-5 la señal "digit" puede ampliarse a "digit(digit)" en la que el género de los datos se define tal como se muestra en la figura 15-1.

15.3.2 Paso D:2 – Parámetros del proceso y del procedimiento

Instrucciones

1) Identificar los géneros de los datos necesarios para los parámetros del proceso (o procedimiento).

2) Determinar el cometido de cada parámetro mediante un comentario en el título del proceso (o procedimiento).

Directrices

Dentro de un proceso, un parámetro se trata como una variable. La única diferencia entre un parámetro de proceso y una variable con valor por defecto es que un parámetro de proceso puede recibir un valor diferente para cada instancia del proceso. Normalmente, este valor es la identidad de alguna otra entidad tal como un Pid o un número de equipo.

La información clasificada y los diseños previos relativos a la creación y supresión proporcionan directrices sobre el cometido de los parámetros que se pasan a un proceso cuando éste se crea. La invocación de un procedimiento puede corresponder a la creación del mismo (y el retorno de procedimiento puede corresponder a la supresión).

Regla 37 – Un parámetro de proceso no debiera contener el Pid padre ya que éste está disponible como el parent.

Resultado: Los parámetros del proceso (o procedimiento) han sido formalizados.

15.3.3 Paso D:3 – Parámetros de entrada

Instrucciones

1) Añadir parámetros a las entradas conforme a las definiciones de las señales.

2) Definir variables según se requieran para recibir los valores de entrada.

3) Establecer el cometido de cada variable en un comentario.

Directrices

Comprobar que cada parámetro definido en una definición de señal se utiliza al menos en una entrada o es necesario para la comunicación con el entorno.

Regla 38 – Un parámetro de señal no debiera incluir el Pid emisor ya que éste está disponible como ***sender****.*

Resultado: Se han formalizado las interfaces de entrada.

15.3.4 Paso D:4 – Transiciones formales

Instrucciones

1) Sustituir el texto informal de las tareas, decisiones y respuestas por asignaciones formales, expresiones formales, expresiones de gama formal y llamadas a procedimientos, utilizando cualquier nuevo operador utilizado en las expresiones formales que deba añadirse a los géneros de los datos en el paso D:6.

2) Definir, cuando sea necesario, variables y sinónimos adicionales.

3) Definir, según las necesidades, procedimientos adicionales.

4) Añadir parámetros a las llamadas de procedimientos de acuerdo con las definiciones de los mismos.

Directrices

El texto informal anterior realizado mediante símbolos puede ser de utilidad como comentarios anexos a los símbolos.

Si el valor de la función utilizada en una expresión sólo es función de los parámetros efectivos, la función puede convertirse en un operador o en un procedimiento. Si el resultado es función de otros datos, debe convertirse en procedimiento. Si el comportamiento de la función depende de datos de otros procesos, puede ser adecuado convertirla en un procedimiento remoto o bien, determinar como se obtiene dicha información. Para obtener la información pueden ser necesarios parámetros adicionales a las señales existentes, así como el almacenamiento de dicha información o la comunicación en el procedimiento, posiblemente con señales adicionales.

La definición de un procedimiento tiene un tratamiento semejante al de un proceso a través de los pasos B:1 a D:9, incluyendo la posibilidad de disponer de un procedimiento que se llama internamente y un procedimiento anidado dentro del procedimiento.

Cuando el comportamiento normativo es independiente de algunos datos de procesos o procedimientos informativos, el constructivo **any** puede utilizarse para crear valores aleatorios para dichos datos o decisiones aleatorias en procesos o procedimientos informativos. Así, las partes informativas pueden modelar situaciones en las que el comportamiento y los datos son impredecibles. Las transiciones espontáneas se utilizan si es impredecible determinar el momento ("cuando") ocurre algo (véase 15.2.2), utilizándose el valor **any** si lo que es impredecible es el hecho en si mismo.

*Regla 39 – Un valor o una decisión que no sea determinística (utilizando **any**) no debe aparecer en una parte normativa del sistema salvo que se marque explícitamente como informativo.*

*Regla 40 – Un valor o una decisión que no sea determinística (utilizando **any**) debe tener siempre un comentario adjunto explicando como se realiza la elección.*

Resultado: Al finalizar este paso se ha eliminado el texto informal.

Ejemplo

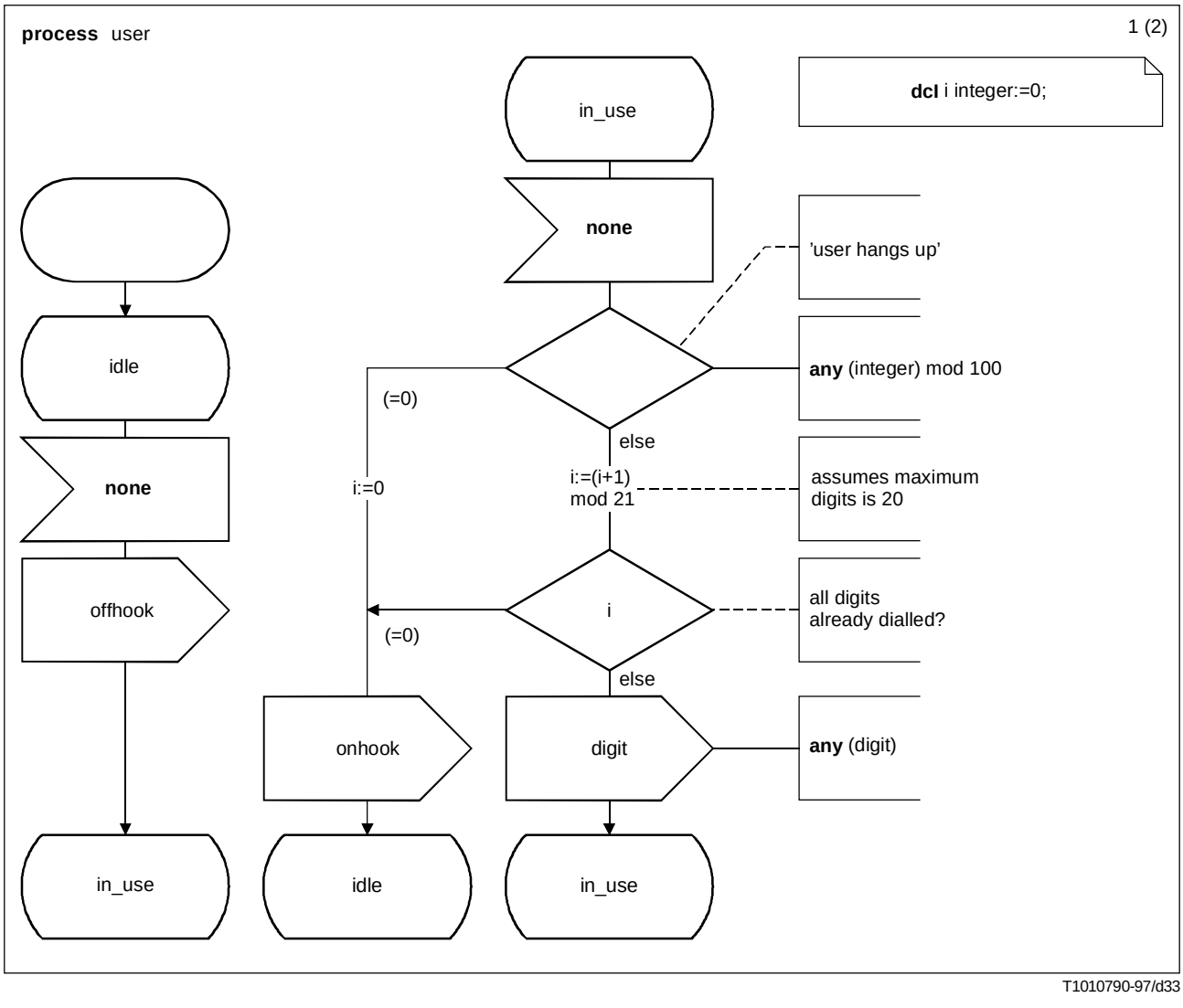


Figura 15-8/Sup. 1 a la Rec. Z.100 – Parte de un proceso completo pero informal para un usuario

Este ejemplo formaliza el proceso de usuario de la figura 15-5 e ilustra la utilización del constructivo **any**.

