Tema 03 – Ingeniería de requisitos

Ingeniería del Software

Rubén Fuentes Fernández
Dep. Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense Madrid

Trabajando con Antonio Navarro, Juan Pavón y Pablo Gervás

Contenidos

- Introducción
- Tipología de requisitos
- El proceso de Ingeniería de Requisitos
  - Casos de uso
  - FAST
- IEEE Std. 830-1998

Objetivos

- Entender las dificultades de capturar las necesidades del cliente
- Determinar qué aspectos clave hay que capturar en los requisitos
- Proporcionar pautas para el proceso de captura
- Examinar guías de documentación de requisitos

INTRODUCCIÓN

El contexto del sistema

- Entender la naturaleza de un problema es por lo general algo difícil.
  - Implica una interacción entre el grupo desarrollador y el contratante.
  - Estos grupos tienen diferentes perspectivas, y una idea imprecisa de qué información es relevante y cómo transmitirla.
- Un *interesado* (*stakeholder*) es cualquier persona que puede tener influencia directa o indirecta sobre los requisitos:
  - Clientes → quien contrata o representa a la organización
  - Usuarios → quien utilizará el sistema
  - Otros → ej. expertos externos

Dificultades en la captura de requisitos

- Un sistema tiene muchos usuarios y ninguno tiene una visión de conjunto.
- Distintos interesados pueden tener distintos requisitos.
- Influencia de factores políticos.
- Entorno de negocio dinámico del sistema informático.
- Los clientes no saben qué partes del trabajo pueden transformarse en software.
- Los interesados pueden desconocer lo que esperan del sistema informático.
- Los interesados no saben cómo hacer más eficiente la operación en su conjunto.
- Los interesados no saben detallar lo que saben de forma precisa.
- Es posible que los ingenieros de requisitos no conozcan el dominio del sistema, familiar para los interesados.

Requisitos

- Los requisitos son una descripción de los servicios y restricciones del sistema a construir.
- La Ingeniería de Requisitos es el proceso que permite identificar dichos requisitos.
  - Sistemáticamente
  - De una forma apropiada para el proceso de desarrollo
    - Discusión con el cliente
    - Punto de partida del resto de fases

Tipos de requisitos

- Requisitos de usuario
  - Describen los servicios que el sistema debe proporcionar y las restricciones bajo las que debe operar.
  - Ej. el sistema debe hacer préstamos
- Requisitos de sistema
  - Determinan los servicios del sistema y las restricciones en detalle.
  - Sirven como contrato.
  - Pueden ser funcionales, no funcionales y de dominio.
  - Ej. función préstamo; entrada: código socio, código ejemplar; salida: fecha devolución; ...
- Son los mismos, pero a distinto nivel de detalle.
  - Ambos se describen con lenguaje natural, diagramas y otros recursos.

Requisitos funcionales vs no funcionales

- Los requisitos *funcionales* describen:
  - Los servicios que proporciona el sistema (funciones).
  - La respuesta del sistema ante determinadas entradas.
  - El comportamiento del sistema en situaciones particulares.
  - En algunos casos, también determinan lo que no debería hacer el sistema.
- Los requisitos *no funcionales* describen:
  - Restricciones de los servicios o funciones que ofrece el sistema.

Requisitos – función préstamo

- Requisitos funcionales
  - prioridad: alta
  - estabilidad: alta
  - descripción: presta un ejemplar a un socio de la biblioteca
  - entrada: código socio, código ejemplar
  - salida: fecha devolución
  - origen: operador, consola
  - destino: sistema
  - necesita: base de datos de socios y ejemplares
  - acción: prestar ejemplar
  - precondición: usuario y ejemplar dados de alta, usuario sin préstamos pendientes, usuario sin límite de préstamos alcanzado
  - postcondición: ejemplar prestado al usuario
  - efectos laterales: retirada del carné durante 7 días si el usuario tiene préstamos pendientes de devolución
- Requisitos no funcionales
  - El tiempo de proceso de una petición será siempre inferior a 1 segundo.
  - El sistema de biblioteca debe ser capaz de atender simultáneamente a todas las bibliotecas de la universidad, estimadas en picos de 50 peticiones/minuto

Requisitos no funcionales: tipos

- Requisitos del producto
  - Especifican el comportamiento del producto.
  - Ej. prestaciones, memoria o tasa de fallos
- Requisitos organizativos
  - Se derivan de las políticas y procedimientos de las organizaciones de los clientes y desarrolladores.
  - Ej. estándares de proceso o lenguajes de programación
- Requisitos externos
  - Se derivan de factores externos al sistema y al proceso de desarrollo.
  - Ej. requisitos legislativos o éticos

