Ingeniería del Software

Resumen

Dorian Wozniak

Índice

Capítulo 1

FASE DE REQUISITOS

1.1 Requisitos

Los **requisitos** son características que deben incluirse en el sistema.

Los requisitos se pueden dividir en:

- **Requisitos de usuario**: Condiciones o capacidades necesarias para que el usuario pueda resolver un problema o alcanzar un objetivo.
- **Requisitos del sistema**: Condiciones o capacidades que debe reunir el sistema para satisfacer un contrato, estándar o cualquier otro documento formalmente impuesto.
  - **Requisitos funcionales**: Establecen la funcionalidad o comportamiento del sistema. Describen las interacciones entre sistema y entorno.
  - **Requisitos no funcionales**: Criterios para juzgar cómo funciona el sistema. Son atributos de calidad.

Todo aquello que no indica qué hace o como opera el software no es un criterio. Cuestiones como tecnología de implementación, entorno de desarrollo, metodología y/o arquitectura son **restriccions**.

1.2 Fase de requisitos

La **fase de requisitos** consiste en la recopilación de información y hechos para representar los requisitos.

Algunas de las técnicas de recopilación de información son:

- **Entrevistas**
  - Fundamental para confeccionar requisitos. Maximizar información por sesión.
  - Objetivo definido, preguntas directas, entrevistado es persona clave.
- **Reuniones**
  - Varios entrevistados y entrevistadores
  - Agenda formal (puntos importantes) + informal (*brainstorming*)
- **Cuestionarios**
  - Contiene todas las preguntas que busca el analista.
  - Útil para obtener la misma información de muchos usuarios, especialmente datos numéricos y opciones símples.
  - Fácil de distribuir

Otras fuentes de información son:

- Formularios y documentos
- Manuales de procedimiento
- Informes
- Programas
- etc...

1.3 Representación de requisitos

1.3.1 Tabla de requisitos

Los requisitos son representados en una **tabla**. Cada requisito tiene un **código** y una **descripción**. Los requisitos se pueden agrupar o ordenar tal que sea mas conveniente.

Requisito	Descripción
RF-1	Requisito funcional 1
RF-2	Requisito funcional 2
NF-1	Requisito no funcional 1

1.3.2 Árboles y tablas de decisión

Si un requisito describe una toma de decisión, se puede especificar mediante un árbol/tabla de decisión.

En un **árbol de decisión** la raíz comienza la secuencia de decisiones, yendo a una rama según las condiciones que cumpla.

Una **tabla de decisión** se divide en dos partes y cuatro secciones:

Tabla de decisiones	
Identificación de condiciones	Entrada de condiciones (1 condición /columna)
Identificación de acciones	Entrada de acciones (reglas de decisión marcadas con cruz)

1.4 Validación de requisitos

La **validación** es un proceso manual donde se comprueba que los requisitos:

- Son precisos y claros
- Es válido
- Consistencia
- Completitud
- Realizable
- Verificable
- Trazable
- Manejable
- Libre de detalles de implementación

1.5 Casos de uso

Un **caso de uso** es una descripción de una secuencia de acciones, mas variantes, que ejecuta un sistema para producir un resultado observable de valor para un actor. Para representarlos se utiliza **notación UML**.

- Cada caso de uso se representa con una **elipse**. Tiene un nombre además de poder incluir una descripción, secuencia de pasos, condiciones, etc...
- Un **actor** es un agente que solicita un servicio al sistema.
- El comportamiento se puede especificar textualmente con un **flujo de eventos**.
- Se puede expresar el flujo de eventos graficamente mediante un **diagrama de secuencia**.

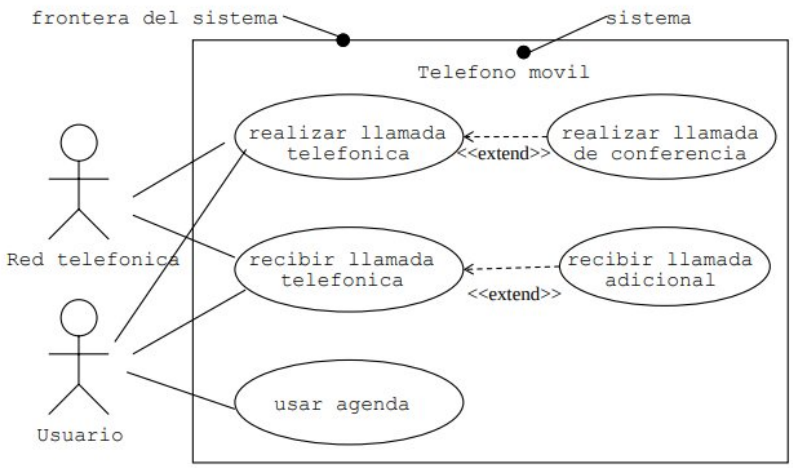

Figura 1.1: Ejemplo de diagrama de caso de uso

Los casos de uso pueden relacionarse entre sí:

