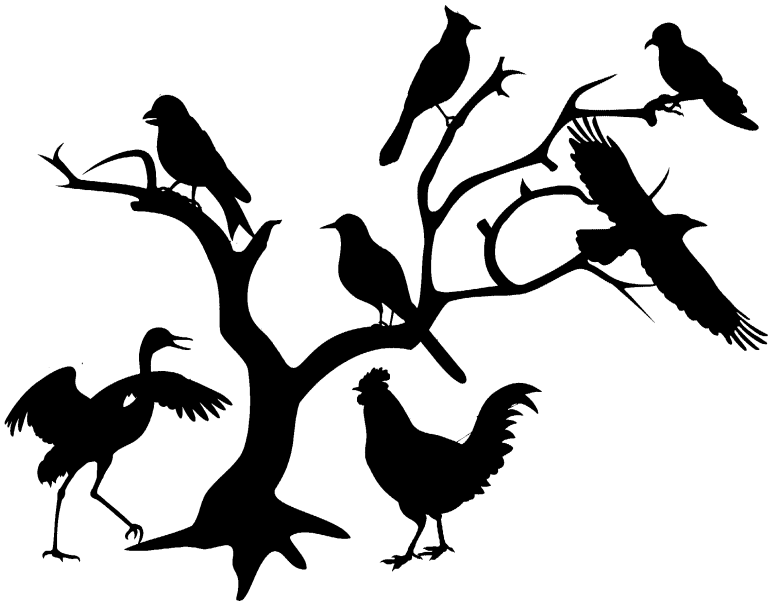
Tema 5º
Análisis Orientado a Objetos

- **El proceso de desarrollo de software**
- **Detalle del proceso de desarrollo de software**
- **Análisis Orientado a Objetos**
- **Documentos de análisis**
- **Especificación de requisitos o requerimientos**
- **Diagramas de casos de uso**
- **Escenarios y sub-escenarios**
- **Prototipos**
- **Otras técnicas de análisis orientado a objetos**
- **Resumen**

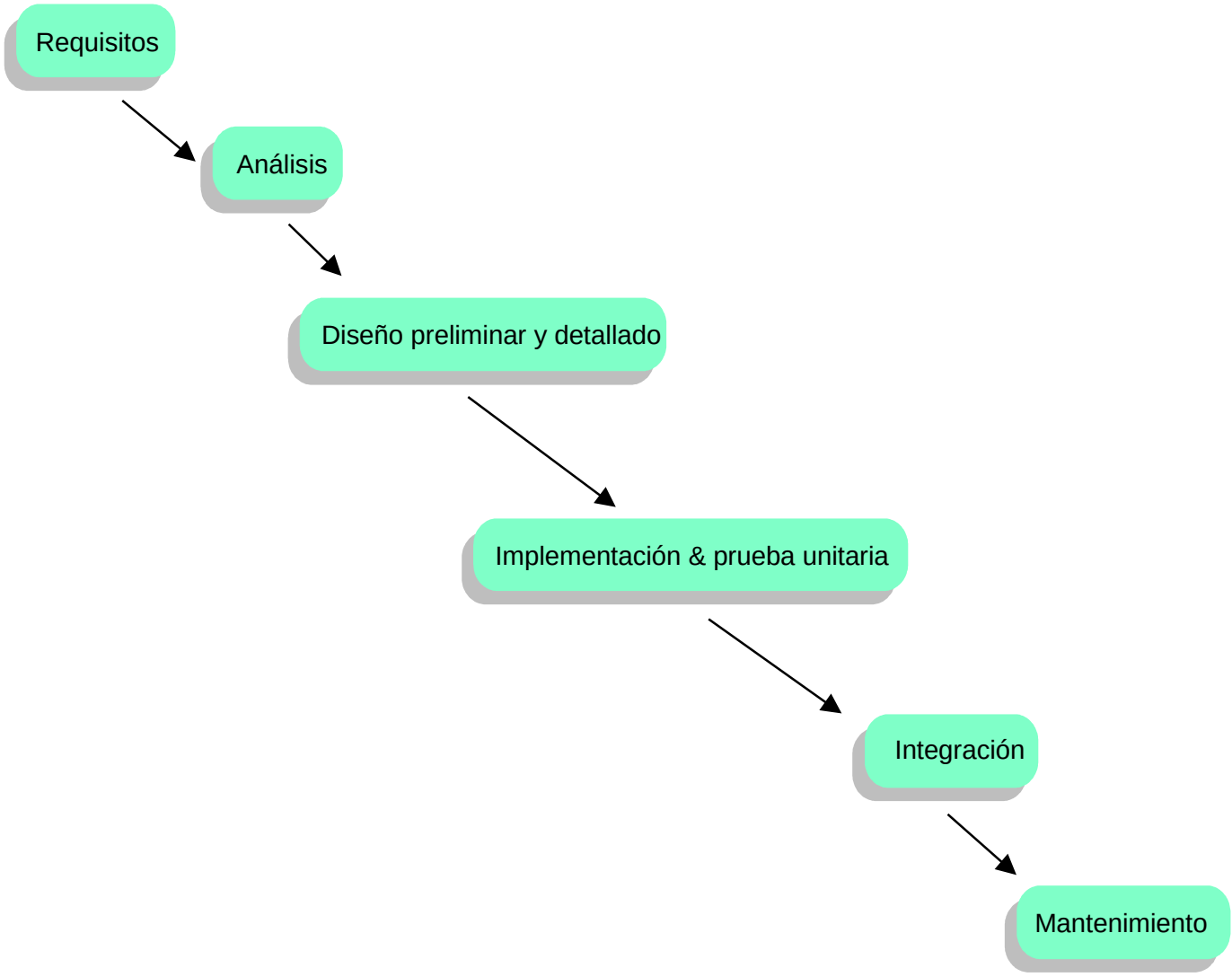


El proceso de desarrollo de software

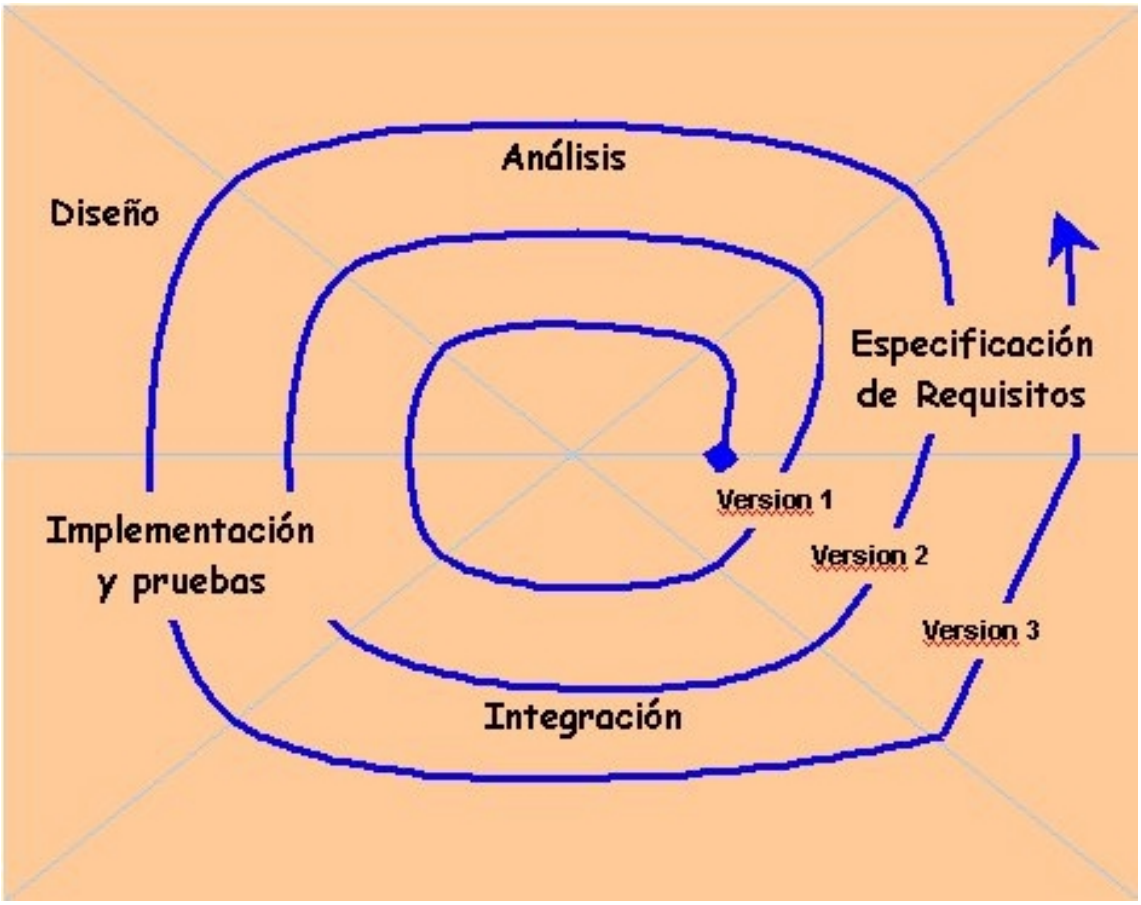
[Booch 94, capítulo 6]