Requisitos del dominio

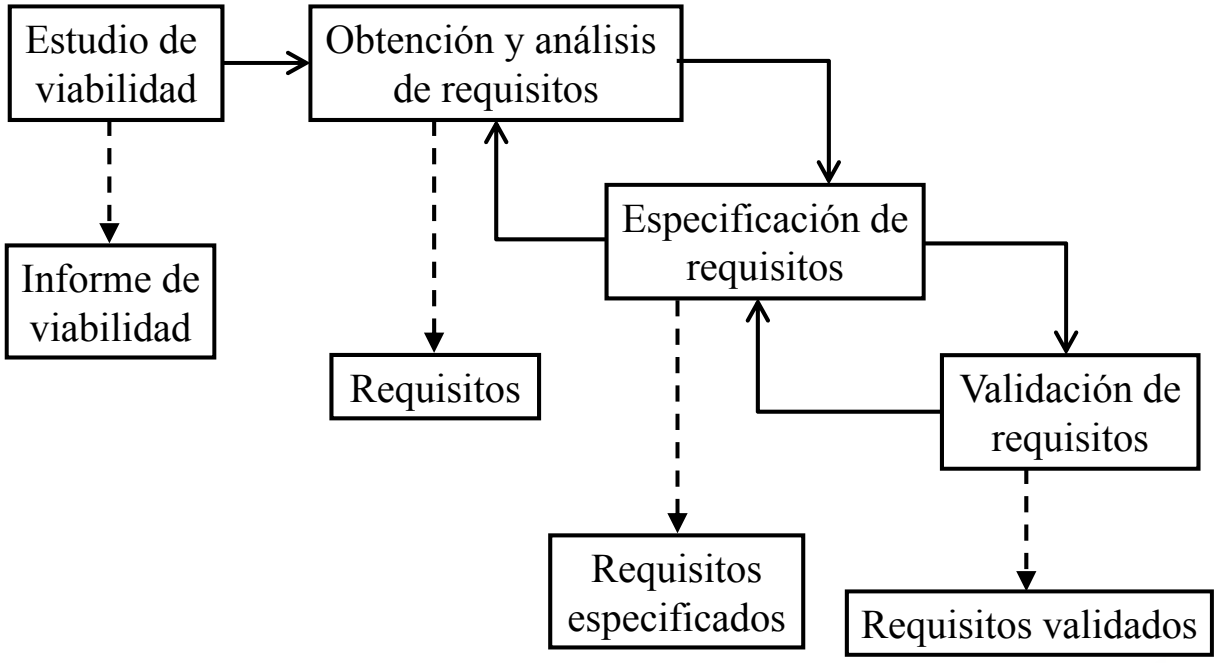
- Se derivan del dominio de la aplicación y reflejan características de dicho dominio.
  - Pueden ser funcionales o no funcionales.
  - Ej. el sistema de biblioteca de la universidad debe ser capaz de exportar datos mediante el Lenguaje de Intercomunicación de Bibliotecas de España (LIBE)
  - Ej. el sistema de biblioteca no podrá acceder a bibliotecas con material censurado

PROCESO

El proceso de Ingeniería de Requisitos

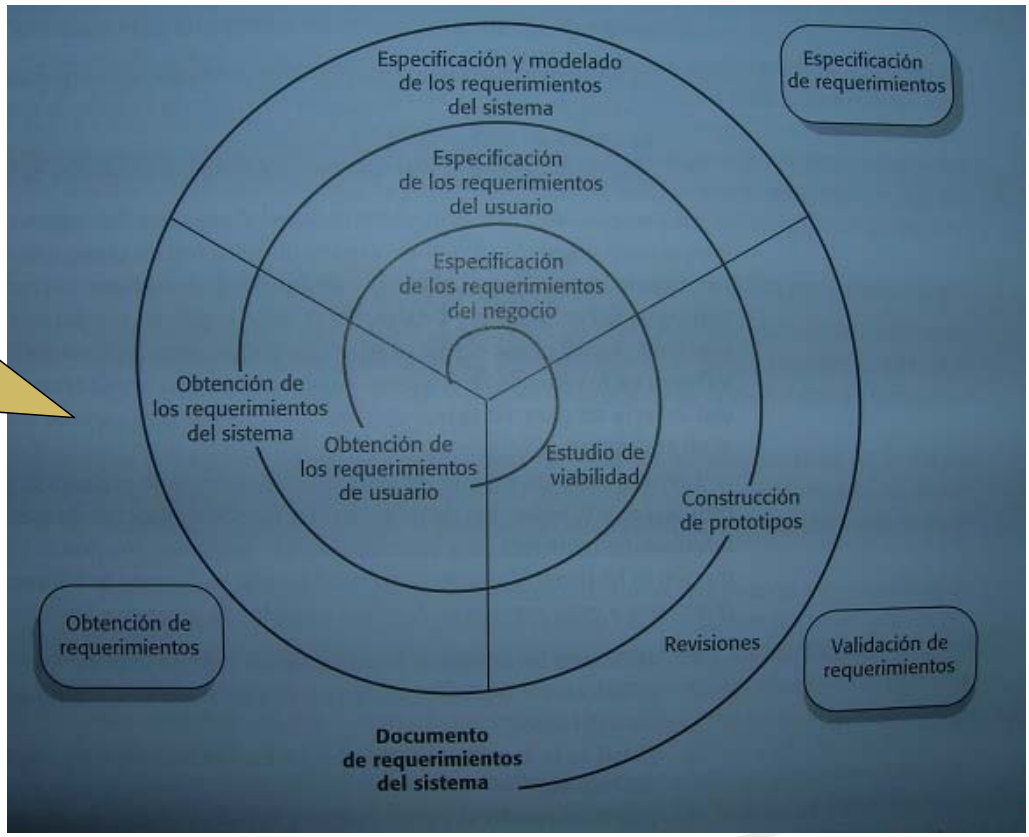
- La Ingeniería de Requisitos debe centrarse en lo que hay que hacer, no en el cómo.
- Para obtener los requisitos es necesario realizar varias tareas.
- También hay que tener en cuenta que los requisitos evolucionan.
  - Es necesario gestionar su cambio.
- Las actividades anteriores se organizan siguiendo modelos de proceso.
  - De la misma manera que el proceso de desarrollo de software global.
  - Aunque existen variantes de este modelo, vienen a cubrir las mismas actividades.

