Tema 4: Análisis de Requisitos

BLOQUE II:
ANÁLISIS Y DISEÑO DE LOS SISTEMAS SOFTWARE

Segundo curso de Grado en Ingeniería Informática
Curso 2014-2015

Javier Sánchez Monedero
jsanchezm@uco.es
http://www.uco.es/users/i02samoj

Índice

Índice

1.

2. Definición y tipos de requisitos

Advertencia

Advertencia
Estos apuntes no contienen todo el material necesario respecto al tema. Intentan dar una visión de conjunto y proporcionar los conocimientos básicos para que podáis consultar la bibliografía y manuales de referencia, así como material complementario colgado en Moodle y los ejercicios realizados en clase.

Introducción

Los requisitos "expresan las necesidades y restricciones que afectan a un producto de software que contribuye a la solución de un problema del mundo real" y nos sirven para delimitar **qué posibles soluciones son adecuadas** para el problema (las que cumplen los requisitos) y cuáles no.

- Para determinar si la solución cumple los requisitos, estos deben referenciar una o varias **características observables** del sistema.
- Para que una característica observable sea un requisito, es necesario que exprese alguna **necesidad o restricción** que afecte al software.

Ejemplo de característica observable. La característica "el sistema debe ser cómodo de usar" no es una característica observable, en el sentido de que su cumplimiento es subjetivo. Por lo tanto, sería necesario encontrar la manera de decir lo mismo haciendo referencia a los hechos observables de nuestro sistema como "se hará un estudio de satisfacción sobre una muestra de cincuenta usuarios y deberá obtener una nota mínima de 4 sobre 5".

Ejemplo de característica que no interesa a nadie. Por ejemplo, "el sistema siempre tardará más de treinta minutos en cargar la página principal del campus" es un hecho observable, pero no habrá nadie que esté interesado en que el sistema que queremos desarrollar lo cumpla; en todo caso, pretenderemos lo contrario, que el sistema tarde menos de N segundos en cargar la página principal del campus.

Definición de requisito
Un requisito es una característica **observable** del sistema que **satisface una necesidad** o expresa una restricción que afecta al software que estamos desarrollando.

Partes interesadas o *stakeholders*

Los *stakeholders* de un proyecto son aquellas personas y entidades que tienen algún impacto o interés en éste.

> *Stakeholder* es un término inglés utilizado por primera vez por R. E. Freeman en su obra: "*Strategic Management: A Stakeholder Approach*", para referirse a «*quienes pueden afectar o son afectados por las actividades de una empresa*».
>
> La traducción de esta palabra ha generado no pocos debates en foros de Internet, aunque son varios los especialistas que consideran que la definición más correcta de "stakeholder"sería parte interesada (del inglés stake, apuesta, y holder, poseedor).[1]

Relación requisitos y parte interesada

Los requisitos nos dicen qué es lo que los diferentes *stakeholders* esperan del nuevo sistema.

Conviene no perder esto de vista.

Ejemplo de desconexión
Las aplicaciones corporativas que incluyen la mayor parte de las empresas de telefonía (o los propios fabricantes) en los móviles no interesan a los usuarios de móviles, y a la vez suponen un coste de desarrollo a estas empresas.

Relevancia de los requisitos

- Si los requisitos no están bien definidos, pueden surgir problemas
- Requisitos ambiguos dan lugar a diferentes interpretaciones por parte de los usuarios y los desarrolladores.
- Lo deseable es que los requisitos fueran completos y consistentes desde el principio:
  - **Completos**: Recogen la descripción de todos los servicios esperados por el usuario.
  - **Consistentes**: No hay contradicciones entre estas especificaciones.
- En la práctica es imposible producir un documento de requisitos completo y consistente.

[1] http://es.wikipedia.org/wiki/Stakeholder

Diferentes interpretaciones

Ejemplo de diferentes interpretaciones ante falta de precisión

El programa proporcionará un visor de documentos.

- Intención del Usuario: visores especiales para cada tipo de documento.
- Interpretación del Desarrollador: proporcionar un visor de texto que muestre el contenido del documento.