15.3.5 Paso D:5 – Parámetros de salida y de creación

Instrucciones

1) Añadir expresiones a salidas utilizando variables y sinónimos ya disponibles.

2) Añadir parámetros efectivos para crear acciones.

Directrices

Comprobar que cada uno de los parámetros enviados a una salida se utiliza al menos en una posible entrada o se necesita para la comunicación con el entorno. En contraste con la comprobación del paso D:3 relativa a que se utiliza un parámetro de definición de señal, en este caso se comprueba que puede utilizarse un parámetro de instancia de señal.

Regla 41 – Si en una salida se omite un parámetro, no debe de haber una entrada que espere un valor para dicho parámetro.

Resultado: La descripción SDL es ahora formal, excepto en lo relativo al comportamiento de las expresiones, que depende de los tipos de datos.

15.3.6 Paso D:6 – Signaturas de datos

Instrucciones

1) Identificar y definir todos los géneros de datos y valores (sinónimos) que es necesario que sean definidos.

2) Si algunos valores dependen de la instalación del sistema, se expresan utilizando sinónimos externos.

3) Para cada género de datos necesario se crea un neotipo (newtype) o un sintipo (syntype) con un nombre significativo o se define un tipo ASN.1.

4) Identificar cualquier nuevo operador que deba ser definido.

   NOTA – Si no hay nuevos operadores, el proceso de formalización queda completo.

5) Para los neotipos con nuevos operadores, listar las signaturas (los literales y los operadores con los géneros de los parámetros) e identificar (en un comentario) un conjunto de operadores y literales que puedan utilizarse para representar todos los valores posibles (los "constructores").

Directrices

Los sinónimos externos pueden utilizarse para:

– proporcionar dimensiones para el número de procesos, el número de ítems0 de datos de una matriz, etc.

– elegir entre las opciones de transición proporcionando funcionalidades facultativas y un comportamiento alternativo.

Heredar el mayor número posible de los datos predefinidos de la Recomendación Z.100 [1], de los "tipos útiles" de ASN.1 y de los datos predefinidos de la Recomendación Z.105 [2]. El objetivo es evitar tener que definir el comportamiento para nuevos operadores. Si los nuevos operadores están definidos, deberían definirse mediante una definición de operador.

La introducción de nuevos operadores puede evitarse como sigue:

– utilizando las definiciones ASN.1 que se definen en la Recomendación Z.105 [2];

– utilizando **struct** para registros;

– utilizando un **generator** predefinido, tal como una matriz o una cadena;

– utilizando un **syntype** si el conjunto de valores de datos debe ser un subconjunto de un género de datos y compatible con el mismo;

– dando un nuevo nombre a un género con **syntype** si el objetivo es introducir un nombre más significativo;

– dando un nuevo nombre a un género con un **newtype** el cual **hereda todo** (**inherits all**) si el objetivo es disponer de un género de datos con las mismas propiedades que uno ya existente, pero no compatible con el género existente;

– utilizando un **generator** definido que evite tener que definir axiomas utilizando otros generadores.

Los operadores se listan normalmente en el neotipo para el género de datos correspondiente al resultado del operador. A veces, un operador produce un valor de un género de datos que está definido en un contexto diferente, por ejemplo, un valor entero o booleano. En este caso, el operador se lista en el newtype del parámetro géneros de datos.

El conjunto de operadores y literales que pueden utilizarse para representar todos los valores posibles es un conjunto de constructores de género de datos. Dichos constructores se utilizan en pasos ulteriores si y solo si, ello es necesario para definir propiedades del operador mediante axiomas SDL. Normalmente el conjunto de constructores no es único, ni ello es siempre

obvio. El conjunto elegido debiera ser suficiente para definir todos los valores de forma que si se suprime un constructor, el conjunto de valores es diferente. La elección puede resultar ardua.

Aunque este paso produce una descripción formal SDL, si se introducen nuevos operadores la funcionalidad puede diferir de la que se pretendía debido a que los géneros de datos definidos por el usuario para dichos operadores puede tener "demasiados" valores. El paso D:7 corrige esta deficiencia pero hace que la interpretación dependa de texto informal. Los pasos D:8 y D:9 hacen que la descripción sea formal.

Resultado: En esta etapa el sistema en su totalidad ha quedado definido formalmente y está completo si no se han introducido nuevos operadores.

15.3.7 Paso D:7 – Descripción informal de datos

Instrucciones

Añadir axiomas informales como texto informal a las definiciones de neotipo.

Directrices

Los nombres de los operadores debieran corresponder a sus funciones, y por lo tanto, ser de ayuda en la descripción de la función.

Aunque el resultado sea un SDL correcto, sólo pueden analizarse completamente las propiedades estáticas. Ello se debe a que las propiedades dinámicas dependen de la interpretación de los axiomas, lo cual solo puede realizarse informalmente. No obstante, en un entorno de soporte real, algunas características o hipótesis pueden permitir que este nivel de descriptivo sea suficiente.

Resultado: Los axiomas informales permiten registrar lo que se pretende conseguir mediante la definición del neotipo, sin una definición formal.

15.3.8 Paso D:8 – Descripción formal de los datos

Instrucciones

1) Para cada signatura de operador añadir, si ello es posible, una definición de operador en forma de diagrama de operador.

   NOTA – Si todos los operadores pueden definirse de esta forma se completa la formalización.

2) Si alguno de los operadores no puede definirse mediante diagramas de operador, se formalizan los axiomas sustituyendo los que son informales por axiomas formales que establecen las propiedades esenciales de los operadores.

3) Utilizar el texto de los axiomas informales como comentarios de los axiomas formales.

Directrices

Los operadores pueden definirse algorítmicamente utilizando un diagrama de operador o de forma axiomática. La forma algorítmica del diagrama de operador es recomendable por ser más fácil de entender. El método axiomático se describe a continuación.

Aunque el método axiomático tiene una mejor base matemática y se utiliza en las Recomendaciones Z.100 [1] y Z.105 [2], es más difícil escribir axiomas correctos que construir un diagrama de operador que haga lo que se desea. Aparentemente, a muchos ingenieros les resulta difícil definir tipos de datos mediante axiomas, por lo que este método provoca fácilmente errores. Las herramientas disponibles manejan con dificultad los axiomas, resultando más difícil encontrar buenas herramientas soporte. Los axiomas no deben por lo tanto utilizarse si no se dispone de la experiencia o la dirección de un usuario experto. Para aplicar un operador en SDL es suficiente tener definida la signatura.