Proceso estilo cascada



Proceso estilo evolutivo

Pese a las figuras, se debe usar "requisitos" en vez de "requerimientos".



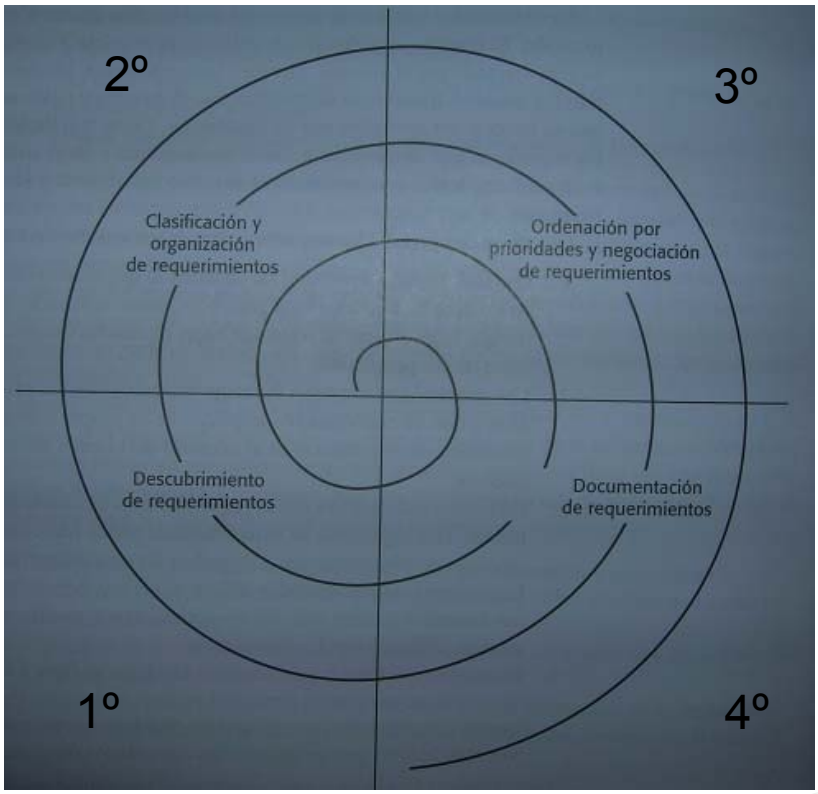
Fases: estudio de viabilidad

- Se trata de una estimación sobre si se puede resolver el problema cumpliendo las restricciones establecidas:
  - Usando las tecnologías software y hardware disponibles.
  - Integrándose en el entorno tecnológico y organizativo.
  - Cumpliendo las restricciones económicas del desarrollo.
  - Creando un sistema rentable en cuanto a operación y mantenimiento.
- Se origina tras una especificación preliminar de los requisitos.
- El estudio debe responder a las preguntas:
  - ¿Contribuye el sistema a los objetivos generales de la organización?
  - ¿Se puede implementar el sistema utilizando la tecnología actual y dentro de las restricciones de coste y tiempo?
  - ¿Puede integrarse el sistema con otros sistemas existentes en la organización?

Fases: análisis de requisitos

- Proceso en el que se interactúa con los interesados para determinar:
  - El dominio de la aplicación
  - Los requisitos funcionales
  - Los requisitos no funcionales

Análisis de requisitos: proceso



Fases: especificación de requisitos

- Se fija una descripción detallada y precisa de los requisitos.
  - Debe servir de base para un contrato entre los clientes y los desarrolladores de software.
- En función del ciclo, los requisitos serán preliminares del negocio, de usuario o de sistema.
  - Dentro de un proceso evolutivo o incremental
- El objetivo es realizar la Especificación de Requisitos Software (SRS).

Lenguajes de especificación de requisitos

- Pueden utilizarse distintos lenguajes para especificar los requisitos:
  - Lenguaje natural
  - Lenguaje natural estructurado
  - Lenguaje de descripción de diseño
  - Lenguaje de especificación de requisitos
  - Notaciones gráficas
  - Especificaciones matemáticas
  - Especificaciones lógicas

Fases: validación de requisitos

- Se encarga de comprobar si los requisitos se ajustan a los deseos y necesidades de los interesados.
- Si la validación es inadecuada, se propagarán errores al diseño e implementación.
- Evidentemente, tiene una fuerte repercusión sobre el coste.

Verificaciones en la validación de requisitos (1/2)

- Corrección
  - Cada requisito identificado es un requisito del sistema.
- No ambigüedad
  - Cada requisito identificado tiene una única interpretación.
- Realismo
  - Comprobar que se pueden implementar los requisitos con las tecnologías y plantilla disponibles.
- Completitud
  - La SRS debe incluir:
    - Todos los requisitos relativos a funcionalidad, prestaciones, restricciones de diseño, atributos o interfaces externas.
    - Definición de las respuestas del sistema a todos los tipos posibles de datos de entrada en todas las posibles situaciones.
    - Todas las etiquetas y referencias a todas las figuras, tablas y diagramas en la SRS, así como definiciones de todos los términos y unidades de medida.

