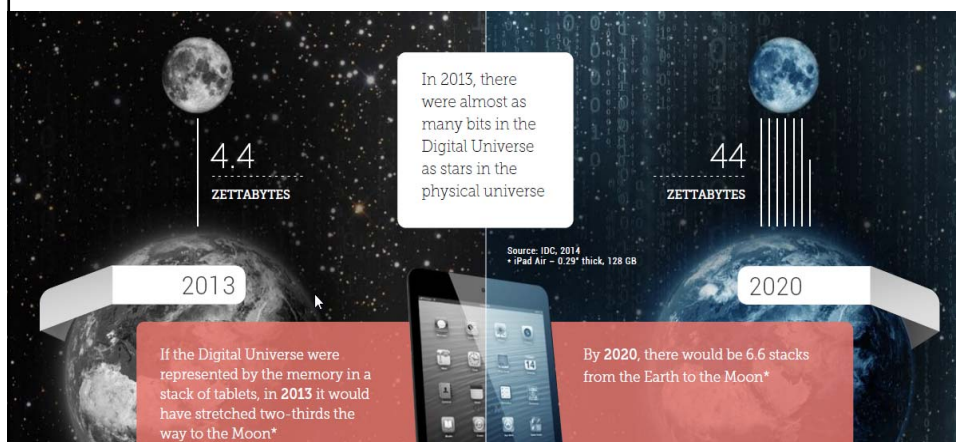


Sistemas de Información

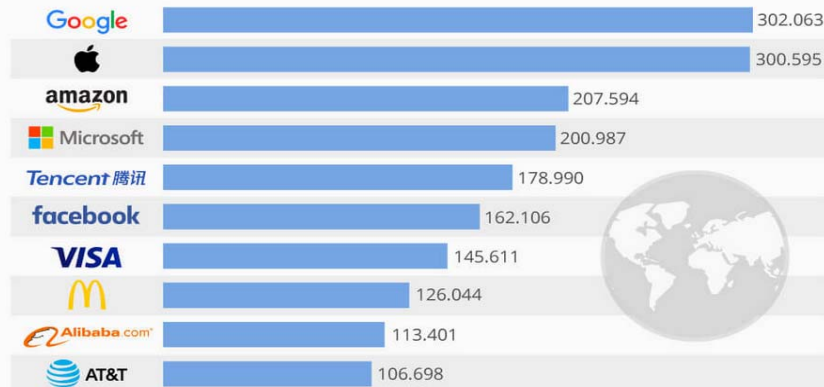
Impacto de la información



Impacto de la información

Las marcas más valiosas de 2018

Valor de las marcas más valiosas en 2018 (en mill. \$)



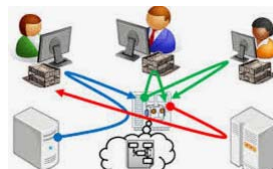
* Dólares estadounidenses

Fuente: WPP/Kantar Millward Brown

statista

¿Qué es un sistema?

- **Sistema:** Conjunto de elementos con un objetivo relacionados entre sí y con su entorno, que responden a entradas para producir salidas. Ente compuesto de estructuras menores interrelacionadas.
- **Análisis de un sistema:**
 1. Descomponer en subsistemas (no analizar su interior: caja negra)
 2. Considerar cada uno un sistema; descomponerlo en sub-subsistemas
 3. Sucesivamente hasta obtener objetos manejables
- **Síntesis o Ingeniería Inversa:** Componer un sistema a partir de sus partes constituyentes



¿Qué es la información?

- **Dato:** Representación de un hecho o fenómeno a través de signos y señales. Los datos describen únicamente una parte de lo que pasa en la realidad y no proporcionan juicios de valor o interpretaciones, y por lo tanto no son orientativos para la acción.
- **Información:** Suma de los datos con relevancia y propósito.
- **Conocimiento:** Capacidad de transformar los datos, la información y la pericia de las personas en acción. Interconecta los datos, la información y las experiencias acumuladas por las personas. El conocimiento puede clasificarse en:
 - **Tácito.** Es el tipo de conocimiento que se encuentra en la mente de las personas (know-how) producto de sus experiencias.
 - **Explícito.** Es el tipo de conocimiento que se puede obtener, codificar y transmitir con facilidad.

Datos, información, conocimiento

- Los datos pueden ser simplemente series de números o de caracteres. Por ejemplo: “260664” ó “Mañana 1952”. Estos caracteres o números, por sí mismos, no constituyen información, ya que para nosotros no significan nada.
- Información: “260664” simboliza la fecha de nacimiento (26 de junio de 1964) de una persona; “Mañana 1952” quiere indicar la cita con el dentista
- La información depende mucho del contexto: “260664” podría significar que una casa cuesta 260.664 euros y “Mañana 1952” podría informar de que mañana una acción de la empresa X se cotizará a 1952 euros.
- Hemos necesitado un procesamiento, completar su significado o situar en un contexto, para que los datos sean significativos.
- Conocimiento para que podamos tomar decisiones: saber si debemos ir a comprar un regalo de cumpleaños o si podemos quedar mañana para ir al cine.