Tipos de requisitos

A grandes rasgos, hay consenso en definir dos tipos de requisitos:

- **Requisitos funcionales**: hacen referencia a las necesidades/funcionalidad que debe satisfacer el sistema (**qué** ha de hacer).
- **Requisitos no funcionales**: expresan **restricciones** sobre el conjunto de soluciones posibles (**cómo** debe hacerlo).

Dependiendo de la bibliografía, encontramos muchos otros tipos de requisitos:

- Requisitos de usuario
- Requisitos del sistema
- Requisitos de dominio
- Requisitos de interfaz
- ...hasta el infinito y más.

Requisitos funcionales

- Los requisitos funcionales nos indican qué cálculos realiza el sistema, qué datos posee, cómo los manipula, etc.
- En general, podemos decir que los requisitos funcionales nos indican cuál es el comportamiento del sistema ante los estímulos que le llegan del exterior.

Requisitos no funcionales

Los requisitos no funcionales **suelen tener forma de restricción** y suelen afectar a gran parte del sistema.

- Por ejemplo. Si decimos que el campus virtual que estamos desarrollando debe ser transportable a otras plataformas, ello no afecta sólo al foro o a la entrega de ejercicios, sino a todo el sistema.

Otra diferencia respecto a los requisitos funcionales es que los requisitos no funcionales **no incluyen comportamiento**. Su finalidad es especificar criterios que evalúen la calidad general del sistema, como seguridad, rendimiento, coste, etc.,

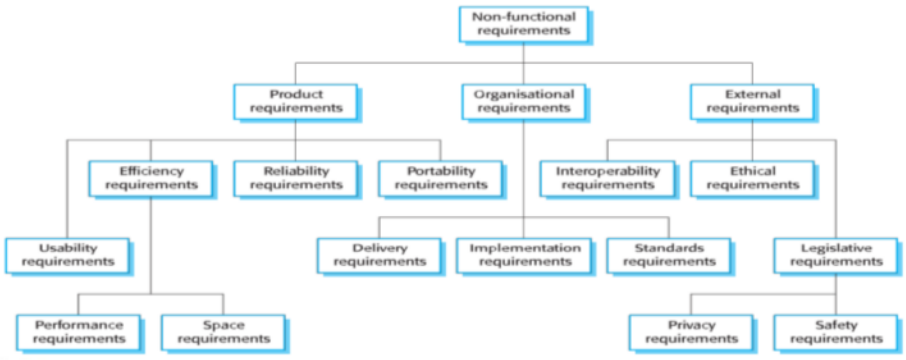

Figura 1: Jerarquía de requisitos no funcionales.

Los requisitos a lo largo del desarrollo

Actividades de los requisitos:

- Obtención de los requisitos.
- Gestión de los requisitos.
- Documentación de los requisitos.
- Verificación del sistema.

Tipos de requisitos no funcionales

Ejemplos de requisitos no funcionales

Algunos tipos de requisitos no funcionales:

- Requisitos Del Producto:
  - Especifican el comportamiento del producto.
  - Ejemplos: rapidez de la ejecución, capacidad de memoria, fiabilidad, etc.
- Requisitos Organizacionales:
  - Derivan de políticas y procedimientos existentes en la organización del cliente y del desarrollador.
  - Ejemplos: Estándares de procesos, métodos de diseño, lenguajes de programación, métodos de entrega, etc.
- Requisitos Externos:
  - Se derivan de factores externos al sistema y a su procesos de desarrollo.
  - Ejemplos: Requisitos de interoperabilidad, legislativos, éticos, etc.

3. Obtención de requisitos

3.1. Objetivo y fases

Objetivo de la actividad

El objetivo de esta actividad es obtener la lista de todos los requisitos que, idealmente, debería cumplir el sistema por desarrollar.

Denominaremos **requisitos candidatos** a aquellos requisitos obtenidos en esta primera etapa, dado que todavía no hemos decidido si los incorporaremos o no al conjunto de requisitos de nuestro sistema.

Fases en la obtención

- Identificar los *stakeholders* del sistema.
  - Modelización de roles de usuario.
  - Representantes de los *stakeholders* (aquella persona que habla en su nombre y que nos debe transmitir sus requisitos).
- Identificar los requisitos de cada uno de los *stakeholders*.

No identificar a todas las partes interesadas seguramente nos lleve a no identificar uno o varios requisitos.