- **Comunicación**: Un actor realiza un caso de uso
- **Generalización**: Un caso de uso hijo es un subtipo de otro padre
- **Inclusión**: Un caso base incorpora explicitamente el comportamiento de otro caso proveedor
- **Extensión**: Un caso base incorpora implicitamente el comportamiento de otro caso proveedor

Figura 1.2: Relaciones en un diagrama de casos de uso

Capítulo 2

FASE DE ANÁLISIS

2.1 Estática del sistema (modelo de objetos)

El **modelo de objetos** muestra la estructura estática de datos correspondiente al sistema del mundo real. Esta estructura proviene de la definición del problema y el conocimiento acerca de este. No existe una representación única correcta del modelo, y depende del problema a resolver.

2.1.1 Diagrama de objetos

Un **objeto** es una entidad o abstracción con fines bien definidos y significado en el dominio del problema.

- Un objeto posee una parte **estática** (atributos), que encapsulan el **estado**, es decir, los valores del objeto en un instante determinado. Este cambia con el tiempo a medida que se aplican acciones sobre el objeto
- Un objeto posee una parte **dinámica** (operaciones), que encapsulan el **comportamiento**, es decir, el conjunto de operaciones que definen las acciones y reacciones disponibles.
- Los objetos poseen **identidad**. Se distinguen por su existencia, no por sus propiedades o por tener un identificador

En un **diagrama de objetos**, se representan estos objetos junto a sus atributos y sus relaciones. Son un mecanismo de **clasificación** e **instanciación** de los elementos de un problema.

2.1.2 Diagrama de clases

Una **clase** describe un grupo de objetos que comparten propiedades, restricciones, relaciones y tienen una semántica común.

- Las clases poseen **atributos**, que describen las propiedades de los objetos y tienen un valor en cada instancia de la clase.
  - Un **atributo de clase** tiene el mismo valor en todas las instancias de una clase
  - El **nombre del atributo** es único dentro de la clase
  - Los atributos **no almacenan objetos**, solo tipos básicos.
  - Una clase tiene **identidad por si misma**. Ni los tipos básicos tienen identidad propia, ni los atributos se usan como identificador

- Una **operación** es una transformación que se pueden aplicar los propios objetos o de forma externa, y son compartidos por los objetos de una clase.
  - Las operaciones poseen **argumentos**. Su numero, tipo y tipo del valor regresado son su **signatura**
  - Una misma operación puede aplicarse en clases distintas mediante **polimorfismo**
  - Si no modifica el estado, se denomina **consulta**. Una consulta sin parámetros es un **atributo derivado**
  - Un **método** es la implementación de una operación para una clase
- La **instanciación** es la operación que permite crear objetos. Cada clase declara una o varias formas de crear y destruir objetos

2.1.3 Relaciones

2.1.3.1 Dependencia

La **dependencia** es una **relación de uso**. Indica que una clase queda afectada por cambios en las especificación, pero no al revés

2.1.3.2 Asociación

La **asociación** es una **relación estructural**. Indica que existe una relación física o conceptual entre objetos. Lo que se representa como **enlaces entre objetos** en un diagrama de objetos se abstrae como una asociación entre clases.

