Tema 5: Especificación de Requisitos

Ingeniería de Requisitos

Raquel Martínez España

Grado en Ingeniería Informática

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

Objetivos

Este tema tiene como objetivos principales:

- Entender las diferentes formas en qué puede describirse un requisito.
- Comprender cómo los requisitos se pueden organizar en un documento de requisitos del software.
- Conocer las principales características de un buen documento de requisitos.
- Conocer el documento de requisitos según el estándar IEEE 830.
- Comprender por qué las técnicas de especificación formal ayudan a descubrir problemas en los requisitos del sistema.
- Conocer el uso de técnicas algebraicas de especificación formal para definir las especificaciones de interfaz.
- Entender cómo las técnicas formales basadas en modelos formales se usan para especificar el comportamiento.

Especificación de Requisitos

Formalmente podemos definir la especificación de requisitos como la producción de un documento que puede ser revisado, evaluado y aprobado.

Siguiendo el glosario de términos estándar de Ingeniería del Software del IEEE (IEEE90), la especificación de requisitos se refiere a la documentación de los requisitos esenciales (funciones, rendimiento, restricciones de diseño y atributos) del software y de sus interfaces externas

La especificación de requisitos tiene diferentes objetivos según el punto de vista del usuario y del desarrollador. Mientras que para el usuario o cliente el objetivo de la especificación de requisitos es definir la necesidad de lo que se quiere y lo que no se quiere hacer, para el desarrollador la especificación de requisitos tiene varios objetivos, entre los que podemos destacar:

- Permitir la comunicación entre clientes, usuarios y desarrolladores: si está bien hecho el usuario estará satisfecho con el producto final. Este acuerdo se realiza al principio, no en las pruebas de aceptación.
- Permitir iniciar la actividad de diseño: los requisitos determinan las características y funcionalidades que el futuro sistema software deberá poseer.
- Soportar las actividades de pruebas del sistema: si la descripción del requisito permite crear escenarios de prueba se puede mostrar que el sistema satisface los requisitos. Si es ambiguo o inconsistente o inestable las pruebas son imposibles.

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

- Controlar la evolución del sistema: manteniendo la especificación actualizada se puede controlar la evolución el sistema: si se añaden nuevos requisitos o si ya se han implementado los que estaban identificados.

Para considerar que una especificación de requisitos es completa y correcta, el documento de requisitos debe contener información acerca del problema, de la interfaz externa del sistema con su entorno (software, hardware, usuarios, puertos de comunicación), incluir las propiedades y comportamiento del sistema, las restricciones de diseño y fabricación del producto, las descripciones acerca de cómo el sistema ayudará a sus usuarios a realizar mejor sus tareas y restricciones relacionadas tanto con la tecnología que será utilizada en el sistema (protocolos, SSOO, etc..), así como restricciones acerca de las propiedades emergentes del sistema, es decir, los requisitos no funcionales.

Téngase en cuenta, que un documento de requisitos no debe incluir información acerca de requisitos del proyecto como la planificación, costes, fases, hitos, etc. De la misma forma, tampoco debe incluir detalles sobre el diseño ni planes de garantía del producto. Esta información queda, por tanto, fuera del documento de requisitos.

Pero, ¿cuál es la mejor forma de escribir los requisitos? Desgraciadamente, no hay una forma perfecta de escribir los requisitos, aunque podemos ayudarnos de algunas guías.

En general, la técnica más utilizada para escribir requisitos es el lenguaje natural utilizando expresiones del tipo: “el sistema hará X…”, “se facilitará Y…”, etc. En numerosas ocasiones el lenguaje natural se complementará con diagramas y notaciones formales. El uso de estas notaciones formales dependerá de quien lee o escribe los requisitos, ya que hay que tener en cuenta que el usuario o el cliente deben participar activamente en la elaboración de los requisitos, por tanto éstos deben estar escritos en un lenguaje que el usuario o cliente puedan entender.