Para especificar las propiedades de un operador mediante axiomas, se especifican en primer lugar las propiedades de los constructores. Ello permite conocer los posibles valores del género. Se especifican a continuación las ecuaciones para el resto de los operadores y literales. Es bastante fácil establecer las propiedades esenciales, pero normalmente resulta difícil garantizar que los axiomas son completos y que no entran en conflicto (véase paso D:9).

Estos axiomas pueden considerarse informativos. Salvo que se prevea utilizar los axiomas como base para una implementación automática, puede considerarse que la formalización se ha completado incluyendo en los axiomas un comentario que indica que son informativos, que no se han demostrado y que pueden ser incompletos. En este caso se omite el paso D:9.

Regla 42 – No se deberían utilizar los axiomas para definir las propiedades de un operador.

Resultado: El resultado es que algunas de las propiedades de los tipos de datos se han definido, aunque probablemente no todos, de manera que, formalmente, cada género tiene muchos más valores de los inicialmente planeados (normalmente muchísimo más).

15.3.9 Paso D:9 – Compleción de la formalización de los datos

NOTA – Este paso sólo se debería utilizar en condiciones excepcionales.

Instrucciones

Añadir axiomas (o definiciones de operador) a las definiciones de neotipos de datos hasta que estén completos (es decir, hasta que todas las expresiones que incluyen operadores no constructores y literales puedan ser escritas de nuevo en forma de expresiones que solo contengan operadores constructores y literales).

Directrices

Las directrices detalladas en este paso se describen en I.5/Z.100 [1].

Evitar la definición de constructores (es decir, operadores que crean valores de un género de datos). En vez de ello, debe garantizarse que hay un literal para todos y cada uno de los valores de un género. Suponiendo que no se utiliza la **noequality** (**desigualdad**), cada literal que se define tiene un valor único.

En el caso de que sean necesarios nuevos operadores que utilicen completamente los géneros definidos en las unidades de campo de aplicación circundante, el operador se define en un neotipo postizo (*dummy*).

En ningún caso (de forma intencionada o no) debe cambiarse el número de valores de un género utilizando axiomas en el neotipo de un género diferente.

Resultado: Si se han seguido todos los pasos anteriores, en esta etapa se ha conseguido una especificación formal SDL, incluyendo definiciones completas e inequívocas para todos los géneros de datos utilizados en la especificación.

15.4 Pasos de tipo (pasos T) (T-steps, *type steps*)

Los pasos de tipo se utilizan para definir las partes del sistema como tipos de tal forma que puedan ser reutilizadas.

Para obtener el máximo beneficio de la aplicación de estos tipos se recomienda que en grandes sistemas se apliquen en paralelo con los pasos precedentes. Ello se debe a que cuando se desarrolla una rama de la jerarquía del sistema, pueden generarse tipos que pueden reutilizarse en otra rama.

Los pasos más importantes son T:1 y T:2, mientras que los pasos T:3 a T:5 pueden considerarse opcionales. Es aconsejable disponer de unos buenos conocimientos de SDL-92 y de orientación a objetos antes de utilizar los pasos T:3 a T:5. En algunos casos en lugar de la especialización puede utilizarse la parametrización, tal como se describe en el paso L:2.

15.4.1 Paso T:1 – Identificación de tipos de SDL

Instrucciones

1) Modificar las definiciones de instancias utilizadas en el sistema para utilizar tipos.

2) Cuando dos instancias (por ejemplo, dos bloques) tienen el mismo comportamiento, se utiliza el mismo tipo para ambas instancias.

Directrices

Permite que los tipos sean claramente reconocidos, pudiéndose eliminar alguna repetición innecesaria en el sistema SDL. Una situación típica es aquella en la que se define un sistema "extremo a extremo" y la terminación en cada extremo se describe mediante bloques que tienen el mismo comportamiento. La definición debe repetirse para ambos bloques salvo que se utilice un tipo de bloque, en cuyo caso la misma definición del tipo de bloque es válida para ambos bloques.

La definición de un tipo de sistema es facultativa pero permite que la definición de sistema se reutilice como base para otras normas semejantes. Esto es especialmente útil en el caso de un conjunto de normas relacionadas.

La adición de bloques o tipos de procesos requiere que se añadan puertas para identificar como la conexión con la definición utilizando el tipo se corresponde con los trayectos de comunicación del tipo.

Un proceso o bloque que sólo se utiliza una vez debería ser inicialmente modelado como un tipo porque es posible sustituirlo en un paso ulterior por un tipo basado en otro tipo existente. Aunque es más fácil producir inicialmente diagramas sin utilizar tipos, la introducción de tipos permite reutilizaciones que pueden ahorrar numerosas descripciones repetitivas así como esfuerzos de ingeniería.

Cabe señalar que todas las definiciones de señales de procedimientos son definiciones de tipos.

Regla 43 – Dos bloques (o procesos) que modelan instancias de lo mismo, deben ser modelados mediante definiciones de bloques (procesos) que se basen en una definición común de tipo de bloque (proceso).

Resultado: Tipos definidos para el sistema, bloques y procesos.

Ejemplo

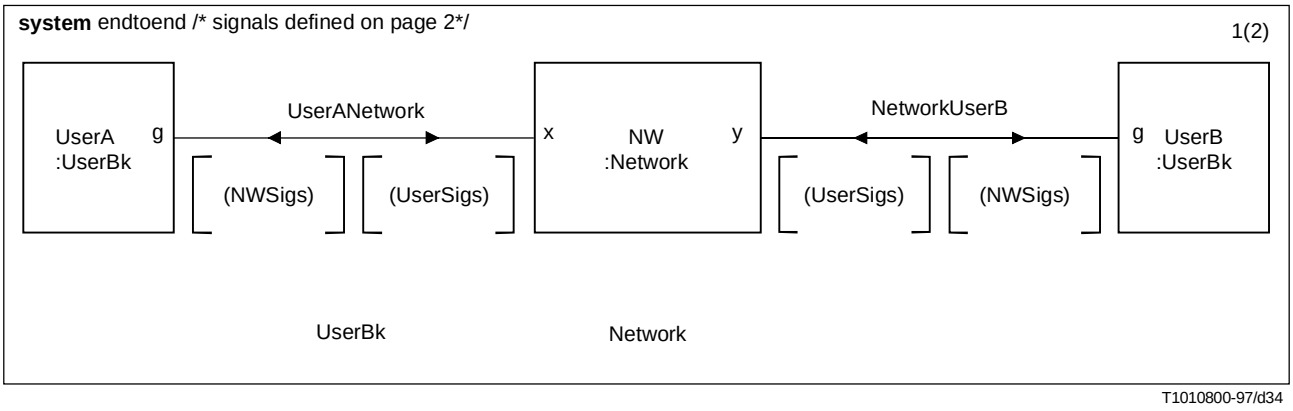


Figura 15-9/Sup. 1 a la Rec. Z.100 – Definición del sistema mediante tipos

15.4.2 Paso T:2 – Especialización de tipos por adición de propiedades

Instrucciones

1) Identificar los casos en los que un tipo tiene el mismo comportamiento que otro tipo cuando recibe los mismos estímulos (con señal o con un procedimiento de llamada remoto) que aquél, pero que además maneja estímulos diferentes.

