UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T X.904

(12/97)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

SERIE X: REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

Procesamiento distribuido abierto

Tecnología de la información – Procesamiento distribuido abierto – Modelo de referencia: Semántica arquitectural

Recomendación UIT-T X.904

(Anteriormente Recomendación del CCITT)

RECOMENDACIONES DE LA SERIE X DEL UIT-T

REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

REDES PÚBLICAS DE DATOS	
Servicios y facilidades	X.1–X.19
Interfaces	X.20–X.49
Transmisión, señalización y conmutación	X.50–X.89
Aspectos de redes	X.90–X.149
Mantenimiento	X.150–X.179
Disposiciones administrativas	X.180–X.199
INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Modelo y notación	X.200–X.209
Definiciones de los servicios	X.210–X.219
Especificaciones de los protocolos en modo conexión	X.220–X.229
Especificaciones de los protocolos en modo sin conexión	X.230–X.239
Formularios para declaraciones de conformidad de implementación de protocolo	X.240–X.259
Identificación de protocolos	X.260–X.269
Protocolos de seguridad	X.270–X.279
Objetos gestionados de capa	X.280–X.289
Pruebas de conformidad	X.290–X.299
INTERFUNCIONAMIENTO ENTRE REDES	
Generalidades	X.300–X.349
Sistemas de transmisión de datos por satélite	X.350–X.399
SISTEMAS DE TRATAMIENTO DE MENSAJES	X.400–X.499
DIRECTORIO	X.500–X.599
GESTIÓN DE REDES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS Y ASPECTOS DE SISTEMAS	
Gestión de redes	X.600–X.629
Eficacia	X.630–X.639
Calidad de servicio	X.640–X.649
Denominación, direccionamiento y registro	X.650–X.679
Notación de sintaxis abstracta uno	X.680–X.699
GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Marco y arquitectura de la gestión de sistemas	X.700–X.709
Servicio y protocolo de comunicación de gestión	X.710–X.719
Estructura de la información de gestión	X.720–X.729
Funciones de gestión y funciones de arquitectura de gestión distribuida abierta	X.730–X.799
SEGURIDAD	X.800–X.849
APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Cometimiento, concurrencia y recuperación	X.850–X.859
Procesamiento de transacciones	X.860–X.879
Operaciones a distancia	X.880–X.899
PROCESAMIENTO DISTRIBUIDO ABIERTO	**X.900–X.999**

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

NORMA INTERNACIONAL 10746-4

RECOMENDACIÓN UIT-T X.904

TECNOLOGÍA DE LA INFORMACIÓN – PROCESAMIENTO DISTRIBUIDO ABIERTO – MODELO DE REFERENCIA: SEMÁNTICA ARQUITECTURAL

Resumen

La presente Recomendación | Norma Internacional forma parte integrante del modelo de referencia del procesamiento distribuido abierto (ODP). Contiene una formalización de los conceptos de modelado del ODP definidos en las cláusulas 8 y 9 de la Rec. UIT-T X.902 | ISO/CEI 10746-2. La formalización se consigue interpretando cada concepto en base a las construcciones de las diferentes técnicas de descripción formal normalizada.

Orígenes

El texto de la Recomendación UIT-T X.904 se aprobó el 12 de diciembre de 1997. Su texto se publica también, en forma idéntica, como Norma Internacional ISO/CEI 10746-4.

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

Prefacio

La presente Recomendación | Norma Internacional forma parte integrante del modelo de referencia del procesamiento distribuido abierto (ODP). Contiene una formalización de los conceptos de modelado del ODP definidos en las cláusulas 8 y 9 de la Rec. UIT-T X.902 | ISO/CEI 10746-2. La formalización se consigue interpretando cada concepto en base a las construcciones de las diferentes técnicas de descripción formal normalizada.

La presente Recomendación | Norma Internacional va acompañada por una enmienda y un informe técnico. La enmienda asociada se centra en la formalización del lenguaje del punto de vista computacional contenido en la Rec. UIT-T X.903 | ISO/CEI 10746-3. El informe técnico asociado contiene ejemplos de cómo puede aplicarse la formalización del modelo de referencia ODP para desarrollar especificaciones.

NORMA INTERNACIONAL

RECOMENDACIÓN UIT-T

TECNOLOGÍA DE LA INFORMACIÓN – PROCESAMIENTO DISTRIBUIDO ABIERTO – MODELO DE REFERENCIA: SEMÁNTICA ARQUITECTURAL

0 Introducción

El rápido crecimiento del procesamiento distribuido ha creado la necesidad de un marco de coordinación para la normalización del procesamiento distribuido abierto (ODP, *open distributed processing*). El modelo de referencia (RM, *reference model*) del ODP proporciona ese marco. Crea una arquitectura dentro de la cual se puede integrar el soporte de la distribución, el interfuncionamiento, la interoperabilidad y la portabilidad.

El modelo de referencia básico del procesamiento distribuido abierto (RM-ODP, *basic reference model of open distributed processing*) (véanse las Recomendaciones UIT-T X.901 a X.904 | ISO/CEI 10746), se basa en conceptos precisos utilizados en trabajos actualmente en curso para el desarrollo del procesamiento distribuido y, en la medida de lo posible, en el empleo de técnicas de descripción formal para la especificación de la arquitectura.

El modelo de referencia de procesamiento distribuido abierto está constituido por:

- La Rec. UIT-T X.901 | ISO/CEI 10746-1: **Visión de conjunto**: Contiene una visión de conjunto de las motivaciones del ODP, que da el alcance, la justificación y la explicación de conceptos esenciales y una descripción de la arquitectura ODP. Este texto no es normativo.

- La Rec. UIT-T X.902 | ISO/CEI 10746-2: **Fundamentos**: Contiene la definición de conceptos y marco analítico y la notación para la descripción normalizada de sistemas de procesamiento distribuido (arbitrarios). La exposición se hace solamente a un nivel de detalle suficiente para la aplicación de la Rec. UIT-T X.903 | ISO/CEI 10746-3 y el establecimiento de los requisitos para las nuevas técnicas de especificación. Este texto es normativo.

- La Rec. UIT-T X.903 | ISO/CEI 10746-3: **Arquitectura**: Contiene la especificación de las características que debe tener un procesamiento distribuido para que sea abierto. Éstas son las constricciones a que deben ajustarse las normas ODP. Emplea las técnicas descriptivas de la Rec. UIT-T X.902 | ISO/CEI 10746-2. Este texto es normativo.

- La Rec. UIT-T X.904 | ISO/CEI 10746-4: **Semántica arquitectural**: Contiene una formalización de los conceptos de modelado ODP definidos en las cláusulas 8 y 9 de la Rec. UIT-T X.902 | ISO/CEI 10746-2 y una formalización de los lenguajes de punto de vista de la Rec. UIT-T X.903 | ISO/CEI 10746-3. La formalización se consigue interpretando cada concepto en términos de las construcciones de las diferentes técnicas de descripción formal normalizada. Este texto es normativo.

La presente Recomendación | Norma Internacional tiene por objeto proporcionar una semántica arquitectural para el ODP, lo cual viene a ser en definitiva una interpretación de los conceptos básicos de modelado y de especificación de la Rec. UIT-T X.902 | ISO/CEI 10746-2 y los lenguajes de punto de vista de la Rec. UIT-T X.903 | ISO/CEI 10746-3, utilizando las diversas características de los diferentes lenguajes de especificación formal. Se elabora una semántica arquitectural en cuatro lenguajes de especificación formal diferentes: LOTOS, ESTELLE, SDL y Z. El resultado es una formalización de la arquitectura del ODP. Siguiendo un proceso de desarrollo iterativo y realimentación, se ha mejorado la coherencia de la Rec. UIT-T X.902 | ISO/CEI 10746-2 y la Rec. UIT-T X.903 | ISO/CEI 10746-3.

Una semántica arquitectural aporta las ventajas adicionales de:

- contribuir a una elaboración idónea y uniforme de descripciones formales de sistemas ODP; y

- de permitir la comparación uniforme y coherente de las descripciones formales de la misma norma en lenguajes de especificación formal diferentes.

En vez de proporcionar la correspondencia de todos los conceptos de la Rec. UIT-T X.902 | ISO/CEI 10746-2, la presente Recomendación | Norma Internacional se centra en los más fundamentales. Se facilita indirectamente una semántica para los conceptos arquitecturales de nivel superior mediante sus definiciones en términos de conceptos básicos del ODP.

Se pueden encontrar ejemplos de la utilización de algunos de los lenguajes de especificación formal en este informe en TR 10167 (sobre directrices para la aplicación de ESTELLE, LOTOS y SDL).

En las cláusulas que siguen, los conceptos se numeran de acuerdo con el esquema utilizado en la Rec. UIT-T X.902 | ISO/CEI 10746-2.

1 Alcance

La presente Recomendación | Norma Internacional establece una semántica arquitectural del procesamiento distribuido abierto (ODP). Esto para:

- facilitar la formalización de los conceptos relativos al modelado del ODP;
- contribuir a una elaboración idónea y uniforme de descripciones formales de normas para sistemas distribuidos;
- actuar a modo de puente entre los conceptos relativos al modelado del ODP y los modelos semánticos de los lenguajes de especificación: LOTOS, SDL, ESTELLE y Z;
- proporcionar una base de comparación uniforme y coherente entre descripciones formales de la misma norma en los lenguajes de especificación que se utilizan para elaborar una semántica arquitectural.

Este texto es normativo.

2 Referencias normativas

Las siguientes Recomendaciones y Normas Internacionales contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación | Norma Internacional. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y Normas son objeto de revisiones, por lo que se preconiza que los participantes en acuerdos basados en la presente Recomendación | Norma Internacional investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y las Normas citadas a continuación. Los miembros de la CEI y de la ISO mantienen registros de las Normas Internacionales actualmente vigentes. La Oficina de Normalización de las Telecomunicaciones de la UIT mantiene una lista de las Recomendaciones UIT-T actualmente vigentes.

- ISO/CEI 8807:1989, *Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour.*
- Recomendación UIT-T Z.100 (1993), *Lenguaje de especificación y descripción del CCITT.*
- ISO/CEI TR 10167:1991, *Information technology – Open Systems Interconnection – Guidelines for the application of Estelle, LOTOS and SDL.*
- ISO/CEI 13568[1)], *Information technology – Programming Languages their Environments and System Software Interfaces, Z Specification language.*
- The Z Notation, *A Reference Manual, J.M. Spivey, International Series in Computer Science, Second Edition, Prentice-Hall International, 1992.*
- ISO/CEI 9074:1997, *Information technology – Open Systems Interconnection – Estelle: A formal description technique based on an extended state transition model.*

1) Actualmente en estado de proyecto.

3 Definiciones

3.1 Definiciones de la Norma ISO/CEI 8807

Esta Recomendación | Norma Internacional emplea los siguientes términos definidos en ISO/CEI 8807:

actualización de parámetros, composición paralela, conformidad, definición de proceso, definición de tipo, definición de tipo parametrizado, deshabilitación, ecuación, elección, enriquecimiento, entrelazado, evento denotación de acción, evento interno observable, expresión de comportamiento, extensión, género, guarda, habilitación, instanciación, lista de parámetros de valores, lista de parámetros formales, lista de puertas formales, ocultación de puerta, operación, predicado de selección, puerta, reducción, sincronización.

3.2 Definiciones de la Recomendación UIT-T Z.100

Esta Recomendación | Norma Internacional emplea los siguientes términos definidos en la Rec. UIT-T Z-100:

activar, ahora, canal, cláusula atleast, condición habilitante, conjunto, crear, entrada, estado siguiente, exportación, finalizar, importación, llamada, nodelay, parada, parámetro de contenido, procedimiento, procedimiento distante, procedimiento exportado, proporcionar, puerto, redefinir, reinicializar, retorno, ruta de señales, salida, sentencia de acción, señal, señal continua, tarea, temporizador, tiempo, tipo de bloque, tipo de proceso, tipo de servicio, tipo de sistema, tipo virtual, transición, variable exportada, variable importada, variable revelada, variable visionada, visión.

3.3 Definiciones de la norma de base Z

Esta Recomendación | Norma Internacional emplea los siguientes términos definidos en la norma de base Z:

cálculo de esquema, composición de esquema, conjunción, descripción axiomática, esquema (operación, estado, alineación de trama), invariante, postcondición, precondición, refinamiento de datos, refinamiento de operación, revocación.

3.4 Definiciones de ISO/CEI 9074

Esta Recomendación | Norma Internacional emplea los siguientes términos definidos en ISO/CEI 9074:

actividad, actividad de sistema, anexar, bloque de transición, canal, cláusula DELAY, cláusula FROM, cláusula PROVIDED, cláusula TO, cláusula WHEN, conectar, declaración de asignación, definición de canal, definición de cuerpo de módulo, definición de encabezamiento de módulo, desconectar, estado de control, función, iniciar, instancia módulo, instancia progenitora, instanciación, interacción, liberar, procedimiento, procedimiento primitivo, proceso de sistema, punto de interacción, punto de interacción externa, transición, rol, salida, separar, variable exportada.

4 Interpretación de los conceptos relativos al modelado

4.1 Semántica arquitectural en LOTOS

LOTOS es un lenguaje de especificación formal (FSL, *formal specification language*) normalizado (ISO/CEI 8807). En la norma se dispone de material docente.

Esta cláusula explica cómo se pueden expresar los conceptos fundamentales relativos al modelado en LOTOS (véase ISO/CEI 8807). Hay que señalar que en LOTOS existen dos maneras principales de modelar los conceptos contenidos en la Rec. UIT-T X.902 | ISO/CEI 10746-2. Se basan en la parte álgebra de proceso del lenguaje y en la parte tipificación de datos ACT ONE del lenguaje. Puesto que la formalización ACT ONE de los conceptos es aplicable también al SDL-92, la formalización ACT ONE se indica en una cláusula aparte. Véase 4.2.

Para evitar confusión entre la terminología de ODP y la de LOTOS, la subcláusula que sigue utiliza letras *cursivas* para indicar términos específicos de LOTOS.

4.1.1 Conceptos básicos de modelado

4.1.1.1 Objeto

Una *instanciación* de una *definición de proceso* de LOTOS que puede ser referenciada de manera exclusiva.

4.1.1.2 Entorno (de un objeto)

La parte de un modelo que no forma parte del objeto. En LOTOS, el entorno de un objeto dentro de una especificación en un momento determinado viene dado por el entorno de la especificación y por las otras *expresiones de comportamiento* que se componen con ese objeto en la especificación en ese momento.

NOTA – El entorno de una especificación está vacío si la especificación no está parametrizada.

4.1.1.3 Acción

Las acciones de LOTOS se modelan como *eventos internos* o como *eventos observables*. Todos los eventos de LOTOS son atómicos. Una acción interna puede venir indicada explícitamente por el símbolo de *evento interno*, **i**, o por una ocurrencia de evento cuya *puerta* asociada está *oculta* con respecto al entorno.

Una interacción se representa en LOTOS mediante una *sincronización* entre dos o más *expresiones de comportamiento* asociadas con objetos en un punto de interacción común (*puerta*). Las interacciones pueden ser de las clases siguientes:

- *sincronización* pura en una *puerta* común sin oferta: no se produce ningún paso de valores entre objetos;
- ! y ! para *sincronización* pura: no se intercambian valores entre los objetos;
- ! y ? para el paso de valores siempre que el evento ? contenga el evento !: otra manera de considerar esto es que el evento ! selecciona un valor de entre diversos valores para el evento ? ;
- ? y ? para establecimiento de valor: el efecto aquí es un acuerdo sobre un valor de la intersección del conjunto de valores. Si la intersección de los valores es el conjunto vacío, no se produce *sincronización*, y, por consiguiente, no hay interacción.