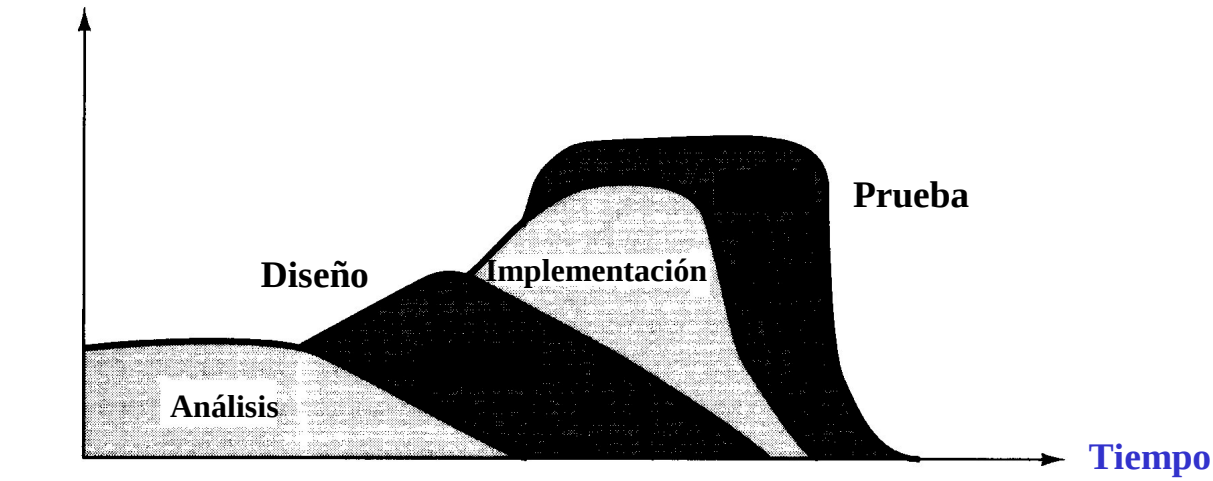
- No hay recetas mágicas, aunque es necesario tener un proceso preceptivo.
- Las características fundamentales de un proyecto con éxito
  - **Buena visión arquitectónica**
    - No existe ningún camino bien definido para idear una arquitectura. Tan sólo se pueden definir los atributos de una buena arquitectura:
      - Capas de abstracción bien definidas
      - Clara separación de intereses entre interfaz e implementación
      - Arquitectura simple
    - Es necesario distinguir entre decisiones estratégicas y tácticas
    - ***Decisiones estratégicas*** *es aquella que tiene amplias implicaciones estratégicas e involucra así a la organización de las estructuras de la arquitectura al nivel más alto*
    - ***Decisiones tácticas*** *son las que sólo tienen implicaciones arquitectónicas locales, es decir sólo involucran a los detalles de interfaz e implementación de una clase*
  - **Ciclo de vida incremental e iterativo**
    - Los ciclos de desarrollo no deben ser anárquicos ni excesivamente rígidos
    - Cada pasada por un ciclo análisis/diseño/evolución lleva a refinar gradualmente las decisiones estratégicas y tácticas, convergiendo en última instancia hacia una solución con los requisitos reales de los usuarios finales (habitualmente no expresados explícitamente por éstos)

El proceso de desarrollo de software
Modelo en cascada



El proceso de desarrollo de software
Modelo en espiral [Jacobson 92]





Detalle de un proceso de desarrollo de software

Aunque el proceso es iterativo el orden de los pasos fundamentales es el siguiente:

- **Análisis**
  - Características comunes de los documentos.
    - Identificación. Título, descripción, versión, fecha, revisión, código del documento..
  - Documentos de análisis
  - Especificación de requisitos o requerimientos
  - Diagramas de casos de uso
  - Escenarios y sub-escenarios
  - Prototipos
- **Diseño (preliminar y detallado)**
  - Modelado de Clases, Objetos y mecanismos de colaboración
    - Diagramas de interacción
      - Diagramas de secuencia
      - Diagramas de colaboración
    - Diagramas de Clases y consulta de patrones de diseño.
    - Diagramas de objetos
  - Modelado del comportamiento de clases y objetos
    - Diagramas de actividades
    - Diagramas de estados
  - Construcción del modelo físico
    - Diagramas de componentes
    - Diagramas de despliegue
- **Implementación**
  - Las decisiones iniciales de implementación se toman a partir de los diagramas de componentes y de despliegue
  - Se implementan las clases de un componente a partir de los diagramas de clases y diagramas de objetos
  - A partir de los diagramas de actividades y de los diagramas de estados se implementa el comportamiento de los métodos de cada clase
- **Prueba**
  - Prueba unitaria de cada clase
  - Prueba de módulos
  - Prueba de integración se realiza siguiendo los escenarios, diagramas de interacción., actividades y estados
- **Mantenimiento**
  - Informes de errores
  - Nueva especificación de requisitos. Nueva versión

Análisis Orientado a Objetos

- **Análisis orientado a objetos (AOO)** *[Booch 94]*
  - *"es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en el vocabulario del dominio del problema"*
- Documentos básicos de análisis orientado a objetos
  - Documentos de análisis
  - Especificación de requisitos o requerimientos
  - Diagramas de casos de uso
  - Escenarios y subescenarios
  - Prototipos y su evaluación
- Todos los documentos deben estar identificados y codificados

Identificación

- Es necesario identificar todos los elementos del proceso de desarrollo de software de una forma unívoca
- Todos los documentos deben estar identificados
- Título
  - debe reflejar de la mejor forma posible sus fines y su funcionalidad
- Descripción
- Autores
- Versión. Notación decimal.
- Revisión. Autores
- Fecha
- Código de cada documento o diagrama

Documentos de análisis

- Contiene la documentación que aporta el cliente que encarga la aplicación
- También contiene las actas de las reuniones de trabajo del grupo de análisis
  - Es necesario un secretario que tome acta
  - Es necesario aprobar el acta de cada reunión por todos los miembros

Ejemplo de documento de análisis

Se debe realizar un sistema capaz de mantener una base de datos con todos los equipos hardware y software de una empresa, de manera que se pueda obtener información acerca del número de licencias instaladas y de los equipos en los que están instaladas dichas licencias.

Además debe ser posible controlar el hardware, las modificaciones efectuadas en los equipos, las averías de dichos equipos, la composición de cada uno de los ordenadores y el software que está instalado en ellos.