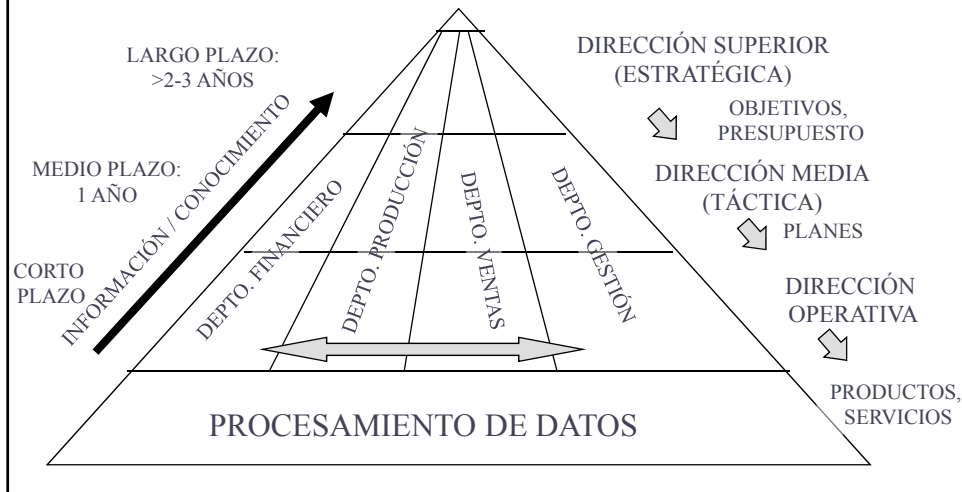
Calidad de la información

- **Valor de la información:**
 - Utilidad para tomar decisiones
- **Calidad de la información:**
 - **Relevante:** útil para decidir y reduce incertidumbre
 - **Precisa:** coherente con la realidad
 - **Completa**
 - **Destinada a la persona o sistema adecuado**
 - **A tiempo:** disponible cuando se necesita
 - **Nivel de detalle adecuado**
 - **Comprensible**

Sistemas de información

- **No automatizados:** Compuestos por personas, información, procedimientos o métodos de trabajo y equipo no informático (papel, lápiz, archivadores,...)
- **Automatizados:** Incluyen un sistema informático. Una red de información formalizada y estructurada según la necesidades y posibilidades de la organización que se apoya en un sistema informático para suministrar información de calidad necesaria para la gestión y dirección de la organización.

Flujo de información en la organización



10

Tipos de SI

- Sistemas basados en Internet (e-commerce, e-learning, etc).
- Sistemas de extracción y de gestión del conocimiento.
 - Por ejemplo sistemas de big data
- Sistemas de información empresarial (ERP, CRM, SCM, etc.)
 - Permiten a las organizaciones integrar y coordinar sus procesos de negocio en distintos ámbitos: sanitario, industrial, comercial,...
- Sistemas de ayuda a la toma de decisiones y de bussines intelligence.
 - Proporcionar conocimiento para la tomad e decisiones empresarial (por ejemplo cuadros de mando): se integran con los dos anteriores

Tarea de un experto en SI

- Actuar de puente entre las necesidades de gestión y las posibilidades que la tecnología ofrece:
 - Analizar los requisitos de la organización
 - Diseñar soluciones eligiendo, adaptando e integrando las herramientas disponibles más adecuadas.
- Identificar las oportunidades de mejora de procesos y de introducir innovaciones, facilitando que la organización utilice sus sistemas de información para competir estratégicamente.

Planificación estratégica de SI

- Análisis para planificar el desarrollo de los sistemas informáticos en consonancia con la estrategia global de la organización. Se hacen a largo plazo (2-5 años) y deben tener una visión global de la organización y coordinar los intereses de los distintos departamentos.
- Resultado: Plan Estratégico de Sistemas de Información (PESI), que contiene:
 - Situación actual de los SI
 - Lista de proyectos de desarrollo y sus prioridades
 - Estudio económico del plan
 - Relación de actividades de la empresa afectadas por el plan
 - Mecanismos de evaluación y actualización del plan
- La metodología MetricaV3 incluye un proceso para elaborar el PESI


¿Cómo afecta un PESI a los informáticos?

- Los directivos de informática intervienen en la elaboración junto con la participación de directivos y representantes de todas las áreas
 - Posible ayuda de consultores
- Informáticos:
 - Los proyectos a desarrollar serán los marcados en el PESI
 - Hay un marco estable de trabajo y objetivos claros

Sistemas estándar

- Los proyectos definidos en el PESI pueden ser proyectos de mantenimiento o desarrollo.
- Los proyectos de desarrollo pueden resolverse desarrollando un sistema nuevo o comprando un sistema estándar.
- Los sistemas estándar cubren necesidades amplias y en muchos campos. Son personalizables y flexibles.
- Existen una gran industria de comercialización, instalación y consultoría de sistemas estándar