Ejemplo
Portal de gestión de matrículas con nombre de letra griega que se olvida de que los discapacitados también estudian en la universidad. A nivel de interfaz de usuario o a nivel de descuentos en matrícula no se contemplan.

3.2. Identificación de requisitos

Técnicas

- *Brainstorming* o lluvia de ideas.
- Entrevistas y cuestionarios.
- Observación y prototipado.
- Listas predefinidas.

Brainstorming o lluvia de ideas

Se aplica cuando no se tiene un punto concreto de partida. Una sesión puede ser de la siguiente forma:

1. Creación del grupo de trabajo.
2. *Brainstorming* inicial.
3. Organización y consolidación del conjunto inicial.
4. Refinamiento.

Se suele utilizar sobre todo en la fase de identificación de partes interesadas y requisitos candidatos.

Entrevistas y cuestionarios

Las claves para que la entrevista sea productiva son:

- **Elegir correctamente a los entrevistados**. Variedad y representatividad de la muestra.
- **Evitar las respuestas condicionadas**. Cualquiera de nosotros, ante la pregunta "¿Querrías que el sistema diera respuesta en menos de un segundo?", respondería que sí, dado que entendemos que si decimos que no, el sistema será muy lento.
- **Evitar respuestas limitadas por el conocimiento actual**. A veces el conocimiento actual puede limitar las opciones exploradas.
- **Aportar información sobre el coste de las alternativas**. Si alguien nos pregunta si queremos que un sistema haga X y no nos dice el coste de ello, lo más probable es que la respuesta sea un *sí*.

Observación y prototipado

Esta técnica consiste en la observación directa de los usuarios mientras usan un sistema de información.

Algunas utilidades:

- Sobre todo para detectar requisitos relacionados con la usabilidad.
- Facilitar la comunicación con el usuario dado que entenderán mejor un prototipo que la descripción de un requisito).
- Explorar alternativas

Se puede efectuar sobre un sistema ya existente, sobre una implementación parcial del sistema final (en el caso del desarrollo incremental) o sobre un prototipo.

Listas predefinidas

Las listas predefinidas o listas de comprobación (*checklist* en inglés) nos pueden ayudar a no pasar por alto algunos requisitos, sobre todo los no funcionales.

Ejemplo de lista predefinida

El **estándar IEEE 830** nos recuerda que siempre debemos documentar, además de los requisitos funcionales, los **requisitos no funcionales** de lo siguiente:

- Rendimiento (volumen de usuarios, volumen de datos, etc.)
- Requisitos lógicos de la base de datos (tipo de acceso, frecuencia de los accesos, entidades, relaciones y restricciones de integridad, etc.)
- Restricciones de diseño (otros estándares, limitaciones de hardware, etc.)
- Otros atributos:
  - fiabilidad, disponibilidad, seguridad (criptografía, informes sobre acciones de usuarios…), mantenibilidad, portabilidad

4. Gestión de requisitos

Una vez obtenida la lista de **requisitos candidatos**, deberemos decidir cuáles son los que queremos prever. Para ello, deberemos valorar el coste de cada requisito en tiempo y recursos de desarrollo. También necesitaremos saber qué valor o importancia tiene cada requisito para los diferentes *stakeholders*. Con estos dos datos ya podremos seleccionar la lista **definitiva de requisitos**.

Una buena gestión de requisitos nos permitirá maximizar el valor obtenido por el proyecto de desarrollo y, por lo tanto, es una parte fundamental de la gestión del proyecto.

Incluye:

1. Valoración de los requisitos.
2. Priorización de los requisitos.
3. Selección según riesgo.

5. Documentación de los requisitos

5.1. Definición y características deseables

Definición

Denominamos ***especificación*** al artefacto, típicamente un documento textual, que documenta el conjunto de los requisitos que se han seleccionado para el proyecto.

La especificación de requisitos presenta el contrato entre los *desarrolladores* y *las partes interesadas* y sirve como herramienta de comunicación para los dos grupos. Por ello es un elemento central de cualquier método de desarrollo, especialmente en cuanto a la gestión del proyecto.