2) Definir el primer tipo como un subtipo que **hereda** (**inherits**) las propiedades del segundo tipo y al que se **añaden** (**adding**) propiedades para poder manejar estímulos adicionales.

Directrices

Un subtipo es un tipo creado especializando otro tipo (llamado el supertipo del subtipo). La especialización puede hacerse de dos maneras: añadiendo propiedades al supertipo o cambiando en el subtipo algunas de las propiedades del supertipo. En este paso se considera la adición de propiedades al supertipo.

Las propiedades que pueden añadirse están limitadas sólo por las ya definidas para el tipo, de forma que las propiedades añadidas no entren en conflicto con la definición heredada. Los canales y los bloques pueden añadirse a un tipo de sistema o a un tipo de bloque que circunden bloques. Las rutas de señal, las definiciones de procesos (tipos), las definiciones de procedimientos y de señales, pueden añadirse a un tipo de bloque que circunde procesos. A un tipo de proceso se pueden añadir nuevas transiciones para nuevas señales, pudiendo las transiciones dar lugar a nuevos estados.

Resultado: Un nuevo subtipo con comportamiento ampliado pero compatible con el comportamiento anterior.

Ejemplo

Dos procesos (suerma y suerma_supertype) son idénticos excepto en que un proceso (suerma) acepta una señal adicional (abort_call). Cuando entra abort_call reinicia el proceso al estado de reposo con independencia de cual sea el estado anterior del proceso. El tipo de proceso para el segundo proceso puede ser un subtipo para el primer proceso (véase la figura 15-10).

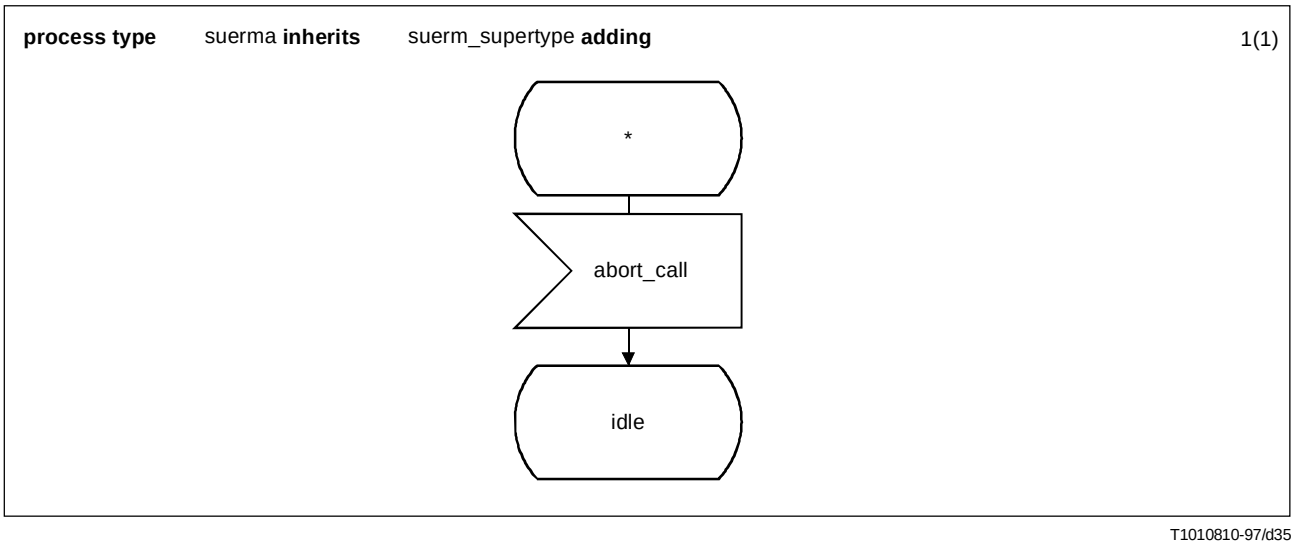


Figura 15-10/Sup. 1 a la Rec. Z.100 – Un subtipo creado añadiendo una propiedad

15.4.3 Paso T:3 – Utilización de "virtual" para generalización de tipos

Instrucciones

1) Si no hay tipos que tengan la misma estructura interna y comportamiento pero que manejen situaciones ligeramente diferentes (son "similares"), el paso queda completado.

2) Para cada conjunto de tipos "similares" (como se indica en la instrucción 1) se aplican las instrucciones 3 a 6.

3) Identificar la parte común y especificarla como un supertipo general que es utilizado por los otros tipos que se convierten en subtipos cuando se aplica el paso T:5.

4) Si el supertipo general es un tipo de bloque, identificar las instancias (bloques y procesos) de los subtipos que tienen que ser diferentes, convertirlas en instancias basadas en tipos, definiendo los tipos como tipos virtuales del supertipo general.

5) Si el supertipo general es un tipo de proceso o de procedimiento, identificar las secuencias de acción parcial, que deberían ser adaptables y:

   5.1 si son transiciones completas, hacerlas transiciones virtuales;

   5.2 en otro caso, sustituir las secuencias de acciones parciales por llamadas de procedimientos y definir los correspondientes procedimientos virtuales;

   5.3 si hay otros procedimientos en el supertipo general, considerar si estos deberían ser virtuales.

6) Para cada tipo virtual del supertipo general, identificar o definir un tipo que tenga la estructura interna y los parámetros adecuados a todas las redefiniciones y utilizar dicho tipo como constricción para el tipo virtual elaborado en el paso T:4.

NOTA – El supertipo definido en este paso no se utiliza en el sistema SDL hasta que se aplican otros pasos.

Directrices

Un supertipo general es un supertipo algunas de cuyas partes pueden redefinirse. Dichas partes se identifican como virtuales. Si el tipo se utiliza sin redefinir dichas partes, son aplicables las definiciones de las partes virtuales. Por lo tanto, las definiciones de las partes virtuales del tipo general se eligen de forma que modelen la variación más común.

Debe evitarse generalizar en exceso los tipos. Existe también el peligro de generalizar cuando dos tipos tienen un comportamiento similar pero conceptos bien diferentes, por ser difícil entender cabalmente el objetivo del supertipo general tal como se aplica a cada caso. El supertipo general debiera recoger las características esenciales de un concepto tal como el comportamiento que no es virtual.

Una transición virtual permite redefinir toda la transición. Considerar la conversión de las partes de la transición en procedimientos virtuales de tal forma que las partes comunes sean fijas y la transición completa no tenga que ser virtual.

Regla 44 – La parte común de un supertipo debiera constituir al menos el 70% del total de cada subtipo.

Resultado: Un tipo generalizado mediante la definición como virtuales de algunos de los tipos locales o transiciones.

Ejemplos

Un caso típico es el de dos procesos que son idénticos salvo en la transición de una señal en un estado. El supertipo es la definición del proceso en la que dicha transición es virtual o contiene una llamada de uno o más procedimientos virtuales.