Ventajas de los sistemas estándar

- Un típico sistema de funcionalidad compleja suele tener más de 50 KLOC:
 - Coste de desarrollo interno excede al de compra de uno estándar
- Desarrollo interno puede llevar años, instalar un sistema comprado puede hacerse en unos meses o un año, dependiendo del entorno. 
- El coste de desarrollo y mantenimiento está controlado.
- Analistas y programadores de empresas especializadas tienen experiencia probada en implementar.
- Los paquetes a la venta están ya instalados en otras empresas: han sido ya probados y corregidos
 - Hay mucha experiencia para implantar y optimizar

16

Adquisición de un sistema estándar

Listar en detalle requisitos presentes y futuros

Investigar sistemas disponibles

Examinar documentación y analizar calidad con herramientas y pruebas

Comprobar su capacidad de personalización

Verificar si hay herramientas CASE para modificarlo y mantenerlo

Examinar al vendedor, su soporte y experiencia

Comprobar enlace con resto de sistemas y base de datos actuales y futuras

Hablar con usuarios y permitir uso temporal de prueba

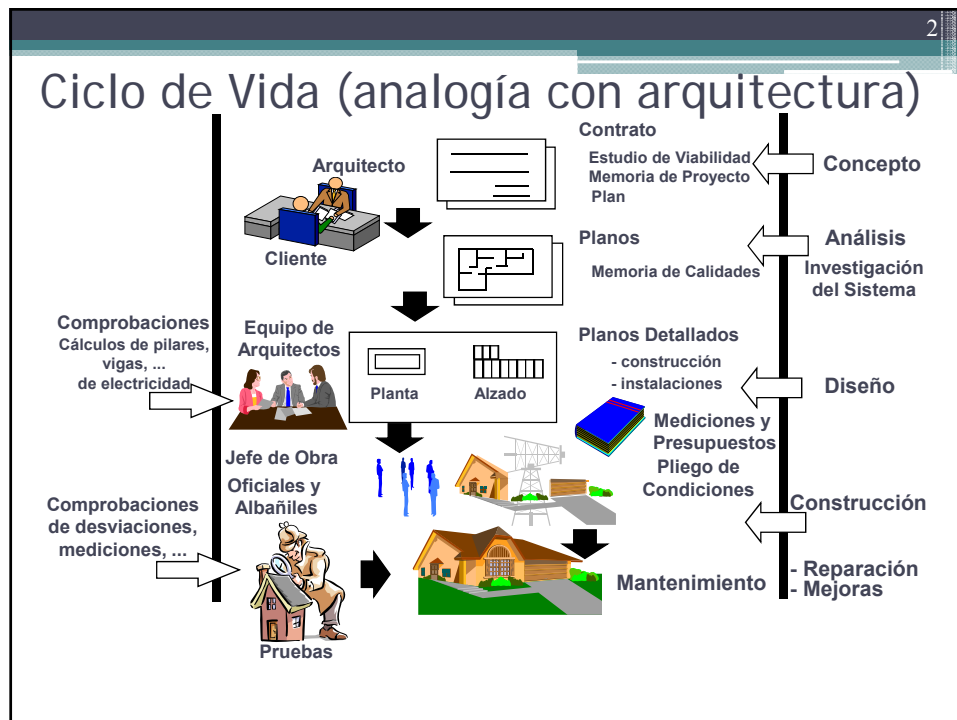
Solución intermedia

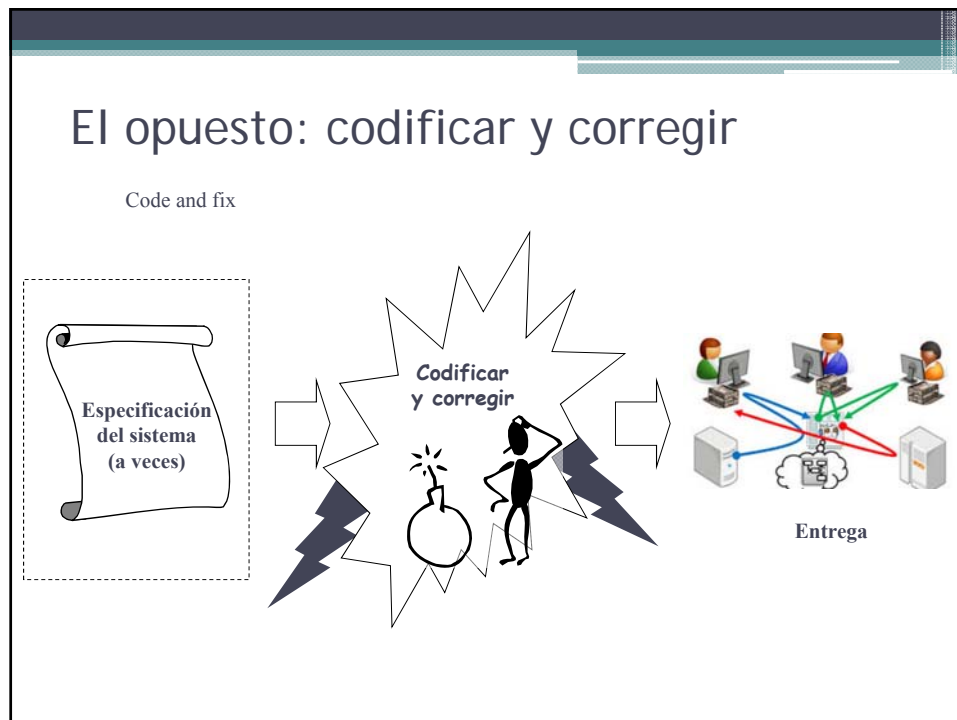
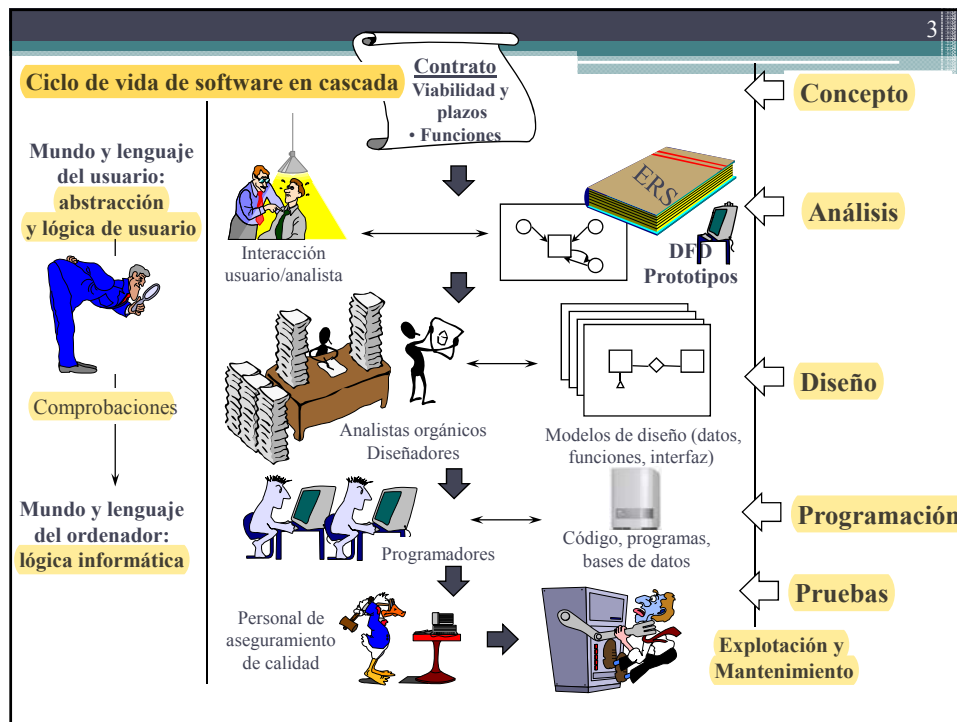
- Posibilidad de comprar un sistema estándar y personalizarlo adaptándolo a necesidades, con cierto desarrollo.
- Hay que elegir cuidadosamente el sistema de base para adoptar esta solución.
- Establecer al inicio las modificaciones y dónde se quiere llegar.
 - Evitar tentación de aumentar sin medida la funciones y acabar desarrollando un software a medida: no es ventajoso.
- Grupo de trabajo con personal interno y de la empresa vendedora o distribuidora para realizar las modificaciones.
 - El personal interno adquiere conocimientos fundamentales para prescindir de los consultores externos en el futuro.

Ejemplos de sistemas estándar

- **ERP (Enterprise Resource Planning).**
 - Gestores de la información de la organización que integran todas las funciones anteriores más importantes.
- **CRM (Customer Relationship Management)**
 - Sistemas para analizar, segmentar y clasificar la relación particular con el cliente.
- **CIM (Computer Integrated Manufacturing)**
 - Sirve ayuda a la fabricación: transporte y almacenamiento de piezas, gestión de stocks y de puestos de trabajo, control de robots industriales.
- **SCM (Supply Chain Management)**
 - Gestionar todas las relaciones existentes de proceso industrial y de integración con clientes y suministradores.