La asociación suele tener:

- Un **nombre descriptivo**.
- **Roles** que juega cada objeto de la asociación.
  - Roles son obligatorios en asociaciones reflexivas
  - Nombres o roles son necesarios en caso de varias asociaciones con los mismos objetos
- **Multiplicidad**, es decir, el número de elementos (en un rango) que participan en la relación

Aunque puedan ser cuestiones de diseño, tambien puede incluir:

- El sentido de **navegación**. En caso contrario se asumen bidireccionales
- La **visibilidad** de los objetos respecto a otros fuera de la asociación
- **Clase asociación**, si esta tiene sus propios metodos y propiedades
  - Las propiedades pueden ser de clase o de asociación
- **Calificación** para hacer más legible y preciso el modelo
- **Asociaciones n-arias**, aunque no son frecuentes y se suelen descomponer
- **Agregaciones**, un caso especial donde se distingue una clase principal (el todo) y un conjunto de subclases con los que está asociada (las partes)
  - La **composición** es una agregación mas estricta donde el todo esta estrictamente compuesto por subclases y no existe sin ellas

2.1.3.3 Herencia

La **herencia** es una **relación de generalización y especialización**. Indica las similitudes compartidas entre clases en forma de una **clase padre** (generalización), mientras que las diferencias se especifican en una **clase hija** (especialización).

En la herencia:

- Cada instancia de la subclase es una instancia de la superclase
- Las subclases heredan atributos y métodos de la superclase, sobre las cuales añade las propias

La herencia puede contener:

- **Discriminadores** que indican la propiedad que está siendo abstraida
- **Restricciones**
  - **Incompleta/completa** según si permite o no añadir clases hijas adicionales
  - **Sobrelapada/disjunta** según si permite tener simultaneamente o no más de una de las clases hijas definidas instanciadas

Los métodos heredados pueden ser **redefinidos**:

- **Por extensión**: Añade un nuevo comportamiento, normalmente sobre los atributos de la clase hija
- **Por restricción**: Aplica una restricción de tipo a los párametros de la clase padre
- **Por optimización**: Cumple la misma función con un código más eficiente

Otros mecanismos de herencia son:

- La **abstacción**, donde solo los hijos se instancian. Se pueden definir operaciones sobre la clase padre pero cada hijo proporciona su implementación
- La **herencia múltiple**, donde una clase hija hereda propiedades de múltiples padres. Solo se hereda la misma característica una vez, y cualquier caso de ambiguedad se debe concretar
- La **delegación**, donde se pasan las operaciones a otra clase para que las ejecute y así evitar herencia múltiple

- **Mediante agregación de roles**: Se dividen componentes de la superclase, donde cada componente sustituye una generalización
- **Heredando la clase mas importante**: El resto se delega a una clase asociada
- **Añidada**: Se mantiene la herencia pero se subdivide aun mas la jerarquía

2.2 Dinámica del sistema (modelo dinámico)

2.2.1 Diagramas de interacción (colaboración)

2.2.1.1 Diagrama de secuencia

2.2.1.2 Diagrama de comunicación

2.2.2 Diagrama de estados

2.2.3 Diagrama de actividades

Capítulo 3

ANEXO: EJERCICIOS

3.1 Fase de requisitos

Ej: Un aula informática de una escuela universitaria tiene la siguiente política de reservas:

- Los alumnos pueden reservar un máximo de 4 horas a la semana.
- Las reservas pueden realizarse hasta el día anterior al uso.
- Cuando un alumno hace una reserva A, se le entrega un justificante de la misma. El formato del justificante es el siguiente.

Reserva Aula Informática
Equipo: 014
Día reservado: 28-06-93
Hora reservada: 16:00
Número de Expediente: 54.321
Nombre: Alicia Kensington Carroll

Figura 3.1: Formato justificante

- Cuando se abre el Aula hay un listado de las reservas realizadas sobre cada equipo, de modo que la gente puede usar los equipos no reservados. El formato del listado de reservas es el siguiente:

Reservas del aula informática — Día: 28-06-93

Equipo	Hora	Alumno
001	10:00	Peter Pan
014	10:00	María Montesa
014	16:00	Alicia Pérez
...	...	...

Figura 3.2: Formato listado

- El horario del aula es de 10:00 a 18:00 todos los días lectivos y las reservas comienzan siempre a una hora en punto

Cod	Descripción
RF-1	El sistema debe permitir reservar equipos del aula informático a alumnos
RF-2	Un alumno puede reservar equipos como máximo 4 horas cada semana
RF-3	Se debe entregar al alumno un justificante al realizar una reserva
RF-4	Las reservas deben ser para un dia posterior al actual
RF-5	El sistema generará un listado con las reservas realizadas de los equipos en un día dado
RF-6	Las reservas de un equipo deben realizarse en el horario entre las 10 y las 18, comenzando en punto
RF-7	Las reservas deben realizarse en horario lectivo
RNF-1	El formato de un justificante debe incluir un número de expediente; el nombre y apellidos de quien realiza la reserva; el número del equipo reservado; y el día y la hora de la reserva