Verificaciones en la validación de requisitos (2/2)

- Consistencia
  - Consistencia interna con todos los requisitos en todos los documentos (ej. requisitos de usuario vs requisitos de sistema).
- Priorizada
  - La SRS debe priorizar la importancia y/o la estabilidad de cada requisito mediante un identificador de importancia y/o estabilidad para cada requisito.
- Verificable
  - Cada requisito identificado es verificable.
    - Es decir, se puede comprobar que el sistema implementa el requisito.
- Modificable
  - La estructura y estilo permiten cambios en los requisitos de forma fácil, completa y consistente manteniendo la estructura y estilo.
- Trazable
  - El origen de cada requisito está claro.
  - Facilita la referencia de cada requisito en desarrollos futuros o en mejoras de la documentación.

Técnicas de validación

- Hay distintas técnicas para validar los requisitos, que pueden utilizarse conjunta o individualmente:
  - Revisión de requisitos
    - Análisis sistemático de los requisitos por parte de un equipo de revisores.
  - Prototipado
    - Creación de un modelo ejecutable del sistema.
  - Generación de casos de prueba
    - Si no se puede diseñar un caso de prueba para un requisito, probablemente el requisito sea problemático.
  - Análisis automático de la consistencia
    - Supuesto que se haya utilizado una herramienta formal de especificación de requisitos.

Gestión de requisitos software

- Los requisitos pueden cambiar y evolucionar.
  - Puede haber requisitos variables en función del usuario.
  - El cliente no suele ser el usuario.
  - El producto y/o el entorno evolucionan.
- La gestión de requisitos es el proceso de entender y controlar los cambios en los requisitos del sistema.
  - Tiene en cuenta aspectos técnicos no contemplados por la Gestión de la Configuración Software (GCS).

Objetivos de la gestión de requisitos software

- En la gestión de requisitos debemos ser capaces de:
  - Identificar los requisitos
  - Tener un proceso de gestión del cambio
  - Disponer de políticas de traza
  - Disponer de soporte CASE

Gestión de requisitos software

- La identificación de cada requisito se supone realizada cuando se dispone de una SRS.
- El proceso de gestión de cambio es común al de cualquier Elemento de Configuración Software (ECS).
- Las políticas de traza deben llevar cuenta de diversa información de traza:
  - De origen → liga el requisito al interesado
  - De requisitos → determina las dependencias entre requisitos
  - De diseño → liga los requisitos a los módulos de diseño

Matriz de traza

- La información de traza de requisitos puede representarse mediante una matriz de traza.

Req. id	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		D	R					
1.2			D			R		D
1.3	R			R				
2.1			R		D			D
2.2								D
2.3		R		D				
3.1								R
3.2							R	

D: depende de R: relacionado con

Tipología de requisitos según su evolución

- Estables
  - Se derivan de la actividad principal del sistema y están directamente relacionados con el dominio.
- Volátiles
  - Probablemente cambiarán durante el desarrollo del sistema o después de su entrega.
  - Se clasifican en:
    - Mutables → Cambian debido a cambios en el entorno en el que la organización opera.
    - Emergentes → Aparecen durante el desarrollo del sistema por el mayor entendimiento por parte del cliente.
    - De consecuencia → Aparecen por la introducción del sistema.
    - De compatibilidad → Dependen de los procesos de negocio de la organización.

Herramientas CASE

- Las herramientas CASE facilitan:
  - El almacenamiento de requisitos
  - La gestión del cambio
  - La gestión de la traza
- Pueden ser herramientas específicas o gestionar el cambio de la SRS como el de cualquier otro ECS.

Técnicas de especificación de requisitos

CASOS DE USO

Modelo de casos de uso

- Visión del sistema tal como se muestra a sus usuarios externos.
  - Lo desarrollan tanto los analistas como los expertos del dominio (los *interesados*).
- Para entenderlo mejor se puede fragmentar en distintos puntos de vista y con diferentes propósitos.
- Representa el acuerdo alcanzado entre clientes y desarrolladores.
  - Sirve como entrada al análisis, diseño y pruebas.

Casos de uso

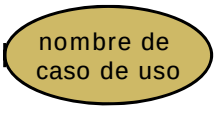
- Un caso de uso describe una forma en que los actores usan el sistema.
  - Es un fragmento de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.
  - Es una transacción (*caso de uso*) que tiene significado para los usuarios (*actores*).
- Un actor es cada tipo de usuario o sistema externo que interactúa con el sistema.
  - Terceros fuera del sistema que colaboran con el sistema.