Proceso de desarrollo



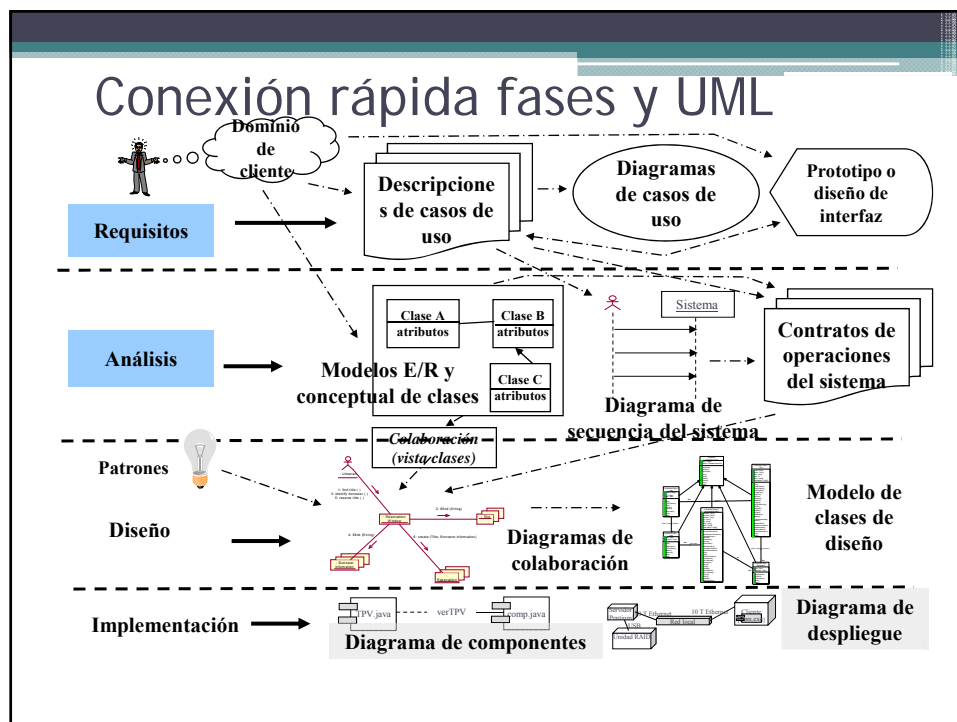


Actividades del ciclo de vida del software

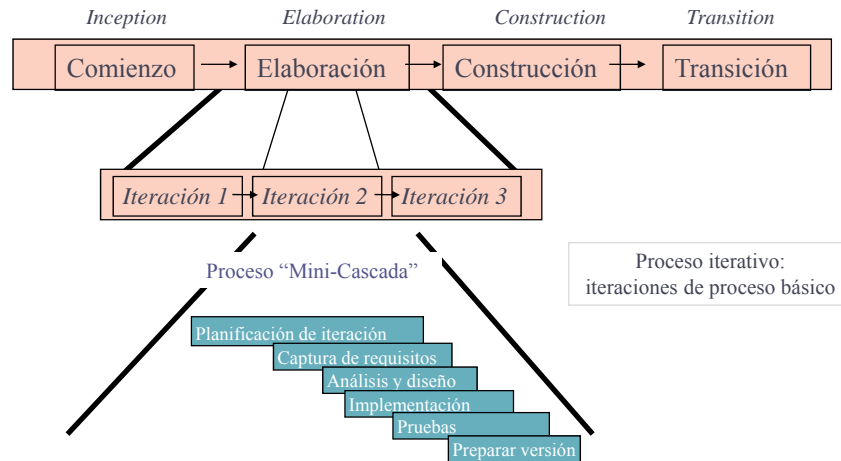
Software development activities



Requirements elicitation	What is the problem?
Analysis	How can we partition the problem?
System design	What is the solution?
Detailed / object design	What are the best mechanisms to implement the solution?
Implementation	How is the solution constructed?
Testing	Is the problem solved?
Delivery	Can the customer use the solution?
Maintenance	Are enhancements needed?

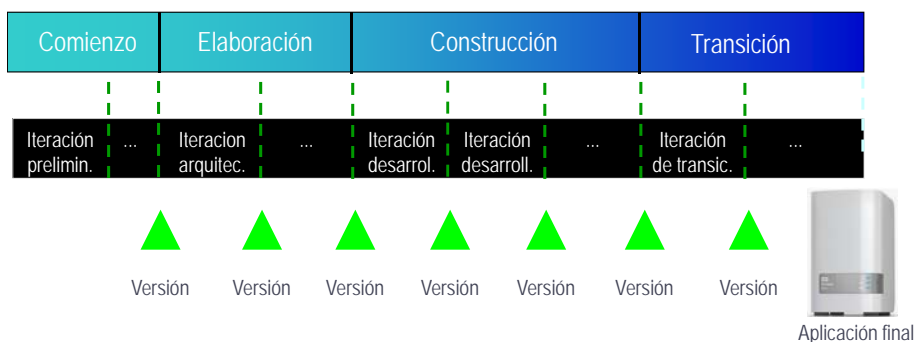


Proceso unificado



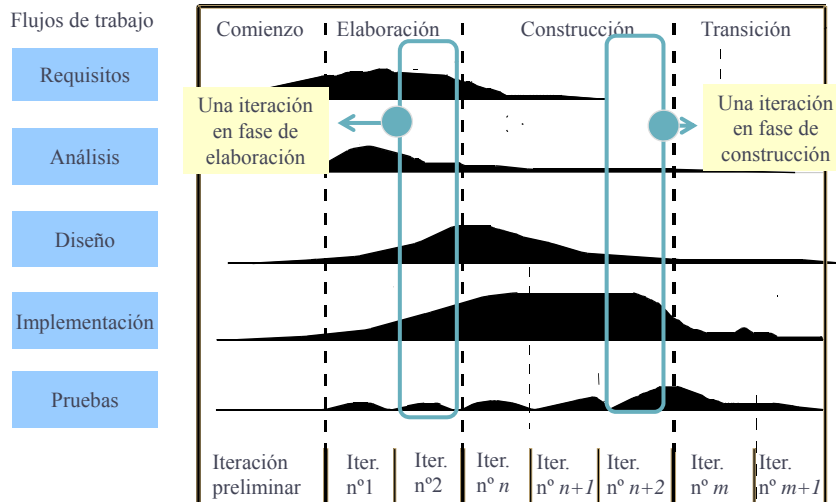
Fases e iteraciones

Proceso iterativo:
entregas que evolucionan hasta la final



Una iteración es una secuencia de actividades con un plan establecido y criterios de evaluación, y que da como resultado una versión ejecutable