Así, en sistemas simples podemos utilizar sólo texto de manera que sea más sencillo de entender. Sin embargo, si la especificación es grande y/o compleja, utilizar sólo texto puede hacerse inmanejable. Por ello, para sistemas de tamaño complejidad media/grande es más recomendable el uso combinado de texto y técnicas (ya sean intuitivas o formales). En general, el uso de sólo técnicas no será recomendable, ya que esto dificulta su legibilidad y rastreo.

Normalmente la definición de los requisitos se realiza en positivo, tratando de recoger las necesidades que se deben satisfacer con el sistema. Sin embargo, en algunos casos establecer lo que el sistema no debe hacer es tan importante como establecer lo que debe hacer. Esto es especialmente importante para sistemas críticos.

El documento de Requisitos

El documento de requisitos es el modo habitual de guardar y comunicar requisitos. Por “Documento” se entiende cualquier medio electrónico de almacenamiento y distribución (procesador de textos, base de datos, herramienta de Gestión de Requisitos, etc.)

Se considera buena práctica utilizar, al menos, dos documentos, a distinto nivel de detalle:

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

- Documento de Requisitos de Usuario (en adelante DRU). En inglés, User Requirements Document, URD.
- Especificación de Requisitos Software (en adelante ERS). En inglés, Software Requirements Specification, SRS.

Documento de Requisitos de Usuario

Se trata de la fase de definición del problema y es un proceso iterativo donde el usuario es el máximo responsable. Su objetivo es obtener una definición de lo que el usuario espera del sistema computacional. Las entradas para la realización de este documento serán los resultados de entrevistas, encuestas, estudios, ejercicios de prototipado, etc. realizados en las fases de captura y análisis. La salida será el documento de requerimientos de usuario (DRU), donde se describen todos los requerimientos con claridad y consistencia

Las fases a llevar a cabo para la elaboración del DRU consistirán en i) especificar los requisitos del usuario, ii) describir los requisitos en términos de sus atributos (identificador, necesidad, prioridad, estabilidad, fuente, claridad y verificabilidad), iii) organizar los requisitos en base a la categoría a la que pertenezcan y iv) revisar y comprobar las salidas de la fase.

Documento de Requisitos de Software

Se trata de la fase de análisis del problema, donde se debe adoptar un método de análisis y en la que el desarrollador es el máximo responsable. El objetivo aquí es obtener una definición del software que se va a construir. En este caso, las entradas consistirán en el documento de requisitos del usuario (DRU) y la salida será el documento de requisitos del software (DRS), donde se representa la visión que el desarrollador tiene del software y la descripción del ámbito del software.

Las actividades a realizar para la elaboración del DRS serán la descripción de los requisitos en términos de sus atributos asegurando la completitud y consistencia, y evitando la duplicidad de los mismos y la organización de los requisitos en base a la categoría a la que pertenezcan. Por último, esto debe ser revisado.

Como vemos, existen diferencias entre el DRU y el DRS. En el DRU se escribe desde el punto de vista del usuario/cliente/interesado. Normalmente los requisitos de usuario, contenidos en la DRU, no poseen demasiado nivel de detalle y se incluye la descripción del problema actual (razones por las que el sistema de trabajo actual es insatisfactorio) y las metas que se espera lograr con la construcción del nuevo sistema.

Por otra parte, el DRS desarrolla mucho más los contenidos de la DRU. Los requisitos del software contenidos en el DRS son, por tanto, más detallados. Contiene la respuesta a la pregunta ¿Qué características debe poseer un sistema que nos permita alcanzar los objetivos, y evitar los problemas, expuestos en la DRU?

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

La diferencia entre la DRU y la DRS no es que la DRU emplee lenguaje natural y la ERS emplee modelos o notaciones formales. Ambas pueden utilizar todo tipo de notaciones. La diferencia entre ambos documentos es más de nivel de detalle. Por ejemplo, el siguiente requisitos en el DRU: "*El software debe acceder a ficheros externos creados por otras herramientas*", quedaría detallado en el DRS como:

1.1. El usuario debe poder elegir el tipo de fichero externo

1.2. Cada tipo de fichero externo se puede asociar con la herramienta que produce ese tipo de fichero

1.3. Cada tipo de fichero externo puede ser representado por un icono en la barra de herramientas

1.4. Este icono puede ser definido por el usuario

1.5. Cuando el usuario selecciona uno de estos iconos el efecto es aplicar la herramienta asociada a ese tipo de fichero

Debemos tener en cuenta que el DRS, tiene diferentes tipos de lectores:

- Clientes. Para comprobar que satisface sus necesidades y para hacer cambios a los requisitos.
- Gestores. Para preparar una oferta por el sistema y planificar el proceso de desarrollo.
- Ingenieros de Desarrollo. Para comprender el sistema a desarrollar.
- Ingenieros de Pruebas. Para elaborar pruebas de validación del sistema.
- Ingenieros de Mantenimiento. Para facilitar la comprensión del sistema y las relaciones entre sus partes.

Características de una buena ERS (DRS)

Según el estándar IEEE 830, el documento ERS tiene que tener la calidad como objetivo. Sin embargo, la calidad es difícil de cuantificar. Por ello el IEEE establece 24 características que una ERS de calidad debería poseer:

1. **No ambigua**: un requisito ambiguo se presta a distintas interpretaciones. Cada característica del producto final debe ser descrita utilizando un término único. Si un término tiene distintos significados en distintos contextos se debe incluir un glosario
2. **Completa.** Si todo lo que se supone que el software debe hacer está incluido en la ERS. Se describen todas las posibles respuestas a todas las posibles entradas y en todas las situaciones posibles, para datos válidos y no válidos. No contendrá secciones de tipo TBD

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

(en este caso quién, y cuándo debe completarlo). Las páginas deben estar numeradas; las figuras y tablas numeradas y referenciadas; las unidades de medida especificadas, el material de referencia descrito y debe contener todas las secciones del estándar utilizado. Esta característica es la más difícil de detectar.

3. **Correcta**. Si todo requisito de la ERS contribuye a satisfacer una necesidad real.

4. **Comprensible.** Si todo tipo de lectores (clientes, usuarios, desarrolladores, equipo de pruebas, gestores, etc.) entienden la ERS.

5. **Verificable.** Si para cada requisito existe un procedimiento de prueba finito y no costoso para demostrar que el futuro sistema lo satisface. Los requisitos son verificables dependiendo de la forma en que estén escritos. Un requisito puede no ser verificable si es ambiguo o si usa medidas no cuantificables, por ejemplo: normalmente, a menudo,…Algunos ejemplos para cuantificar requisitos no funcionales serían los descritos en la siguiente tabla:

Propiedad	Medida
Rendimiento	Transacciones procesadas por segundo Tiempo de respuesta a un evento Tiempo de refresco de pantallaKbytes
Tamaño	Nº de chips de RAM
Fácil de uso	Tiempo de aprendizaje Nº de marcos de ayuda
Fiabilidad	Tiempo medio entre fallos Probabilidad de no estar disponible Tasa de ocurrencia de fallos Disponibilidad
Robustez	Tiempo de recuperación ante fallos Porcentaje de eventos causantes de fallo Probabilidad de corrupción de datos en un fallo
Portabilidad	Porcentaje de declaraciones dep. del objetivo Nº de sistemas objetivo

6. **Consistente Internamente**. Si no existen subconjuntos de requisitos contradictorios. En este sentido pueden existir distintos tipos de conflictos:

   - Conflicto de comportamiento: por ejemplo, “La luz se encenderá sólo si se presiona el botón” y “Cuando se suelta el botón la luz se enciende”
   - Conflicto de términos: si dos términos en contextos diferentes significan lo mismo.
   - Características en conflicto: por ejemplo “Todas las entradas se harán por selección de una opción en un menú desplegable” y “El lenguaje de comandos para el usuario consta de las siguientes órdenes,…”
   - Inconsistencia temporal: por ejemplo, “A ocurre cuando B está ocurriendo” y “A ocurre después de 10 segundos de que ocurra B”.

7. **Consistente Externamente**. Si ninguno de los requisitos está en contradicción con lo expresado en documentos de nivel superior.