Cod	Descripción
RNF-2	El formato de un listado para un día dado debe incluir el día al que pertenecen las reservas; y para cada entrada de la lista, el equipo, hora y nombre del alumno que realiza la reserva

Ej: A un cliente se le dará trato preferencial si cumple una de estas tres condiciones:

- Compra más de 10.000 € por año y tiene una buena historia de pagos
- Compra más de 10.000 € por año y ha comerciado con nosotros por más de 20 años
- Compran 10.000 € o menos por año, pero tiene una buena historia de pagos

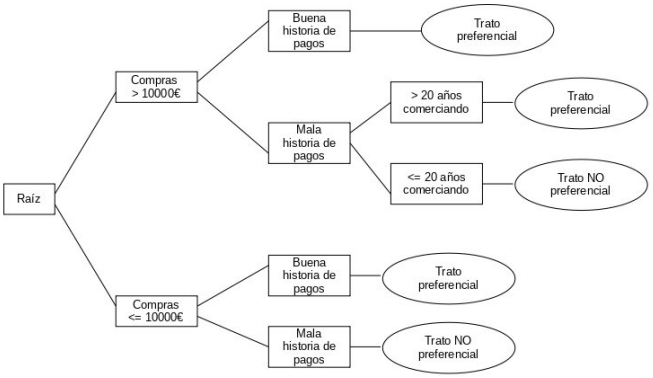

Figura 3.3: Árbol de decisión

Ej: (cont anterior)

C1: ¿El cliente tiene mas de 10000€ de compras al año?	S	S	S	S	N	N	N	N
C2: ¿El cliente tiene buena historia de pagos?	S	S	N	N	S	S	N	N
C3: ¿EL cliente ha comerciado durante mas de 20 años?	S	N	S	N	S	N	S	N
A1: Trato normal				x			x	x
A2: Trato preferencial	x	x	x		x	x		

Ej: El Centro de Vacaciones del Alto Pirineo, dependiente de la DGA, tiene un conjunto de plazas reservadas en una serie de zonas de acampada para disfrute exclusivo de los niños descendientes de habitantes del alto Pirineo Aragonés

Se desea construir un sistema de reservas de plazas que pueda operarse en un PC con la siguiente política de reservas:

1. Hay tres zonas de acampada con las siguientes capacidades:
    1. Campo principal: 50 campistas máximo
    2. Campo secundario: 30 campistas máximo
    3. Campo terciario: 30 campistas máximo
2. Cada zona de acampada esta dividida en el siguiente número de áreas:
    1. Campo principal: 5 áreas
    2. Campo secundario: 3 áreas
    3. Campo terciario: 3 áreas
3. Algunas áreas pueden ser cerradas al público cuando se necesitan para eventos especiales de Agencia.
4. Máximo 10 personas por reserva.
5. Las reservas se realizan con no más de un mes de adelanto. Un periodo de espera de un mes debe transcurrir entre reservas, comenzando a contar desde el día que la llave de acceso es devuelta.
6. Una reserva consiste en una lista de nombres de adultos y niños. Por lo menos el 30% deben ser niños.
7. Las reservas están limitadas a:
    1. 7 días consecutivos durante el periodo escolar
    2. 4 días / 3 noches durante vacaciones escolares
8. Las reservas pueden ser canceladas llamando con al menos dos días de adelanto. Una no presentación es un aviso. El segundo aviso implica una suspensión de tres meses de derecho de reserva. El tercer aviso son seis meses de suspensión.

9. Se solicita un depósito de 9 € a la entrega de la llave. Este depósito es devuelto cuando se devuelve la llave. Las llaves deben ser recogidas en día laborable y no antes de una semana antes de la reserva. Las llaves deben ser devueltas el primer día laborable después de la estancia. El retraso en la devolución de la llave implica la pérdida del depósito. La perdida de una llave implica tres meses de suspensión y pérdida del depósito.

- Problemas de ambiguedad:
  - 1) ¿campo = zona? ¿campista = tienda?
  - 3) no utiliza artículos indeterminados
  - 6) ¿hasta qué edad se considera adulto?
- Problemas de consistencia:
  - 2) con 1): ¿área = 10 campistas?
  - 5) con 3): ¿podemos reservar áreas con reservas ya realizadas?
- Problemas de completitud:
  - 4) ¿cuál es el mínimo?
  - 5) ¿a quién afecta la espera?
  - 6) ¿hay un procentaje mínimo/máximo de adultos?