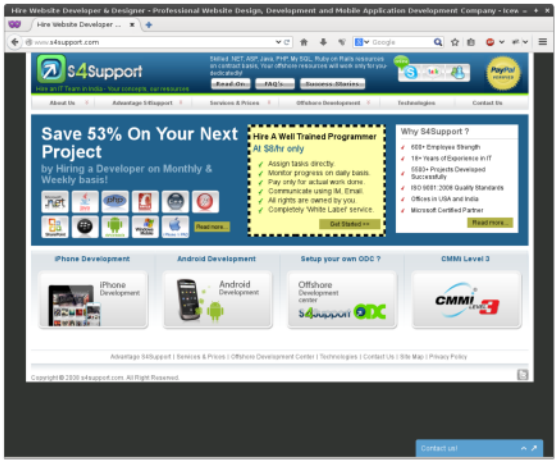

Figura 2: Web que se dedica a proporcionar trabajadores indios globalmente.

Forma y detalle

El estilo y la formalidad de la especificación dependerán del proyecto, pero también del contexto en el que se desarrolla y de las personas que participan en él.

Ejemplos de escenarios

- Aplicación móvil para una PYME desarrollada por otra PYME de la misma ciudad. La documentación de requisitos tal vez no necesite tener excesivos detalles.
- Aplicación de gestión corporativa desarrollada por una multi-nacional que subcontrata el desarrollo a otra empresa. Típicamente, el cliente final sólo se relacionará con la empresa que subcontrata el desarrollo.
- Software para un satélite en órbita en la tierra. Por ejemplo, en los satélites se utilizan memorias de un solo uso que cuestan varias decenas de miles de euros; *grabar* una versión nueva del programa puede costar varios miles de euros cada vez que se haga. Así, el coste de una comprobación de requisitos, por ejemplo de eficiencia en escenario real, es grande.

Programadores *low-cost*

Calidades según IEEE

El estándar **IEEE 830-1998** Committee [1998], titulado "Práctica recomendada por IEEE para las especificaciones de requisitos de software", define una buena especificación de requisitos como:

- **Correcta**. Todos los requisitos que documenta lo son realmente; es decir, si todos los requisitos enumerados son necesidades que el software debe satisfacer.
- **No ambigua**. Cada requisito enumerado tiene una única interpretación posible.
- **Completa**. Debe contener todos los requisitos de todas las partes interesadas. Si bien esto es casi imposible, además a menudo no se seleccionarán todos los requisitos para implementar.
- **Consistente**. No hay conflictos entre requisitos.
- **Requisitos etiquetados** con información relevante para la gestión de requisitos, como puede ser la importancia y el coste.
- **Verificable**, es decir, se refiere a cualidades observables y medibles.
- **Trazable**. Cada requisito enumerado está claramente identificado y facilita la referencia en los artefactos desarrollados a lo largo del proyecto (documentos, software, pruebas...).

5.2. ¿Cómo logramos la calidad en la obtención?

Algunas buenas prácticas

¿Cómo logramos la calidad en la obtención y documentación requisitos?

1. **Identificadores de requisitos** que permitan referencias cruzadas entre documentos. Un identificador no numérico puede ayudar a recordar el requisito que se referencia más rápidamente.
2. No perder el **punto de vista global**. A veces se identifican como requisitos necesidades técnicas. Por ejemplo, validación de información y almacenamiento en la base de datos, es necesaria técnicamente pero no aporta valor a la parte interesada.
3. **Granularidad**. Compromiso entre nivel de detalle y brevedad. En desarrollo iterativo e incremental, en general, los requisitos que se vayan a implementar de inmediato deben estar más detallados que el resto.
4. No dar por supuesta una **interfaz gráfica de usuario**. Por ejemplo, si imponemos una lista desplegable dentro de un requisito se pasará por alto la opción de un cuadro de texto de búsqueda incremental, que tal vez sea más cómodo para la persona usuaria.
5. **Condiciones de aceptación.** Para cada requisito deben escribirse claramente las condiciones de aceptación.
6. **Uso de plantillas.** El estándar IEEE 830-1998 ya mencionado o la plantilla de especificación de requisitos Volere[2] son dos buenos puntos de partida.