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

8. **Realizable.** Si, dados los actuales recursos, la ERS es implementable.
9. **Concisa.** La ERS debe ser lo más breve posible, sin que esto afecte al resto de atributos de calidad.
10. **Independiente del diseño.** Si existe más de un diseño e implementación que realizan la ERS.
11. **Trazable.** Si cada requisito se puede referenciar de forma unívoca.
12. **Modificable.** Si los cambios son fáciles de introducir de forma completa y consistente. Esto implica la existencia de tabla de contenidos, índice y referencias cruzadas.
13. **Electrónicamente almacenada**
14. **Ejecutable/Interpretable/Prototipable/Animable**. Si existe una herramienta software que, recibiendo como entrada la ERS, realice un modelo ejecutable de la misma (aplicable tan sólo a ciertas notaciones).
15. **Anotada por importancia relativa:** "Obligatorio, Deseable u Opcional".
16. **Anotada por estabilidad relativa.** Será la probabilidad de cambio (p.ej. "Alta, Media o Baja").
17. **Anotada por versión.** Si un lector de la ERS puede determinar qué requisitos serán satisfechos por qué versión del producto
18. **No redundante.** Cada requisito se expresa en un solo lugar de la ERS (se podrán establecer excepciones si se aumenta la legibilidad del documento).
19. **A un nivel de detalle adecuado.**
20. **Precisa.** Una ERS es precisa si hace uso de valores numéricos para precisar las características del sistema. En la práctica es muy difícil.
21. **Reutilizable.** Si ciertas secciones de la ERS se pueden reutilizar.
22. **Trazada.** Si está claro el origen de cada requisito (quién o qué lo pide y porqué).
23. **Organizada.** Si el lector puede fácilmente encontrar la información buscada.
24. **Con referencias cruzadas.** Si se utilizan referencias entre requisitos relacionados (trazabilidad intra-ERS) o entre secciones relacionadas.

Existen diversos estándares para documentar los requisitos. Por ejemplo el del IEEE Std. 830, IEEE Std. 1362 (ConOps) o el de PSS-05 de la Agencia Espacial Europea (ESA).

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

Uno de los más extendidos es el de ANSI/IEEE Std 830 (1998) (IEEE Recommended Practice for Software Requirements Specifications). Es la revisión del 830-1993, y se ha adaptado al ISO/IEC 12207 (procesos del ciclo de vida del software). Este estándar describe el contenido y cualidades de un buen documento SRS y propone diversas estructuras para el documento dependiendo de la forma en que se agrupan los requisitos.

La estructura definida por este estándar es la siguiente:

1. INTRODUCCIÓN
 1.1. Propósito
 1.2. Ámbito
 1.3. Definiciones, acrónimos y abreviaturas
 1.4. Referencias
 1.5. Visión general del resto del documento
2. DESCRIPCIÓN GENERAL
 2.1. Perspectiva del producto
 2.2. Funciones del sistema
 2.3. Características de los usuarios
 2.4. Restricciones generales
 2.5. Suposiciones y dependencias
3. REQUISITOS ESPECÍFICOS
 3.1. Requisitos de interfaces externos
 3.2. Requisitos funcionales
 3.3. Requisitos de rendimiento.
 3.4. Atributos del sistema
 3.5. Otros requisitos.

La sección 1 explica el contenido del documento. La sección 2 define la relación del software con su entorno (2.1), da una visión general de las funciones del producto (2.2.), describe las características generales de los potenciales usuarios (2.3), indica limitaciones de hardware, interfaces, lenguaje de programación (2.4).

Por último la sección 3 contiene todos los requisitos del software. Para cada requisito, se debe incluir: identificar único, descripción de cada entrada (el estímulo) en el sistema, cada salida (la contestación) del sistema, y proceso, error. Por ejemplo:

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

Requisito: 1.1 Dar de Alta Tarjeta	
Entrada	Los datos del cliente (nombre, apellidos, dirección, teléfono y e-mail)
Proceso	Generar un nuevo identificador de tarjeta. Añadir en el almacén TARJETAS una entrada con este identificador, la fecha de alta (fecha actual), los puntos iniciales de la tarjeta (0) y los datos del cliente.
Salida	Una entrada nueva en el almacén TARJETAS y la tarjeta física que se entrega al cliente.
Error	El usuario deberá introducir obligatoriamente el nombre y los apellidos del cliente. De no hacerlo, el sistema emitirá el mensaje de error "Imposible dar alta tarjeta: no se han proporcionado el nombre y los apellidos del cliente". Asimismo, para poder ponerse en contacto con el cliente, será obligatorio introducir la dirección y el teléfono del cliente. De no hacerse, el sistema mostrará el mensaje de error "Imposible dar alta tarjeta: no se han proporcionado la dirección y el teléfono del cliente".