Si se necesita una granularidad no atómica de las acciones, se puede emplear el refinamiento de los eventos. Esto habilitará a continuación el modelado de acciones no instantáneas y superpuestas. Téngase en cuenta que el refinamiento de eventos no es un problema trivial, sobre todo cuando se ha de mantener la compatibilidad en comportamiento.

No existe en LOTOS un constructivo con el que expresar relaciones de causa a efecto, si bien en algunas ocasiones se podría representar esto de manera informal.

4.1.1.4 Interfaz

Una abstracción del comportamiento de un objeto que consta de un subconjunto de acciones observables de ese objeto. Puesto que todas las acciones observables de un objeto en LOTOS requieren *puertas* con las que sincronizar con el entorno, el subconjunto de acciones observables se efectúa normalmente dividiendo las *puertas* dadas en la *definición de proceso* asociada con el objeto. Para obtener una interfaz puede efectuarse la *ocultación* de las *puertas* no requeridas por la interfaz objeto de consideración. De manera alternativa, se puede emplear la *sincronización* sólo con un subconjunto de las *puertas* asociadas con un objeto. En este caso, las acciones que se realicen en las *puertas* de la *definición de proceso* ausentes del conjunto con el que se sincroniza, pueden considerarse como acciones internas del objeto por lo que se refiere al entorno sincronizador con las *puertas* que constituyen la interfaz.

Obsérvese que esta definición exige que las interfaces de un objeto empleen nombres de *puerta* diferentes, es decir, no es posible distinguir entre interfaces que utilizan la misma *puerta*.

4.1.1.5 Actividad

Una actividad es un gráfico acíclico de acciones, dirigido y de una sola cabeza, en el que cada nodo del gráfico representa un estado del sistema y cada arco representa una acción. Para que ocurra una acción debe satisfacer las precondiciones del estado del sistema.

4.1.1.6 Comportamiento (de un objeto)

El comportamiento de un objeto viene definido por la *expresión de comportamiento* de LOTOS asociada con la *definición de proceso* que constituye la plantilla del objeto. Una *expresión de comportamiento* puede constar de una secuencia de ofertas de eventos visibles externamente y de *eventos internos.* El comportamiento real de un objeto, tal como podría ser registrado en un rastro, depende de la *expresión de comportamiento* asociada con el objeto y de cómo se configura éste con el entorno. El comportamiento real que muestra el objeto depende de la *expresión de comportamiento* del objeto y de cómo se sincroniza con su entorno. Un objeto puede mostrar un comportamiento no determinístico.

4.1.1.7 Estado (de un objeto)

La condición de un objeto que determina el conjunto de todas las secuencias de acciones en las que el objeto puede participar. Esta condición viene regida por la *expresión de comportamiento* definida en la plantilla del objeto a partir de la cual se creó el objeto y, posiblemente, por las vinculaciones vigentes de cualesquiera variables locales existentes.

4.1.1.8 Comunicación

El transporte de la información (vía paso de valores) entre dos o más objetos que interactúan. No es posible escribir directamente las relaciones de causa a efecto. Hay que señalar además que la propia *sincronización* se puede interpretar como comunicación.

4.1.1.9 Ubicación en el espacio

LOTOS se abstrae de la noción de ubicación en el espacio. Sólo es posible igualar espacio con la estructura del modelo de especificación. La ubicación de un evento – la ubicación estructural con respecto al modelo de especificación – viene dada por una *puerta* para interacciones en LOTOS. LOTOS se aleja de la noción de ubicación en el espacio en la que un *evento interno* puede ocurrir. Este alejamiento se efectúa implícitamente utilizando el constructivo *ocultación ... en* de LOTOS que hace que las *puertas* utilizadas internamente dentro de un proceso resulten invisibles al entorno del proceso, o explícitamente utilizando el símbolo de *evento interno,* **i**.

Se puede utilizar la misma ubicación en el espacio para más de un punto de interacción. Esto es posible en LOTOS disponiendo de una sola *puerta* con diferentes *denotaciones de acciones.*

La ubicación de un objeto viene dada por la unión de las localizaciones de las *puertas* asociadas por ese objeto, es decir, la unión de todas las ubicaciones de las acciones en las que ese objeto puede tomar parte.

4.1.1.10 Ubicación en el tiempo

LOTOS se abstrae del concepto de tiempo considerando sólo un orden temporal, de modo que no haya una *ubicación absoluta en tiempo métrico relativo.* La ubicación en el tiempo sería posible, sin embargo, si se utilizara una forma ampliada de LOTOS con aspectos de tiempo incorporados.

4.1.1.11 Punto de interacción

Una *puerta* con una lista vacía posiblemente de valores asociados.

NOTA – En una especificación, los cambios de ubicación se pueden reflejar mediante cambios en los valores asociados.

4.1.2 Conceptos de especificación

4.1.2.1 Composición

– **de objetos:** Un objeto compuesto es un objeto descrito mediante la aplicación de uno o más operadores de combinación de LOTOS. Dichos operadores son como sigue:

  - *operador de entrelazado* (|||);
  - *operadores de composición paralela* (|| y |[*lista de puerta*]|);
  - *operador habilitante* (>>);
  - *operador deshabilitante* ([>);
  - *operador de elección* ([]).

– **de comportamientos:** La composición de las *expresiones de comportamiento* asociadas con los objetos componentes en la creación de un objeto compuesto mediante composición. Los operadores de la composición de comportamientos son los mismos que los de la composición de objetos.

4.1.2.2 Objeto compuesto

Un objeto descrito utilizando uno o más *operadores de entrelazado, composición paralela, deshabilitantes, habilitantes* y *de elección* de LOTOS.

4.1.2.3 Descomposición

- **de objetos:** La expresión de un objeto determinado como un *objeto compuesto*. No obstante, puede haber más de una manera de descomponer un objeto compuesto.

- **de comportamientos:** La expresión de un comportamiento determinado como un comportamiento compuesto. Puede haber más de una manera de descomponer un comportamiento compuesto.

NOTA – También podría considerarse que la noción de descomposición de comportamientos la aportan inherentemente las *operaciones* ACT ONE y las ecuaciones asociadas con un *género*. Es decir, estas *operaciones* y *ecuaciones* proporcionan todas las combinaciones posibles de los comportamientos. Así por ejemplo, una composición secuencial podría generarse mediante *operaciones* aplicadas secuencialmente. Cada aplicación de una *operación* de la secuencia debe satisfacer las ecuaciones necesarias para su ocurrencia. Se puede discutir, de todos modos, si esto es una composición de comportamientos porque las *operaciones* y ecuaciones ya existían y definían todos los comportamientos posibles.

4.1.2.4 Compatibilidad en comportamiento

En LOTOS se han elaborado teorías específicas para verificar la compatibilidad de los comportamientos. No hay características sintácticas específicas del lenguaje LOTOS con las que construir y garantizar la compatibilidad en comportamiento en general. La norma LOTOS, no obstante, desarrolla la noción de *conformidad* que sirve de base para la consideración de la compatibilidad en comportamiento.

Para determinar si los comportamientos de dos objetos son compatibles o no, se ha de introducir la noción de *conformidad*. La *conformidad* se refiere a la evaluación de la funcionalidad de una implementación frente a su especificación, en donde el término implementación debe considerarse como una descripción menos abstracta de una especificación.

Si **P** y **Q** son dos procesos de LOTOS, la declaración **Q** es conforme a **P** (escrito de la siguiente manera: **Q conf P**) equivale a decir que **Q** es una implementación válida de **P**. Esto significa que si **P** puede efectuar un rastro σ y a continuación comportarse como un proceso **P'**, y si **Q** puede también efectuar un rastro σ y a continuación comportarse como **Q'**, se deben satisfacer las siguientes condiciones a propósito de **P'** y **Q'**: cuando quiera que **Q'** pueda negarse a realizar cualquier evento de un conjunto A dado de acciones observables, **P'** debe también tener la posibilidad de negarse a realizar cualquier evento de A.

Así pues sólo, si **Q conf P** se halla en cualquier entorno cuyos rastros se limitan a los de **P**, no puede bloquearse **Q** cuando **P** no pueda bloquearse. Otra manera de formular esto sería diciendo que **Q** tiene los bloqueos de **P** en un entorno cuyos rastros se limitan a los de **P**.

Un objeto se puede hacer compatible en comportamiento con un segundo objeto tras alguna modificación de su comportamiento, lo que podría incluir la *extensión* del comportamiento del objeto (añadiendo comportamiento adicional) o una *reducción* del comportamiento del objeto (restringiendo el comportamiento del objeto). Este proceso de modificación de un objeto se conoce como refinamiento (véase 4.1.2.5).

4.1.2.5 Refinamiento

Refinamiento es un proceso por el cual un objeto puede ser modificado, bien extendiendo o bien reduciendo su comportamiento, o mediante una combinación de ambas cosas, de tal modo que se conforme a otro objeto. Suponiendo que **P** y **Q** son procesos de LOTOS, una *extensión* de **P** por **Q** (escrito de la siguiente manera: **Q extends P**) significa que **Q** no tiene menos rastros que **P**, pero, en un entorno cuyos rastros se limitan a los de **P**, **Q** tiene los mismos bloqueos. Una *reducción* de **P** por **Q** (escrito de la siguiente manera: **Q reduces P**) significa que **Q**, no tiene más rastros que **P** pero, en un entorno cuyos rastros se limitan a los de **Q**, **P** tiene los mismos bloqueos.

4.1.2.6 Rastro

El rastro del comportamiento de un objeto desde su estado instanciado inicial hasta algún otro estado es un registro de la secuencia finita de interacciones (*eventos observables*) entre el objeto y su entorno.

4.1.2.7 Tipo de una <X>

Los tipos que se pueden representar explícitamente en LOTOS para objetos e interfaces son tipos de plantilla. No hay un constructivo explícito en LOTOS que permita el modelado de tipos de acción como tales. Una especificación LOTOS consta de una *expresión de comportamiento* que se compone de *denotaciones de acción* (plantillas de acción). Estas plantillas de acción se producen como parte del comportamiento del sistema, en cuyo caso su ocurrencia puede contemplarse aproximadamente como la instanciación de la plantilla de acción, o pueden no producirse, en cuyo caso la plantilla de acción permanece no instanciada. Las propias plantillas de acción se pueden indicar mediante el símbolo de *evento interno*, **i**, u ofertas de eventos en *puertas* que pueden tener o no una secuencia finita de declaraciones de valor y/o variable.

LOTOS no ofrece facilidades con las que caracterizar acciones directamente, sin embargo, una forma limitada de caracterización de acciones se basa en la característica de *sincronización* de LOTOS. Esto es, podría considerarse que las *denotaciones de acción* (plantillas de acción) sincronizadas deben satisfacer el mismo tipo de acción para que se realice la acción. No obstante, LOTOS no clasifica los rasgos caracterizados de estas *denotaciones de acción* arbitrarias y, por tanto, no es posible asignar un tipo formal a ninguna acción dada. Podría ocurrir que, informalmente, se diera a las ofertas de eventos implicadas en una interacción, un rol de causa y efecto, pero éste no es el caso por lo general. Véase 4.1.1.8.

El símbolo *evento interno* se puede utilizar para representar un tipo de acción, en donde la característica común de esta colección de acciones es que no tienen características.

Obsérvese que, al establecer que el único predicado posible en LOTOS para objetos (e interfaces) es que satisfacen su tipo de plantilla, los conceptos de tipo y tipo plantilla dados en la Rec. UIT-T X.902 | ISO/CEI 10746-2 se reducen a la misma técnica de modelado en LOTOS. Así pues, no hay distinción en LOTOS entre un tipo en su sentido amplio de caracterización y un tipo de plantilla en su sentido más restrictivo de instanciación de plantilla.

4.1.2.8 Clase de una <X>

La noción de clase depende del predicado de tipo caracterizador que satisfacen los miembros de la clase. Objetos, interfaces y acciones pueden satisfacer muchos predicados de tipo caracterizadores arbitrarios. Un tipo que se puede escribir es un tipo de plantilla. Cuando tal es el caso, la clase de objetos, interfaces y acciones asociada con ese tipo es la clase de plantilla.

NOTA – Obsérvese que, al establecer que la única clasificación posible en LOTOS para objetos, interfaces y acciones es que satisfacen su tipo de plantilla, los conceptos de clase y clase de plantilla dados en la Rec. UIT-T X.902 | ISO/CEI 10746-2 se reducen a la misma técnica de modelado en LOTOS. Así pues, no hay distinción en LOTOS entre una clase en su sentido general de clasificación y una clase de plantilla en su sentido más restrictivo como el conjunto de instancias de un tipo de plantilla dado.

4.1.2.9 Subtipo/supertipo

Puesto que los tipos que pueden representarse en LOTOS para objetos como interfaces y, en menor medida, acciones, son tipos de plantilla, una relación de subtipos en LOTOS es una relación que puede existir entre tipos de plantilla. En LOTOS, sin embargo, no existe una característica directa para representar relaciones de subtipificación directamente. Si se necesita la subtipificación, se pueden utilizar *extensiones* para dar una relación de subtipos basada en la sustituibilidad; no obstante, ésta no es una característica proporcionada explícitamente en LOTOS.

4.1.2.10 Subclase/superclase

Puesto que los tipos que pueden escribirse en LOTOS para objetos, interfaces y, en menor medida, acciones, son tipos de plantilla, existe una relación de subclase entre dos clases cuando existe una relación de subtipificación entre sus correspondientes tipos de plantilla.

4.1.2.11 Plantilla <X>

– **Plantilla de Objeto:** Una *definición de proceso* con algún modo de identificación exclusiva mismo una vez instanciado. Si no se da una lista de parámetros de valor, no será posible la identificación de objeto para más de un objeto instanciado a partir de la plantilla de objeto.

  Con respecto a la combinación de plantillas de objeto en LOTOS, no existen operadores de combinación excepto para una forma limitada de fijación del alcance en la que se utiliza el término "*donde*" de LOTOS.

– **Plantilla de Interfaz:** Cualquier comportamiento obtenido a partir de una *definición de proceso* considerando sólo las interacciones en un subconjunto de las *puertas* asociadas con la *definición de proceso*. Esta subconjuntación de las *puertas* se consigue *ocultando* las *puertas* que no se requieren para las interacciones que se consideran.

  Con respecto a la combinación de plantillas de interfaz en LOTOS, no existen operadores de combinación excepto para una forma limitada de fijación del alcance en la que se utiliza el término "*donde*" de LOTOS.

– **Plantilla de Acción:** Una *denotación de acción* que puede ser un símbolo de *evento interno*, un identificador de puerta o un identificador de puerta seguido por una secuencia finita de declaraciones de valor y/o variable.

NOTA – La definición que aquí se hace de *denotación de acción* es artificial ya que LOTOS no soporta realmente el concepto de plantilla de acción. En LOTOS, los comportamientos posibles se especifican dando *denotaciones de acción* combinadas de alguna forma. Relacionar una plantilla con una *denotación de acción* es lo máximo que se puede conseguir con LOTOS. Sin embargo, el texto de Rec. UIT-T X.902 | ISO/CEI 10746-2 exige una plantilla de acción para agrupar las características de las acciones. Esto no forma parte de LOTOS ya que existen ofertas de evento (*denotaciones de acción*) de forma aislada y no es posible reunirlas y aplicar una plantilla para caracterizarlas.