Por otro lado cada equipo y cada software que posee la empresa tiene asociados una serie de manuales de los que es necesario seguir la pista, pudiendo, en cada momento, saber qué manuales tiene cada equipo y también cada programa.

Por tanto existen tres elementos importantes implicados en el sistema:

1.El software.

2.El hardware.

3.Los manuales.

Es necesario seguirles la pista a estos tres elementos y saber en todo momento las relaciones entre ellos para poder localizar, mediante el ordenador el manual de un componente instalado en un ordenador.

Para el Software es necesario saber el nombre del producto, la versión, la marca o casa que lo fabrica, la fecha de compra, el precio de compra, el proveedor, el soporte (disquetes, CD-ROM, etc.), el número de elementos del soporte, la localización física del soporte, el número de instalaciones, los equipos en los que está instalado, el número de licencias adquiridas, los manuales que acompañan el producto y la localización física de dichos manuales.

Las localizaciones físicas pueden ser sustituidas por los códigos si se codifican tanto los soportes físicos como los manuales.

El sistema debe ser capaz de contestar a las preguntas:

1.Licencias existentes, en uso y necesarias (si procede) de cada una de las aplicaciones que se estén usando en la empresa.

2.Ordenador u ordenadores (si hay varios) en que reside una aplicación.

3.Composición del paquete original (disquetes, CD-ROM, etc. y manuales).

4.Proveedor que sirvió el programa, fecha y precio de adquisición.

Un detalle importante a tener en cuenta es que existe la posibilidad de que exista software llave-en-mano, y en este caso además hay que saber si se dispone o no de los códigos fuente, la casa que lo desarrolló, quién posee los derechos de copia, el precio, el tiempo de desarrollo y el nombre de la persona responsable del proyecto.

Los software se quedan obsoletos y, por tanto, es necesario actualizarlos. Se debe tener en cuenta que es necesario que el sistema ofrezca una reseña histórica del producto (versiones anteriores) y por tanto es necesario saber el estado de cada uno de ellos (activo, actualizado, desechado, en preparación, pedido, etc.)

En el caso de los antivirus y otros programas similares, es necesario obtener regularmente una adaptación, por lo que es importante que el sistema nos avise de la inminencia de la caducidad de dichos sistemas.

Ejemplo de Documento de Análisis (continuación ...)

De cada ordenador se necesita saber su composición (monitor, teclado, ratón y unidad central). De esta última es necesario saber su composición (VGA, disco duro, disquete, placa madre, procesador(es), memoria RAM, memoria caché, etc.).

Cada uno de los cuatro componentes principales estará codificado adecuadamente para permitir el intercambio de dichos equipos entre los diferentes puestos de ordenador, de manera que la asociación no sea fija. De ellos es necesario saber (cuando exista) la marca, el modelo, el número de serie, y otras características particulares (por ejemplo, del monitor la resolución, si es o no *Energy*, etc.)

Además de ordenadores existen otros equipos: impresoras, plotters, scaners y unidades de almacenamiento (ZIP, CD y Magneto-ópticos) de los cuales es necesario saber, al menos, la marca, el modelo, el número de serie y una breve descripción de sus características.

De cada uno de los equipos es necesario tener un reseña histórica de sus averías y cambios, así como una estimación de su precio (en función del precio de compra de cada uno de sus componentes).

En el caso de los ordenadores es necesario saber el software que tienen instalado (comenzando por el sistema operativo) y debe ser posible seguir la pista de los manuales de cada una de sus partes componentes.

De los manuales sólo es necesario controlar su código, su ISBN (si lo posee), su precio (si es aparte del paquete) y su título. Pero es imprescindible poder obtener información del software y hardware que está relacionado con ellos.

Los manuales deben de ser actualizados según se vaya cambiando el software y el hardware .

El programa debe ser capaz de gestionar el sistema desde diferentes puntos de vista: responsable de informática, mantenimiento y usuario.

El responsable de informática es el encargado de comprar todos los componentes (equipos, software y manuales). Además da estos equipos de alta y debe poder apoyarse en el sistema para gestionar los pedidos.

Para esta última labor debe poder anotar en el sistema que ha pedido un componente a un proveedor (por tanto que está pendiente de recibir), confirmando en la recepción este pedido, además tendrá la capacidad de poder anular un pedido o si se da el caso anularlo

Ejemplo de Documento de Análisis (continuación ...)

Además debe poder obtener informes de inventario de los equipos tasados por el precio de compra menos una amortización del 25% anual (que dejaría al equipo sin valor pasados cuatro años) para los procesadores y del 10% anual para el resto de los equipos.

Además debe poder obtener informe de la composición de cada equipo, del estado de disponibilidad de cada uno de ellos y de el estado con respecto a la garantía del equipo.

El responsable de informática es la única persona que puede dar de alta, modificar y dar de baja los equipos.

La baja de un equipo se dará en el momento en que se avise de la avería y si el equipo no tiene arreglo lo daremos de baja permanente.

Además debe poder obtener toda la información que tienen el resto de los usuarios del sistema (responsable de mantenimiento y usuarios), y tendrá acceso a un buzón de sugerencias sobre el sistema.

El responsable de informática se encarga además de reservar un equipo cuando se solicita por cualquier usuario, para ello tiene que obtener los informes de disponibilidad y composición de equipos.

El responsable de mantenimiento debe poder anotar en todo momento las averías de cada equipo (fecha, hora de comienzo, hora de fin, nombre del mecánico y descripción). Debe poder anotar que un equipo no tiene reparación. Además debe poder obtener informes acerca del tiempo de inactividad de los equipos y acceso al buzón de sugerencias.

El usuario debe poder obtener información de los manuales, software y hardware asociado y disponibilidad de un equipo en concreto . Tendrá acceso al buzón de sugerencias.

Se debe tener en cuenta que:

* El sistema trabajará en un entorno multiusuario.
* Las bases de datos serán un modelo estándar.
* Cada equipo estará compuesto por un monitor, un teclado, un ratón, y una unidad central.
*La unidad central estará formada por una serie de componentes que se describirán de manera textual en un campo al efecto.

Especificación de requisitos o requerimientos

- *"La captura de requisitos es el proceso de averiguar, normalmente en circunstancias difíciles, lo que que se debe construir"* [Jacobson 1999, capítulo 6]
- La captura de requisitos es complicada
  - Los usuarios habitualmente no saben expresar exactamente lo que quieren
  - Es difícil tener una visión global del problema a resolver
- La especificación de requisitos es un documento más técnico y elaborado de los documentos de análisis
- Es importante codificar los requisitos para poder seguirlos a lo largo del proceso de desarrollo de software
- Se puede utilizar una especificación jerárquica
  - Están todos codificados por niveles, al igual que las leyes.
  - Se desea que en las actas quede reflejado lo más exactamente posible el problema a resolver, y que en las reuniones de análisis se determine exactamente que requisitos se añaden o se eliminan
  - Los requisitos relacionados se organizan dentro de un mismo nivel
  - Cada nivel 1 se puede hacer corresponder posteriormente con un caso de uso
  - Cada nivel 2 se puede hacer corresponder posteriormente con un escenario
  - Cada nivel 3 se puede hacer corresponder posteriormente con un sub-escenario

Ejemplos de requisitos

R.0 Requisitos generales

R.0.1 Tendremos en cuenta trabajar con fechas que codifiquen el año con cuatro cifras.

R.0.2 Las unidades monetarias deberán poder trabajar con cifras decimales con vistas al inmediato cambio de moneda que se nos aproxima.

R.1 Gestión de clientes

R.1.0 Requisitos generales de los clientes

R.1.0.1 Los clientes pueden ser fijos o eventuales

R.1.0.2 A los clientes se les asigna un número identificativo

R.1.0.3 Los clientes se definen por D.N.I., nombre, dirección, ciudad, teléfono, y departamento.

R.1.1 Añadir clientes

R.1.1.1 Solamente los Usuarios con permiso de Administrador podrán añadir clientes fijos.