Otra forma de definir cada requisito sería especificando la validación de las entradas, la secuencia de las operaciones, los mensajes de error y control de excepciones, el efecto de los parámetros, la secuencia de las salidas y las fórmulas para la conversión de las entradas en salidas.

La sección 3 se puede organizar de diversas formas ya que diferentes tipos de sistemas requieren diferente énfasis en sus requisitos. Algunas alternativas en la especificación de requisitos específicos, según IEEE 830-1998, son:

1. Modos de operación
2. Clases de usuarios
3. Objetos (en sentido OO)
4. Características, o servicios (deseados externamente, que pueden requerir una secuencia de inputs para obtener el resultado deseado)
5. Estímulo/Respuesta
6. Jerarquía funcional: por cada elemento del DFD (flujos, procesos, entidades, ficheros) cada sección describe un DFD o descripción funcional equivalente (ligado a AE)
7. Múltiples organizaciones: combinando dos o más de las anteriores según necesidades de la organización

Por último, es importante destacar algunos problemas prácticos que podemos encontrarnos con el documento de requisitos. En primer lugar un problema común es el manejo de gran cantidad de requisitos, para ello la mejor solución es el uso de herramientas CARE.

Otro problema común es la no consideración de requisitos "óbvios". En este caso no se trata de especificar todo, si no de utilizar el sentido común. Otro fallo frecuente es el nivel de detalle en que debe expresarse un requisito (relacionado con el problema anterior, evitando detallar cuestiones que sean obvias). Dependerá del nivel de abstracción en que nos encontremos (requisitos de empresa, de usuario, de software). Para solucionar este problema debemos mantener la trazabilidad entre requisitos para ligar niveles de abstracción.

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

Por último, es importante determinar la correcta ubicación de un requisito en el apartado que mejor corresponda del documento. A veces un mismo requisito pertenece a varias categorías y puede por tanto ser asignado) a una o más secciones del documento. En estos casos utilizaremos referencias cruzadas o tratamiento basado en aspectos software.

Especificación formal

Podemos definir una especificación formal como una especificación escrita y aprobada de acuerdo con algunos estándares establecidos. Según el glosario de términos del IEEE podemos definirla como: "*una especificación escrita en una notación formal, a menudo para ser usada en pruebas de corrección*".

La especificación formal es una especificación expresada en un lenguaje cuyo vocabulario, sintaxis y semántica están formalmente definidas. Los lenguajes de especificación formal están basados en conceptos matemáticos como la teoría de conjuntos, lógica y álgebra. La especificación formal es parte de una colección de técnicas conocidas como "métodos formales". Los métodos formales incluyen la especificación formal, el análisis y prueba de la especificación y la verificación de programas.

El uso de métodos formales nos facilita encontrar errores en la especificación y permite producir una especificación no ambigua que facilita las pruebas y reduce su coste. Además introduce rigor e "ingeniería" en el desarrollo de sistemas, permite la documentación de requisitos, la comprensión de propiedades intrínsecas del software y, en general, facilita y ayuda en todo el ciclo de vida (diseño, implementación, tests y mantenimiento).

A pesar de las ventajas de los métodos formales, existen una serie de razones por las que su uso no está muy extendido. En primer lugar, las técnicas para desarrollo rápido de software no se llevan bien con la especificación formal. En segundo lugar, los métodos formales tienen un alcance limitado, no se adaptan para especificar las interfaces de usuario y la interacción con el usuario. Y por último tienen una escalabilidad limitada, se han usado con éxito en sistema críticos relativamente pequeños, pero el problema crece al no haber herramientas de soporte.