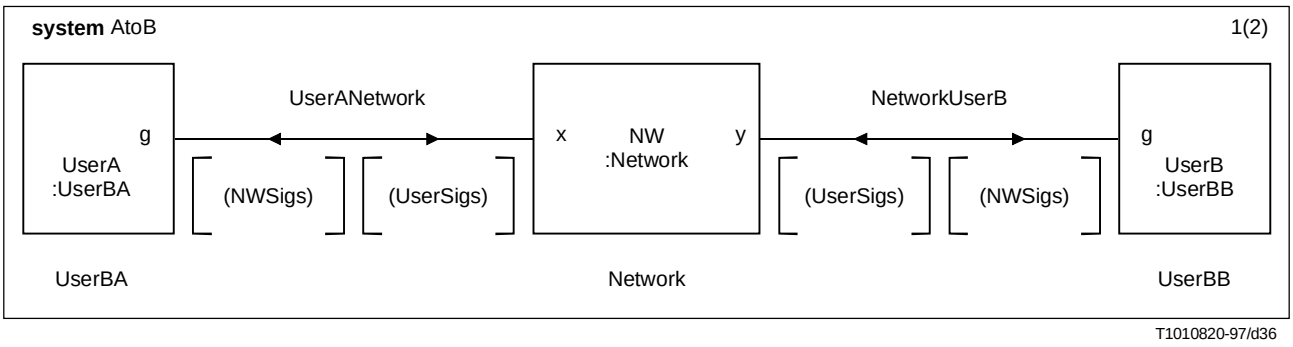


Figura 15-11/Sup. 1 a la Rec. Z.100 – El sistema AtoB (de A a B)

Considérese otro ejemplo en el que un sistema "AtoB" representa la comunicación de A a B a través de una red. A y B son bloques casi idénticos que contienen un proceso; cada uno se llama User. El proceso User es ligeramente diferente en cada caso. El supertipo general puede ser un tipo de bloque UserBk.

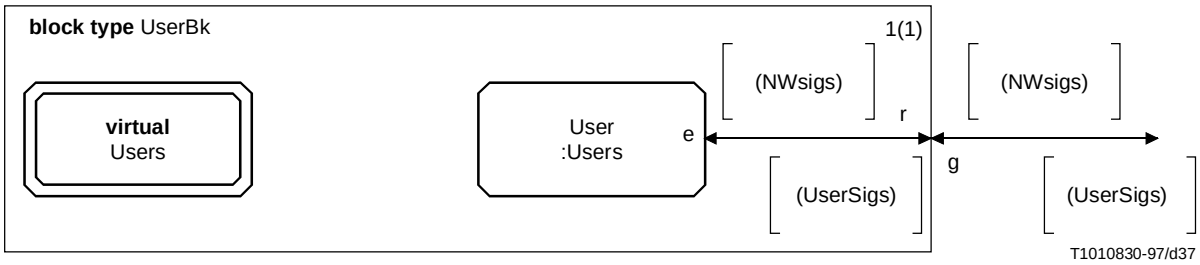


Figura 15-12/Sup. 1 a la Rec. Z.100 – El tipo de superbloque general para "users"

Debido a que sólo hay un proceso en cada bloque y éste tiene un comportamiento diferente en cada caso, este proceso se basa en el tipo de proceso **virtual** Users. Los tipos de bloque UserBA y UserBB pueden definirse como herederos del supertipo general UserBk y redefiniendo el proceso virtual Users.

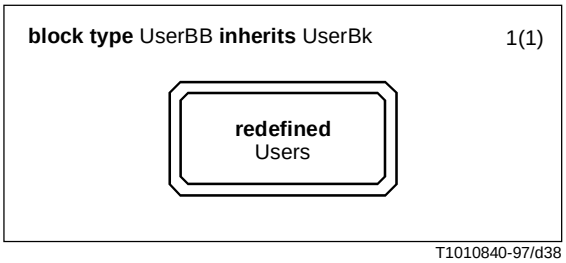


Figura 15-13/Sup. 1 a la Rec. Z.100 – Definición de UserBB utilizando el supertipo general UserBk

15.4.4 Paso T:4 – Constricción de los tipos virtuales

Instrucciones

1) Decidir cuales son las propiedades de un tipo virtual que deben aplicarse a todas las redefiniciones de dicho tipo virtual.

2) Definir (o identificar) un tipo que tenga dichas propiedades (llamado el "tipo de constricción").

3) Añadir **al menos** (**atleast**) un "tipo de constricción" a las definiciones de los tipos **virtuales** (**virtual**) del supertipo general.

4) Definir un tipo **redefined** (basado en el tipo de constricción virtual) para los casos en que se utiliza el tipo virtual.

Directrices

Un supertipo general circunda tipos virtuales que pueden ser diferentes en cada ocasión que el supertipo general se utiliza para definir un subtipo. Es necesario que haya una definición del tipo virtual circundado por el supertipo general y una definición del tipo (correspondiente al tipo virtual) circundado por cada subtipo (del tipo general). El tipo de constricción:

– define las propiedades comunes del tipo virtual en el supertipo general y en cada una de las redefiniciones del tipo virtual;

– constriñe las definiciones de forma que cada definición sea un subtipo del tipo de constricción y, por lo tanto, debe heredar el tipo de constricción.

Si la definición del tipo virtual circundado en el subtipo coincide con el tipo de constricción, éste constituye el valor por defecto y se omite la redefinición en el subtipo.

Regla 45 – El tipo de constricción debe fijar las propiedades de:

– *los parámetros de un procedimiento virtual de forma que todas las llamadas tengan el mismo número de parámetros y del mismo tipo;*

– *parámetros, puertas y comportamiento común para un proceso virtual;*

– *puertas y bloques internos o procesos conectados a la puerta para un bloque virtual.*

Resultado: Una definición de tipo (el "tipo de constricción") para cada tipo virtual del supertipo general y un subtipo (del tipo de constricción) para cada comportamiento real.