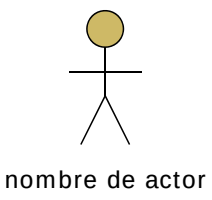
Especificación del modelo de casos de uso

- El estándar de UML [OMG, 2010] especifica estos modelos con:
  - Diagramas de casos de uso
  - Descripción de los casos de uso
    - Mediante plantillas de texto
    - Acompañados de diagramas de interacción

Diagrama de casos de uso: entidades

- Caso de uso
  - Secuencia de acciones, incluyendo variantes, que puede realizar el sistema interaccionando con los actores del sistema
- Actor
  - Un conjunto coherente de roles que juegan los usuarios cuando interaccionan con los casos de uso
- Límite del sistema
  - Representa el límite entre el sistema físico y los actores que interaccionan con el sistema





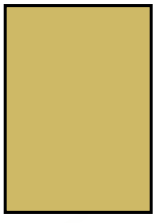
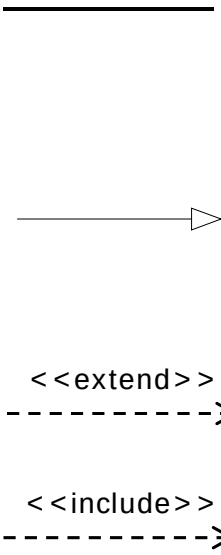
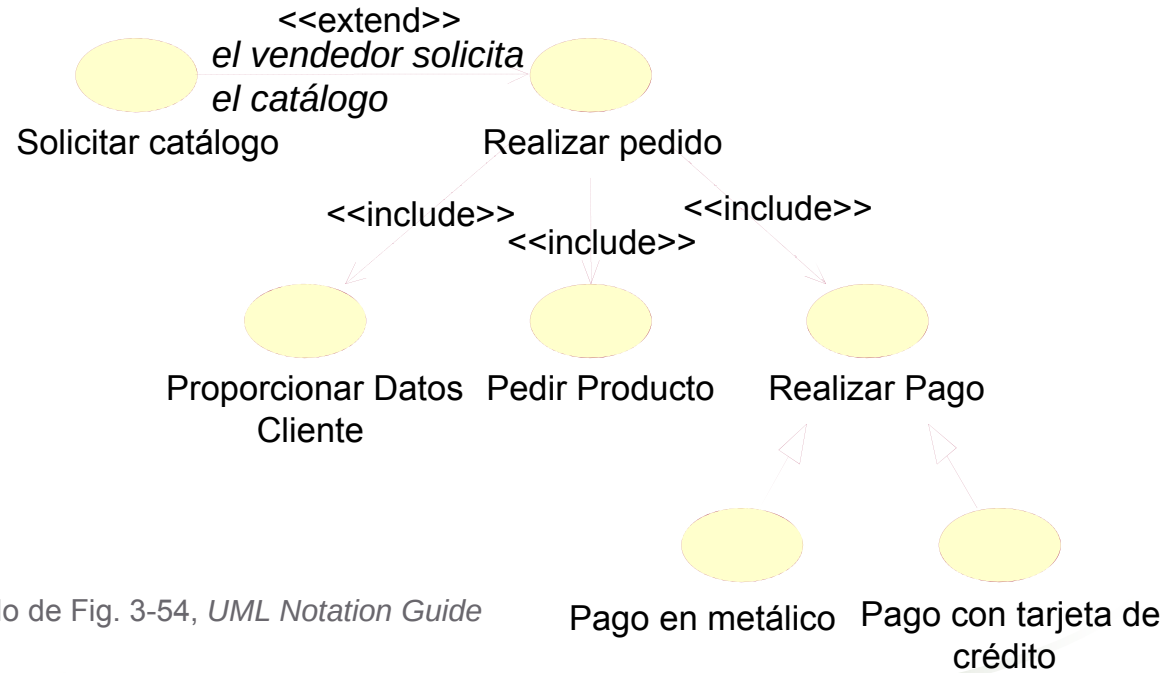


Diagrama de casos de uso: relaciones

- Asociación
  - La participación de un actor en un caso de uso
  - La instancia de un actor se comunica con instancias de un caso de uso
- Generalización
  - Relación taxonómica entre un caso de uso más general y otro más específico
- Dependencia
  - <<extend>> el comportamiento de un caso de uso origen extiende el de un caso de uso destino
  - <<include>> el comportamiento de un caso de uso origen incluye el de un caso de uso destino