Por esto, la utilización de métodos formales se justifica para seguridad y fiabilidad muy altas. Por ejemplo en sistemas de control de tráfico aéreo, de control médico, sistemas espaciales, etc. Al estar basados en formalismos matemáticos es posible verificar la corrección, completitud y consistencia de los requisitos (de forma automática). Sin embargo, son difíciles de usar y comprender, lo que hace que la intervención del cliente disminuya al avanzar la especificación.

Existen básicamente dos enfoques para realizar especificación formal:

- Algebraico: en el que el sistema se especifica en términos de sus operaciones y las relaciones entre ellas.

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

- Basado en modelo: en el que el sistema se especifica en términos de un modelo de estado que se construye usando constructores matemáticos como conjuntos y sucesiones. Las operaciones se definen según cómo modifican el estado del sistema.

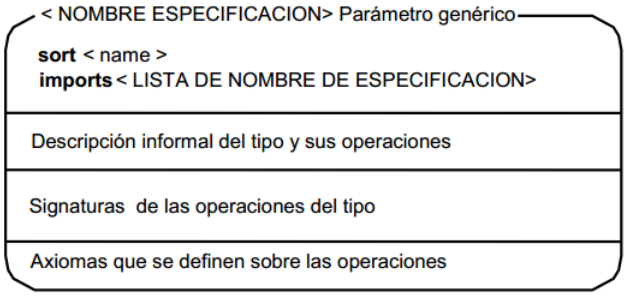
En los siguientes apartados se describen ambos enfoques.

Especificación algebraica

Se utiliza especialmente para la especificación de interfaces de subsistemas. La especificación de interfaces entre subsistemas permite el desarrollo independiente de los subsistemas y consiste en dar información sobre qué servicios están disponibles para otros subsistemas y cómo pueden ser utilizados.

Las interfaces se pueden definir como tipos abstractos de datos (TADs) o clases de objetos. El TAD se define especificando las operaciones del tipo más que la representación del tipo. Así, la especificación algebraica define el tipo abstracto de datos en términos de las relaciones entre las operaciones del tipo.

La estructura de una especificación algebraica tiene la siguiente forma:

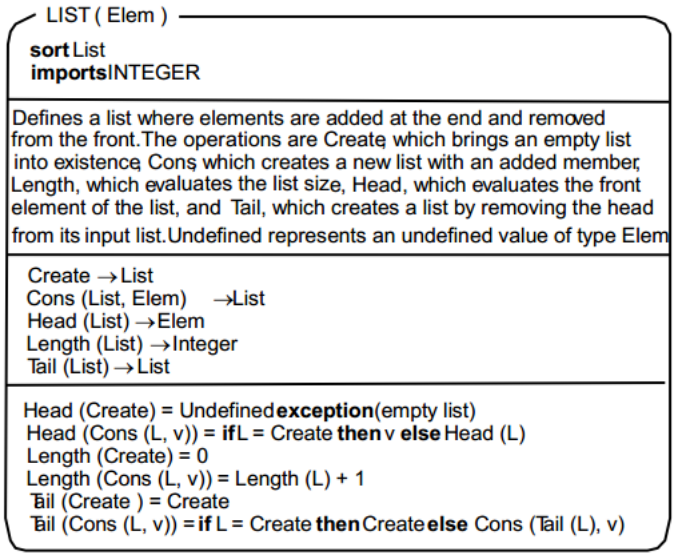


Dónde tenemos los siguientes componentes:

- Introducción. Define el nombre del tipo y declara otras especificaciones que se usan.
- Descripción. Describe informalmente las operaciones del tipo. La notación completa esta descripción con una sintaxis y semántica no ambiguas para las operaciones del tipo.
- Signatura. Define la sintaxis de las operaciones, número y clase de sus parámetros, y las clases de los resultados.
- Axiomas. Define la semántica de las operaciones mediante la definición de axiomas que caracterizan su comportamiento.