Ejemplo

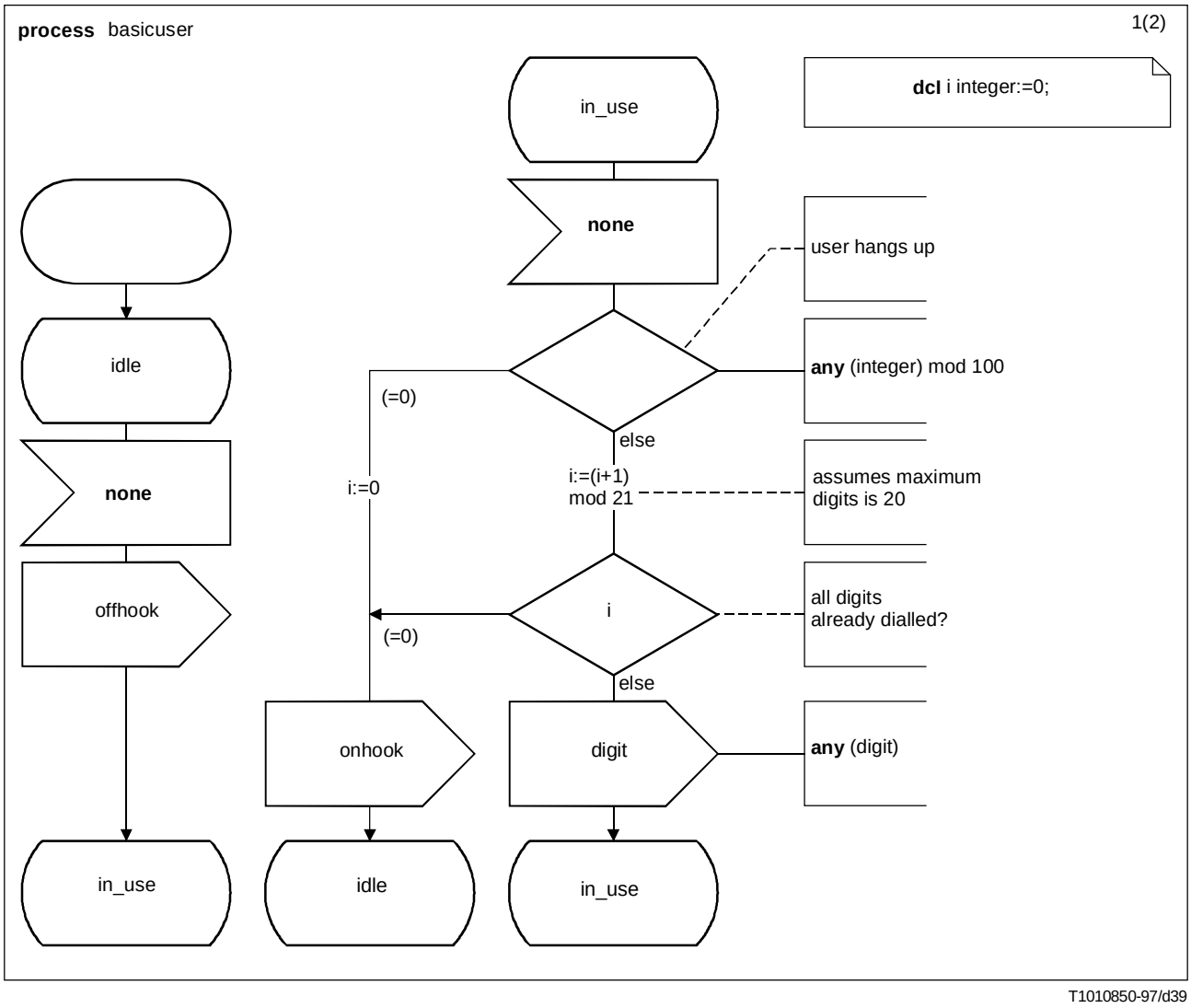


Figura 15-14/Sup. 1 a la Rec. Z.100 – El tipo de proceso basicuser

Se supone que cada vez que se utiliza UserBk, el tipo de proceso Users tiene al menos las propiedades de un basicuser. La definición del tipo de proceso virtual para Users en el tipo de super bloque general UserBk se define que sea **atleast** basicuser.

15.4.5 Paso T:5 – Especialización mediante la redefinición de tipos

Instrucciones

1) Sustituir cada subtipo identificado en el paso T:3 por un tipo que **inherits** del supertipo general, y en este subtipo.

2) Redefinir las transiciones virtuales del contexto.

3) Redefinir los tipos virtuales del contexto utilizando **redefinitions** del paso T:4.

Directrices

Este paso es necesario para producir un tipo que describa el comportamiento requerido de los tipos más abstractos generados por pasos T anteriores.

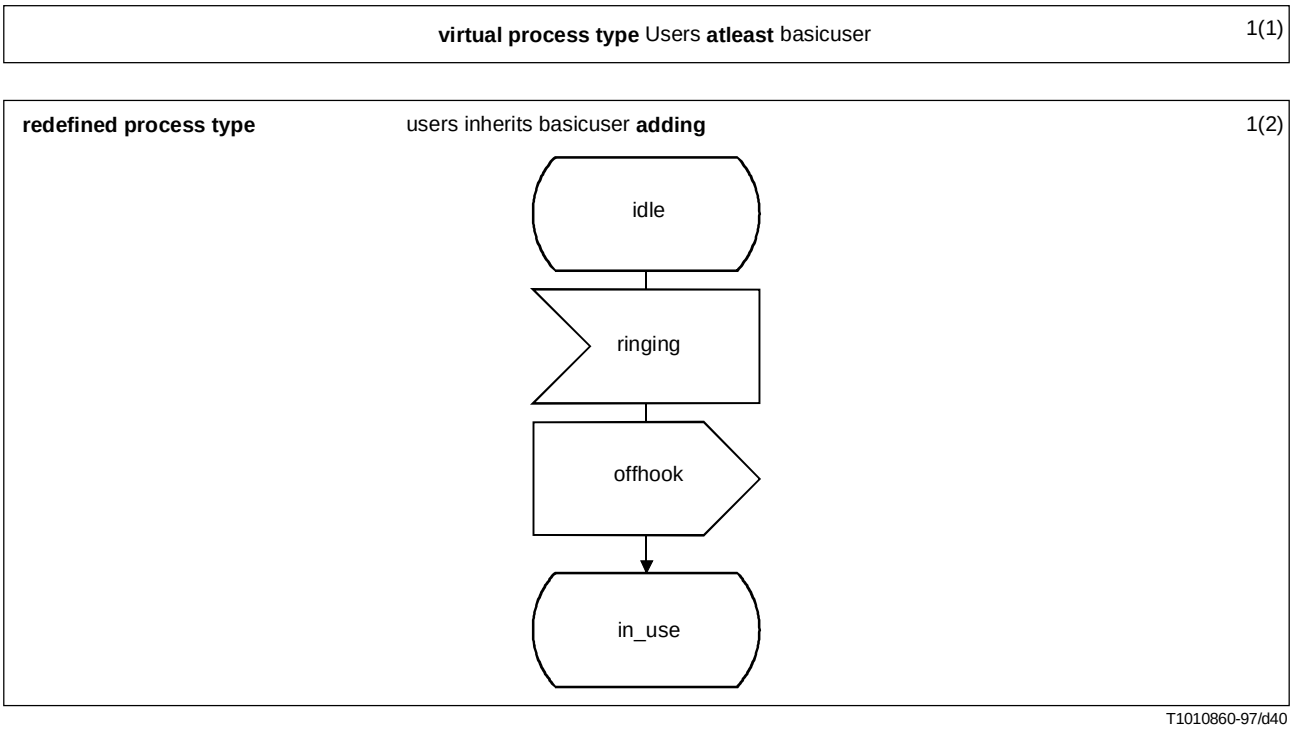
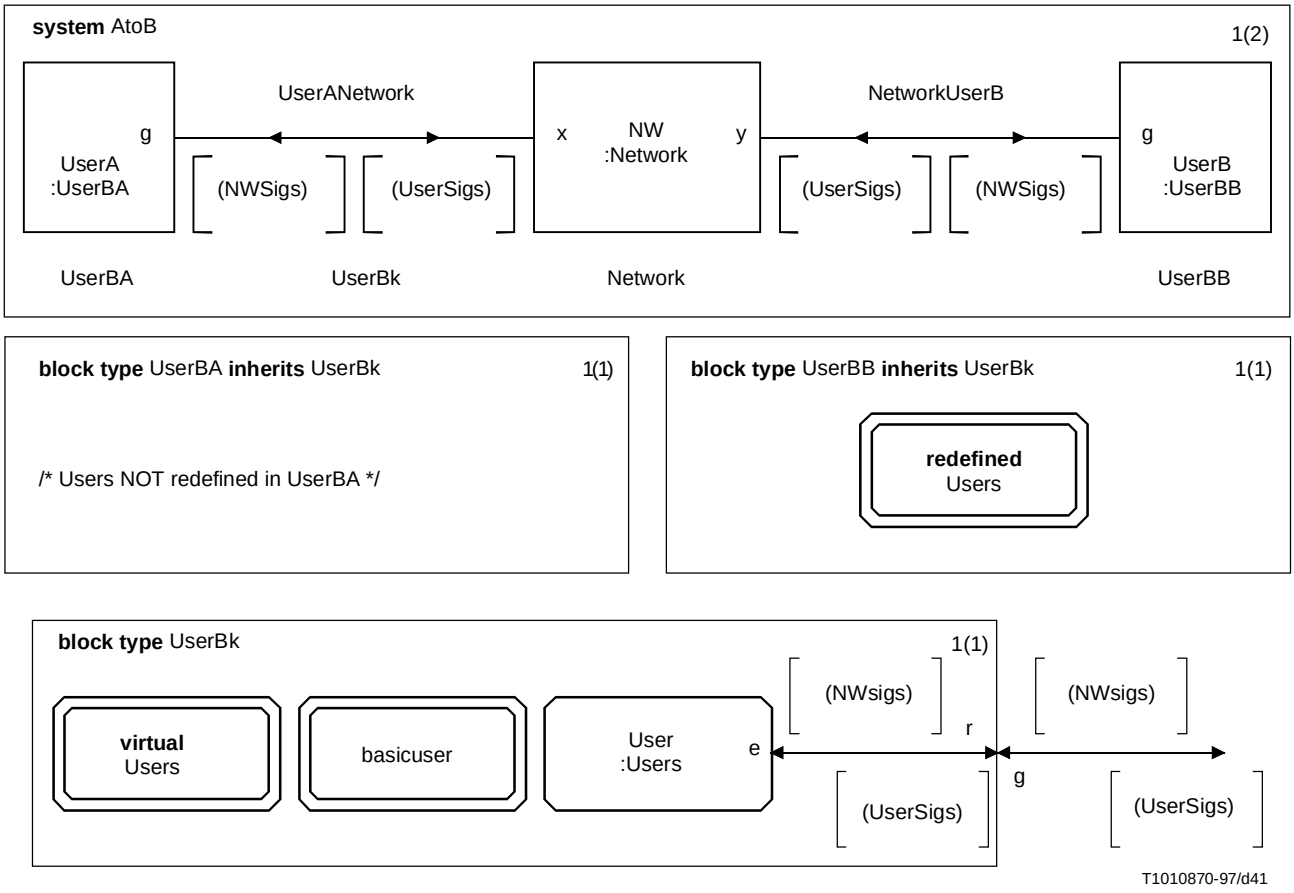