[2] http://www.volere.co.uk/template.htm

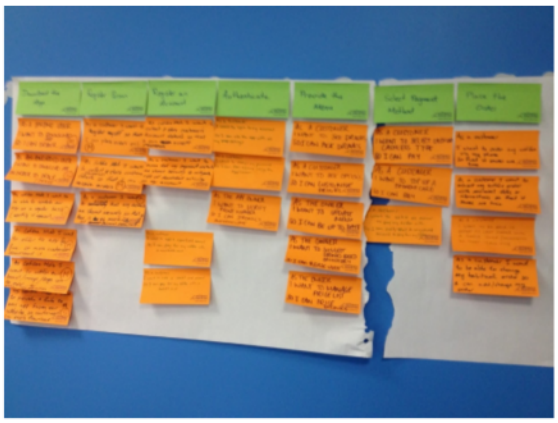

Figura 3: Ejemplo de mapa de historias de usuario (fuente Hastie and Wick [2014]).

Herramientas de captura y documentación de requisitos

Las técnicas más populares son:

- Los ***casos de uso***, dentro del Proceso Unificado.
- Las ***historias de usuario***, dentro de las metodologías ágiles Kniberg [2007], Miquel and Martos [2010].

En ambas metodologías estas técnicas se consideran una parte central y esencial. Aunque son herramientas equivalentes dentro de las metodologías, muchas personas hacen hincapié en sus diferencias Hastie and Wick [2014], García [2012].

5.3. Historias de usuario

Historias de usuario en Scrum

La **Pila de Producto** es el *corazón* de Scrum.

La **Pila de Producto** es, básicamente, una lista priorizada de *requisitos*, o **historias**, o funcionalidades, o lo que sea. Cosas que el cliente quiere, descritas usando la terminología del cliente. Llamamos a esto *historias*, *historias de usuario*, o a veces simplemente *elementos de la Pila*.

Mapa de historias de usuario

Información recogida

Campos básicos

- Una descripción escrita de la historia (sirve como recordatorio de que existe la historia y es útil para planificar, etc.).

- Una serie de conversaciones que sirven para definir y aclarar los detalles de la historia.
- Un conjunto de pruebas que documentan los detalles y que permiten determinar cuándo está completamente implementada la historia.

Para tener una mejor idea ver *Capítulo 2: Cómo hacemos Pilas de Producto*, en Kniberg [2007].

Ejemplo

Ejemplo de historia de usuario (fuente Miquel and Martos [2010])
Un ejemplo de historia de usuario sería "Como alumno, quiero poder entregar un ejercicio por medio del campus virtual" (esto sería la descripción, lo que pondríamos a la tarjeta). Esta historia debería ir acompañada de una serie de conversaciones entre los desarrolladores y los *stakeholders* para ir perfilando detalles, como qué formatos de documento son aceptados, si los profesores pueden o no imponer un formato, si es necesario que el sistema verifique las fechas límite automáticamente, etc.

Ejemplo de historia de usuario (continuación)
Todos estos detalles, finalmente, se deberán recopilar en un conjunto de pruebas que nos han de permitir determinar si la historia se ha implementado con éxito o no:

- Verificar que si se adjunta un archivo en formato PDF, DOC, DOCX u ODT, éste se guarda y se asocia al usuario.
- Verificar que si el profesor limita los formatos, sólo se aceptan entregas en los formatos indicados por el profesor.
- Verificar que si se adjunta un archivo en un formato incorrecto, se avisa al usuario.
- Verificar que si se intenta hacer la entrega fuera de las fechas de la actividad, se muestra un error al usuario.
- Verificar que si el usuario sale sin adjuntar el archivo, el sistema le avisa y le pide confirmación.
- Verificar que sólo se puede hacer la entrega en aquellas asignaturas en las que está matriculado el usuario.

Ejemplo de ficha para historia de usuario

5.4. Casos de uso

¿Qué es un caso de uso?

Los **casos de uso** son una **técnica de documentación de requisitos** muy extendida, entre otros motivos porque UML le da apoyo. Se trata de un enfoque a la manera de documentar requisitos que permite usar varios grados de detalle y de formalismo, lo que los hace adecuados en escenarios muy variados. Miquel and Martos [2010]

Definición original

"Descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable y de valor para un actor" Booch et al. [2006].

Un caso de uso recoge el contrato entre el sistema y las partes interesadas mediante la descripción del comportamiento observable del sistema.