Siguiendo esta estructura podríamos, por ejemplo definir el TAD lista como:

María Teresa García Valverde - Tlf: (+34) 968 27 86 57



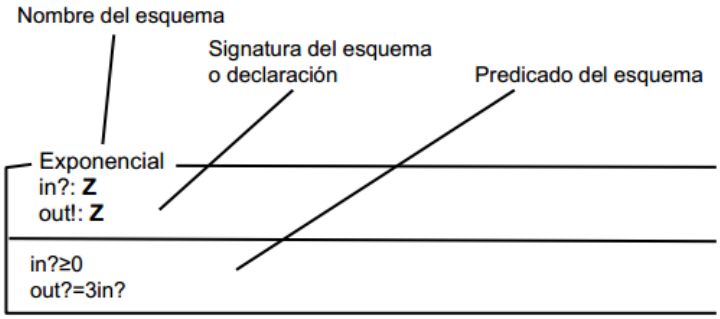
Especificación basada en modelos

La especificación basada en modelos permite la especificación del comportamiento del sistema. La especificación del sistema se expresa como un modelo de estados del sistema y las operaciones se especifican definiendo cómo afectan al estado del sistema.

Existen diversas notaciones para la especificación basada en modelos, como por ejemplo VDM (Jones, 1980;1986), B (Wordsworth, 1996) y Z (Hayes, 1987; Spivey,1992). En este tema nosotros nos centraremos en los aspectos más relevantes de la notación Z.

En Z los sistemas se modelan usando conjuntos y relaciones entre conjuntos. Las especificaciones se presentan como texto informal complementado con descripciones formales para aumentar la legibilidad. Las descripciones formales se incluyen como pequeños trozos fáciles de leer, que reciben el nombre de esquemas. Los esquemas introducen variables de estado y definen restricciones y operaciones en el estado.

Por ejemplo, un esquema en Z para función exponencial quedaría como:

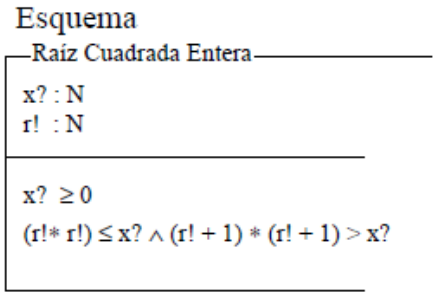


María Teresa García Valverde - Tlf: (+34) 968 27 86 57

Como vemos, el esquema se compone de dos partes: la signatura y el predicado. La signatura define las entidades que constituyen el estado del sistema y el predicado establece las condiciones que siempre deben cumplirse para esas entidades.

Los esquemas pueden ser manipulados utilizando operaciones tales como composición de esquemas, renombrado de esquemas y ocultación de esquemas. Estas operaciones permiten manipular los esquemas y definir esquemas más complejos

Cuando un esquema define una operación, el predicado puede establecer precondiciones y postcondiciones. Estás definen el estado antes y después de la operación. La diferencia entre ambas define la acción especificada en el esquema de la operación. Por ejemplo, el esquema de la raíz cuadrada en Z quedaría como:



Para analizar los componentes más importantes en Z vamos a seguir un ejemplo sobre una entidad bancaria. En ella se tienen en cuenta las siguientes características:

- Cada cliente se identifica por su número de documento y puede tener sólo una caja de ahorro.
- No se guardará el historial de transacciones de las cajas; sólo se preservará el saldo de cada una.
- Los clientes sólo pueden extraer y depositar dinero en efectivo, y solicitar el saldo de su caja.
- Un cliente puede solicitar el cierre de su caja de ahorro sólo si su saldo es cero.

En primer lugar definiremos los tipos elementales:

[DNI]

En Z el único tipo básico nativo es Z, es decir los enteros. Los naturales se pueden definir: N=={n:Z /0 ≤n}. Así, también tendremos:

DINERO == N

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

El estado del banco lo podemos definir como:

Banco

$$ca : DNI \pfun DINERO$$

Esto quiere decir que el banco “es" sus clientes y sus cajas de ahorro, donde existe relación funcional entre los clientes y sus cajas de ahorro. Esta relación es una función parcial, ya que no todos los elementos de DNI son clientes del banco. DNI representa a todos los DNIs de todos los potenciales clientes del banco, y no a todos los clientes actuales.