La composición de plantillas de acción se puede equiparar aproximadamente a la *sincronización* con el paso de valores o la generación de valores. En este caso, dos (o más) plantillas de acción acuerdan una plantilla de acción común para que se produzca la *sincronización*, es decir, una plantilla de acción con las características comunes a todas las plantillas de acción que participan en la *sincronización* (composición).

4.1.2.12 Firma de interfaz

Una firma de interfaz, en tanto que conjunto de plantilla de acción asociadas con las interacciones de una interfaz, se representa en LOTOS mediante un conjunto de *denotaciones de acción*. Los miembros de este conjunto son las *denotaciones de acción* que requieren *sincronización* con el entorno para ocurrir.

4.1.2.13 Instanciación (de una plantilla < *X* >)

– **de una Plantilla de Objeto:** El resultado de un proceso que utiliza una plantilla de objeto para crear un nuevo objeto en su estado inicial. Este proceso implica la *actualización* de la *lista de puertas formales* y *parámetros formales* de una *definición de proceso* reetiquetándolos uno por uno a partir de una lista de puertas especificadas y una lista de parámetros efectivos. Las características del objeto creado serán regidas por la plantilla de objeto y por cualesquiera parámetros utilizados para instanciarlo.

– **de una Plantilla de Interfaz:** El resultado de un proceso por el cual se crea una interfaz a partir de una plantilla de interfaz. La interfaz creada puede ser utilizada en adelante por el objeto con el que está asociada para interactuar con el entorno. Las características de la interfaz creada las determinará la plantilla de interfaz y cualesquiera parámetros utilizados para instanciarla.

– **de una Plantilla de Acción:** Esto se da como ocurrencia de acción en LOTOS. Puede implicar la reescritura de expresiones ACT ONE.

4.1.2.14 Rol (papel)

Un nombre asociado con una *definición de proceso* en la plantilla para un objeto compuesto (es decir, composición LOTOS de *expresiones de comportamiento*). Los roles, en tanto que tales, no se pueden utilizar como parámetros. Sin embargo, es posible asignar valores de datos a cada rol en una composición para distinguirlos o direccionarlos específicamente.

4.1.2.15 Creación (de una < *X* >)

– **de un objeto:** La instanciación de una plantilla de objeto como parte del comportamiento de un objeto existente.

– **de una interfaz:** Puesto que los objetos y las interfaces se modelan de la misma manera en LOTOS (vía *definiciones de proceso*), la creación de objetos corresponde a la creación de interfaces. Así pues, la definición de creación de interfaces se da mediante la creación de objetos.

4.1.2.16 Introducción (de un objeto)

La *instanciación* del comportamiento asociado con la especificación LOTOS.

4.1.2.17 Supresión (de una < *X* >)

- **de un objeto:** La terminación de la *instanciación* de un proceso. Se puede efectuar utilizando el *operador inhabilitante* de LOTOS, la *expresión de comportamiento* (*parada*) de inacción de LOTOS que no permite el paso del control o la *expresión de comportamiento* (*salida*) de terminación satisfactoria cuando el paso del control sea posible vía el *operador habilitante*.

- **de una interfaz:** El proceso mediante el cual el comportamiento futuro de un objeto se limita al comportamiento que no necesita la participación de la interfaz dada que ha de ser suprimida.

4.1.2.18 Interfaz de un tipo

- **de una Plantilla de Objeto:** Una instancia de una plantilla de objeto determinado se representa en LOTOS mediante una instanciación de esa plantilla de objeto o una sustitución aceptable de una instanciación de esa plantilla de objeto. El sustituto aceptable debe captar, aquí, las características que identifican a ese tipo. Por ello, un sustituto aceptable podría ser otra plantilla que fuese compatible en comportamiento con la primera. Esto podría conseguirse a través de la *extensión* definida en 4.1.2.4. Utilizando esta relación se garantiza que todas las características del tipo que se considera quedan incluidas. Podría ocurrir, no obstante, que se encontrara una forma más débil de relación de satisfacción de tipo que no requiriese la inclusión de todas las características asociadas con una plantilla dada, sino algún subconjunto de todas esas características.

- **de una Plantilla de Interfaz:** Puesto que una plantilla de interfaz se representa del mismo modo que una plantilla de objeto (vía *definición de proceso* en LOTOS), el texto anterior es aplicable igualmente (es decir, sustitución de todas las ocurrencias de objeto por interfaz) para una instancia de plantilla de interfaz.

- **de una Plantilla de Acción:** Una instancia de una plantilla de acción (*denotación de acción*) se representa en LOTOS mediante otra *denotación de acción* que presenta una oferta de evento equivalente.

4.1.2.19 Tipo de plantilla (de una < *X* >)

Un predicado que expresa que una < *X* > es una instancia de una plantilla dada, en donde < *X* > puede ser un objeto, una interfaz o una acción.

4.1.2.20 Clase de plantilla (de una < *X* >)

La clase de plantilla de una < *X* > es el conjunto de todas las < *X* > que son instancias de esa plantilla < *X* >, en donde < *X* > puede ser un objeto, una interfaz o una acción.

NOTA – La noción de clase de plantilla de una acción está limitada en su aplicación a LOTOS, ya que LOTOS no prevé de manera explícita plantillas de acción, instanciaciones de plantillas de acción o tipos de plantillas de acción.

4.1.2.21 Clase derivada/clase de base

Si la plantilla de una clase es una modificación incremental de la plantilla de una segunda clase, la primera clase es una clase derivada de la segunda y la segunda es una clase de base de la primera.

Las plantillas LOTOS se pueden modificar de manera incremental *extendiendo, enriqueciendo* y modificando los *tipos de datos* o modificando el comportamiento. Surgen problemas, no obstante, con las modificaciones de comportamiento; en concreto:

- caso subtipificación: se puede introducir el no determinismo en el sistema cuando las iniciales de la plantilla heredada y las de su modificación sean las mismas, con lo que no se puede garantizar la subtipificación;

- necesidad de una redirección de autorreferencia: cualquier referencia a una plantilla derivada procedente de una plantilla progenitora deberá redireccionarse a la plantilla derivada, lo que no siempre es posible.

En la norma LOTOS no hay una solución satisfactoria a estos problemas.

4.1.2.22 Invariante

En LOTOS, los únicos invariantes que se pueden representar son *definiciones de proceso*. No hay manera de anexar una invariante a una *definición de proceso* que no sea la propia *definición de proceso*.

4.1.2.23 Precondición

Una precondición es un predicado que una especificación requiere que sea verdadero para que ocurra la acción y que puede expresarse directamente en LOTOS utilizando una o más:

- secuenciación de acciones,
- *guardas* y *predicados de selección*.

4.1.2.24 Postcondición

En LOTOS, la ocurrencia de una acción es independiente del estado del sistema después de ocurrencia de la misma. Por ello, LOTOS no proporciona los medios con los que expresar postcondiciones directamente.

4.2 Semántica arquitectural en ACT ONE

Esta subcláusula proporciona una formalización de los conceptos básicos de modelado y especificación contenidos en la Rec. UIT-T X.902 | ISO/CEI 10746-2. Aunque ACT ONE no es en sí misma un FSL normalizado, se utiliza en la normalización de los FSL LOTOS y SDL-92 y representa una manera alternativa de formalizar los conceptos antes mencionados. Por ello, la formalización ACT ONE de los conceptos contenidos en la Rec. UIT-T X.902 | ISO/CEI 10746-2 se presenta en su propia cláusula aparte.

4.2.1 Conceptos básicos de modelado

4.2.1.1 Objeto

Una instancia de un *género* a la que se puede hacer referencia de manera exclusiva. Obsérvese que los objetos modelados de esta manera deben ser especificados de modo que tengan alguna forma de existencia. Esto puede hacerse mediante un estilo de especificación del álgebra de proceso. Ejemplos del mismo son la recursión en las *definiciones de proceso*, en donde el objeto es un elemento de la *lista de parámetros de valor* asociada con esa *definición de proceso*. De manera alternativa, se pueden utilizar las cláusulas **let...in** para modelar objetos con una forma de existencia. En ambos estilos se requieren *guardas* y/o *predicados de selección* para asegurar que las instanciaciones de las definiciones de *género* son únicas.

4.2.1.2 Entorno de un objeto

En ACT ONE no está previsto el entorno de un objeto. Esta noción sólo puede considerarse a través del álgebra de proceso y las expresiones de ACT ONE que ahí existen. En efecto, puede considerarse que el entorno de un objeto es toda el álgebra de proceso distinta de las expresiones de ACT ONE que representan el objeto en cuestión y las *operaciones* en ese objeto. Esto es, el entorno se utiliza para hacer que se produzcan *operaciones* en un objeto. La noción de entorno referida no exige que las *operaciones* en un objeto sean invocadas por otros objetos. Esto tiene consecuencias en nociones tales como la de interacción, a saber, que la interacción no tiene lugar aquí entre objetos sino entre un objeto y un organismo externo, en este caso, el álgebra de proceso.

4.2.1.3 Acción

La ocurrencia de una *operación*. Obsérvese que, por lo general, no hay distinción inherente entre una interacción y una acción interna simplemente desde la perspectiva ACT ONE. Esto es, las posibles acciones se modelan mediante *operaciones* en la firma de un *género* ACT ONE y pueden ocurrir o no, dependiendo de la ocurrencia de las expresiones ACT ONE existentes en el álgebra de proceso. Así pues, las acciones internas no están previstas de manera explícita en ACT ONE. Podría ocurrir, no obstante, que se pudiera modelar una forma de acción interna mediante *géneros* definidos localmente en el álgebra de proceso. De manera alternativa, todas las *operaciones* declaradas en el álgebra de proceso pueden considerarse como interacciones. Las operaciones utilizadas para satisfacer estas *operaciones*, es decir, las *ecuaciones* asociadas con las *operaciones* consideradas, pueden contemplarse como acciones internas. Por ejemplo, si hay procesos que invocan una *operación pop2* que elimina dos elementos de una cola y ésta utiliza la *operación pop* dos veces en sus *ecuaciones* asociadas, *pop2* puede considerarse como una interacción mientras que *pop* puede considerarse como una acción interna. El problema con este tratamiento de una acción interna es, no obstante, que no existe la noción de transiciones espontáneas como tales, tal como, por ejemplo, con el símbolo de *evento interno* **i** en el álgebra de proceso.

Obsérvese que esta forma de interacción no requiere que dos o más objetos interactúen en el sentido del álgebra de proceso, es decir, mediante la *sincronización* en una *puerta* común. La interacción se puede interpretar aquí más bien como algo causado indirectamente por el entorno pero no necesariamente por un objeto, es decir, no por otra instancia de un *género* que modela un objeto. Así pues, podría suceder que una ocurrencia de oferta de evento que no implica expresiones ACT ONE, provocara una interacción, por ejemplo, por la ocurrencia de una oferta de evento resultante de la *instanciación* de una *definición de proceso* cuya *lista de parámetros de valor* contuviera una *operación* (interacción) en un objeto (u objetos).

4.2.1.4 Interfaz

Las *operaciones* y *ecuaciones* asociadas con un objeto.

4.2.1.5 Actividad

Una secuencia de aplicaciones de *operación* en un *género* determinado. Estas *operaciones* deben satisfacer las *ecuaciones* asociadas con el *género*. Cada *operación* de la secuencia de *operaciones* que tiene lugar, es decir, cada operación de la actividad, debe tener precondiciones que satisfagan las postcondiciones de la ocurrencia de *operación* previa. Las precondiciones y las postcondiciones a las *operaciones* se definen en 4.2.2.23 y 4.2.2.24 respectivamente.

4.2.1.6 Comportamiento (de un objeto)

El comportamiento de un objeto modelado en ACT ONE depende de las *operaciones* asociadas con la plantilla del objeto y del valor vigente del estado al que el objeto está vinculado. Esto es, el valor del estado al que un objeto está vinculado puede utilizarse para limitar las *operaciones* que pueden tener lugar, por ejemplo, excluyendo la ocurrencia de determinadas *operaciones* cuyas *ecuaciones* no son válidas para ese valor del estado.

ACT ONE no prevé de manera explícita la aplicación de constricciones a ocurrencias de *operación*, tales como secuenciación, no-determinísmo, concurrencia y constricciones en tiempo real como tales. ACT ONE prevé más bien *operaciones* y *ecuaciones* que denotan en sí mismas todas las constricciones posibles, es decir, todos los posibles comportamientos con sus constricciones asociadas.

No existe en ACT ONE la característica con la que modelar acciones internas de manera específica. Esto es, no hay noción de transiciones espontáneas que pudieran ocurrir en el álgebra de proceso con el símbolo de *evento interno* **i** por ejemplo.

Cabe señalar que no hay noción real de comportamiento ocurriendo de hecho, de manera aislada en ACT ONE. La noción de ocurrencia de comportamiento de ACT ONE está asociada normalmente a las expresiones de ACT ONE que se evalúan en el álgebra de proceso.

4.2.1.7 Estado (de un objeto)

El valor vigente al que está vinculada una instancia de un *género* que modela un objeto. Cabe señalar que un *género* que modela un objeto debería contener un identificador utilizado para distinguir entre diferentes instancias del *género*. El valor de ese identificador no forma parte, no obstante, del estado, es decir, ese valor deberá permanecer inmutable en las *operaciones* y *ecuaciones* asociadas con el *género*.

4.2.1.8 Comunicación

Esta noción no se soporta en ACT ONE. Podría ocurrir que una forma abstracta de comunicación se pudiera modelar mediante un estilo de especificación del álgebra de proceso. No obstante, esto no refleja el texto de la Rec. UIT-T X.902 | ISO/CEI 10746-2, es decir, no ocurre que un objeto transporte información a otro objeto. Más bien, es el entorno (el álgebra de proceso) utilizado para comunicar y no otros objetos.

4.2.1.9 Ubicación en el espacio

La noción de ubicación en el espacio no se soporta de manera explícita en ACT ONE. Si es preciso, se puede incorporar esta noción en el modelo de especificación, por ejemplo, mediante un *género* que modele una ubicación en el espacio utilizado en *operaciones* cuya ubicación en el espacio se ha de establecer.

4.2.1.10 Ubicación en el tiempo

La noción de ubicación en el tiempo no se soporta de manera explícita. Sin embargo, si la noción de tiempo está relacionada con el estado vigente de un determinado objeto, es decir, con los cambios de estado que han ocurrido y los que pueden ocurrir, la ubicación en el tiempo en el que una determinada acción puede realizarse se puede determinar, en cierta medida, por el estado vigente de un objeto dado.

La ubicación en el tiempo en el que una acción puede realizarse también se puede incorporar en una especificación, por ejemplo, mediante un *género* que modele una ubicación en el tiempo utilizado en *operaciones* cuya ubicación en el tiempo se ha de establecer.

4.2.1.11 Punto de interacción

Esta noción no es soportada directamente en ACT ONE. Podría ocurrir, no obstante, que este concepto se pudiera incorporar en una especificación. Por ejemplo, mediante un *género* utilizado en las *operaciones* que están en el conjunto de interfaces de la misma ubicación. Esto es, todas las *operaciones* del conjunto de interfaces de la misma ubicación requieren un parámetro de entrada *(género)* que indique la ubicación en la que existe esta *operación*. Si hace falta, se pueden modelar varios puntos de interacción de manera que existan en la misma ubicación. Esto podría realizarse mediante una *operación* de creación de un *género* de ubicación que necesite varios *géneros* de punto de interacción a modo de entradas. Las *operaciones* y *ecuaciones* asociadas con estos *géneros* deberían habilitar la identificación de puntos de interacción y ubicaciones diferentes.