Fases (RUP)



10

Gestión del proyecto

- Herramienta **JIRA**. Basado en Scrum
- **3 Iteraciones**
- **Artefactos**. El mismo puede formar parte de varias iteraciones.
- **Varias tareas** en cada iteración.
- Una **tarea** se aplica solo a un artefacto, que tendrá **varias tareas**.
- Las tareas se **pueden dividir** en subtareas.
- Una subtarea es el **trabajo que realiza una persona dentro de una tarea**.

SCRUM. Reuniones

Reunión diaria. Cada miembro responde a tres preguntas:

1. Trabajo realizado ayer → **SEGUIMIENTO DEL SPRINT**
2. Trabajo que se va a realizar hoy
3. Impedimentos para desarrollar el trabajo

Se genera la “sprint backlog” o lista de tareas que se van a realizar, y el “objetivo del sprint”: lema que define la finalidad de negocio que se va a lograr.

PLANIFICACIÓN DEL SPRINT



Ciclo de trabajo

Sprint
(15 – 30 días)

Presentación de lo realizado. Análisis y revisión del incremento generado

REVISIÓN DEL SPRINT



Incremento

Aplicado a nuestro proyecto

El grupo, semanalmente, al inicio de la clase de laboratorio, revisa y redefine tareas y subtareas y carga el trabajo realizado y los comentarios

SEGUIMIENTO DEL SPRINT



Ciclo de trabajo

Sprint
(15 – 30 días)

PEC: Presentación pública de lo realizado

REVISIÓN DEL SPRINT



Incremento

El grupo, al inicio de cada iteración, define tareas y subtareas de la iteración y les asigna un responsable

PLANIFICACIÓN DEL SPRINT



Definición de requisitos

Proceso

- **Extracción o determinación de requisitos.** Proceso mediante el cual los clientes o futuros usuarios del software descubren, revelan, articulan y comprenden los requisitos que desean.
- **Análisis de requisitos.** Proceso de razonamiento sobre los requisitos obtenidos en la etapa anterior, detectando y resolviendo posibles inconsistencias o conflictos, coordinando los requisitos relacionados entre sí, etc.
- **Especificación de requisitos.** Proceso de redacción o registro de los requisitos. Suele recurrirse a un lenguaje natural, lenguajes formales, modelos, gráficos, etc.
- **Validación de los requisitos.** Confirmación, por parte del usuario o el cliente de que los requisitos especificados son válidos, consistentes, completos, etc.

Aunque estas actividades no tienen por qué realizarse en secuencia, ya que hay muchas iteraciones y solapamientos entre ellas, sí marcan un proceso general para la fase de definición de requisitos.

¿Una mala captura de requisitos?



Técnicas de recogida de información

- **Entrevistas**
- **Reuniones**
- **Prototipado**
- **Observación**
- **Cuestionarios**
- **Estudio de documentación**
- **Tormenta de ideas (brainstorming)**

METRICA V3 cuenta con guías para algunas de estas técnicas

Cuestionarios

- Para usuarios dispersos, numerosos, poco disponibles, etc.
 - Distinto de encuesta: no hay entrevistador presente
- Algunas indicaciones de diseño:
 - Tener claros objetivos de información que deseamos obtener
 - Preguntas muy claras y precisas (no presentes para aclarar) y una sola cuestión en cada pregunta
 - Cortos y sencillos (evitar aburrimiento)
 - Preguntas cerradas con unas pocas abiertas
 - Introducción con propósito y que agradezca colaboración
 - Alguna pregunta final de sugerencias, olvidos, etc.