Así el estado inicial del banco, será el conjunto vacío:

BancoInicial

$$Banco$$

$$ca = \varnothing$$

Vamos ahora a definir una operación, por ejemplo, Nuevo Cliente. Cuando una persona se transforma en cliente del banco se le crea una caja de ahorro. El saldo inicial de la cuenta será cero. Esta operación podemos definirla de la siguiente forma:

NuevoClienteOk

$$\Delta Banco$$
$$d? : DNI$$
$$rep! : MENSAJES$$

$$d? \notin \mathrm{dom}\ ca$$
$$ca' = ca \cup \{d? \mapsto 0\}$$
$$rep! = ok$$

Dónde (?) indica que son variables de entrada, (!) indica que son variables de salida y Δ indica que tras esta operación se producirá un cambio de estado.

Para esta operación tendremos que indicar también, que ocurre en caso de error:

NuevoClienteE

$$\Xi Banco$$
$$d? : DNI$$
$$rep! : MENSAJES$$

$$d? \in \mathrm{dom}\ ca$$
$$rep! = numeroClienteEnUso$$

Dónde Θ indica que tras esta operación no se producirá un *cambio de estado y* MENSAJES habría que definirlo.

Así, la operación total Nuevo cliente quedaría como:

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

$$NuevoCliente == NuevoClienteOk \lor NuevoClienteE$$

De la misma forma podríamos definir la operación Depositar Dinero:

DepositarOk

$\Delta Banco$
$d? : DNI$
$m? : DINERO$
$rep! : MENSAJES$

$d? \in \text{dom } ca$
$0 < m?$
$ca' = ca \oplus \{d? \mapsto (ca\ d?) + m?\}$
$rep! = ok$

Y sus correspondientes casos de error:

DepositarE1

$\Xi Banco$
$d? : DNI$
$rep! : MENSAJES$

$d? \notin \text{dom } ca$
$rep! = clienteInexistente$

DepositarE2

$\Xi Banco$
$m? : DINERO$
$rep! : MENSAJES$

$m? \leq 0$
$rep! = montoNulo$

Quedando la operación completa:

$$DepositarE == DepositarE1 \lor DepositarE2$$
$$Depositar == DepositarOk \lor DepositarE$$

María Teresa García Valverde - Tlf: (+34) 968 27 86 57

Puntos clave

En este apartado se muestran aquellos puntos más importantes que el alumno debería tener claros al finalizar este capítulo:

- La especificación de requisitos puede realizarse mediante texto, técnicas o una combinación de ambas.
- Los documentos de requisitos de software son la declaración acordadas de los requisitos del sistema.
- Se deben organizar de tal forma que puedan ser utilizados tanto por los clientes del sistema como por los desarrolladores del software.
- El estándar IEEE para los documentos de requisitos es un punto de partida útil para estándares más específicos de especificación de requisitos.
- Los métodos de especificación formal complementan a las técnicas de especificación informal de requisitos. Pueden utilizarse con la definición de requisitos mediante lenguaje natural para clarificar cualquier área de ambigüedad.
- Las especificaciones formales son precisas y no ambiguas, evitando problemas de mala interpretación del lenguaje. Sin embargo, los no especialistas pueden encontrar estas especificaciones difíciles de entender.
- Una ventaja fundamental de usar métodos formales es que fuerza a un análisis de los requisitos del sistema en una etapa inicial. Corregir errores en esta etapa es menos costoso.
- Las técnicas algebraicas de especificación formal son especialmente adecuadas para especificar interfaces en donde la interfaz se define como un conjunto de clases de objetos o tipos abstractos de datos. Éstas técnicas ocultan el estado del sistema y especifican el sistema en función de las relaciones entre las operaciones de la interfaz.
- Las técnicas basadas en modelos modelan el sistema utilizando construcciones matemáticas tales como conjuntos y funciones. Éstas pueden mostrar el estado del sistema, lo que simplifica algunos tipos de especificación del comportamiento.

María Teresa García Valverde - Tlf: (+34) 968 27 86 57