Ej: *(ver enunciado Aulas Informaticas)*

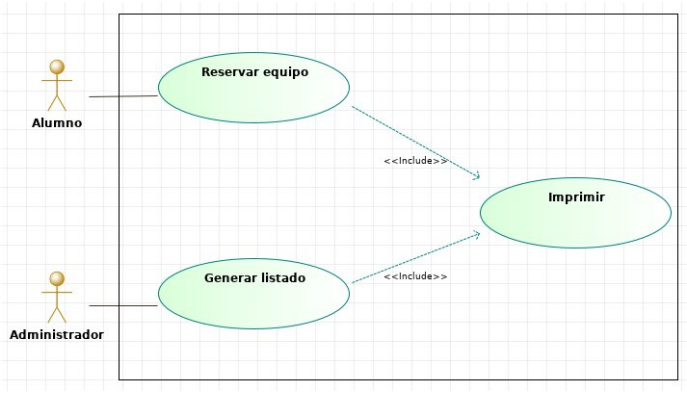

Figura 3.4: Diagrama casos de uso de ejemplo "Aulas Informáticas"

Caso de uso "Reservar equipo"

1. El caso de uso comienza cuando el alumno introduce el nº de expediente
2. El alumno rellena día, hora, equipo
3. El sistema verifica restricciones: máximo de 4 horas semanales, entre 10:00 y 18:00, hora en punto, día lectivo, día posterior a fecha actual
4. *include* Imprimir (nº expediente + nombre + equipo + fecha y hora)

Caso de uso "Generar listado"

1. El caso de uso comienza cuando el administrador introduce la fecha actual
2. El sistema recupera todas las reservas de la fecha del paso 1
3. *include* Imprimir (fecha + lista <nombre, equipo, hora>)

Ej: Se quiere construir una aplicación para gestionar notas en un dispositivo móvil y se han identificado los siguientes requisitos funcionales:

- RF 1. El usuario puede crear una nota.
- RF 2. El usuario puede modificar una nota existente.
- RF 3. El usuario puede eliminar una nota existente.
- RF 4. El usuario puede consultar el listado de notas ya creadas.
- RF 5. Las notas constan de título y cuerpo.

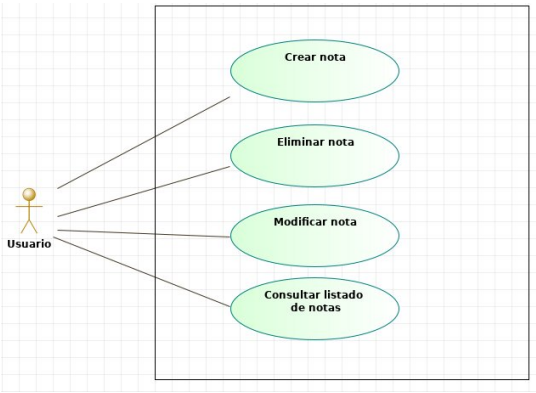

Figura 3.5: Diagrama casos de uso de ejemplo “Gestión notas”

Caso de uso “Nueva nota”

1. El caso de uso comienza cuando el usuario solicita la creación de una nota
2. El usuario da un título a la nota, e introduce un texto para el cuerpo
3. El sistema guarda la nota

Caso de uso “Modificar nota”

1. El caso de uso comienza cuando el usuario selecciona una nota existente y solicita su modificación
2. El usuario modifica título y cuerpo
3. El sistema guarda la nota

Caso de uso “Eliminar nota”

1. El caso de uso comienza cuando el usuario selecciona una nota existente y solicita su eliminación
2. El sistema borra la nota seleccionada

Caso de uso “Consultar notas”

Flujo de eventos principal:

1. El caso de uso comienza cuando se arranca la aplicación
2. El sistema recupera todas las notas
3. El sistema muestra las notas al usuario

Flujos de eventos alternativos:

1. El caso de uso comienza cuando el usuario acaba de crear/modificar/eliminar una nota
2. El sistema recupera todas las notas, incluyendo la que se acaba de crear/modificar, o excluyendo la que se acaba de eliminar
3. El sistema muestra las notas al usuario

3.2 Fase de análisis