R.1.2 Borrado de clientes.

R.1.2.1 Solamente los Usuarios con permiso de Administrador podrán borrar clientes.

R.1.2.2 Para poder borrar clientes es necesario que este no tenga ningún albarán pendiente de facturación y que estos tengan una antigüedad mayor a cinco años.

R.1.2.3 También es necesario que no tenga ninguna máquina reparándose

R.1.3 Modificar clientes.

R.1.3.1 Solo los usuarios con permiso de Administrador pueden modificar los datos de un cliente.

R.1.4 Buscar clientes.

R.1.5 Paso de cliente fijo a cliente eventual y viceversa.

R.1.5.1 Solo los usuarios con permiso de Administrador pueden hacer clientes fijos.

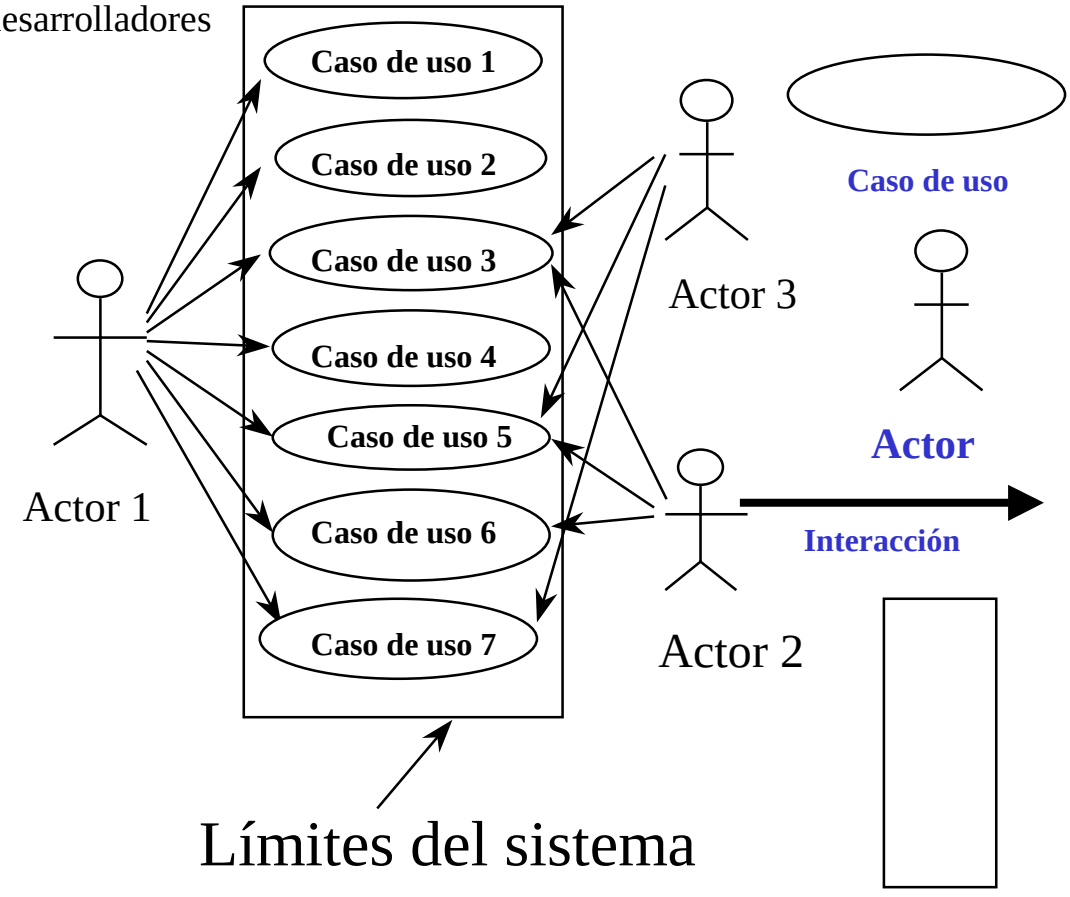
Diagramas de Casos de Uso (I)

[Booch 1999, capítulo 17] [Jacobson 1999, capítulo 3] [Schneider 1998]

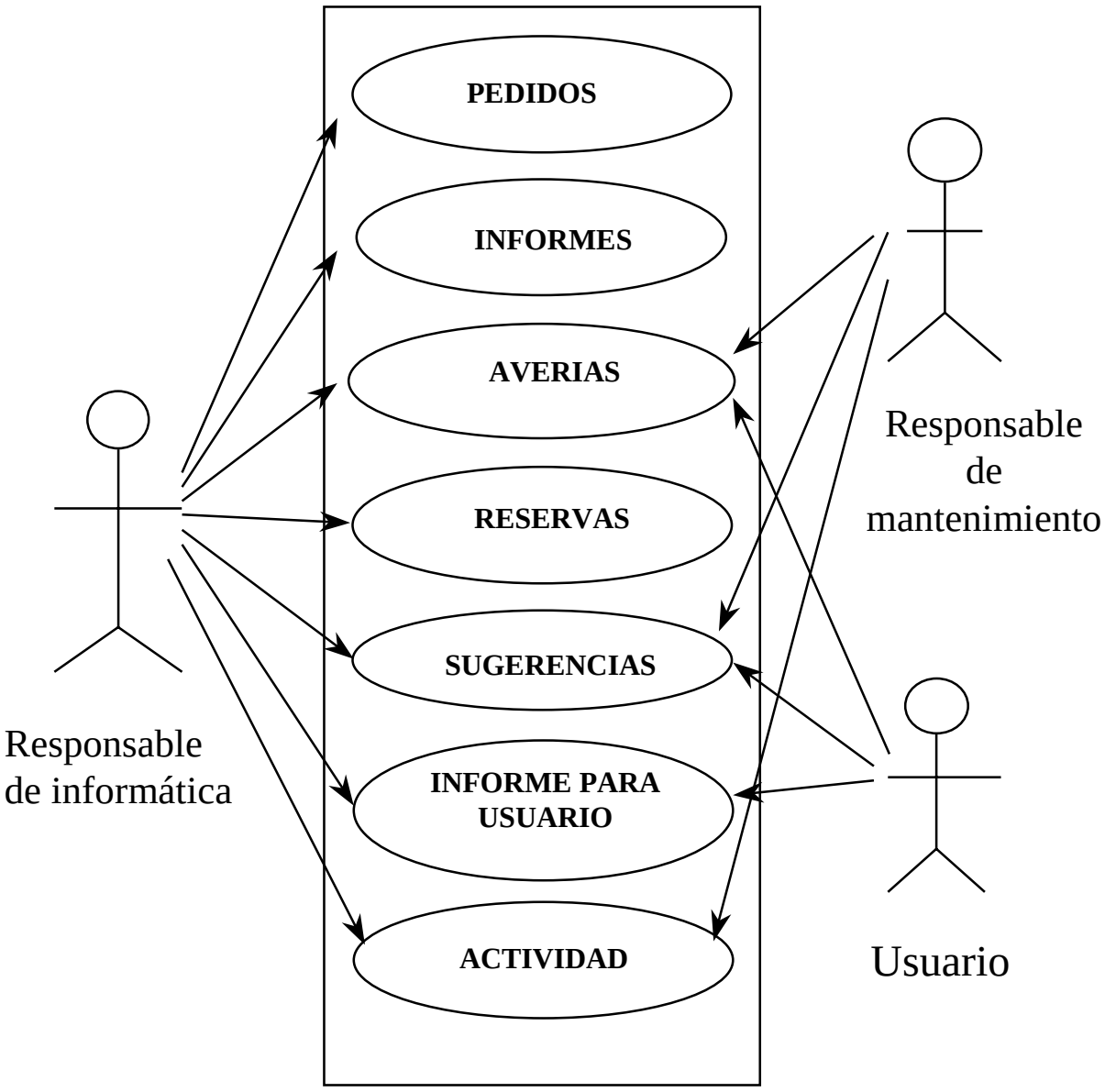
- Es uno de los cinco tipos de diagramas de UML que se utilizan para el modelado de los aspectos dinámicos de un sistema.
- Se corresponden inicialmente con requisitos de primer nivel. Posteriormente se van modelando requisitos de los siguientes niveles.
- Se suelen codificar con el mismo código que el requisito, para hacer más patente su correspondencia.
- Un caso de uso es una técnica de modelado utilizada para describir lo que un nuevo sistema debe hacer o lo que un sistema existente ya hace.
- Los casos de uso representan una vista externa del sistema
- Un modelo de casos de uso se construye mediante un proceso iterativo durante las reuniones entre los desarrolladores del sistema y los clientes (y/o los usuarios finales) conduciendo a una especificación de requisitos sobre la que todos coinciden.
- Un caso de uso captura algunas de las acciones y comportamientos del sistema y de los actores
- El modelado con casos de uso fue desarrollado por Ivar Jacobson [Jacobson 1992]