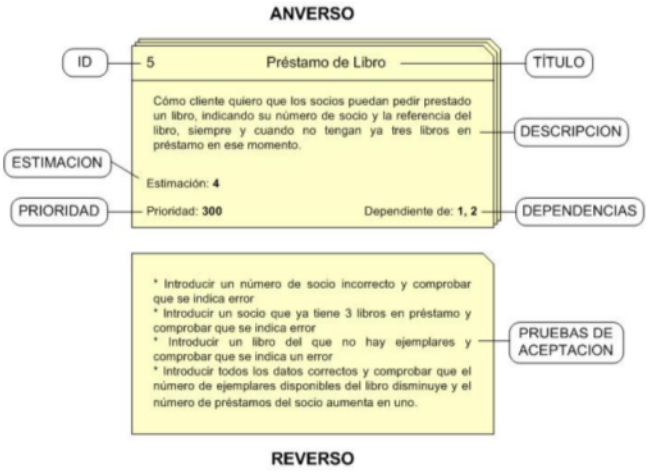

Figura 4: Ejemplo de ficha para historia de usuario García [2012]

Atención

Un Caso de Uso **NO** es un Diagrama, **NO** es un símbolo dentro de un diagrama:

- Es una forma de describir un escenario de interacción usuario sistema
- Los diagramas vienen después (o antes) y son una forma detener una visión general de los casos de uso, sus relaciones con los actores y con otros casos de uso

Actor principal

- No todas las partes interesadas tienen por qué interactuar directamente con el sistema. Por ejemplo en el sistema público de salud normalmente somos una parte interesada pero no interaccionamos directamente con el sistema informático, sino a través de nuestro médico de cabecera y los administrativos.
- En los casos de uso llamaremos **actor principal** al *stakeholder* que realiza una petición al sistema para recibir uno de los servicios y satisfacer un objetivo.

- En un mismo caso de uso además del actor principal pueden aparecer uno o más **actores de apoyo**, también denominados **secundarios**. Éstos son actores externos al sistema que proporcionan un servicio al sistema. Por ejemplo un actor que autoriza un pago con tarjeta de crédito.

Componentes de un caso de uso

- **Nombre**. Cada caso de uso debe tener un nombre que indique qué consigue el actor principal en su interacción con el sistema.
- **Actores**. Partes interesadas que interactúa con el sistema, o servicios externos al sistema.
- **Objetivos y ámbito.** Diferenciar objetivos generales de específicos. Considerar todas las partes interesadas, no sólo las que interaccionan.
- **Precondiciones y garantías mínimas**. Condiciones necesarias para que se pueda dar la interacción del caso. Las garantías se refieren a lo mínimo que el sistema debe garantizar para considerar un éxito la interacción. Comentarios:
  - Condiciones de error: se consideran extensiones del caso de uso.
  - Otras condiciones: no se "activa" el caso de uso.
- **Escenarios**. Hay un escenario (el principal) que es la secuencia de acontecimientos que espera el actor principal cuando pone en marcha la ejecución del caso de uso. Es un escenario de éxito (se cumplen las garantías mínimas). El resto de los escenarios (sean de éxito o de error) forman el conjunto de escenarios alternativos o extensiones. Una característica importante de las extensiones es que, al describirlas, siempre lo hacemos en referencia a la secuencia de acontecimientos del escenario principal: la extensión empieza porque, en algún paso del escenario principal, se da una cierta condición que da paso a la ejecución de la extensión.

Describimos todos los escenarios como una secuencia de acciones que pueden describir:

- Interacciones entre el usuario y el sistema: qué hace el usuario en este punto de la interacción ("El usuario indica el tema y el contenido del mensaje") o qué respuesta da el sistema ("El sistema muestra las prácticas para entregar este mes").
- Validaciones por parte del sistema: "El sistema valida que el usuario tenga acceso al foro".
- Cambios en el estado interno del sistema: "El sistema graba que el usuario ha leído el mensaje".