4.2.2 Conceptos de especificación

4.2.2.1 Composición

– **de objetos:** Por lo general no ocurre que dos objetos cualesquiera puedan combinarse en ACT ONE y tengan un resultado significativo, es decir, un objeto compuesto con su propio comportamiento, etc. La forma más probable de composición en ACT ONE es a través de un constructivo *operación* que tenga dos o más objetos como parámetros de entrada y disponga de algún sistema de identificación única. Considérese la siguiente *operación* de constructivo ACT ONE:

makeCO: Id, Ob1, Ob2 -> CO

Aquí se crea un objeto compuesto a partir de otros dos objetos que tienen sus propias *operaciones* y *ecuaciones* asociadas, es decir, sus propios comportamientos. Mientras que *co: CO = makeCO (id1,ob1,ob2)* se compone de los objetos *ob1* y *ob2*, el objeto *co* no tiene un comportamiento como tal. Esto es, los comportamientos asociados con *ob1* y *ob2* no son aplicables a instancias de *CO*.

Para resolver este problema, se pueden especificar para el objeto compuesto *operaciones* y *ecuaciones* adicionales. La forma de estas *operaciones* y *ecuaciones* y su relación con los objetos componentes determina a continuación la forma de la composición. Por ejemplo, si las *operaciones* y *ecuaciones* del objeto compuesto habilitan simplemente el acceso a objetos componentes aislados se realiza una forma de delegación, representando la composición de los objetos componentes una agregación. Si las *operaciones* y *ecuaciones* del objeto compuesto se especifican de manera que modifican el comportamiento de los objetos componentes, se realiza una forma de composición que se asemeja más al texto de la Rec. UIT-T X.902 | ISO/CEI 10746-2. Obsérvese, no obstante, que la noción de composición que se da en la Rec. UIT-T X.902 | ISO/CEI 10746-2 no exige que se haga una especificación ulterior, es decir, entraña la composición de objetos y comportamientos existentes para generar nuevos objetos y comportamientos, y no la especificación de un comportamiento adicional que haga posible una composición significativa de los objetos componentes.

– **de comportamientos:** Puesto que ACT ONE no proporciona operadores de composición específicos, la noción de composición de comportamientos no está prevista de manera explícita. Debe señalarse que existe una forma de composición ACT ONE: la de *enriquecimiento*. Sin embargo, esta forma no puede clasificarse por lo general como composición, ya que no proporciona una composición explícita de comportamientos (u objetos) como tales. Al contrario, el *enriquecimiento* no ofrece por sí mismo, es decir, sin más especificación, características de composición. Todos los *tipos de datos* existen independientemente cuando se aplica el *enriquecimiento*. Sólo cuando se especifican *operaciones* y *ecuaciones* que utilizan los *géneros* puestos a disposición mediante el *enriquecimiento*, se puede aplicar la noción de composición. No obstante, esto es algo que quizá no refleje adecuadamente el texto de la Rec. UIT-T X.902 | ISO/CEI 10746-2, es decir, no ocurre que dos comportamientos se combinen simplemente. Se requiere una especificación ulterior para combinar los comportamientos. Debe añadirse, con todo, que el *enriquecimiento* pone a disposición todas las *operaciones* y *ecuaciones* existentes de los *géneros* que se combinan. Por ello, la nueva especificación necesaria para combinar comportamientos no incluye las *operaciones* y *ecuaciones* de los *géneros* introducidos mediante el *enriquecimiento*.

Se podría realizar también una forma de composición de comportamientos mediante la *actualización* de *tipos de datos parametrizados.* Para ello es necesario que las *actualizaciones de tipos de datos* satisfagan cualesquiera *géneros, operaciones* y *ecuaciones formales* del *tipo de datos* que se *actualiza*. Esto es lo máximo que se puede conseguir en ACT ONE para captar la noción de composición de comportamiento expuesta en la Rec. UIT-T X.902 | ISO/CEI 10746-2. No obstante, es discutible si esto representa composición de comportamientos, ya que puede que no haya comportamiento de un *tipo de datos parametrizados* hasta que haya sido *actualizado*, es decir, no es cierto que pueda ocurrir una instancia de este *género* en el álgebra de proceso y en las *operaciones* aplicadas.

4.2.2.2 Objeto compuesto

El resultado de una composición de objetos.

4.2.2.3 Descomposición

– **de objetos:** En ACT ONE se pueden descomponer los objetos siempre que existan *operaciones* en la firma asociada con el objeto que hagan posible la descomposición. Por ejemplo, el *tipo de datos* siguiente permite descomponer un objeto compuesto en sus objetos componentes.

```
type Z is X, Y, IdType
    sorts Z
    opns makeZ: Id, X, Y → Z
         getX: Z → X
         getY: Z → Y
    eqns forall x: X, y: Y, id: ID
         ofsort X
              getX (makeZ(id,x,y)) = x;
         ofsort Y
              getY (makeZ(id,x,y)) = y;
endtype (* Z *)
```

Así, dado *z: Z = makeZ(id1,x,y)*, en donde *x* e *y* son instancias de *géneros* que modelan objetos e *id1* es un identificador único, se puede descomponer en *x* e *y*, es decir, en sus objetos componentes, mediante *getX(Z)* y *getY(Z)* respectivamente.

NOTA – Esta interpretación se basa en la idea de que un objeto compuesto se puede descomponer en sus partes componentes (objetos). Sin embargo, el texto que figura en la Rec. UIT-T X.902 | ISO/CEI 10746-2 sólo exige que la descomposición especifique un objeto determinado como combinación de dos o más objetos, es decir, una composición. En ACT ONE siempre ocurre que los objetos compuestos se especifican a partir de combinaciones de objetos componentes. Por ello, la distinción entre composición y descomposición que se da en la Rec. UIT-T X.902 | ISO/CEI 10746-2 resulta un tanto confusa cuando se representa en ACT ONE.

– **de comportamientos:** La noción de descomposición de comportamientos depende de la especificación de composición de comportamientos. Este concepto no está previsto de manera explícita en ACT ONE (véase 4.2.2.1). Esto es, los comportamientos se representan mediante *operaciones* y *ecuaciones* que actúan en un *género*. No ocurre que dos comportamientos de *género* cualesquiera puedan combinarse y se genere un nuevo comportamiento.

NOTA – También podría considerarse que la noción de descomposición de comportamientos viene dada inherentemente por las *operaciones* y *ecuaciones* de ACT ONE asociadas con un *género*. Es decir, que esas *operaciones* y *ecuaciones* proporcionan todas las combinaciones posibles de comportamientos. Así por ejemplo, podría generarse una composición secuencial mediante *operaciones* aplicadas secuencialmente. Cada aplicación de una *operación* de la secuencia debe satisfacer las *ecuaciones* necesarias para su ocurrencia. Se puede discutir, de todos modos, si esto es una composición de comportamientos, porque las *operaciones* y *ecuaciones* ya existían y definían todos los comportamientos posibles.

4.2.2.4 Compatibilidad en comportamiento

En LOTOS, la equivalencia en comportamiento de *tipos de datos* se basa en la equivalencia de nombres de *géneros* y, posiblemente también, en el valor al que estos *géneros* están vinculados. Como resultado de esto, no ocurre por lo general que un objeto pueda sustituir a otro objeto en algún entorno si los objetos se obtienen de diferentes plantillas de objeto, es decir, son instancias de *géneros* diferentes. No obstante, cabe la posibilidad de sustituir, algunas veces, un objeto por otro derivado de una plantilla de objeto diferente. Para ello es necesario que el entorno ofrezca operaciones que sean aplicables a ambos *géneros* y que los resultados de esas *operaciones* sean los mismos. Por ejemplo, *géneros* que representen una pila de enteros y una cola de enteros podrían ser compatibles en comportamiento en algún entorno si el entorno ofreciera solamente una *operación tope* y la cola y la pila tuvieran el mismo número de enteros acumulados en ellas. Esto es, el resultado en ambos casos sería un entero. Si un entorno ofrece una *operación pop* o *push*, no existirá compatibilidad en comportamiento entre los objetos, ya que las *operaciones* retornan *géneros* de pila y *géneros* de cola. Puesto que, por lo general, el entorno de un objeto puede invocar cualquier *operación* de la firma, esta forma de compatibilidad en comportamiento está limitada.

4.2.2.5 Refinamiento

Si bien la noción de refinamiento se ha previsto de manera explícita en el álgebra de proceso de LOTOS, por ejemplo, mediante la prueba de *conformidad*, y las relaciones de equivalencia, se ha trabajado poco a propósito del refinamiento en ACT ONE. No obstante, por intuición cabe decir que el refinamiento en ACT ONE podría tomar muchas formas. Por ejemplo, ampliando la firma de un *género* dado, es decir, facilitando más *operaciones*. Esta forma de refinamiento debe generar compatibilidad en comportamiento natural, es decir, que las *operaciones* y *ecuaciones* permanezcan inalteradas. También son posibles otras formas de refinamiento, por ejemplo, la modificación de las *ecuaciones* asociadas con las *operaciones* en un *género*. Asegurar la compatibilidad en comportamiento en este caso probablemente no sea un asunto trivial.

4.2.2.6 Rastro

Puesto que las interacciones no están previstas de manera explícita en ACT ONE, la noción de un rastro está limitada, es decir, no se puede garantizar que no contenga acciones internas. Si las interacciones se consideran como las *operaciones* que ocurren en el álgebra de proceso y las acciones internas como las *operaciones* utilizadas para evaluar las *ecuaciones* asociadas con esas *operaciones*, un rastro puede ser modelado sólo en cierta medida. En este caso corresponde a la secuencia de aplicaciones de *operación* asociadas con una instancia de un *género* que modela un objeto. Debe señalarse que si las *ecuaciones* asociadas con *operaciones* que modelan interacciones se reescriben, el registro de las interacciones de un objeto, es decir, el rastro, probablemente sea incorrecto. Por ejemplo, las aplicaciones de *operación* de un *push* seguido por un *pop* en una *cola* probablemente se tengan que reescribir como *cola* al contrario que la expresión *pop(push(x,q))*. De aquí que la noción de rastro esté limitada en ACT ONE.

4.2.2.7 Tipo de una <X>

Los objetos, las interfaces y las acciones especificadas en ACT ONE pueden satisfacer numerosos y diferentes predicados caracterizadores arbitrarios. Los tipos que se pueden representar de manera explícita son tipos de plantilla.

4.2.2.8 Clase de una <X>

La noción de clase depende del predicado de tipo caracterizador que satisfacen los miembros de la clase. Objetos, interfaces y acciones pueden satisfacer muchos predicados de tipos caracterizadores arbitrarios. Un tipo que se puede representar es un tipo de plantilla. Cuando tal es el caso, la clase de objetos, interfaces y acciones asociadas con ese tipo es la clase de plantilla.

NOTA – Obsérvese que, al establecer que la única clasificación posible en LOTOS para objetos, interfaces y acciones es que satisfacen su tipo de plantilla, los conceptos de clase y clase de plantilla dados en la Rec. UIT-T X.902 | ISO/CEI 10746-2 se reducen a la misma técnica de modelado en LOTOS. Así pues, no hay distinción en LOTOS entre una clase en su sentido de clasificación general y una clase de plantilla en su sentido más restrictivo, como conjunto de instancias de un tipo de plantilla dado.

4.2.2.9 Subtipo/supertipo

La noción de subtipo y supertipo no es sustentada, por lo general, en ACT ONE ya que LOTOS utiliza la equivalencia de nombres cuando verifica tipos. Por ejemplo, dos tipos de objeto sólo son el mismo cuando están representados por el mismo *género*. Así pues, por lo general no ocurre que un *género* pueda reemplazar a otro *género*. Podría ocurrir, no obstante, que una forma limitada de subtipificación existiera entre dos *géneros* diferentes si el predicado de tipo caracterizador se basara en algún aspecto de los *géneros* que no fuese su nombre, por ejemplo, esta *operación* es válida en este *género* y retorna este resultado. De todos modos, se trata de una forma limitada de tipificación y probablemente no exista en la mayoría de los casos.

4.2.2.10 Subclase/superclase

La noción de subtipos y supertipos se sustenta sólo en medida muy reducida en ACT ONE, a saber, cuando el predicado de tipo caracterizador se basa en algún aspecto del *género*. Como resultado de ello, las nociones de subclase y superclase no son sustentadas plenamente en ACT ONE. Si existe una relación subtipo/supertipo entre dos *géneros*, también existe una relación subclase/superclase entre las instancias de los *géneros* en el álgebra de proceso.

4.2.2.11 Plantilla <X>

- **Plantilla de Objeto:** Una definición de *género* con *operaciones* y *ecuaciones* asociadas que modelan un objeto.

- **Plantilla de Interfaz:** El conjunto de *operaciones* y *ecuaciones* asociadas con una definición de *género* que modela un objeto. Debe señalarse que las nociones de plantilla de interfaz y plantilla de objeto se pueden considerar idénticas ya que las *operaciones* y *ecuaciones* deben actuar siempre en una definición de *género*. También ocurre que la declaración de una instancia de un *género* en el álgebra de proceso lleva asociadas implícitamente las *operaciones* y *ecuaciones* especificadas en la parte ACT ONE de la especificación.

- **Plantilla de Acción:** Una *operación* con *ecuaciones* asociadas.

4.2.2.12 Firma de interfaz

Las *operaciones* aplicables a una variable declarada como instancia de un *género* que modela un objeto.

4.2.2.13 Instanciación (de una plantilla < *X* >)

- **de una plantilla de objeto:** La instanciación de una plantilla de objeto requiere la inicialización de un *género* que modele un objeto a un estado inicial válido. Esta inicialización de *género* debe asegurar que la instancia del *género* puede ser referenciada de manera exclusiva.

- **de una plantilla de interfaz:** La instanciación de una plantilla de interfaz se produce cuando se instancia una plantilla de objeto. Un objeto tiene, como tal, una sola interfaz determinada por las *operaciones* y *ecuaciones* que actúan en el *género* desde el que el objeto es instanciado.

- **de una plantilla de acción:** La ocurrencia de una *operación* de ACT ONE en el álgebra de proceso. Esta *operación* debe satisfacer las *ecuaciones* asociadas con esa *operación*.

4.2.2.14 Rol (papel)

La mejor manera de modelar la noción de un rol es mediante un *género* en ACT ONE. Esto es así porque un rol representa un identificador de un comportamiento. Es decir, con la declaración de un *género* se hacen accesibles las *operaciones* y *ecuaciones* aplicables a ese *género*.

4.2.2.15 Creación (de una < *X* >)

- **de un objeto/interfaz:** Puesto que los objetos y las interfaces sólo tienen una forma de existencia cuando se utiliza ACT ONE junto con el álgebra de proceso, no se puede utilizar ACT ONE por sí misma para modelar una creación. Se puede utilizar ACT ONE junto con el álgebra de proceso para modelar en cierta medida la creación de objetos e interfaces, siempre que se siga un determinado estilo de especificación. Por ejemplo, una *operación* asociada con un *género* que modela un objeto, es decir, una *operación* en un objeto ya existente, que da lugar a la generación de un nuevo objeto. Debería ser posible hacer referencia de manera exclusiva al objeto recién generado. Este nuevo objeto debería utilizarse también en el álgebra de proceso de modo que tenga alguna forma de existencia (véase 4.2.1.1 para más detalles sobre cómo puede lograrse esto).