Diagramas de Casos de Uso (II)

- El sistema que se desea modelar se representa encerrado en un rectángulo
- Los actores son los que interactúan con el sistema. Representan todo lo que necesite intercambiar con el sistema.
  - Un actor es una clase
- Se diferenciará entre actores y usuarios.
  - Un usuario es una persona que utiliza el sistema
  - Un actor representa el papel (rol) que una persona desempeña
  - Por ejemplo una persona puede ser usuario y administrador en un sistema, unas veces actuará como usuario y otras como administrador, pero deben contemplarse ambos actores.
- Los Casos de Uso es un camino específico para utilizar el sistema
- Para cada Caso de Uso, Actor y Sistema se realiza una descripción detallada
- Los Casos de Uso tan sólo indican opciones generales
- El diagrama de Casos de Uso es un diagrama sencillo que tiene como finalidad dar una visión global de toda la aplicación de forma que se pueda entender de una forma rápida y gráfica tanto por usuarios como por desarrolladores



Ejemplo de Casos de Uso



Ejemplo de descripción de los Casos de Uso

1.PEDIDOS

Escenario general donde se realizan todas las operaciones relativas a pedidos: hacer, recibir, anular y devolver pedidos. Todo es realizado por el Responsable de Informática.

2.INFORMES

Todos los informes que son necesarios para el funcionamiento de la empresa: informe de pedido, de amortizaciones, de inactividad, de composición de equipos básicos, de composición de otros equipos, de inventario software y manuales, de garantías y de disponibilidad. Estos informes son realizados para el Responsable de Informática.

3.AVERIAS

Engloba todas ls operaciones relativas a las averías tanto el aviso que puede ser realizado por cualquier actor (Responsable de Informática, de Mantenimiento o Usuario) , como el parte de avería que es realizado por el Responsable de Mantenimiento y entregado al Responsable de Informática.

4.RESERVAS

Es tanto la petición de reserva de un equipo con unas características determinadas, que puede ser realizada por cualquier usuario al Responsable de Informática, como la concesión de una reserva que realiza este último a un usuario.

5.SUGERENCIAS

Es una línea de comunicación entre los diferente agentes que interaccionan con el sistema.

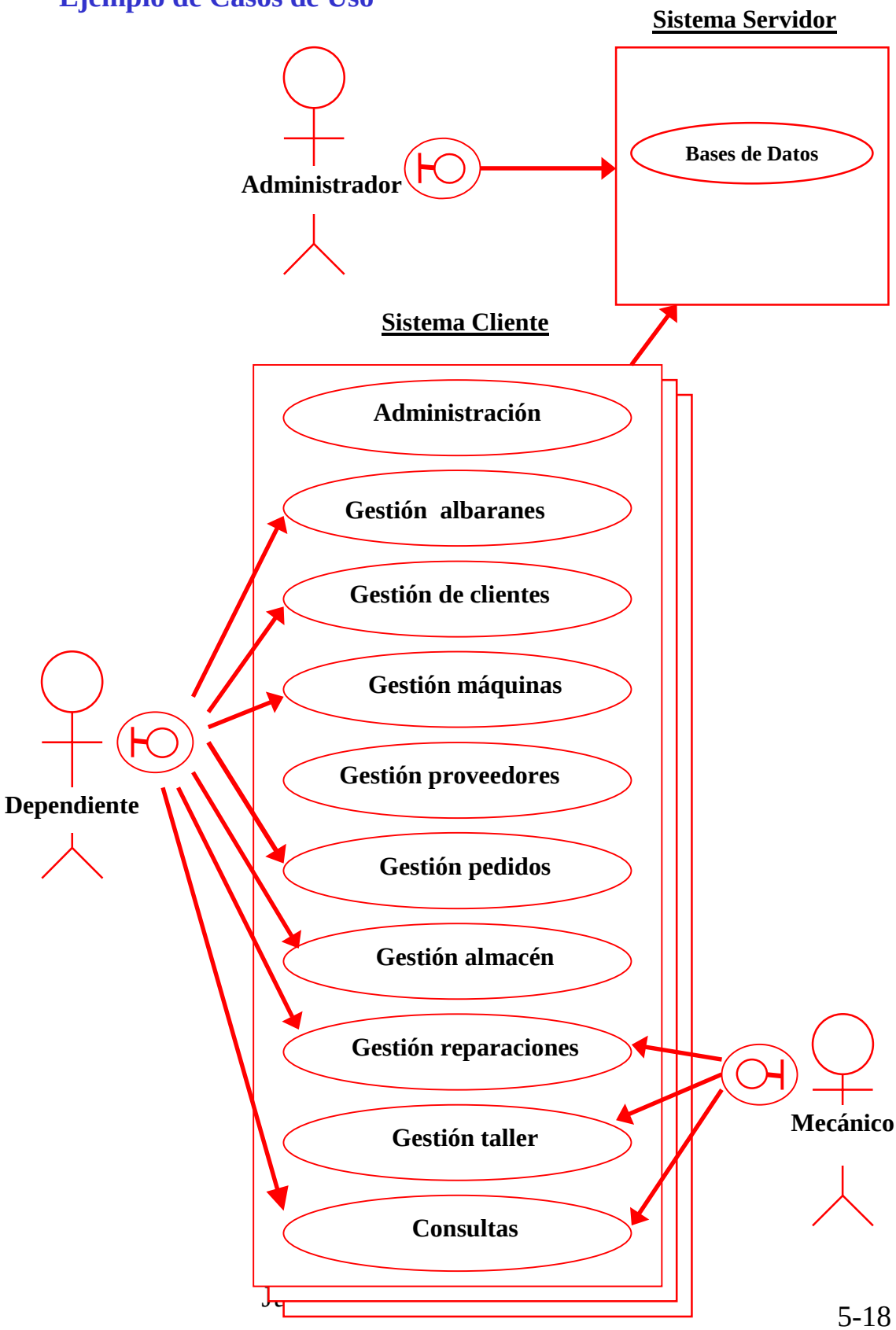
6.INFORMES PARA EL USUARIO

Es un informe especialmente realizado para el usuario donde este puede encontrar toda la información que pueda necesitar en un momento determinado sobre un equipo, su disponibilidad, software o un manual.

7.ACTIVIDAD

Realizado por el Responsable de Informática engloba todo lo relativo al buen funcionamiento del material de la empresa: dar de baja temporalmente un equipo cuando esta en reparación, dar de baja permanentemente un equipo cuando no tiene arreglo y actualizar tanto software como los manuales.

Ejemplo de Casos de Uso



Descripción de actores

Nombre de Actor: Administrador

Definición: Es el encargado de administrar el sistema. Tendrá todos los permisos y libertad de movimientos por el sistema.