Caso de uso: resolver una duda al foro.
Actor principal: el estudiante con la duda.
Actores de soporte: otros estudiantes, profesor.
Nivel: general.
Ámbito: campus virtual.
Escenario principal de éxito:
1. El estudiante con la duda escribe un nuevo mensaje en el foro.
2. Otros estudiantes del grupo leen el mensaje y las respuestas que pueda haber publicadas.
3. Algunos de estos estudiantes escriben respuestas al mensaje original o escriben respuestas a las respuestas.
4. El profesor del aula lee el mensaje original y las respuestas.
5. El profesor del aula decide que hay que intervenir con una respuesta "oficial" y escribe una respuesta al mensaje original o a alguna de las respuestas con su respuesta oficial.
6. El estudiante con la duda y el resto de los estudiantes del grupo leen el mensaje con la respuesta "oficial" del profesor.

Escenarios alternativos:
5a. El profesor decide que no hay que intervenir con una respuesta oficial y se acaba el caso de uso.
6b. Algún estudiante decide que la respuesta oficial no es satisfactoria.
6b1. El estudiante escribe un mensaje de respuesta a la respuesta del profesor pidiendo más indicaciones y volvemos al paso 2.

Figura 5: Ejemplo de componer un caso de uso en casos de uso más concretos, fuente Miquel and Martos [2010]

Clasificación de casos de uso

Podemos clasificar los casos de uso según dos grandes ejes: el **nivel** en el que se describen sus objetivos y el **ámbito** que se considera al describir el caso de uso.

Niveles

Cockburn Cockburn [2001] propone tres niveles:

- **Usuario** (*user goals*): son los más importantes, dado que son los objetivos concretos que los actores principales quieren conseguir al usar el sistema (por ejemplo, "Escribir un mensaje al foro" o "Entregar una práctica").
- **General** (*summary goals*): son el nivel más general y nos sirven para dar contexto y agrupar los objetivos de usuario (por ejemplo, "Resolver una duda en el foro" o "Cursar una asignatura").
- **Tarea** (*subfunctions*): son el nivel más concreto y sólo ofrecen una parte del valor que el actor espera (por ejemplo, "Adjuntar un archivo a un mensaje" o "Poner una calificación de una entrega a un alumno").

Ejemplo de caso de uso (niveles) I

Ejemplo de caso de uso (niveles) II

Si necesitamos concretar más, podemos ver que este caso de uso incluye varios pasos que podemos considerar, cada uno, un caso de uso a escala de usuario:

Caso de uso: matrícula.
Actor principal: estudiante.
Nivel: general.
Ámbito: universidad (organización).
Escenario principal de éxito:
1. El estudiante elige las asignaturas de las que se quiere matricular y el grupo de cada asignatura.
2. La universidad confirma que la selección es correcta e indica el precio de matrícula.
3. El estudiante efectúa el pago de la matrícula.
4. La universidad registra la matrícula del estudiante y da confirmación al estudiante.

Escenarios alternativos:
2a. La selección de asignaturas no es correcta (porque el estudiante no cumple los requisitos necesarios).
2a1. El estudiante rehace su selección y volvemos al paso 2.
4b. El pago no se efectúa dentro del límite establecido por la universidad.
4b1. La matrícula se registra como errónea y se ofrece al estudiante una moratoria de 30 días.
4b2. Si el estudiante realiza el pago correctamente dentro del nuevo límite, vamos al paso 4. En caso contrario, la matrícula queda anulada.

Figura 6: Un caso de uso (bastante simplificado) de matrícula, fuente Miquel and Martos [2010]

- Escribir un mensaje al foro.
- Leer un mensaje o respuesta del foro.
- Escribir una respuesta a un mensaje del foro.

Ámbito

Cuando describimos un caso de uso es importante tener claro cuál es el **ámbito de la descripción** que estamos haciendo: **qué queda dentro** como elemento documentado, y qué queda fuera.

- **Ámbito de organización**: estamos modelizando como caso de uso el comportamiento de la organización como un todo. Todos los sistemas (informáticos o no) y las personas no aparecen como actores, sino que los actores son externos a la organización.
- **Ámbito de sistema**: estamos modelizando un sistema informático. Todos los componentes (de software o hardware) del sistema son internos y, por lo tanto, no aparecen como actores.
- **Ámbito de subsistema**: estamos modelizando una parte del sistema informático, como un componente. Los otros componentes y todos los usuarios directos del componente serán actores en nuestro modelo.