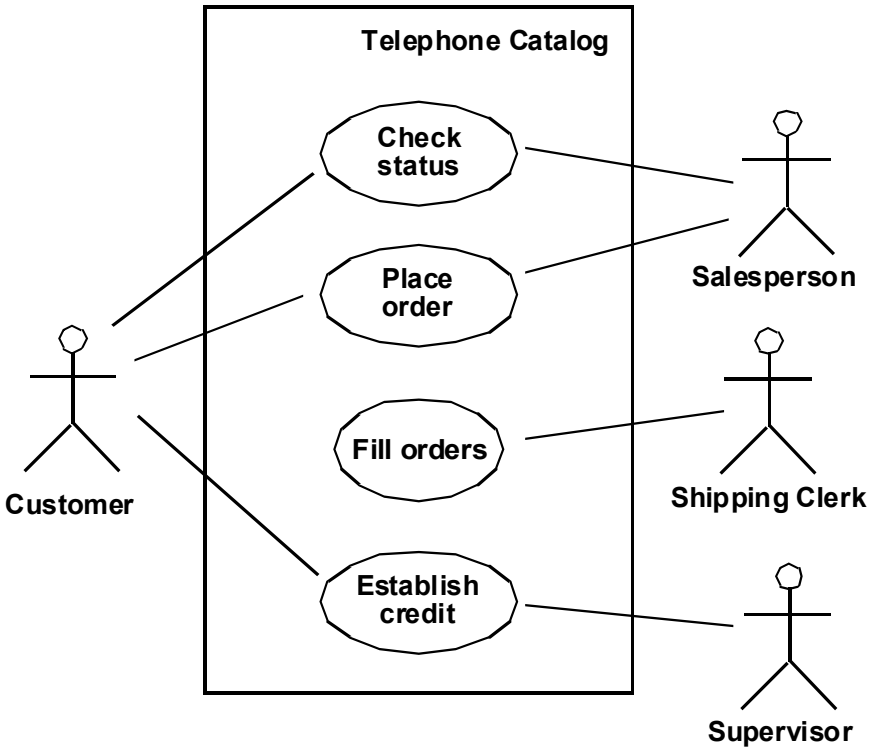


Ejemplo: asociaciones de dependencia y generalización



Adaptado de Fig. 3-54, *UML Notation Guide*

Ejemplo: límite del sistema



CASO DE USO #1	ArranqueSistema	
Objetivo en contexto	Se encarga de crear los objetos que componen el sistema e inicializarlos adecuadamente, leyendo archivos de configuración y cargando los casos de una base de datos	
Entradas		
Precondiciones	Los archivos de configuración deben existir y ser válidos Debe ser posible establecer la conexión con la base de datos, y en ésta se han definido los esquemas adecuados	
Salidas		
Postcondición si éxito	Es posible empezar a interactuar con el sistema	
Postcondición si fallo	El sistema no se ha podido inicializar adecuadamente y la aplicación acaba	
Actores	El usuario y la base de datos	
Secuencia normal	Paso	Acción
	1	Leer los archivos de configuración. Si error S-1
	2	Construir la función de similitud
	3	Construir el conector a la base de datos. Si error S-2
	4	Construir la base de casos, a partir de la función de similitud y el conector a la base de datos. Si error S-3
Secuencias alternativas	Paso	Acción
	S-1	Error al leer los archivos de configuración. Se informa del error y termina la aplicación.
	S-2	Error al establecer la conexión con la base de datos. Se informa del error y termina la aplicación.
	S-3	Error al leer los casos de la base de datos. Se cierra la base de datos, se informa del error y termina la aplicación.

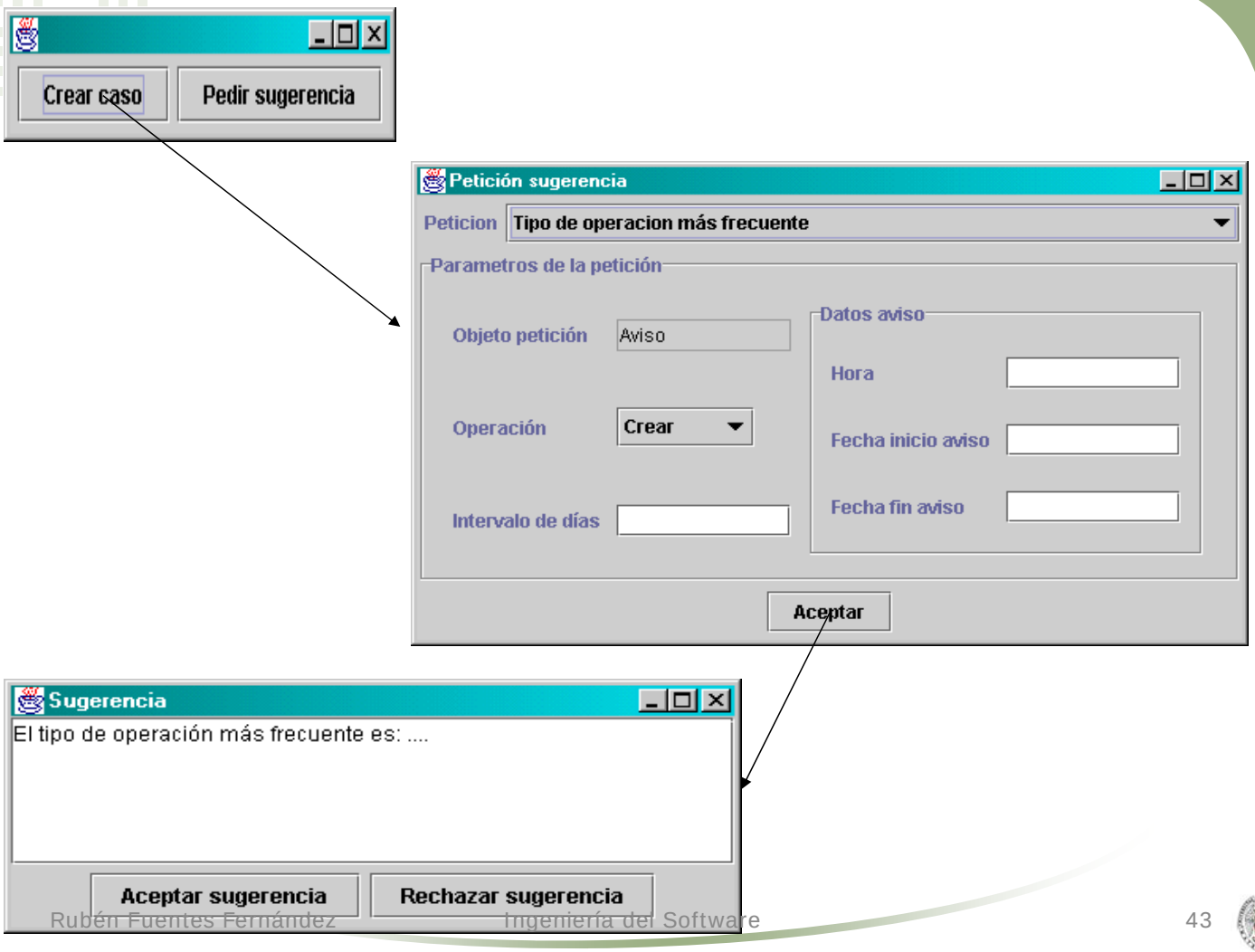
¿Hay algún problema en esta descripción?