Notas:

- El administrador es el encargado de manipular la información contenida en el sistema.
- Tiene acceso a toda la información del sistema y es el único que puede modificar todo lo que le de la gana..

Nombre de Actor: Mecánico

Definición: Es el encargado de realizar las oportunas reparaciones en las máquinas y a su criterio y valoración queda el tomar las decisiones oportunas respecto a que reparación y si es necesario o no el ingreso de la máquina en el taller.

Notas:

- No lo encajamos en la figura de una persona concreta sino que pueden ser varias personas las que puedan encargarse de esta tarea.
- El mecánico solo tiene acceso a la parte del sistema referente a las máquinas a las reparaciones y al las consultas, el acceso al resto le está vedado.
- Dentro de la parte del sistema al que puede acceder no se le permite el borrado de información, solo, añadir y modificar.

Nombre de Actor: Dependiente

Definición: Es la persona que está en contacto directo con los clientes. Tiene acceso limitado a las operaciones del sistema.

Notas:

- El dependiente no podrá dar de alta a clientes fijos, pero si a clientes eventuales.
- No podrá borrar clientes, ni máquinas, ni artículos, ni reparaciones.
- No tiene acceso a la gestión de proveedores ni del taller

Sistemas

SISTEMA SERVIDOR

El Sistema Servidor, (en nuestro caso particular) estará formado por una máquina Windows NT Server (también podría ser una máquina Unix) que será atacado por sistemas Clientes constituidos por máquinas Windows 95 ó Windows NT.

El gestor de la base de datos (CTSQL de MultiBase...), junto a la propia base de datos, se encuentran en el servidor UNIX o Windows NT (solo para el CTSQL).

Existirá también la posibilidad de configurarlo en una sola máquina en Windows 95 ó Windows NT, en cuyo caso los programas de la aplicación, el gestor de la base de datos y la base de datos residirían en la misma máquina Windows.

SISTEMAS CLIENTES

Los programas de la aplicación residen en la máquina «cliente» Windows95 o Windows NT que atacan al servidor donde se encuentra instalada el gestor de la base de datos junto con la base de datos correspondiente.

INTERFACES

INTERFAZ ADMINISTRADOR

El interfaz del Administrador le permite acceder a todas las opciones que presenta la aplicación

INTERFAZ DEPENDIENTE

El Dependiente solamente tendrá acceso a algunas de las funciones que soporta la aplicación y dentro de estas su capacidad de maniobra estará limitada

INTERFAZ MECÁNICO

El mecánico tendrá acceso solamente a las reparaciones y a la gestión del taller además de las consultas.

Ejemplo de descripción de casos de uso

Nombre del Caso de Uso 1: Gestión Clientes

Definición: Actualizaciones de la información acerca de los clientes de Comercial Quirós

Notas:
- Los clientes vienen definidos por: número_cliente, DNI, nombre, dirección, ciudad, teléfono y el departamento para el que trabajan.
- En el caso de que no pertenezca a un departamento determinado este aparecerá en blanco.
- Los clientes pueden ser fijos o eventuales.

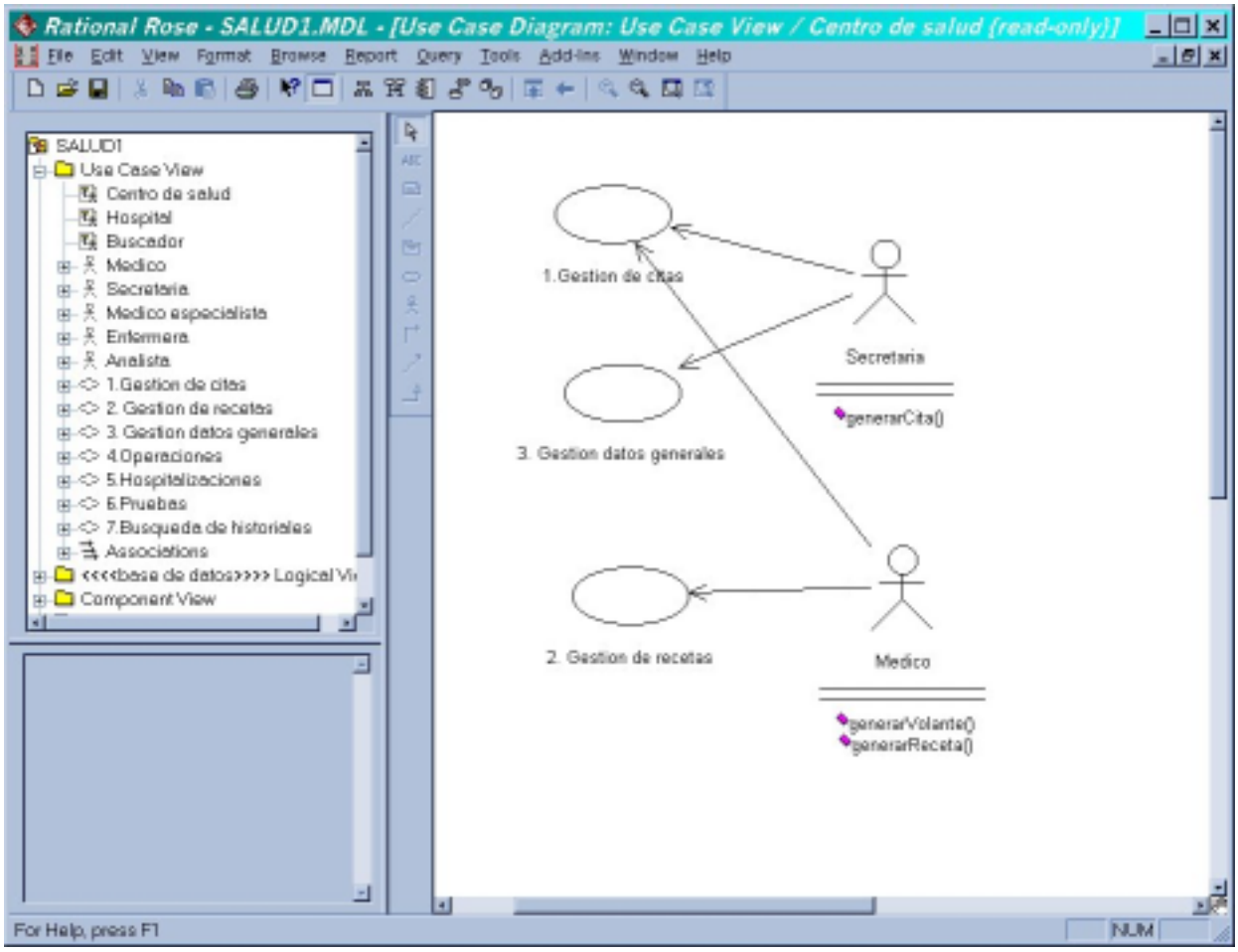
Nombre del Caso de Uso 2: Gestión Máquinas

Definición: Actualizaciones de la información acerca de las máquinas de los clientes de Comercial Quirós, y de las cuales nos encargamos de su mantenimiento y reparación.

Notas:
- Las máquinas vienen definidas por: *nº identificatorio, número_cliente, tipo, marca y modelo.*
- Con *número_cliente* relacionamos la máquina con el cliente al que pertenece.
- Un mismo cliente puede tener varias máquinas.
- El *tipo* nos dice a que clase de máquina pertenece (fotocopiadora, fax...)