Ejemplo de caso de uso: ámbito

6. Conclusiones

Casos de Uso vs. Historias de Usuario

CONCEPTO	CASOS DE USO	HISTORIAS DE USUARIO
Objetivo	Modelar la interacción entre un 'actor' y el Sistema	Redactar una descripción breve de una funcionalidad tal y como la percibe el usuario
Estructura	Texto detallado donde se sigue una plantilla predefinida a completar con conceptos técnicos (objetivo, resumen, actor, evento disparador, extensiones, etc.)	Corta y consistente en una o dos frases escritas en el lenguaje del usuario.
Planificación	No se utilizan para planificar	Se utilizan para planificar
Agilidad	Requieren tiempo para análisis y la redacción de plantillas predefinidas	Se pueden escribir en pocos minutos
Comprensión	Suelen ser de difícil comprensión, incluso para personal técnico	Fáciles de leer y comprender
Mantenimiento	Suelen pertenecer a documentos con cientos de páginas. Difíciles de mantener.	Muy fáciles de mantener
Comunicación	Modelo textual asociado con diagramas: todo tiene que estar escrito	Basada en la comunicación verbal y orientada a la colaboración y discusión para clarificar detalles
Soporte	Escritos en documentos con el objetivo de que estos sean archivados como documentos de referencia.	Escritas en tarjetas (teóricas o reales) con el objetivo de que sean usadas directamente
Duración	Puede ser implementada en varias iteraciones	Debe ser implementada y probada en una única iteración
Autores	Definidos por 'intérpretes' (analistas, consultores, etc.)	Posibilidad de ser definidas por usuarios y clientes
Pruebas	La definición de pruebas se redacta en documentación separada	Contienen 'Pruebas de Aceptación' en el reverso de la tarjeta
Contexto	Proporcionan una visión más general del Sistema y la integración en él.	Proporciona una visión menos obvia, por eso las pruebas y el *feedback* de los usuarios son tan importante en las metodologías agiles.
Metodología	Asociado con RUP	Asociado con Programación Extrema (aunque pueden usarse en RUP)

Figura 7: Comparativa entre Casos de Uso e Historias de Usuario (obviamente a favor de las historias), fuente García [2012].

Conclusiones

Nosotros deberíamos hacer nuestra propia tabla comparativa (más objetiva) cuando conozcamos mejor estas herramientas.

7. Bibliografía

Fuentes

El contenido de estos apuntes se basa a su vez en otros apuntes Miquel and Martos [2010] y Ruiz [2013], y en las referencias citadas.

Para ampliar y contrastar información sobre análisis de requisitos se recomienda consultar, además de las referencias, los apuntes del Tema 4 de la Prof. Irene T. Luque Ruiz [2013].

Bibliografía y enlaces

Referencias

G. Booch, J. Rumbaugh, and I. Jacobson. *El Lenguaje Unificado de Modelado*, volume Segunda edición. Pearson Education. Addison-Wesley, 2006.

A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley, 2001.

C. S. . S. E. S. Committee. 830-1998 - ieee recommended practice for software requirements specifications. Technical report, Institute of Electrical and Electronics Engineers, 1998. URL http://standards.ieee.org/findstds/standard/830-1998.html.

A. E. García. Historias de Usuario vs. Casos de Uso, June 2012. URL http://aegxxi-desarrollo.blogspot.com.es/2012/06/historias-de-usuario-vs-casos-de-uso.html. Consultado el 3 de noviembre de 2014.

S. Hastie and A. Wick. User Stories and Use Cases - Don't Use Both!, March 2014. Consultado el 3 de noviembre de 2014.

H. Kniberg. *Scrum y XP desde las trincheras. Como hacemos Scrum.* InfoQ.com. C4Media Inc, 2007. ISBN 978-1-4303-2264-1. URL http://www.proyectalis.com/2008/02/26/scrum-y-xp-desde-las-trincheras/.

J. Pradel i Miquel and J. A. R. Martos. *Requisitos. Asignatura Ingeniería de ingeniería del Software*. Universitat Oberta de Catalunya, 2010. FUOC PID_00213633.

I. T. L. Ruiz. Apuntes de la Asignatura Ingeniería del Software. Tema 4: Análisis de Requisitos. 2013.