Flujo de sucesos

- El flujo de sucesos es una especificación textual de la secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores.
  - Incluye las alternativas dentro de la secuencia.
- Estas secuencias de acciones deben poder ser modificadas, revisadas, diseñadas, implementadas y probadas como un conjunto significativo.
  - Pueden ser descritas como una sección del manual del usuario.

Prototipo de interfaz de usuario

- Los casos de uso se centran en los aspectos funcionales.
  - No son los únicos requisitos
- Los prototipos de interfaz de usuario ayudan a comprender y especificar las interacciones entre actores humanos y el sistema.
- Pueden usarse modelos de interfaz gráfica y esquemas de pantallas.



Requisitos especiales

- Aquellos que no pueden asociarse a ningún caso de uso
  - Requisitos no funcionales
- Se capturan como lista de requisitos.
  - De interfaz → interacción con un elemento externo que impone restricciones en formatos, tiempos u otros factores
  - Legales (normativas)
  - Físico → característica física que debe poseer un sistema (material, forma, tamaño, peso)
  - Restricción de diseño → limita el diseño (extensibilidad, mantenibilidad, reutilización de sistemas heredados)
  - Restricción de implementación → especifica o limita la codificación o construcción del sistema (estándares, lenguajes, políticas para la integridad de base de datos...)

Glosario

- Define términos comunes importantes que los analistas utilizan para describir el sistema.
- Centrado en el sistema que se va a construir, en lugar de su contexto.

Captura de los requisitos

1. Identificar actores
2. Identificar casos de uso
3. Describir brevemente cada caso de uso
4. Priorizar los casos de uso
5. Detallar los caso de uso
6. Prototipar la interfaz de usuario
7. Identificar los requisitos adicionales
8. Describir y estructurar el modelo de casos de uso completo
   - Incluye la elaboración del glosario

- El proceso no es necesariamente secuencial.

Cómo identificar casos de uso

- A partir de los actores
  - Identificar los actores relacionados con el sistema o la organización.
  - Para cada actor, identificar los procesos que inicia o en los que participa.
- A partir de los eventos
  - Identificar los eventos externos a los que puede responder el sistema.
  - Relacionar los eventos con actores y casos de uso.

- Criterios para identificar los actores relevantes:
  - Al menos un usuario que pueda representar al actor candidato.
  - Coincidencia mínima entre distintos actores.

Pautas de elaboración

- Asegurarse que cada caso de uso describe una parte significativa del funcionamiento del sistema.
- Evitar un número excesivo de casos de uso.
  - Un caso de uso no es un paso, operación o actividad individual en un proceso.
  - Un caso de uso describe un proceso completo que incluye varios pasos.
- Los casos de uso tienen que ser entendibles tanto por desarrolladores como por expertos del dominio.
  - Es una descripción de alto nivel del sistema.
  - Evitar conceptos de diseño.

Cuándo definir un modelo de casos de uso

- Para identificar los requisitos funcionales del sistema
  - y para identificar escenarios de prueba
- Normalmente en las primeras etapas de desarrollo
  - En metodologías dirigidas por casos de uso:
    - Empezar por los casos de uso y derivar de ellos los modelos estructural y de comportamiento
  - y si no:
    - Asegurarse que el modelo de casos de uso es consistente con los modelos estructural y de comportamiento

Técnicas de especificación de requisitos

FAST

Técnica de facilitación de la especificación de aplicaciones

- La técnica de facilitación de la especificación de aplicaciones (FAST) es un método específico para gestionar entrevistas.
  - Múltiples versiones
- Está diseñado para poner a clientes y desarrolladores a trabajar en equipo.

Proceso fundamental

- Se realiza una reunión previa con el cliente.
  - Establecer el alcance y la descripción básica.
- Se redacta una petición de producto breve.
  - 1 o 2 páginas
- Se convoca una reunión FAST.
  - Se elige un moderador.
  - Se reparte la petición de producto a todos los asistentes.
  - Se realiza la reunión.

Reunión: entorno

- Se celebra en sitio neutral.
- Asisten clientes y desarrolladores
- Hay reglas claras para la preparación y la participación.
- Hay un orden del día.
  - Suficientemente formal para que se cubra todo.
  - Suficientemente informal para que haya flexibilidad.
- Hay un moderador.
  - Cliente o desarrollador
- Hay un mecanismo de definición.
  - Ej. pizarra, proyector o fichas.
- El objetivo es identificar el problema y especificar los requisitos básicos de la solución.

Deberes para la reunión

- Cada asistente tiene que traer preparadas a la reunión las siguientes listas:
  - Objetos → elementos que forman parte del entorno del sistema, produce el sistema o utiliza el sistema
  - Servicios
  - Restricciones → coste, tamaño, reglas de negocio…
  - Criterios de rendimiento
- Las listas no tienen que ser exhaustivas pero deben reflejar la visión que cada participante tiene del sistema.