Casos de uso en Rational Rose ®

• Tiene una sección para ir introduciendo los Casos de Uso (*Use Case View*)
• Permite el manejo de actores, que se traducirán al sistema como clases
• Cada sistema recibe un nombre (no aparece el rectángulo) y está ligado a una ventana



Escenarios y sub-escenarios

- Cada caso de uso da lugar múltiples escenarios
- Se codifican siguiendo la codificación de los casos de uso
- Se estudia cada escenario utilizando guiones como los que se usan en el cine
- Cada equipo que pasa por un escenario identifica los objetos y sus responsabilidades, así como los mecanismos que relacionan los objetos
- De los escenarios iniciales se puede pasar a otros escenarios secundarios
- Los escenarios también se pueden utilizar para probar el sistema en la fase de pruebas
- El estudio de los escenarios con detalle permitirá enriquecer el Diccionario de clases
- No es necesario hacer todos los escenarios y sub-escenarios posibles si se observa que no enriquecen el diccionario de clases

Numeración: 1.1.
Titulo: Hacer pedido
Precondiciones: Sugerencias de compra, caducidad de licencias, bajas permanente hardware, …
Quien Lo Comienza: Responsable de Informática.
Quien Lo Finaliza: Responsable de Informática.
Postcondiciones:
Excepciones:
Descripción: Son las operaciones de compra de todos los componentes (hardware, software y manuales) que realiza el responsable de informática. Además da estos equipos de alta y debe apoyarse en el sistema para gestionar los pedidos correctamente para lo que debe anotar en el sistema que ha pedido un componente a un proveedor (por tanto que está pendiente de recibir).

Numeración: 1.2.
Titulo: Anular pedido
Precondiciones: Cambio de precio, cambio de necesidades de la Empresa.
Quien Lo Comienza: Responsable de Informática
Quien Lo Finaliza: Responsable de Informática
Postcondiciones:
Excepciones
Descripción: Esta operación la realiza el responsable de informática cuando toma la decisión de anular un pedido que había realizado con anterioridad.
de informática confirma la recepción de los pedidos.

Ejemplo de escenarios

Caso de uso 1: Gestión de Clientes

Nombre de Escenario 1.1: Dar de alta un cliente eventual

Precondiciones:
- No existe ficha de cliente.

Postcondiciones:
- Todos los datos se han introducido correctamente.
- El numero de clientes se incrementa en uno

Excepciones:

Iniciado por: Dependiente/Administrador.

Finalizado por: Dependiente/Administrador.

Detalle operaciones:
- Cliente acude a una tienda de la compañía.
- Dependiente (ó Administrador) obtiene datos de cliente.
- Dependiente (ó Administrador) introduce ficha en el sistema con los datos *número, dni, nombre, dirección, ciudad, teléfono,* y *departamento.*

Nombre de Escenario 1.2: Dar de alta un cliente fijo.

Precondiciones:
- No existe ficha de cliente.

Postcondiciones:
- Todos los datos se han introducido correctamente.
- El número de clientes se incrementa en 1

Excepciones

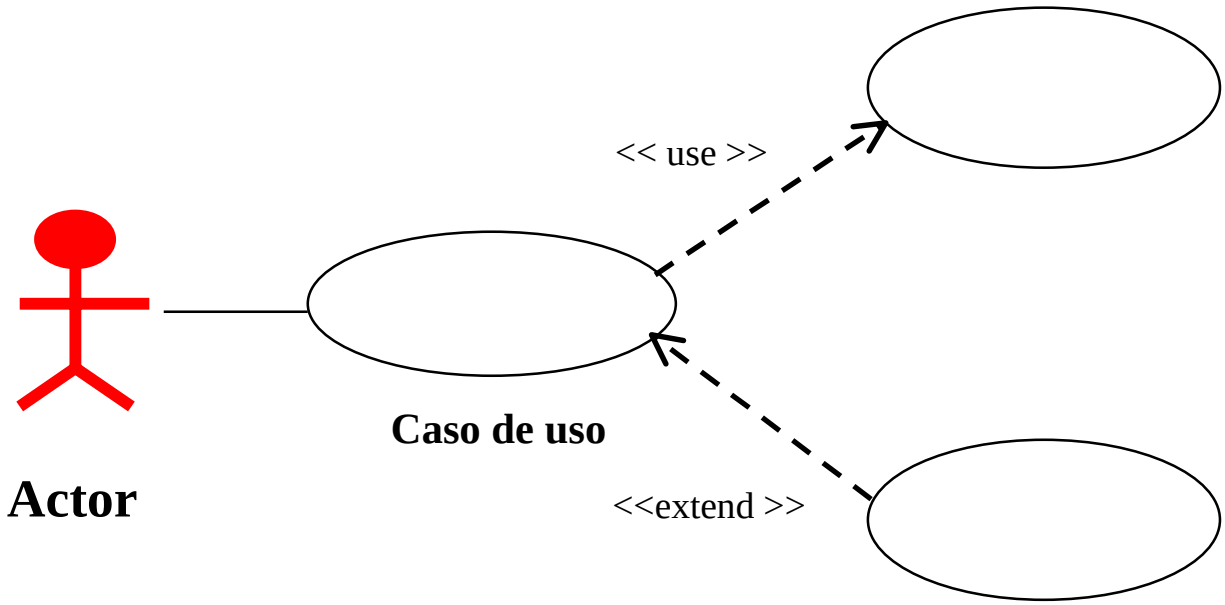
Iniciado por: Administrador.

Finalizado por: Administrador.

Detalle operaciones:
- Cliente acude a una tienda de la compañía.
- Administrador obtiene los datos del cliente.
- Administrador introduce ficha en el sistema con los datos *número, dni, nombre, dirección, ciudad, teléfono,* y *departamento.*

Diagramas de Casos de Uso (III)

- Un **caso de uso** es la típica interacción entre un usuario y un sistema informático
- Un **actor** es el papel que el usuario juega con respecto al sistema. Un actor no tiene que ser un humano, puede ser por ejemplo otro sistema externo que pide información al sistema actual
- La relación <<***extend***>> se utiliza cuando un caso de uso es similar a otro caso de uso pero se le añade alguna característica nueva
- La relación << ***use*** >> se utiliza cuando se tiene una parte del comportamiento común a más de un caso de uso, y no se desea almacenar una copia en cada caso de uso de la descripción de este comportamiento.



Prototipos

[Piattini 96]

- El prototipado consiste en la elaboración de un modelo o maqueta del sistema que se construye para evaluar mejor los requisitos que se desea que cumpla
- Es particularmente útil cuando:
  - El área de la aplicación no está bien definida, bien por su dificultad o bien por falta de tradición en su automatización.
  - El coste del rechazo de la aplicación por los usuarios, por no cumplir sus expectativas, es muy alto.
  - Es necesario evaluar previamente el impacto del sistema en los usuarios y en la organización.
- Estos modelos o prototipos suelen consistir en versiones reducidas, demos o conjuntos de pantallas (que no son totalmente operativos) de la aplicación pedida. Existen tres razones principales para emplear prototipado, ordenadas por frecuencia de uso:
  - **Prototipado de la interfaz de usuario** para asegurarse de que esta bien diseñada, que satisface las necesidades de quienes deben usarlo. Este tipo de prototipado es bastante frecuente, no cuesta mucho y puede consistir en simples modelos de pantallas en papel, simuladas con programas de dibujo o presentación o auténticas simulaciones muy elaboradas de la interfaz. No suele resultar más caro que el trabajo tradicional y es muy efectivo para evitar los múltiples cambios que suelen solicitar los usuarios en este aspecto.
  - **Modelos de rendimiento** para evaluar el posible rendimiento de un diseño técnico, especialmente en aplicaciones críticas en este aspecto. Estos modelos tienen un carácter puramente técnico y, por lo tanto, no son aplicables al trabajo de análisis de requisitos.
  - **Prototipado funcional**. Cada vez más utilizado, está relacionado con un ciclo de vida iterativo. En este caso, en vez de seguir el procedimiento habitual (tirar el prototipo una vez probado y empezar a desarrollar la aplicación), el prototipo supone una primera versión del sistema con funcionalidad limitada. A medida que se comprueba si las funciones implementadas son las apropiadas, se corrigen, refinan o se añaden otras nuevas hasta llegar al sistema fina

Otras técnicas de análisis orientado a objetos

- **Análisis OO mediante fichas CRC (Clases/Responsabilidades/Colaboradores)**
- **Descripción informal en lenguaje natural**