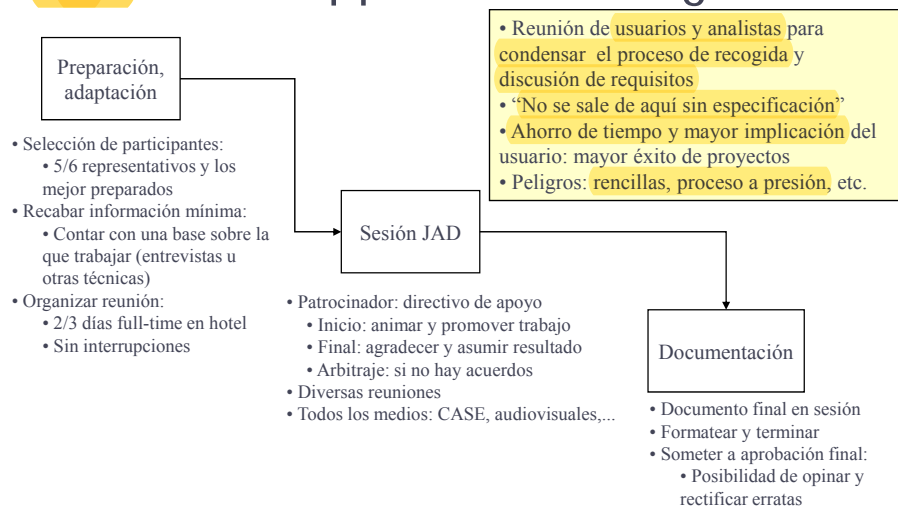
Entrevistas - Resumen

- Entrevistas
 - Obtener información de forma individual
 - Preparar un guion previo que se remite al entrevistado
 - Obtener la máxima información sin provocar rechazo
 - Al final resumir las conclusiones para aclarar malentendidos
 - Enviar el acta al entrevistado para fijar ideas por escrito

Reuniones

- Procedimiento
 - Obtener información dispersa entre varios usuarios, comunicar información, tomar decisiones
 - Preparar el orden del día y convocar la reunión
 - Al inicio resumir objetivos y método de trabajo
 - Importancia de la persona que dirige la reunión
 - Al final resumir las conclusiones para aclarar malentendidos, destacar puntos pendientes y fijar siguiente reunión o actividades/responsables
 - Enviar el acta a los participantes para fijar ideas por escrito
- Diversos tipos

JAD: Joint Application Design



JRP (Joint Requirements Planning)

- Potenciar la participación activa de la alta dirección. Muy útil en el desarrollo del Plan Estratégico de SI
- Las características de las sesiones JRP y JAD son comunes en cuanto a la dinámica del desarrollo de las sesiones y la obtención de los modelos con el soporte de las herramientas adecuadas.
- Diferencias: nivel más alto en la organización en cuanto a visión global del negocio y capacidad de decisión. Tipo de información de salida :
 - Modelos de procesos de la organización.
 - Modelo de información.
 - Modelo de sistemas de información, etc.

Prototipado

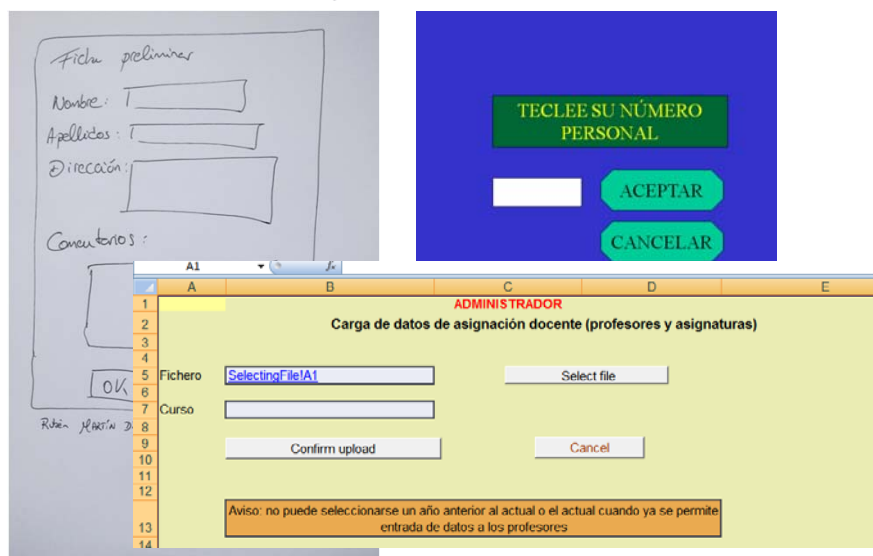
- Creación de “maquetas” de sistemas
 - para evaluación de usuario y/o desarrolladores
 - según objetivo, requiere herramientas distintas (Office, entornos visuales, programación, etc.)
 - coste/esfuerzo alto (¿10% de proyecto?)
- Aplicación:
 - área poco definida: dificultad o sin tradición
 - coste alto de rechazo
 - evaluar impacto previamente
- Tipos:
 - interfaz de usuario: más habitual
 - prototipado funcional evolutivo
 - reaprovecha código (ciclo de vida evolutivo)

Prototipado y métodos afines

- **Maquetas (Mock Ups)**
 - Visión simplificada de la aplicación a desarrollar
 - Muestra para el usuario de como quedará y a que se parecerá la interfaz del software sin haber programado aplicación ni funcionalidad
 - Representa la interfaz gráfica del sistema con trazos sobre papel o con herramienta visual.
- **Storyboards***
 - Se encargan de presentar secuencia de interfaz gráfica con las actividades del sistema
 - Papel o herramientas como Visio, Excel, PowerPoint,...