4.2.2.16 Introducción (de un objeto)

La introducción de un objeto puede efectuarse de varias maneras en ACT ONE cuando se utiliza junto con el álgebra de proceso. Por ejemplo, mediante ocurrencias de ofertas de eventos cuyas *denotaciones de acción* dan lugar a una nueva instancia de un *género* que modela un objeto que se genera. Estas nuevas instancias deberán estar en un estado inicial válido, ser referenciables de manera exclusiva y tener alguna forma de existencia en el álgebra de proceso. Alternativamente, se pueden introducir objetos mediante cláusulas **let ... in**. También aquí deberán tener un estado inicial válido, deberán ser utilizados en el álgebra de proceso de modo que tengan alguna forma de existencia y deberá ser posible identificarlos de manera exclusiva.

4.2.2.17 Supresión (de una < *X* >)

- **de un objeto/interfaz:** La supresión de un objeto o de una interfaz puede realizarse en ACT ONE cuando se utiliza junto con el álgebra de proceso, mediante la reescritura que se produce con las *ecuaciones* asociadas con las *operaciones* en los *géneros* que modelan objetos. Por ejemplo, una *operación* que elimina un elemento de un conjunto se puede utilizar para modelar la supresión de un objeto con un determinado identificador, que se elimina (se suprime) del conjunto de objetos instanciados existentes como parte de la *lista de parámetros de valor* de una *definición de proceso* recursiva.

4.2.2.18 Instancia de un tipo

- **de una Plantilla de Objeto/Interfaz/Acción:** Una instancia de un tipo depende del predicado caracterizador que define el tipo. Si el predicado de tipo es el tipo de plantilla para objetos e interfaces, una instancia del objeto o tipo de interfaz corresponde a una ocurrencia del *género* que modela los objetos e interfaces que se consideran en el álgebra de proceso. De manera similar, si el predicado de tipo es el tipo de plantilla para acciones, una instancia de un tipo de acción viene dada por la ocurrencia de una *operación* que modela el tipo de acción que se considera en el álgebra de proceso.

4.2.2.19 Tipo de plantilla (de una < *X* >)

- **de un Objeto/Interfaz:** Un predicado en instanciaciones del *género* utilizado para modelar un objeto. Todas las instanciaciones (ocurrencias) de la plantilla (*género*) en el álgebra de proceso tienen las *operaciones* y *ecuaciones* que están asociadas con ese *género*. Puesto que una plantilla para un objeto y una interfaz se modelan del mismo modo, es decir, mediante una definición de *género* con *operaciones* y *ecuaciones* asociadas, el tipo de plantilla de un objeto y el tipo de plantilla de la interfaz a ese objeto son sinónimos en ACT ONE. Esto es, ambos corresponden a la ocurrencia de un *género* en el álgebra de proceso.
- **de una Acción:** Un predicado en ocurrencias de *operación* en el álgebra de proceso. Esto es, todas las instanciaciones (ocurrencias) de la plantilla (*operación*) en el álgebra de proceso deben cumplir los requisitos de la plantilla, es decir, deben tener las mismas entradas y producir los mismos resultados que genera la definición de *operación*, y la evaluación de la *operación* se rige por las *ecuaciones* asociadas con esa *operación*.

4.2.2.20 Clase de plantilla (de una < *X* >)

- **de un Objeto:** Es el conjunto de instancias de un *género* determinado que modela un objeto en el álgebra de proceso.
- **de una Interfaz:** Es el conjunto de instancias de un *género* determinado que modela una interfaz a un objeto en el álgebra de proceso.
- **de una Acción:** Es el conjunto de instancias de una *operación* determinada en el álgebra de proceso.

4.2.2.21 Clase derivada/clase de base

En ACT ONE no se sustentan clases derivadas ni de base. Esto se debe a que las clases en ACT ONE normalmente vienen dadas sólo mediante clases de plantilla, es decir, para objetos, el conjunto de instancias de un *género* determinado que modela un objeto en el álgebra de proceso. No ocurre que los *géneros* puedan ser modificados de manera incremental. Esto es, los *géneros* y las etiquetas anexadas a los mismos, o sea, los nombres de los *géneros*, no permiten hacer referencia a otro *género* ya que siempre existe autorreferencia. Así pues, las *operaciones* y *ecuaciones* asociadas con un *género* determinado sólo son aplicables a ese *género* y no a otro *género*.

Debe señalarse también que aunque la noción de *actualización* de las clases parametrizadas induce a pensar que poseen las características de las relaciones clase derivada/clase de base, no representan de hecho esas relaciones. Esto es así porque no ocurre que las instancias de una clase parametrizada puedan producirse en el álgebra de proceso, es decir, que han de ser *actualizadas* para que una clase pueda existir.

4.2.2.22 Invariante

Esta noción es implícita dentro de ACT ONE, es decir, los objetos deben satisfacer siempre las *operaciones* y *ecuaciones* que les son aplicables.

4.2.2.23 Precondición

En ACT ONE, todas las *operaciones* deben satisfacer todas las *ecuaciones* (y cualesquiera *guardas* asociadas) que les sean aplicables antes de que puedan ocurrir.

4.2.2.24 Postcondición

Esta noción es implícita dentro de ACT ONE, es decir, la ocurrencia de una *operación* (acción) determinada requiere que se definan (verdadero) las *ecuaciones* asociadas.

4.3 Semántica arquitectural en SDL-92

SDL-92 es un FSL normalizado. Se adjunta material docente a la Rec. UIT-T Z.100. Existe un cierto número de libros de texto relativos al SDL, y también instrumentos de tipo comercial que sustentan diferentes aspectos del SDL, desde el manejo de gráficos al análisis y generación de códigos de programación basados en SDL.

SDL modela un *sistema* como un conjunto de máquinas de estados finitos ampliadas que comunican mediante mensajes llamados *señales*. El concepto de datos de SDL se basa en ACT ONE. Las máquinas de estados se amplían en el sentido de que pueden definir variables locales para retener parte de su historial. Las señales se comunican de manera asíncrona, con lo que se ofrece un acoplamiento débil entre componentes de un sistema distribuido.

En esta subcláusula se presenta una manera de expresar conceptos modeladores en SDL. No se considera que la representación sea única. No obstante, establece que casi todos los conceptos fundamentales se pueden expresar en SDL. Debe señalarse que existe además un procedimiento alternativo para el modelado de muchos de los conceptos que aquí se dan. Esto se refiere a la utilización de ACT ONE. En 4.2 figuran los detalles relativos a este otro procedimiento.

La versión utilizada es SDL-92, definida en la Rec. UIT-T Z.100. SDL-92 contiene un cierto número de ampliaciones en comparación con SDL-88. Las más importantes son:

- constructores orientados al objeto;
- posibles *canales* no retardadores;
- no determinismo;
- posible inclusión de conceptos de datos alternativos y llamadas de *procedimiento distante*.

La característica más significativa de la tipificación de datos alternativos es que permite la utilización combinada de ACT ONE y ASN.1 dentro de SDL. La semántica de la combinación de SDL-92 con ASN.1 se define en la Rec. UIT-T Z.105.

Para evitar confusiones, se ha utilizado letra *cursiva* en el texto para indicar conceptos de SDL, cuando así se ha considerado necesario.

4.3.1 Conceptos básicos de modelado

4.3.1.1 Objeto

Los objetos en SDL son instancias de *tipo de sistema, tipo de bloque, tipo de proceso, tipo de servicio, temporizador, canal* y *ruta de señales*. Instancias que se caracterizan por un estado y un comportamiento.

Cada instancia está encapsulada, es decir, cualquier cambio en su estado sólo puede ocurrir como resultado de una acción interna o de una interacción con su entorno.

Las referencias para algunos tipos de objetos se han de indicar de manera explícita.

4.3.1.2 Entorno (de un objeto)

El entorno de un objeto depende de la clase de objeto. Véase el cuadro 1.

4.3.1.3 Acción

Una acción en SDL es una *entrada* o *conservación* simple, una *sentencia de acción simple*, una *transición* completa o la ejecución completa de un *procedimiento*. Posibles *declaraciones de acción* simples son:

- *tarea, importación, exportación, visión;*
- *salida;*
- *crear;*
- *inicializar, reinicializar, activar;*
- *llamada de procedimiento;*
- *parada/retorno;*
- *estado siguiente.*

Cuadro 1 – Entorno de objeto

Clase de objeto	Constricciones del entorno
sistema	– *señales* entrantes de canales
bloque	– tipo de datos globales
	– *señales* entrantes de canales
	– llamadas de *procedimientos exportados*
	– variables importadas
proceso	– tipos de datos globales
	– *señales* entrantes de *rutas de señales* implícitas o explícitas
	– llamadas de *procedimientos exportados*
	– variables visionadas/importadas
	– constricciones de tiempo para acciones de *entrada*
servicio	– tipos de datos/*temporizadores*/variables globales, memoria tampón de señales compartida (perteneciente a instancia de *proceso* abarcadora)
	– *señales* entrantes de *rutas de señales* implícitas o explícitas
	– llamadas de *procedimientos exportados*
	– variables visionadas/importadas
	– constricciones de tiempo para acciones de *entrada*
temporizador	– llamadas de *inicializar* y *reinicializar, parada* del *proceso* del propietario
canal	– *señales* entrantes procedentes de los *bloques* conectados (del entorno de *sistemas*)
ruta de señales	– *señales* entrantes procedentes de las instancias de *proceso*/*servicio* conectadas

La transmisión de una *señal* por un *canal* o una *ruta de señales* también es una acción, como lo es la generación de una *señal de temporizador*. Las interacciones son la *entrada/salida* de una *señal*, la *llamada* y la acción de *retorno* de un *procedimiento distante*, y la utilización de variables compartidas (variables de *servicios* de *procesos* globales, variables *reveladas/visionadas* y de *importación/exportación* de *procesos*). La secuencia de acciones de envío, transporte y, en su caso, recepción de una *señal* (*salida/entrada*) se puede considerar como una interacción.

4.3.1.4 Interfaz

Hay diferentes maneras de describir las interfaces en SDL, dependiendo de la clase de objeto, como se muestra en el cuadro 2.

En el caso de un objeto *bloque*, el conjunto de *procedimientos distantes exportados/importados* al/del exterior del *bloque* así como el conjunto de *señales* enviadas/recibidas por los *procesos* de ese *bloque* deberán ser encapsulados en uno o más procesos. Estos *procesos* actúan a continuación como las interfaces del objeto *bloque*. Con esas descripciones de interfaces sólo se definen las interacciones potenciales de un objeto, describiendo cada interfaz un subconjunto de interacciones potenciales de un objeto.

Cuadro 2 – Interfaces de objeto

Clase de objeto	Interfaz caracterizada por
sistema	– *puertas de sistema* con sus listas de *señales*
	– *canale*s entrantes/salientes con sus listas de *señales*
bloque	– *puertas de bloque* con sus listas de *señales*
	– *canales* entrantes/salientes con sus listas de *señales*
	– *rutas de señales* entrantes/salientes con sus listas de *señale*
	– conjunto de *procedimientos distantes* (*exportados/ importados* al/del exterior del *bloque*)
	– conjunto de variables distantes (*exportadas/importadas* al/del exterior del *bloque*)
proceso	– *puertas de proceso* con sus listas de *señales*
	– conjunto de todas las *señales de entrada/salida* válidas
	– conjunto de todos los *procedimientos exportados/importados*
	– conjunto de variables compartidas (*reveladas/visionadas* al/del exterior del *proceso*)
servicio	– *puertas de servicio* con sus listas de *señales*
	– conjunto de todas las *señales de entrada/salida* válidas
	– conjunto de todos los *procedimientos exportados/importados*
temporizador	– identificación de *temporizador*
canal	– conjuntos de todas las *señales* llevadas en cada sentido
ruta de señales	– conjuntos de todas las *señales* llevadas en cada sentido

4.3.1.5 Actividad

Por lo general no se puede denotar de manera explícita una actividad, ya que puede abarcar varios objetos.

Un caso especial de actividad es la ejecución de un *procedimiento distante* o local en el que la acción de *llamada* es la cabecera de la actividad y las acciones de retorno potenciales son las acciones de cola.

4.3.1.6 Comportamiento (de un objeto)

El comportamiento de un *proceso/servicio* es el conjunto de todas las transiciones de ese *proceso/servicio*. Las acciones de *entrada* imponen constricciones a las circunstancias en las que las transiciones pueden ocurrir. Se puede introducir constricciones adicionales utilizando el constructivo *proporcionar* o el constructivo *señal continua*. Un objeto puede mostrar un comportamiento no determinístico.

El comportamiento de un *bloque* se agrega a partir del comportamiento de los *procesos* contenidos en ese *bloque*. Un sistema o bien el comportamiento de los *bloques* contenidos en ese *sistema*.

El comportamiento de un *canal* o una *ruta de señales* es el transporte de *señales* (de manera instantánea o retardada cuando así se especifique).

El comportamiento de un *temporizador* se predefine en SDL en el sentido de un despertador.

4.3.1.7 Estado (de un objeto)

El conjunto de todas las secuencias de acciones en las que un objeto puede participar en un instante dado viene determinado, en el caso de un *proceso/servicio*, por el estado SDL vigente en ese instante, los valores de las variables locales y el contenido de puerto de entrada.

El estado de un *bloque* o un *sistema* es el conjunto de los estados de todos los *procesos*. Un sistema o bien los *bloques* más todos los *canales* o *rutas de señales* contenidos.

El estado de un *canal* viene dado implícita o explícitamente por un *bloque*. El estado depende de si el *canal* tiene o no una propiedad de *retardo*.

El estado de una *ruta de señales* siempre se da implícitamente.

El estado de un *temporizador* es activo o no activo. El estado de un *temporizador* activo viene determinado por el tiempo que queda para el envío de una *señal* de temporización.

4.3.1.8 Comunicación

El transporte de información entre dos o más objetos se hace mediante *canales* o *rutas de señales* explícitos o implícitos (en el caso de *procedimientos* distantes o variables *importadas/exportadas*). La información se lleva mediante *señales*.

4.3.1.9 Ubicación en el espacio

Las acciones ocurren dentro de instancias de *proceso* e instancias de *servicio*. Las acciones de transmisión ocurren dentro de *canales* o *rutas de señales*.

4.3.1.10 Ubicación en el tiempo

Cada acción se caracteriza mediante la fecha en que comienza y la fecha en que termina. Las acciones pueden ser instantáneas. La duración de una acción no puede ser denotada explícitamente. Las acciones pueden ser programadas para una fecha concreta o para después de un plazo especificado.

NOTA –

a) En SDL hay un tiempo global accesible mediante *ahora*. Nada se dice a propósito de las unidades de tiempo.

b) Para dos acciones consecutivas *a1* y *a2*, la relación aplicable es *ahora(a1) <= ahora(a2).*

c) Las únicas maneras de direccionar tiempo explícitamente son la acción inicializar para un *temporizador* y la aplicación de *ahora* en *condiciones habilitantes* y *señales continuas*. Ha de evitarse la programación de acciones para un determinado instante en el tiempo.

4.3.1.11 Punto de interacción

Los puntos de interacción son las puertas de las instancias de *bloque/proceso/servicio* y los puntos extremo de *canales* (posiblemente implícitos) y *rutas de señales*. Las variables compartidas son también puntos de interacción. Tienen una ubicación. Un objeto puede tener varios puntos de interacción.