Análisis mediante fichas CRC (Clases/Responsabilidades/Colaboradores)

[Wirfs-Brock 1990] [Wilkinson 1995, capítulo 4] [Bellin 1997]

- Es una forma simple de analizar escenarios
- Son muy útiles para la enseñanza del AOO, DOO y POO
- Facilitan las “tormentas de ideas” y la comunicación entre desarrolladores
- Se crea una ficha para cada clase que se identifique como relevante en el escenario
- A medida que el equipo avanza puede dividir las responsabilidades
- Las fichas CRC pueden disponerse espacialmente para representar relaciones de colaboración
- Las fichas CRC también se pueden colocar para expresar jerarquías de generalización/especialización
- No están contempladas en UML

Ficha CRC (anverso y reverso)

Clase
Responsabilidades
Colaboraciones

Clase
Superclase Subclase
Atributos

Ejemplo de ficha CRC

Clase: Reunión
Responsabilidades Planificar Comprobar la sala asignada Conocer hora de comienzo Conocer la fecha Conocer número de asistentes Conocer equipamiento necesario
Colaboraciones Sala de conferencias Organizador de reuniones

Clase: Reunión
Superclase: Subclases: Reunión de trabajo, Junta de Escuela, Clase de un curso
Atributos: Orden del día Lugar Fecha Hora de inicio Asistentes Equipo necesario

Descripción informal en lenguaje natural

[Abbott 1983]

• Subrayar los nombres y los verbos de la descripción del problema

•Los **nombres** representan los **objetos** candidatos

•Los **verbos** representan las **operaciones** candidatas

–Ventajas

- •Obliga al desarrollador a trabajar sobre el vocabulario del espacio del problema
- •Es sencillo
- •Es didáctico

– Inconvenientes

- •No es riguroso, al ser el lenguaje natural ambiguo
- •Es compleja su aplicación a grandes proyectos

Resumen

- No existen recetas fáciles para el análisis de software
- La correcta definición de los requisitos y su seguimiento en el proceso de desarrollo es uno de los factores fundamentales de la calidad del software
- Los escenarios son una potente herramienta para el análisis orientado a objetos, y pueden utilizarse para guiar los procesos de análisis clásico, análisis del comportamiento y análisis de dominios.
- Las abstracciones clave reflejan el vocabulario del dominio del problema y pueden ser descubiertas en el dominio del problema, o bien ser inventadas como parte del diseño.
- Los mecanismos denotan decisiones estratégicas de diseño respecto a la actividad de colaboración entre muchos tipos diferentes de objetos
- El único artefacto que tiene UML a nivel de análisis son los Diagramas de Casos de Uso.

EJERCICIOS PROPUESTOS

- 5.1 Realizar el análisis del juego del ajedrez. Se puede jugar dos personas entre sí o una persona contra el ordenador. En este último caso debe ser posible seleccionar el nivel de dificultad entre una lista de varios niveles. El juego de ajedrez permitirá al jugador elegir el color de las piezas. La aplicación deberá permitir detectar los movimientos ilegales de las piezas, tiempos utilizados cada jugador, registro de jugadas y piezas perdidas. También determinará si se alcanzan tablas, y permitirá abandonar la partida a un jugador.
- 5.2 Realizar el análisis de una aplicación que realiza estudios de mercado para situar grandes superficies (hipermercados). Se supone que cada gran superficie necesita un mínimo de población que pueda acceder a dicho hipermercado en menos de un tiempo dado. La red de carreteras se puede representar mediante un grafo.
- 5.3 Realizar el análisis para gestionar los fondos bibliográficos y de socios de una biblioteca por Internet.
- 5.4 Realizar el análisis para gestionar un centro de enseñanza (profesores, asignaturas, alumnos, matriculación, calificaciones de cada asignatura, expediente,...) por Internet.

Referencias Bibliográficas

- [Abbott 1983] Abbott, R.J.Program Design by Informal English Descriptions. *Comunications of the ACM, 26(11),882-894. 1983.*
- [Bock/Odell 1994] C. Bock and J. Odell, "A Foundation For Composition," *Journal of Object-oriented Programming*, October 1994.
- [Booch 1994] G.Booch. *Object-oriented analysis and design with applications.* Benjamin Cummings (1994). Versión castellana: *Análisis y diseño orientado a objetos con aplicaciones.* 2ª Edición. Addison-Wesley/ Díaz de Santos (1996).
- [Booch 1999] G. Booch, J. Rumbaugh, I. Jacobson. *The unified modeling language user guide.* Addison-Wesley (1999). Versión castellana *El lenguaje unificado de modelado.* Addison-Wesley (1999)
- [Bellin 1997] D. Bellin and S. Suchman Simone.*The CRC Card book.* Addison-Wesley, 1997
- [Coad 1991] P. Coad, E. Yourdon. *Object-Oriented Analysis*. Second Edition .Prentice-Hall, 1991.
- [Cook 1994] S. Cook and J. Daniels, *Designing Object Systems: Object-oriented Modelling with Syntropy*, Prentice-Hall Object-Oriented Series, 1994.
- [Eriksson 1998] H-E Eriksson & M. Penker. *UML Toolkit. Wiley, 1998.*
- [Fowler 1997] M. Fowler with K. Scott, *UML Distilled: Applying the Standard Object Modeling Language*, ISBN 0-201-32563-2, Addison-Wesely, 1997. Versión castellana *UML gota a gota*, Addison-Wesley 1999.
- [Jacobson 1992] I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard. *Object-Oriented Software Enginneering. A use Case Driven Approach*., ISBN 0-201-54435-0, Addison-Wesely, 1992.
- [Jacobson 1999] I. Jacobson, G. Booch, J. Rumbaugh. *The unified software development process*. Addison-Wesley (1999).Versión Castellana. *El Proceso Unificado de Desarrollo de Software*. Prentice-Hall, 2000.
- [Lee 1997] R. C.Lee & W. M. Tepfenhart. *UML and C++, Prentice-Hall, 1997*
- [Piattini 1996] M.G. Piattini, J.A. Calvo-Manzano, J. Cervera, L. Fernández. *Análisis y diseño detallado de aplicaciones de gestión*. RA-MA (1996)
- [Rumbaught 1991] Rumbaught J., Blaha M., Premerlani W., Wddy F., Lorensen W. *Object-oriented modeling and design.* Prentice-Hall (1991). Versión castellana: *Modelado y diseño orientado a objetos. Metodología OMT*. Prentice-Hall (1996)
- [Rumbaught 1999] Rumbaught J., I. Jacobson, G. Booch. *The Unified Modeling Language Reference Manual.* Addison-Wesley (1999). Versión castellana. *El Lenguaje Unificado de Modelado. Manual de Referencia.*Addison-Wesley, 2000.
- [Reenskaug 1996] T. Reenskaug. *Working with Objects. The Ooram Software Engineering Method.*Prentice-Hall, 1996
- [Schneider 1998] G. Schneider, J. Winters. *Applying Use Cases: A Practical Approach.* Addison-Wesley, 1998.
- [Wilkinson 1995] N. M. Wilkinson. *Using CRC Cards. An Informal Approach to Object-Oriented Development*, 1995, SIGS BOOKS, ISBN 1-884842-07-0
- [Wirfs-Brock 1990] R. Wirfs-Brock, B. Wilkerson y L. Wiener. *Designing Object-Oriented Software.*Pentice-Hall, 1990.

Referencias en la web

- [Coad] Peter Coad http://www.togethersoft.com/
- [OMG] Object Management Group, http://www.omg.org
- [ROSE] Herramienta Rational Rose http://www.rational.com
- [Shlaer-Mellor] Shlaer-Mellor Object-Oriented Analysis http://www.projtech.com
- [UML] UML en www.rational.com y en http://www.omg.org