Figura 15-15/Sup. 1 a la Rec. Z.100 – El tipo de proceso virtual Users y una redefinición de Users

Resultado: Un sistema basado en definiciones de tipos.

Ejemplo



NOTA – UserBA es el mismo que UserBk y por lo tanto puede eliminarse. Igualmente ya que basicuser sólo se utiliza en UserBk, la definición (virtual) de Users podría también definir el tipo de constricción.

Figura 15-16/Sup. 1 a la Rec. Z.100 – El sistema AtoB (con un tipo general superbloque UserBk)

15.5 Pasos de localización (pasos L) (L-steps, *localization steps*)

El objetivo de los pasos de localización es garantizar que los tipos se definen en los lugares más adecuados para evitar definiciones innecesarias de nuevos tipos.

15.5.1 Paso L:1 – Tipos no parametrizados

Instrucciones

1) Determinar la unidad de campo de aplicación adecuada para la definición de cada tipo.
2) Mover cada tipo (y las definiciones asociadas) a dicha unidad de campo de aplicación o (si ello es posible) a un lote.
3) Añadir **use** (**uso**) a cada nuevo lote en un símbolo de texto de referencia de lote del diagrama del sistema.

Directrices

Un tipo que no depende de ninguna definición puede moverse a un lote. Debe tenerse especial cuidado al identificar tipos mutuamente dependientes o definiciones que puedan mover un lote junto.

Al mismo tiempo puede revisarse la unidad de campo de aplicación para otras definiciones. Cada definición se sitúa de forma que pueda ser reutilizada en distintos lugares pero asimismo con la menor visibilidad para todos sus usos. Es oportuno mover las definiciones de las señales del sistema a un lote debido a que ello redunda en que el sistema tenga un mayor parecido con un bloque.

Un tipo debe ser local si depende de definiciones o de otros tipos de la misma unidad de campo de aplicación (suponiendo que no es adecuado ni factible mover todas las definiciones relacionadas). Un tipo es local si su visibilidad debe restringirse al contexto local. La necesidad de restringir la visibilidad de un tipo ocurre a menudo por la redefinición de un tipo virtual con el mismo nombre. Un tipo debe ser local si sólo tiene sentido en dicho contexto.

Es útil ubicar un conjunto de definiciones en un lote de forma que dicho conjunto pueda ser reutilizado en distintos sistemas, pero SDL-92 sólo permite que al diagrama del sistema se adjunten referencias a lotes. La utilización de un lote entra pues en conflicto con la restricción del campo de aplicación local de las definiciones. En el paso L:3 se considera con más detalle la localización en lotes.

Regla 46 – El ámbito de aplicación que se desea para un lote debe añadirse como un comentario a la cláusula de referencia del lote.

Resultado: Todas las definiciones tienen un campo de aplicación local adecuado o se definen en un lote con un comentario sobre su campo de aplicación objetivo.

15.5.2 Paso L:2 – Definición de parámetros de contexto

Instrucciones

1) Identificar cuando dos o más definiciones tipo son idénticas excepto en la utilización que hacen de algunos ítems designados (tales como señales o procedimientos).
2) Determinar las constricciones que existen (definición común mínima) sobre los tipos de los ítems designados (por ejemplo, una señal puede tener la constricción de que debe tener dos parámetros enteros) e identificar o definir un tipo que debe utilizarse en una cláusula **atleast**.
3) Definir una nueva definición de tipo con los ítems designados como parámetros de contexto y con cláusulas de constricción **atleast** en los parámetros.
4) Sustituir la utilización de tipos que son prácticamente idénticos por la utilización del nuevo tipo con parámetros de contexto, de forma que los parámetros efectivos sean los ítems designados originalmente.
5) Colocar una referencia a una nueva definición de tipo nuevo en la unidad del campo de aplicación que incluya todas las utilizaciones.

Directrices

Los parámetros de contexto son parámetros de tipos que se sustituyen por parámetros efectivos que son definiciones de ítems. La sustitución es estática y tiene lugar antes de la interpretación del sistema SDL. Los parámetros efectivos que se dan en el contexto del uso del tipo definen un tipo en el que se han sustituido los parámetros.

Los restantes parámetros del SDL (por ejemplo, parámetros de procesos, procedimientos y señales) son sustituidos por los parámetros efectivos que son valores; dicha sustitución es dinámica y tiene lugar cuando se interpreta el sistema. Los valores de dichos parámetros se pasan a las instancias del tipo.

La utilización de parámetros de contexto es, en algunas ocasiones, una alternativa a la construcción de algunas partes de un tipo virtuales de forma que éste pueda ser reutilizado. En el caso de señal, temporizador, variable, sinónimo y parámetros de contexto de género, no hay elección posible: SDL no tiene tipos virtuales para dichos ítems. Para los procedimientos se recomienda utilizar un procedimiento virtual si el procedimiento efectivo es siempre local. Se recomienda utilizar un parámetro de contexto de procedimiento si el procedimiento efectivo es, en ocasiones, de naturaleza más global. Para un proceso es preferible utilizar el parámetro de contexto de proceso de un bloque en caso de que los tipos de procesos para los parámetros efectivos puedan definirse, sin perder utilidad, como externos a los bloques. Sistema y bloque no pueden ser parámetros de contexto.