4.3.2 Conceptos de especificación

4.3.2.1 Composición

– **de objetos:**

a) un objeto *sistema* puede ser una composición concurrente de objetos *bloque* que pueden ser interconectados por *canales*;

b) un objeto *bloque* puede ser una composición concurrente de objetos *bloque* que pueden ser interconectados por *canales* o una composición concurrente de objetos *proceso* que pueden ser interconectados por *rutas de señales*;

c) un objeto *proceso* puede ser una composición entrelazante de objetos *servicio*;

d) un canal puede ser una composición de *bloques* interconectados por canales.

– **de comportamientos:**

a) el comportamiento de un *sistema* es una composición concurrente del comportamiento de sus *bloques*;

b) el comportamiento de un *bloque* es una composición concurrente del comportamiento de sus *subbloques* o de sus *procesos*;

c) el comportamiento de un *proceso* es una composición entrelazante del comportamiento de sus *servicios* o una composición secuencial de las acciones de su gráfico de *proceso*;

d) el comportamiento de un *servicio* es una composición secuencial de las acciones de su gráfico de *servicio*.

4.3.2.2 Objeto compuesto

De acuerdo con 4.3.2.1, los objetos siguientes se pueden expresar como una composición:

– *sistema;*

– *bloque;*

– *proceso;*

– *canal.*

4.3.2.3 Descomposición

– **de objetos:** La especificación de un objeto determinado como una composición.

– **de comportamientos:** La especificación de un comportamiento determinado como una composición.

4.3.2.4 Compatibilidad en comportamiento

No hay un procedimiento general con el que describir explícitamente la compatibilidad en comportamiento en SDL. No obstante, la base semántica del lenguaje en lo relativo a sistemas de transición permite la definición de compatibilidad en comportamiento y su verificación.

Una instancia de una clase derivada puede considerarse compatible en comportamiento restringido con una instancia de la clase de base correspondiente y una instancia de un tipo *redefinido* puede considerarse compatible en comportamiento restringido con una instancia del *tipo virtual* correspondiente. Las cláusulas *Atleast* se pueden utilizar para exigir una compatibilidad en comportamiento restringido.

4.3.2.5 Refinamiento

Hay dos maneras de refinar la especificación SDL de un objeto:

– subestructurando (a nivel de *bloque* o *sistema*);

– utilizando las características orientadas al objeto (herencia, virtualidad y parámetros genéricos).

4.3.2.6 Rastro

No hay ninguna manera explícita de especificar rastros en SDL. Los rastros se pueden obtener como resultado de la interpretación de una especificación SDL de acuerdo con la semántica dinámica de SDL.

NOTA – Los gráficos de secuencias de mensajes (MSC, *message sequence charts*) proporcionan una sintaxis y una semántica apropiadas para la representación de rastros de especificaciones SDL. Existe una estrecha relación entre la sintaxis y la semántica de los MSC y la sintaxis y la semántica del SDL. Los MSC se definen y normalizan en la Rec. UIT-T Z.120.

4.3.2.7 Tipo (de una <X>)

No hay predicado explícito general en SDL.

4.3.2.8 Clase (de una <X>)

En general no se soporta este concepto, sólo para tipos de plantilla.

4.3.2.9 Subtipo/supertipo

En general no se soporta este concepto.

4.3.2.10 Subclase/superclase

En general no se soporta este concepto, sólo para tipos de plantilla.

4.3.2.11 Plantilla <X>

- **Plantilla de Objeto:** Las plantillas de objeto son definiciones de tipo para la clase de objeto apropiada (*sistema, bloque, proceso, servicio*). Para *temporizadores, canales* y *rutas de señales*, las plantillas de objeto son las declaraciones correspondientes.
- **Plantilla de Interfaz:** Dependiente de la clase de interfaz, una plantilla de interfaz puede ser dada implícitamente por una declaración (*canal, rutas de señales*) o explícitamente por una definición de tipo para un *proceso* (véase 4.3.1.4).
- **Plantilla de Acción:** Una plantilla de acción viene especificada por la definición de un *gráfico* de *proceso/procedimiento/servicio.* Las plantillas de acción atómica son *entrada, salida, conservación, inicializar, reinicializar, crear, tarea, parada, retorno, estado siguiente, llamada, importación, exportación, visión*.

Las plantillas se pueden especificar utilizando parámetros (parámetros formales o parámetros de contexto formal). Los parámetros pueden tener constricciones. Las plantillas pueden ser combinadas (es decir, una definición de tipo puede contener otras definiciones de tipo).

4.3.2.12 Firma de interfaz

El conjunto de *tipos de señal* y *tipos de procedimiento distante* que son válidos para la interfaz.

4.3.2.13 Instanciación (de una plantilla <X>)

En SDL hay dos maneras de instanciar plantillas:

- instanciación implícita (*sistema, bloque, canales, rutas de señales, procesos, servicios*) que se hace mediante declaración de objeto;
- instanciación explícita, que utiliza *crear* (sólo para *procesos*).

Las instanciaciones son siempre el resultado de una acción para instanciar una plantilla. Los parámetros de contexto formal tienen que ser actualizados antes de que pueda obtenerse la instanciación (mediante una especialización de tipo de *proceso* o una declaración de *proceso*).

4.3.2.14 Rol (papel)

No hay ninguna manera general de especificar roles.

Los roles se pueden describir como parámetros de contexto formal.

NOTA – Se pueden utilizar cláusulas *Atleast* para una mayor cualificación de un rol.

4.3.2.15 Creación (de una <X>)

Hay dos clases de creación (véase 4.3.2.13):

- instanciación implícita;
- instanciación explícita, interpretación de una acción *creación*.

4.3.2.16 Supresión (de una <X>)

Sólo se pueden suprimir los objetos *proceso*. Un *proceso* sólo se puede suprimir a sí mismo. Esto se hace mediante la interpretación de la acción *parada*. Si un *servicio* interpreta una acción *parada*, se produce la supresión de ese *servicio*, la supresión de todos los demás *servicios* pertenecientes al mismo *proceso* y la supresión del *proceso*.

NOTA –

a) La supresión de un proceso por otro proceso se puede modelar utilizando la salida de una señal especial cuyo eventual consumo por el receptor hace que éste interprete una parada.

b) La supresión de todos los procesos de un bloque se puede considerar como una supresión de ese bloque.

4.3.2.17 Introducción (de un objeto)

La instanciación implícita (véase 4.2.3.13) se puede considerar que es una introducción.

4.3.2.18 Instancia de un tipo

Un objeto es una instancia de un *tipo de sistema*, *tipo de bloque*, *tipo de proceso* o *tipo de servicio* X si hay una instanciación explícita o implícita de esa X o de un sustituto de X. Un sustituto es una instancia de una plantilla de tipo que es una especialización de tipo SDL.

4.3.2.19 Tipo de plantilla (de una <X>)

El hecho de que una <**X**> sea una instanciación de una plantilla <**X**> se puede expresar para *procesos*, *servicios*, *bloques* y *sistemas* mediante la denotación de que el objeto es una instancia SDL de la definición de tipo.

4.3.2.20 Clase de plantilla (de una <X>)

No hay una notación explícita general para caracterizar la clase de plantilla de una <**X**>; sin embargo, la clase de plantilla es el conjunto de todas las instancias de *procesos*, *bloques*, *servicios* o *sistemas* de una definición de *tipo de proceso*, definición de *tipo de bloque*, definición de *tipo de servicio*, o una definición de *tipo de sistema* respectivamente.

4.3.2.21 Clase derivada/clase de base

Una definición de tipo se puede derivar de otra definición de tipo mediante *especialización*, lo que incluye:

- la vinculación de parámetros de contexto y la adición de nuevos parámetros de contexto;
- la herencia de definiciones;
- la redefinición de componentes virtuales;
- la adición de otras definiciones.

Se pueden aplicar constricciones utilizando los constructivos *por lo menos* y *finalizado*.

NOTA – La herencia múltiple no se sustenta.

4.3.2.22 Invariante

En SDL no hay notación para invariantes.

4.3.2.23 Precondición

Se pueden utilizar *condiciones habilitantes*, *señales continuas* y *señales* para especificar las precondiciones de una transición.

4.3.2.24 Postcondición

En SDL no hay notación para postcondiciones.

4.4 Semántica arquitectural en Z

La notación Z es una notación de especificación basada en una teoría de conjuntos muy tipificada y en el cálculo de predicados de primer orden. Z todavía no es un FSL de ISO sino una norma que está siendo preparada por el WG19 de SC22 de ISO. La versión más reciente de la norma está contenida en la norma de base Z. Véase la cláusula 2.

Existe una norma de facto para Z (notación Z de Spivey). Véase la cláusula 4. Esta versión de la notación es estable y muy próxima a la norma de base Z y se soportan instrumentos de verificación de la sintaxis y la semántica estática. Es la versión de la notación que se utilizará para la semántica arquitectural del RM-ODP hasta que se disponga ampliamente de una versión ISO de la documentación.

Para evitar confusiones en la terminología relativa al ODP y a Z, en las subcláusulas que siguen se utiliza letra *cursiva* para denotar términos específicos de Z.

4.4.1 Conceptos básicos de modelado

4.4.1.1 Objeto

En Z se puede describir un objeto mediante una colección de fragmentos de especificación. Dichos fragmentos deberán contener una colección de *esquemas de operación* (que representan la interfaz del objeto) incluyendo los *esquemas de estados* apropiados. Estos esquemas de estado pueden incluir predicados que se utilizan para representar (fragmentos de) las invariables de objeto. Los fragmentos de especificación deberán también disponer de algún medio por el que puedan ser referenciados de manera exclusiva (representando la intensidad del objeto). Esto puede conseguirse mediante un identificador en el(los) *esquema(s) de estado* del objeto que permanezca constante en todas las operaciones definidas para ese objeto. Por último, ha de existir un estado inicial válido de ese objeto, lo que puede realizarse utilizando un esquema de inicialización que dé vinculaciones legales a las variables declaradas en el *esquema de estado* con un predicado que asegure que el objeto es único dentro de la especificación.

NOTA – Cuando se especifiquen objetos en Z se ha de actuar con precaución, ya que el lenguaje Z no posee características de encapsulación (que son esenciales para describir objetos) como se ve en la nota de 4.4.1.3.

4.4.1.2 Entorno (de un objeto)

El entorno de un objeto en una especificación Z se describe en base a su entrada y su salida. La entrada a un objeto viene del entorno. La salida de un objeto va al entorno. El entorno de un objeto se puede especificar directamente o se puede dejar sin especificar. Si no se especifica, puede haber ocurrencias de *esquemas de operación* que produzcan salidas o requieran entradas con el entorno de la especificación en su conjunto, ya sea proporcionando las entradas o recibiendo las salidas, respectivamente. Sin embargo, si se especifica el entorno de un objeto, ello significa que para cada *esquema de operación* asociado con el objeto existe otro *esquema de operación* (posiblemente asociado con otro objeto) que requiere entradas o salidas del mismo tipo que el objeto considerado. A estos dos *esquemas de operación* se les une otro a continuación, con lo que las entradas/salidas de la operación considerada se denominan respectivamente, a partir de ese momento, salidas y entradas de la operación que representa el entorno.

El entorno de un objeto también se puede dar mediante variables referenciadas por un objeto que tenga un alcance global, por ejemplo, alguno de los que figuran en las *descripciones axiomáticas*.

4.4.1.3 Acción

En Z se modela una acción mediante la realización de una operación especificada en un *esquema de operación*. El efecto es el cambio instantáneo de estado (o el cambio nulo) de los objetos con los que está asociada la acción. Una acción puede producir un resultado no determinístico.

Puesto que no hay notación explícita de encapsulación en Z, no es usual determinar si una acción es observable o interna en Z, por lo que la distinción entre interacciones y acciones internas no está claramente definida. Esta Recomendación | Norma Internacional utilizará el convenio de que un *esquema de operación* que representa una acción que tiene entradas, salidas o variables globales a lo largo de la especificación, interactúa con su entorno. El entorno puede ser especificado o puede no serlo (véase 4.4.1.2). Las acciones que requieran entradas de un entorno no especificado y que no producen salidas pueden considerarse como acciones no observables invocadas externamente. Las acciones que producen salidas que van a un entorno no especificado pueden considerarse como acciones observables (espontáneas) invocadas internamente. Las acciones que requieren entradas de un entorno no especificado y producen salidas que van a ese entorno pueden considerarse como acciones observables invocadas externamente.

Sin embargo, si se especifica el entorno de un objeto, ello significa que para cada *esquema de operación* que requiera entradas o salidas y que esté asociado con una interfaz a un objeto, es decir, una acción observable, existe otro *esquema de operación* (posiblemente asociado con otro objeto) que requiere entradas o salidas del mismo tipo que el objeto considerado. A estos dos *esquemas de operación* se les une otro a continuación, con lo que las entradas/salidas de la operación considerada se denominan respectivamente, a partir de ese momento, salidas y entradas de la operación que representa el entorno.

De manera alternativa, la ocurrencia de operaciones que hagan referencia a variables globales a lo largo de la especificación se pueden considerar como interacciones.

Todas las operaciones de Z son atómicas. Esto es, los *esquemas de operación* de Z se producen en su totalidad o no se producen en absoluto. Así pues, no es posible tener en Z acciones que no sean atómicas.

Un objeto que interactúa consigo mismo puede ser modelado informalmente mediante la composición de *esquemas de operación* de Z. Por ejemplo, la operación OpA con salida *a!: A* se puede componer con la operación OpB, con entrada *b?: A*, y se puede añadir una *conjunción* de predicado para indicar que *a! = b?*.

La noción de relaciones de causa a objeto no se hallan en sentido estricto dentro del ámbito de aplicación de Z. Sin embargo, si una operación necesita que se produzca una entrada, podría considerarse que es el entorno el que provoca la ocurrencia de la operación, es decir, que el entorno actúa como el productor y el *esquema de operación* como el consumidor. De manera similar, si un *esquema de operación* produce una salida, podría considerarse que es el entorno el que actúa como el consumidor y la operación como el productor. Si un *esquema de operación* dado requiere tanto entradas como salidas determinadas, o no tiene entradas o salidas, no es posible dar una relación de causa a efecto a esa acción particular.

NOTA – Debe señalarse que este convenio sintáctico para distinguir entre acciones internas y acciones observables es limitado ya que no hay distinción semántica entre operaciones que se han de interpretar como espontáneas o internas y aquellas que requieren participación ambiental; esto se deja para el comentario de lenguaje natural que deberá acompañar a todas las especificaciones de Z. Como consecuencia de lo anterior, la definición indicada más arriba trata una cola con pérdidas como un subtipo de una cola. No obstante, la intención evidente del funcionamiento "extra lose" en una cola con pérdidas es que ocurra de manera no determinística.

4.4.1.4 Interfaz

Una abstracción del comportamiento de un objeto obtenida identificando las operaciones asociadas con ese objeto que han de formar la sustancia de la interfaz. En todos los *esquemas de operación* restantes, se ocultan todas las entradas y salidas y la ocurrencia de las operaciones definidas en esos *esquemas de operación* se consideran como acciones internas, es decir, no requieren o no implican la participación del entorno del objeto. El texto Z resultante que representa ese objeto es una plantilla de interfaz. Cualquier instancia de la plantilla de interfaz es una interfaz.

4.4.1.5 Actividad

La noción de actividad como un gráfico acíclico, dirigido y de una sola cabeza, de acciones, no existe directamente en el lenguaje Z. Sin embargo, el concepto de actividad puede ser modelado en cierta medida teniendo en cuenta que si la acción *x* precede a la acción *y* en alguna actividad, la *postcondición* de la acción *x* debe implicar la *precondición* para la acción *y*.