Reunión: fases

- Primera fase → exposición
  - Se exponen las listas para un área concreta.
  - En este punto no se admiten críticas ni discusión.
  - Se elabora una lista combinada.
  - Cuando están las listas combinadas para todas las áreas, se acuerda una versión negociada de cada una.
- Segunda fase → elaboración
  - Se separan los asistentes por equipos.
  - Cada grupo se encarga de hacer una mini especificación de unas cuantas propuestas de la lista.
  - Cada equipo presenta su mini especificación a todos los participantes.
  - En función de esas especificaciones se rehacen las listas.
  - Se asigna a alguien la tarea de redactar un documento de especificación.
- Tercera fase → iteración de las anteriores

ESTÁNDARES - IEEE STD. 830-1998

Introducción

- El IEEE Std. 830-1998 [IEEE, 1998] se encarga de proporcionar unas normas para la creación de la SRS.
- Objetivos de la SRS:
  - Establecer la base para un acuerdo entre clientes y desarrolladores sobre qué debe hacer el software.
  - Reducir el esfuerzo de desarrollo.
  - Proporcionar una base para la estimación de costes y planificación.
  - Proporcionar una guía para la validación y verificación.
  - Facilitar la transferencia de software.
  - Servir como base para mejoras posteriores.

Ámbito de la SRS

- La SRS debe centrarse en qué hace el sistema, no en cómo lo hace ni en cómo se construirá.
- Una SRS debería identificar las siguientes características del sistema:
  - Funcionalidad
    - Lo que hace el sistema.
  - Interfaces externas
    - Forma de interactuar del sistema con las personas, el hardware del sistema, otro hardware y otro software.
  - Prestaciones
    - Entre otras, velocidad, disponibilidad y tiempo de respuesta del sistema.
  - Atributos
    - Entre otros, portabilidad, corrección, mantenibilidad y seguridad.
  - Restricciones de diseño impuestas a la implementación
    - Entre otras, estándares, lenguajes de implementación y entornos operativos.

Límites del ámbito de la SRS

- Hay aspectos que deben excluirse de la SRS:
  - Detalles de diseño o implementación
  - Restricciones adicionales sobre el software
- Salvo que hayan sido contempladas como parte de los aspectos anteriores.

Interfaces

- Las interfaces entre distintos sistemas software pueden especificarse en un documento de especificación de interfaces que sea referenciado por diferentes SRSs.
- Entre estas interfaces se consideran las de comunicación, software o bases de datos.

IEEE Std. 830-1998: tabla de contenidos (1/2)

1. Introducción
   - (1.1) Propósito
   - Alcance
   - Definiciones, acrónimos y abreviaturas
   - Referencias
   - Resumen
2. Descripción general
   - (2.1) Perspectiva del producto
   - Funciones del producto
   - Características del usuario
   - Restricciones
   - Supuestos y dependencias
   - Requisitos futuros

IEEE Std. 830-1998: tabla de contenidos (2/2)

3. Requisitos específicos
   - (3.1) Interfaces externas
   - Funciones
   - Requisitos de rendimiento
   - Requisitos lógicos de la base de datos
   - Restricciones de diseño
   - Atributos del sistema software
- Apéndices
- Índice

Lenguaje de especificación de requisitos funcionales

- El estándar no determina el lenguaje en el que se debe describir cada requisito funcional.
- Supuesto que elijamos Lenguaje Natural Estructurado, para cada requisito funcional deberíamos incluir:
  - Prioridad y nivel de estabilidad
  - Descripción de la función
  - Descripción de la entrada y su origen
  - Descripción de la salida y su destino
  - Indicación de otras entidades utilizadas
  - Acción detallada de la función
  - Pre y post condiciones
  - Efectos laterales, si hay

CONCLUSIONES

Conclusiones

- Los requisitos son una pieza clave del desarrollo.
  - La base del contrato entre clientes y desarrolladores
  - El punto de partida de todas las demás fases del desarrollo
- Los requisitos tienen distintos:
  - Ámbitos de uso → Requisitos de usuario y del sistema
  - Tipos → Requisitos funcionales, no funcionales y del dominio
  - Estabilidad → Requisitos estables y volátiles
- La Ingeniería de requisitos es fundamental en la IS.
  - Capturar los requisitos es una tarea difícil.
  - Se requieren guías para su obtención y gestionar su evolución.
- El IEEE Std. 830-1998 como modelo de SRS.

Glosario

- CASE = *Computer-Aided Software Engineering*
- ECS = Elemento de Configuración Software
- FAST = *Facilitated Application Specification Techniques*
- GCS = Gestión de Configuración Software
- IEEE = *Institute of Electrical and Electronics Engineers*
- IS = Ingeniería del Software
- OMG = *Object Management Group*
- SRS = *Software Requirements Specification*
- UML = *Unified Modeling Language*

Referencias

- R. Pressman: Ingeniería del Software. Un enfoque práctico, 7ª edición. McGraw-Hill, 2010.
  - Capítulos 10 y 11
- I. Sommerville: Ingeniería del Software, 7ª edición. Addison Wesley, 2007.
  - Capítulos 5 y 6
- IEEE: IEEE Std. 830-1998. Recommended Practice for Software Requirements Specification. IEEE, 1998.
- OMG: OMG Unified Modeling Language (OMG UML), Superstructure, version 2.3. OMG, 2010.