Cuadro 4/Sup. 1 a la Rec. Z.100 – Uso de parámetros de contexto

	Puede ser parámetro de contexto	**Puede tener parámetro de contexto**	**Puede ser tipo virtual**
sistema, bloque	no	sí	sí
proceso	sí	sí	sí
procedimiento	sí	sí	sí
señal, temporizador	sí	sí	no
género	sí	sí	no
variable, sinónimo	sí	no	no

En ocasiones, el parámetro de contexto no tiene constricciones, omitiéndose entonces la cláusula **atleast**.

Regla 47 – Los tipos con parámetros de contexto sólo deberían utilizarse cuando mejora la comprensión del sistema mediante la reutilización de conceptos.

Resultado: Un tipo menos dependiente del contexto incluido en el sistema y el uso de dicho tipo en sustitución de dos o más definiciones dependientes del contexto.

15.5.3 Paso L:3 – Definición de lote

Instrucciones

1) Identificar los grupos de ítems relacionados que son específicos del sistema y que pueden separarse en un lote adecuado de uso general.
2) Definir un lote donde los ítems identificados se representan mediante tipos.
3) Registrar información que permita localizar al lote durante las búsquedas de tipos adecuados.
4) Registrar en la biblioteca la utilización del lote.
5) Cuando un lote se reutiliza, registrar su utilización en la biblioteca y hacer un registro especial de todos los cambios del lote que lo hagan reutilizable.

Directrices

Los lotes son particularmente útiles cuando hay un conjunto de sistemas SDL que están relacionados, tales como las descripciones de distintos servicios suplementarios en varias normas relacionadas. Se supone que los lotes se almacenan en una biblioteca de reutilizaciones para su ulterior uso.

Cuando se buscan definiciones en la biblioteca de reutilización, la información clasificada se utiliza como base de las claves de búsqueda a fin de establecer la correspondencia con la información de la biblioteca. La información (nombres definidos, palabras clave) puede utilizarse para reconocer en la biblioteca grupos de conceptos relacionados. A fin de disponer de un conjunto de tipos útiles para un dominio de aplicación, es recomendable utilizar un lote en lugar de encerrar dichos tipos en (por ejemplo) un tipo de bloque, especializando el tipo de bloque para cada uso. Un tipo de bloque es más apropiado para una parte funcional de un sistema.

En ocasiones, un lote de la biblioteca es prácticamente lo que se requiere, pero no se ajusta demasiado bien a la información clasificada. En este caso, el lote de la biblioteca puede modificarse para abarcar el nuevo sistema. Antes de modificar un lote deben comprobarse otras utilizaciones del mismo para evitar introducir incompatibilidades.

Aunque el manejo de un lote no está completamente definido en la norma SDL, se confía en que el entorno soporte lo maneje de forma que esté disponible para cualquier lote o especificación del sistema.

Resultado: Lotes de definiciones de tipos para ser reutilizados en la biblioteca y utilizados en el sistema SDL.

16 Referencias

[1] Recomendación UIT-T Z.100 (1993), *Lenguaje de especificación y descripción del CCITT*.

[2] Recomendación UIT-T Z.105 (1995), *Lenguaje de especificación y descripción combinado con la notación de sintaxis abstracta uno*.

[3] Recomendación UIT-T Z.120 (1996), *Gráficos de secuencias de mensajes*.

[4] Recomendación X.208 del CCITT (1988) (equivalente a ISO/CEI 8824: 1990), *Especificación de la notación de sintaxis abstracta uno*.

[5] Recomendación UIT-T X.680 (1994)/enm. 1 (1995) (equivalente a ISO/CEI 8824-1: 1996), *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de la notación básica – Enmienda 1: Reglas de extensibilidad.*

[6] BRECK (R.), HAUGEN (O.): Engineering Real Time Systems, Prentice-Hall, 1993.

[7] OLSEN (A.) *et al:* Systems Engineering Using SDL-92, North Holland, 1994.

[8] Recomendaciones UIT-T X.900 – X.905 (próxima publicación), *Tecnología de la información – Procesamiento distribuido abierto* (ISO/IEC 10746).

[9] Recomendación UIT-T Z.500 (1997), *Marco de los métodos formales en las pruebas de conformidad*.

[10] Recomendaciones UIT-T X.290 – X.292, *Metodología y marco de las pruebas de conformidad de interconxión de sistemos abiertos de las Recomendaciónes sobre los protocolos para aplicaciones del UIT-T* (ISO/CEI 9646).

[11] RUMBAUGH (J.) *el al:* Object-Oriented Modeling and Design, Prentice-Hall, 1991.

[12] JACOBSEN (I.) *et al:* Object-Oriented Software Engineering, Addison-Wesley, 1992.

[13] Recomendación UIT-T Q.1200 (1993), *Estructura de las Recomendaciones de la serie Q sobre la red inteligente.*

[14] Recomendación UIT-T Z.400 (1993), *Estructura y formato de los manuales de calidad para soporte lógico de telecomunicaciones.*

[15] HOGREFE (D.): Validation of SDL Systems, Computer Networks and ISDN Systems, North Holland, 1996.

[16] Recomendación I.130 del CCITT (1988), *Metodo de caracterización de los servicios de telecomunicación soportados por una RDSI y de los capacidades de red de una RDSI*.

[17] Recomendación UIT-T Q.65 (1997), *Metodología funcional y unificada para la caracterización de servicios y capacidades de red.*

[18] BELINA (F.), HOGREFE (D.) y TRIGILA (S.): Modelling OSI in SDL (in Turner: Formal Description Techniques), North-Holland, 1988.

[19] BELINA (F.), HOGREFE (D.) y SARMA (A.): SDL with Applications from Protocol Specification, Prentice Hall, 1991.

[20] Recomendación UIT-T X.200 (1994), *Tecnología de la información – Interconexión de sistemas abiertos – Modelo de referencia básico: El modelo básico.*

[21] Recomendación UIT-T X.210 (1993), *Tecnología de la información – Interconexión de sistemas abiertos – Modelo de referencia básico: Convenios para la definición de servicios en la interconexión de sistemas abiertos.*

[22] Recomendación X.219 del CCITT (1988), *Operaciones a distancia: Modelo, notación y definición del servicio*.

[23] Recomendación X.722 del CCITT (1992), *Tecnología de la información – Interconexión de sistemas abiertos – Estructura de la información de gestión: Directrices para la definición de objetos gestionados*.

[24] REED (R.) (editor): Specification and Programming Environment for Communication Software, North Holland, 1993.

[25] OLSEN (A.) *et al*: Systems Engineering Using SDL-92, North Holland, 1994.

[26] WITASZECK (D.) *et al*: Development Method for SDL-92 Specifications based on OMT, in SDL '95 with MSC in CASE, Proceedings of the Seventh SDL Forum, North Holland, 1995.

[27] GUO (F.) y MACKENZIE (T.W.): Translation of OMT to SDL-92, in SDL '95 with MSC in CASE, Proceedings of the Seventh SDL Forum, North Holland, 1995

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 Z	**Lenguajes de programación**