4.4.1.6 Comportamiento (de un objeto)

El comportamiento de un objeto en un estado determinado es el conjunto de todas las actividades que pueden tener lugar a partir de ese estado. La secuencia efectiva de las acciones que pueden realizar, puede verse afectada por el entorno del objeto y las constricciones expresadas en las *precondiciones*.

4.4.1.7 Estado (de un objeto)

Una vinculación de las variables del estado declaradas en el(los) *esquema*(s) *de estado* asociado(s) con la plantilla del objeto utilizada para calcular *precondiciones*.

4.4.1.8 Comunicación

En Z se puede modelar la comunicación mediante entradas y salidas de operaciones. Las entradas a y las salidas de *esquemas de operación* se consideran normalmente como comunicaciones con el entorno de un objeto. Puesto que la comunicación se produce entre objetos, se debe especificar el entorno de un objeto (véase 4.4.1.2) para modelar la comunicación. A continuación tiene lugar la comunicación normalizando primero los *esquemas de operación* asociados con los objetos interactuantes y uniéndolos seguidamente; a partir de ese momento, las salidas de una operación se denominan entradas al otro *esquema de operación*. Este modelado de la comunicación requiere que las entradas y salidas de los *esquemas de operación* asociados sean del mismo tipo.

De manera alternativa, la ocurrencia de *esquemas de operación* que hacen referencia a variables globales a lo largo de la especificación representan comunicaciones, comunicándose el valor de la variable global que sigue a la ocurrencia de la operación con todos los *esquemas de operación* que hacen referencia a esa variable.

4.4.1.9 Ubicación en el espacio

El concepto de espacio no se considera primitivo en Z. La ubicación en el espacio en la que se produce una acción sólo puede darse en Z en base al modelo de especificación, más bien que en base al sistema del mundo real que se modela. Así pues, una ubicación en el espacio podría ser introducida como un tipo Z. De esta manera, se pueden especificar relaciones asociando *esquemas de operación* con ubicaciones dadas en el espacio, lo que permite a continuación, analizar cuáles son las ubicaciones en el espacio en las que pueden realizarse acciones.

4.4.1.10 Ubicación en el tiempo

El concepto de tiempo no se considera primordial en Z. La ubicación en el tiempo en que se realiza una acción sólo puede darse en Z en base al modelo de especificación, más bien que en base al sistema del mundo real que se modela. Así pues, una ubicación en el tiempo podría introducirse como un tipo Z que puede ser asociado con determinadas acciones, por ejemplo, mediante alguna relación. De esta manera es posible cuantificar el tiempo en el que las acciones pueden realizarse.

4.4.1.11 Punto de interacción

El concepto de punto de interacción depende de las definiciones de interacción y de ubicaciones en el espacio y en el tiempo. Véanse 4.4.1.3, 4.4.1.9 y 4.4.1.10.

4.4.2 Conceptos de especificación

4.4.2.1 Composición

- **de Objetos:** La composición de objetos no es una característica ofrecida explícitamente por el lenguaje Z, debido entre otras cosas a la falta de encapsulación. Sin embargo, es posible modelar algunas características de la composición mediante la inclusión de esquemas y la redefinición de operaciones a través de la promoción.
- **de Comportamientos:** Puesto que un comportamiento en el caso más degenerado debe considerarse como una acción, y una acción en Z es la realización de una operación definida por un *esquema de operación*, la composición de acciones equivale a la combinación de *esquemas de operación* en Z. Los esquemas de operación se pueden combinar en Z de varias maneras, tales como:
  - *cálculo de esquemas*;
  - *composición de esquemas* (;);
  - *revocación* (⊕).

Por lo general, se pueden derivar de la composición características del comportamiento resultante que no pueden derivarse de los comportamientos individuales que se combinan. Además, se pueden suprimir detalles y relevantes de los comportamientos combinados.

4.4.2.2 Objeto compuesto

Véase la interpretación de la composición más arriba.

4.4.2.3 Descomposición

- **de objetos:** Véase la interpretación de la composición de objetos más arriba.
- **de comportamientos:** Véase la interpretación de la composición de comportamientos más arriba.

4.4.2.4 Compatibilidad en comportamiento

La compatibilidad en comportamiento se basa en la noción de sustituibilidad en un entorno dado. La extensión es una posible manera de conseguirla. La extensión de una plantilla de base puede tener componentes adicionales en el *esquema de estado* asociado, un invariante de estado más fuerte, condiciones iniciales más fuertes y más *esquemas de operación*. Los *esquemas de operación* asociados con la extensión del tipo de plantilla pueden tener *precondiciones* más débiles y *postcondiciones* más fuertes que los *esquemas de operación* correspondientes en el tipo de plantilla de base.

4.4.2.5 Refinamiento

El refinamiento es el proceso de transformación de una especificación en otra especificación más detallada. Puesto que Z trata de abstracciones de sistemas en las que los datos y las operaciones sobre los mismos se utilizan para representar el sistema que se considera, se han identificado dos formas principales de refinamiento:

- *refinamiento de operación*; y
- *refinamiento de datos.*

Para refinar una especificación, el refinamiento debe asegurar la compatibilidad en comportamiento entre la especificación y el refinamiento. Para tener esto en cuenta existen ciertas condiciones con las que se garantiza que el refinamiento de una especificación de Z genera una especificación válida más detallada. Dichas condiciones son las de *seguridad* y *vitalidad*. La condición de seguridad respecto al refinamiento de una especificación consiste en que cualquier circunstancia que sea aceptable para la especificación debe serlo también para el refinamiento. La condición de vitalidad aplicada al refinamiento de una especificación quiere decir que, cualquiera que sea la circunstancia aceptable para la especificación, esta última debe permitir el comportamiento del refinamiento.

Las condiciones de seguridad y vitalidad han de ser aplicables tanto a los refinamientos de operación como a los de datos.

4.4.2.6 Rastro

El modelado de un rastro en Z está limitado por dos razones. En primer lugar no hay manera de registrar las acciones de un objeto, y, en segundo lugar, no existe distinción semántica entre acciones internas y acciones observables como se ve en la nota de 4.4.1.3.

4.4.2.7 Tipo (de una < *X* >)

Un objeto, una interfaz o una acción pueden tener muchos tipos de ODP, diferentes entre sí. Los tipos de ODP corresponden a conjuntos en Z, en donde el predicado caracterizador viene dado por la condición de miembro del conjunto.

4.4.2.8 Clase (de < *X* >)

El conjunto de todas las < *X* > tal que el predicado de miembro del conjunto, es decir el tipo de ODP, es verdadero.

4.4.2.9 Subtipo/supertipo

Los subtipos y supertipos en ODP corresponden, respectivamente, a los subconjuntos y superconjuntos en Z.

4.4.2.10 Subclase/superclase

Las subclases y superclases en ODP corresponden, respectivamente, a las relaciones de subconjunto y superconjunto en Z.

4.4.2.11 Plantilla de < *X* >

- **Plantilla de Objeto:** Fragmentos de una especificación que representa un estado, tienen una identidad única (inmutable) que puede ser referenciada y un conjunto asociado de *esquemas de operación* que actúan en ese estado. Si la plantilla de objeto es una plantilla genérica, la forma precisa de la plantilla sólo se dará cuando se dé el tipo de los parámetros parametrizadores.
- **Plantilla de Interfaz:** Un conjunto de *esquemas de operación* derivados del texto Z que representa una plantilla de objeto del modo descrito en la interpretación de la interfaz (véase 4.4.1.4). Si la plantilla de objeto es un plantilla genérica, la forma precisa de la plantilla de interfaz sólo se dará cuando se dé el tipo de los parámetros parametrizadores. Las plantillas de interfaz pueden combinarse utilizando las operaciones Z para la combinación de esquemas.
- **Plantilla de Acción:** Un *esquema de operación*. Las plantillas de acción pueden combinarse utilizando las operaciones Z para la combinación de esquemas. Si la plantilla de acción es una plantilla genérica, la forma precisa de la plantilla de acción sólo se dará cuando se dé el tipo de los parámetros parametrizadores.

4.4.2.12 Firma de interfaz

El conjunto de plantillas de acción asociadas con las interacciones de una interfaz.

NOTA – Debe señalarse que el texto de la Rec. UIT-T X.902 | ISO/CEI 10746-2 se refiere a una firma de interfaz como un conjunto de plantillas de acción asociadas con las interacciones de una interfaz. Puesto que una plantilla de acción representa las características comunes de una colección de acciones, es probable que esa definición sea incorrecta. Esto es, una plantilla de acción probablemente incluya información tanto semántica como sintáctica. No obstante, las interpretaciones comunes de una signatura de interfaz se refieren sobre todo al nivel sintáctico. Si se considera entonces una noción sintáctica de firma de interfaz, ello corresponde al conjunto de plantillas de acción normalizadas asociadas con las interacciones de una interfaz con todas las variables de no entrada/salida declaradas en la firma del *esquema de operación* que se cuantifica existencialmente en la parte predicado del *esquema de operación*.

4.4.2.13 Instanciación (de una plantilla < *X* >)

- **de una Plantilla de Objeto:** Especificación de los valores iniciales de las variables referenciadas en la plantilla de objeto. A menudo esto está previsto explícitamente en Z mediante un esquema o conjunto de esquemas de inicialización. Los valores de estas variables después de la inicialización deben corresponder a un estado válido, es decir, un estado que satisfaga cualesquiera *invariantes* que pudieran estar presentes.

  NOTA – Estos *invariantes* pueden hacer referencia a otros objetos.

- **de una Plantilla de Interfaz:** Construcción de los fragmentos de plantilla de interfaz de la especificación Z mediante especificación de valores de los parámetros genéricos y suministrando predicados apropiados sobre las variables referenciadas en la plantilla de interfaz.

- **de una Plantilla de Acción:** La realización de una operación especificada en un *esquema de operación*.

4.4.2.14 Rol (cometido)

Un rol puede ser representado por un nombre que identifique un determinado objeto, por ejemplo, mediante un identificador que figura en el *esquema de estado* asociado con el objeto. El nombre se puede utilizar a continuación en un *esquema de alineación de trama* para promover las operaciones del objeto especificado de modo que se efectúe el sistema (la especificación) como un todo.

NOTA – Puede que esta descripción no capte la retención del texto asociado de la Rec. UIT-T X.902 | ISO/CEI 10746-2.

4.4.2.15 Creación (de una < *X* >)

- **de un Objeto:** La creación de un objeto en Z se efectúa proporcionando un estado inicial válido para el texto Z asociado con la plantilla de objeto. Esto es, proporcionando una vinculación de las variables dadas en el *esquema de estado* del objeto a los valores iniciales que mantienen. A menudo esto está previsto explícitamente en Z mediante un esquema de inicialización. En este caso la acción de creación viene representada por la realización de la operación dada en el esquema de inicialización.

- **de una Interfaz:** La creación de una interfaz en Z está ligada inherentemente a la creación de objetos. Esto es, cuando se inicializa el texto asociado con una plantilla de objetos, se inicializan también cualesquiera plantillas de interfaz que pudieran estar presentes.

En la creación, es necesario asegurarse de que la identidad del objeto que se crea es única dentro de la especificación. Esto puede hacerse mediante un *esquema de alineación de trama* con un predicado apropiado que garantice que las identidades de todos los objetos creados son únicas. El *esquema de alineación de trama* se puede utilizar a continuación para promover la inicialización de un objeto de modo que se efectúe la especificación como un todo.

NOTA –

a) El texto que aquí se da implica la creación de un objeto, porque se da un esquema de inicialización como parte de una especificación. Sin embargo, en Z no hay noción de la especificación que realmente aplica este esquema de inicialización pues en si mismo Z no es ejecutable. Así es de hecho la introducción de un objeto, es decir, el objeto es instanciado por un mecanismo no abarcado por el modelo. Parece como si la noción de inicialización de una especificación Z fuese parcialmente creación (puesto que se da un esquema de inicialización) y parcialmente introducción (puesto que el modelo no abarca la aplicación del esquema de inicialización).

b) Normalmente ocurre que, a la aplicación de un esquema de inicialización sigue una obligación de prueba, para tener la seguridad de que el objeto se encuentra en un estado inicial válido.

4.4.2.16 Introducción (de un objeto)

Véase creación (de un objeto) (véase 4.4.2.15).

4.4.2.17 Supresión (de una < *X* >)

Es posible tener una representación abstracta de la supresión cuando se utilice un *esquema de alineación de trama*. La supresión se puede modelar entonces como la promoción de una operación con la que eliminar un estado y una identidad de objeto del sistema como un todo.

También podría ocurrir que se modelara una forma de supresión basada en la inactividad. Por ejemplo, un objeto cuyos comportamientos futuros asociados puede que no ocurran más debido a la violación de *invariantes*, podría considerarse suprimido en cierto modo. Sin embargo, esta forma de supresión quizá no refleje de manera precisa la definición dada en la Rec. UIT-T X.902 | ISO/CEI 10746-2, es decir, que no hay destrucción en sentido estricto.

4.4.2.18 Instancia de un tipo

Una instancia de un tipo ODP en Z se representa mediante un elemento del conjunto cuyos miembros satisfacen el predicado, es decir, el tipo ODP. Dada una noción de tipo más específica, por ejemplo tipo de plantilla, una instancia de un tipo de objeto o interfaz corresponde a la inicialización del texto Z (o una ampliación del mismo) que representa el objeto o la interfaz que se considera, de tal modo que existe un estado inicial válido para ese objeto o esa interfaz. El predicado caracterizador viene dado aquí por el texto de la especificación y los predicados asociados sobre posibles vinculaciones legales (*invariante*), que debe ser satisfecho por el esquema de inicialización, para que pueda clasificarse como una instancia del tipo de objeto o interfaz.

Puesto que un *esquema de operación* da las características de un tipo de acción, una instancia de tipo de acción viene dada por la ocurrencia del *esquema de operación* o por la ocurrencia de otro *esquema de operación* que sea una ampliación de ese *esquema de ampliación*, es decir, la ampliación incluye los rasgos caracterizadores del tipo de acción.

4.4.2.19 Tipo de plantilla (de una < *X* >)

En Z, un tipo de plantilla de objeto o interfaz es un predicado de que un esquema de inicialización deja el objeto y las interfaces asociadas en un estado inicial válido. Así pues, todas las variables de estado deberían estar vinculadas y todos los predicados necesarios (*invariantes*) deberían ser satisfechos. A menudo, la verificación de un tipo de plantilla de objeto o interfaz requiere que se efectúe una prueba.

Un tipo de plantilla de acción corresponde a un predicado de que una plantilla de acción, dada por un *esquema de operación*, puede ocurrir, es decir, es una instanciación de un *esquema de operación* determinado. Así pues, se prevé que todas las instanciaciones satisfagan los predicados sobre vinculaciones legales de variables, dados en los predicados del *esquema de operación*.

4.4.2.20 Clase de plantilla (de una < *X* >)

Una clase de plantilla de una < *X* > es el conjunto de todas las < *X* > que son instancias de esa plantilla < *X* >, en donde < *X* > puede ser un objeto, una interfaz o una acción.

4.4.2.21 Clase derivada/clase de base

Dadas dos plantillas A y B en Z, en donde A es una modificación incremental de B y las instancias de A y B están en la relación clase derivada/clase de base, respectivamente, las modificaciones incrementales de B para producir A pueden incluir:

- la adición o supresión de parámetros de estado;
- la adición, supresión o modificación de operaciones; o
- el fortalecimiento o debilitamiento de *invariantes*.

4.4.2.22 Invariante

Un predicado que una especificación requiere siempre que sea verdadero. Z permite que los *invariantes* se representen directamente en esquemas y *descripciones axiomáticas*. A menudo, los *invariantes* imponen restricciones a las posibles vinculaciones que las variables de los esquemas o las *descripciones axiomáticas* puedan adoptar. Por eso se utiliza con frecuencia un *invariante* para restringir los posibles comportamientos dados en una especificación.

4.4.2.23 Precondición

La condición sobre el estado del sistema, antes de la ocurrencia de una operación definida por un *esquema de operación*, y sus entradas, de modo que exista un posible estado después de la realización de la operación y salidas que satisfagan las *postcondiciones*. Z permite que las *precondiciones* se representen directamente.

4.4.2.24 Postcondición

Un predicado que describe el conjunto de estados en los que se puede encontrar un determinado sistema después de realizar una operación definida por un *esquema de operación*. Z permite que las *postcondiciones* se representen directamente.

4.5 Semántica arquitectural en ESTELLE

ESTELLE es un FSL normalizado (véase ISO/CEI 9074). En la Norma ISO se incluye material didáctico. Existen diversos instrumentos con los que sustentar la simulación así como la implementación distribuida de las especificaciones de ESTELLE.

ESTELLE se basa en máquinas automáticas de estados finitos ampliados. Un sistema se modela mediante un conjunto estructurado jerárquicamente de instancias de módulo que comunican en un modo asíncrono vía intercambio de mensajes por canales. La sintaxis del lenguaje y la definición de los tipos de datos y variables se basa en ISO Pascal.

En una especificación ESTELLE, la interfaz visible externamente de un módulo se define en el *encabezamiento del módulo*, mientras que el *cuerpo del módulo* describe la estructura interna y el comportamiento del módulo. Las *instancias de módulo* definen *puntos de interacción* a través de los cuales pueden enviar y recibir mensajes. *Dos puntos de interacción* pueden ser conectados si han sido definidos para los roles opuestos de la misma *definición de canal*. Una *definición de canal* contiene dos *roles* para los respectivos extremos del *canal*. Para cada *rol* se definen los mensajes (llamados en ESTELLE *interacciones*) que pueden ser enviados. *Esa definición de interacción* consta de un nombre junto con un conjunto de parámetros. A cada *punto de interacción* se le asigna una cola en la que se almacenan los mensajes entrantes. Una *instancia de módulo* puede tener también una cola común que es compartida por algunos de los *puntos de interacción* o por todos ellos.

La estructura de una especificación ESTELLE es dinámica, es decir, se pueden instanciar y liberar *instancias de módulo* y se pueden conectar puntos de interconexión y desconectarlos dinámicamente. Las instancias de módulo de una especificación ESTELLE sustentan un orden jerárquico fuerte. Cada instancia de módulo puede instanciar o liberar *instancias de módulo* vástago, o conectar o desconectar sus *puntos de interacción*. La única manera que tiene una *instancia de módulo* de acceder a instancias hermanas (o a otras *instancias de módulo* que no sean sus propios vástagos) es mediante el intercambio de *interacciones* por *canales*.

ESTELLE se elaboró inicialmente para especificar servicios y protocolos de comunicación. Soporta la encapsulación pero no contiene las características orientadas al objeto de herencia o subtipificación. A pesar de eso, ESTELLE permite la expresión de la mayoría de los conceptos relativos al ODP. Las especificaciones de ESTELLE son de fácil lectura y, puesto que ESTELLE es una técnica constructiva, se presta bien a la simulación y la implementación.

En las subcláusulas que siguen se muestra cómo pueden expresarse los conceptos básicos del ODP utilizando ESTELLE.

En el texto, se utilizan letras *cursivas* para indicar conceptos ESTELLE definidos en la Norma ISO. Se señala que ESTELLE utiliza la noción de *interacción* para indicar los mensajes intercambiados entre *instancias de módulo* comunicantes.

4.5.1 Conceptos básicos de modelado

4.5.1.1 Objeto

Un objeto es modelado en ESTELLE por una *instancia de módulo*.

4.5.1.2 Entorno (de un objeto)

La parte de la especificación que no forma parte de la *instancia de módulo*; sobre todo la *instancia progenitora* y las otras instancias que están conectadas a los *puntos de interacción* de las instancias de módulo vía *canales*.

4.5.1.3 Acción

En ESTELLE se representa una acción mediante la ejecución de una *cláusula WHEN*, la ejecución de una declaración de acción, una *transición* completa o la ejecución de un *procedimiento*. Son posibles las siguientes declaraciones de acción:

- *salida,*
- *iniciar,*
- *conectar,*
- *anexar,*
- *liberar,*
- *desconectar,*
- *separar,*
- *declaración de asignación.*

Hay varios tipos de **interacciones**. La ejecución de una declaración *salida* es una interacción, lo mismo que la ejecución de una *cláusula WHEN*. También, la secuencia de acciones formada por la *salida* de una *interacción* a través de un *punto de interacción* y su consumo subsiguiente por la ejecución de una *cláusula WHEN* se puede considerar como una interacción. Otro tipo de interacción es la actualización de una *variable exportada*. Las demás acciones son **internas**.

NOTA – Puesto que los *canales* de ESTELLE poseen infinitas colas, el entorno está siempre preparado para participar en *interacciones*.

4.5.1.4 Interfaz

En ESTELLE hay dos clases de interfaces:

- la primera está formada por el conjunto de todas las *interacciones* definidas para el *rol* asignado (en la *definición de canal* correspondiente) en un *punto de interacción externa* de un objeto;
- la segunda está formada por el conjunto de todas las *variables exportadas* de un objeto.

En el primer caso, el conjunto de interacciones (que constituyen la interfaz) contiene el conjunto de declaraciones *salida* y/o *cláusulas WHEN* que el objeto ejecuta en el *punto de interacción*. En el segundo caso, el conjunto de interacciones consta de todas las declaraciones que leen o escriben las *variables exportadas*. En esta clase de interfaz, las interacciones sólo son posibles entre una *instancia de módulo* y su *instancia progenitora*.

4.5.1.5 Actividad

Por lo general, una actividad no puede ser denotada explícitamente ya que puede abarcar varios objetos. Una actividad es una cierta secuencia relacionada de acciones, por ejemplo, una *transición*, o un *procedimiento* o una *función*, si las declaraciones simples se consideran acciones.

4.5.1.6 Comportamiento (de un objeto)

El comportamiento de un objeto viene determinado por el conjunto de todas las *transiciones* de ese objeto. En las *cláusulas de transición* (por ejemplo, la *cláusula FROM*, la *cláusula PROVIDED*) se definen las constricciones a las circunstancias en las que las acciones especificadas se pueden realizar. Un objeto puede mostrar un comportamiento no determinístico.

4.5.1.7 Estado (de un objeto)

El estado de un objeto tiene los siguientes aspectos:

- el *estado de control* de las *instancias de módulo*;
- el contenido de las colas de los *puntos de interacción*;
- los valores de las *variables exportadas* e internas, y los parámetros formales de la *instancia de módulo*;
- el estado de las *instancias* vástago existentes, su estructura de conexión y sus *variables exportadas*.

Estos aspectos juntos determinan el conjunto de todas las secuencias y acciones (*transiciones*) en las que la *instancia de módulo* puede tomar parte.

4.5.1.8 Comunicación

La información es transportada entre objetos de dos maneras diferentes:

- a través de la *salida* y subsiguiente recepción de una *interacción* (una secuencia de acciones forma una interacción);
- a través de la lectura y actualización de una *variable exportada*.

4.5.1.9 Ubicación en el espacio

Las acciones se realizan dentro de *instancias de módulo* o en los *puntos de interacción* de las *instancias de módulo*. Así pues, la ubicación en el espacio de una acción corresponde a la ubicación en el espacio de la *instancia de módulo* asociada.

4.5.1.10 Ubicación en el tiempo

En ESTELLE, el tiempo sólo se representa en las *cláusulas DELAY* asociadas posiblemente con *transiciones*. La única hipótesis establecida con respecto al tiempo es que aumenta uniformemente a medida que avanza la ejecución.

4.5.1.11 Punto de interacción

Un punto de interacción viene representado por un *punto de interacción* o por el conjunto de todas las *variables exportadas* de un objeto.

4.5.2 Conceptos de especificación

La expresión de los conceptos de especificación dada en ESTELLE resulta difícil porque su soporte de conceptos orientados al objeto es limitado.

4.5.2.1 Composición

- **de objetos:** Una *instancia de módulo* puede estar compuesta de un conjunto de *instancias de módulo* vástago. Al nivel máximo, una especificación puede estar compuesta por un conjunto de *instancias de sistema*;
- **de comportamientos:** Cuando una *instancia de módulo* es una composición de *instancias* vástago, su comportamiento es:
  - compuesto en paralelo, entrelazado, si al *módulo* progenitor se le atribuye *actividad* o *actividad de sistema*; o
  - compuesto en paralelo, síncrono, si al *módulo* progenitor se le atribuye *proceso* o *proceso de sistema*.

Al nivel máximo, el comportamiento de las *instancias* vástago es compuesto en paralelo, si la especificación está compuesta por un conjunto de instancias de sistema.

4.5.2.2 Objeto compuesto

Una *instancia de módulo* que se describe mediante un conjunto de *instancias de módulo* vástago.

4.5.2.3 Descomposición

- **de objetos:** La especificación de un determinado objeto es una composición.
- **de comportamientos:** La especificación de un determinado comportamiento es una composición.

4.5.2.4 Compatibilidad en comportamiento

No hay ninguna manera directa de expresar la compatibilidad en comportamiento en ESTELLE. Sin embargo, la base semántica del lenguaje con respecto a los sistemas de transición permite la definición de la compatibilidad en comportamiento y su verificación.

4.5.2.5 Refinamiento

Un objeto puede ser refinado subestructurándolo en *instancias* vástago cooperantes.

4.5.2.6 Rastro

Los rastros se pueden obtener a partir de la interpretación dinámica (ejecución/simulación) de una especificación ESTELLE.

4.5.2.7 Tipo (de una < *X* >)

No hay ninguna manera de formular predicados explícitamente en ESTELLE.

4.5.2.8 Clase (de una < *X* >)

No se soporta.

4.5.2.9 Subtipo/supertipo

No se soporta.

4.5.2.10 Subclase/superclase

No se soporta.

4.5.2.11 Plantilla < *X* >

Un plantilla de objetos se representa mediante una *definición de cuerpo de módulo* junto con la correspondiente *definición de encabezamiento de módulo*. Las plantillas de acción son:

- *salida;*
- *iniciar;*
- *liberar;*
- *conectar;*
- *desconectar;*
- *anexar;*
- *separar;*
- *declaración de asignación*; y
- *cláusula WHEN.*

Puesto que hay dos clases de interfaces, una plantilla de interfaz se da:

- mediante la correspondiente *definición de canal* (para conjuntos de *interacciones* en un *punto de interacción*). En este caso, el comportamiento de la interfaz asociada se puede especificar mediante una *instancia de módulo* vástago que es *instanciada* y anexada al *punto de interacción* cuando se crea la interfaz. Si tal es el caso, la plantilla de interfaz contiene el *encabezamiento de módulo* y la *definición de cuerpo* de este módulo;
- en la *definición de encabezamiento de módulo* del objeto (para *variables exportadas*).

4.5.2.12 Firma de interfaz

Una firma de interfaz se representa de las dos maneras siguientes:

- en las definiciones de *interacciones* contenidas en la interfaz (para un conjunto de *interacciones* en un *punto de interacción*);
- en las acciones de asignación para las *variables exportadas*.

4.5.2.13 Instanciación (de una plantilla < *X* >)

Una instanciación de un objeto es una *instancia de módulo* de la *definición de módulo* correspondiente. Una instanciación de una interfaz es un *punto de interacción* particular (posiblemente con una *instancia de módulo* vástago específica anexada a él, representando el comportamiento de la interfaz) o el conjunto de *variables exportadas* de una *instancia de módulo* particular.

4.5.2.14 Rol (papel)

Con cada declaración de *punto de interacción* está asociado un *rol*. Los *puntos de interacción* que se han de conectar deben tener *roles* opuestos en la misma *definición de canal*. ESTELLE soporta también la especificación de *cuerpos de módulo* diferentes para el mismo *encabezamiento de módulo*. De esta manera es posible elegir entre diferentes comportamientos cuando un objeto es instanciado. Las interfaces del objeto son las mismas con independencia del *cuerpo de módulo* seleccionado.

4.5.2.15 Creación (de una < *X* >)

Los objetos se crean mediante la acción *iniciar*. Las interfaces se crean implícitamente, cuando se crea el objeto. No hay creación dinámica de interfaces. Sin embargo, la creación de una interfaz como *punto de interacción* se puede modelar seleccionando un *punto de interacción* (de un conjunto de puntos de interacción) con un *canal* apropiado y un *rol* asociado. Si se especifica un *módulo* vástago para representar el comportamiento de la interfaz, una *instancia* del *módulo* es *instanciada* y asociada al *punto de interacción* proporcionado.

4.5.2.16 Supresión (de una < *X* >)

Un objeto se suprime explícitamente mediante la acción *liberar*. Las interfaces se suprimen implícitamente junto con el objeto. Si se modela la creación dinámica de interfaces como se indica más arriba (véase 4.5.2.15), la supresión de una interfaz corresponde a la *liberación* de la *instancia de módulo vástago* anexada (si hay alguna) y a la marcación del *punto de interacción* como no representativo de una interfaz.

4.5.2.17 Introducción (de un objeto)

No hay ninguna manera de introducir objetos. Por lo general, los mecanismos no abarcados por el modelo se pueden captar utilizando procedimientos *primitivos*, *procedimientos externos* o *funciones*.

4.5.2.18 Instancia de un tipo

Puesto que en ESTELLE no hay subtipificación/subclasificación, las instancias de una plantilla se dan mediante las instanciaciones de esa plantilla.

4.5.2.19 Tipo de plantilla (de una < *X* >)

Para un objeto, el predicado que se puede formular es que un objeto es una instancia de la *definición de módulo* correspondiente. Una interfaz (un conjunto de *interacciones* en un *punto de interacción*) es una instancia de la *definición de canal* correspondiente. Una interfaz representada por el conjunto de todas las *variables exportadas* es una instancia de la *definición de encabezamiento de módulo* correspondiente. Una acción es una instancia de la plantilla de acción correspondiente.

NOTA – Lo que se puede formular en ESTELLE es que un objeto, interfaz o acción es una instanciación de una plantilla determinada. Puesto que el concepto de subclase no se soporta verdaderamente, sólo las instanciaciones de una plantilla se pueden identificar como tales, pero no las instancias.

4.5.2.20 Clase de plantilla (de una < *X* >)

La clase de plantilla de un objeto es el conjunto de todas las *instancias* del mismo módulo. La clase de plantilla de una interfaz es el conjunto de todos los *puntos de interacción* definidos utilizando la misma *definición de canal* y el mismo *rol*.

4.5.2.21 Clase derivada/clase de base

No se soporta.

4.5.2.22 Invariante

No hay ninguna manera de formular invariantes explícitamente en ESTELLE.

4.5.2.23 Precondición

Las precondiciones de la ejecución de una *transición* se formulan en las *cláusulas de transición* de la transición. Las precondiciones de una acción dentro de un *bloque de transición* vienen dadas por las precondiciones de la transición junto con las acciones contenidas en el *bloque de transición* que precede a la acción de que se trata.

4.5.2.24 Postcondición

La postcondición de una transición se define mediante la *cláusula TO* de la transición junto con las acciones del *bloque de transición*.

SERIES DE RECOMENDACIONES DEL UIT-T

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