* Guía para la creación de storyboards en el diseño de aplicaciones
https://www.researchgate.net/profile/Monica_Forero3/publication/316845657_Guia_para_crear_storyboard_en_el_proceso_del_diseno_de_interfaz/links/591358e9a6fdcc963e7ee05e/Guia-para-crear-storyboard-en-el-proceso-del-diseno-de-interfaz

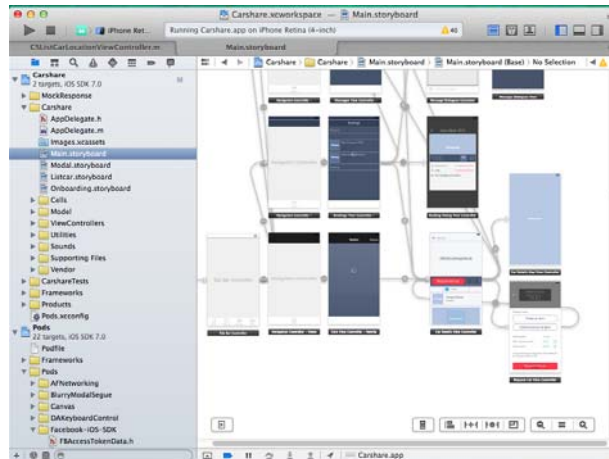
Ejemplos (I)



Ejemplos (II)

Tutorial para crear interfaces usando storyboard para desarrolladores IOS (Apple) :

<https://www.efectoapple.com/introduccion-los-storyboards-parte-1/>



Otras técnicas

- **Observación:**
 - Captar mejor la realidad del trabajo diario
 - “Pasar una mañana en el puesto”, “pasar por cliente”, *mystery shopping*, etc.
- **Estudio de documentación:**
 - Siempre necesario: no se necesita interlocutor
 - Imprescindible como fase previa a otras técnicas
 - Análisis: procesos formales/documentados, normas, impresos, formularios, legislación, etc.
 - Mejor tener siempre muestras de documentos rellenos: apreciar uso real
- **Brainstorming:**
 - Crear nuevas ideas cuando no hay tradición del problema o de software existente
 - Técnica creativa muy usada en publicidad, etc.
 - Ronda de ideas en grupo sin evaluar su bondad
 - Al parar, leer lo dicho previamente y empezar otra ronda

Dificultad de comunicación



Resistencia al cambio

- Ser humano: animal de costumbres
 - Se resiste incluso a cambios buenos
- Causas / Soluciones:
 - Resistencia al personal informático
 - Equipos conjuntos, coordinación de directivos, comprensión del trabajo informático, formación de usuarios
 - Percepción de que “el proyecto no es bueno”
 - Involucrar en el proyecto, análisis de coste/beneficio
 - No percibir necesidad de cambio (de forma de trabajo, de sistema, ...)
 - Lista de ventajas del nuevo sistema frente al antiguo
 - Miedo a perder poder o puesto. Computerfobia: miedo tecnológico
 - Formación, integración en el proyecto
 - Sistemas mal diseñados para usuarios
 - Tener en cuenta la usabilidad y accesibilidad en el desarrollo

Análisis de requisitos

- **Compleitud.**
 - **No hay omisiones:** no faltan requisitos (propiedad global) y no faltan detalles en la especificación de cada requisito (propiedad individual).
 - **Difícil de determinar:** contrastar con el cliente, comparar con proyectos semejantes, buscar la visión de conjunto, detectar huecos o partes infra-especificadas,...
- **Detección de Conflictos e Inconsistencias**
 - Los requisitos se agrupan por categorías y se organizan en sub-conjuntos, se estudia cada requisito en relación con el resto y se clasifican en base a las necesidades de los clientes/usuarios.
 - Es corriente en clientes y usuarios solicitar más de lo que puede realizarse o proponer requisitos contradictorios.

Preguntas planteadas en una Especificación de requisitos del software (ERS)

- La **funcionalidad.**
 - ¿Qué tiene que hacer el software ?
- Las **interfaces externas.**
 - ¿Cómo el software actúa recíprocamente con las personas, el hardware y otros sistemas hardware o software?
- La **actuación.**
 - ¿Cuál es la velocidad, la disponibilidad, tiempo de respuesta, tiempo de la recuperación, etc.?
- Los **atributos de calidad.**
 - ¿Que fiabilidad, mantenibilidad, portabilidad, seguridad, etc. necesita el sistema?
- Las **restricciones del diseño.**
 - ¿Hay alguna restricción de idioma, hardware, recursos, etc.?

Características de una buena ERS

- IEEE Std. 830:
 - No ambigua
 - Completa
 - Fácil de verificar
 - Consistente
 - Fácil de modificar
 - Ordenación por prioridades
 - Facilidad para identificar fuente y efectos de cada requisito (trazabilidad)
 - Facilidad de uso en explotación y mantenimiento



Requisitos funcionales y no funcionales (IEEE std. 610)

- Requisito funcional:
 - Requisito que especifica una función que un sistema o componente debe ser capaz de realizar
 - Ejemplos de un sistema de gestión de notas:
 - Obtener estadísticas de notas
 - Introducir nota de prácticas
 - Gestionar datos anagráficos del alumno
- Requisito no funcional:
 - Requisito que especifica una característica del sistema
 - Seguridad, rendimiento, facilidad de uso, capacidad, etc.
 - Ejemplos de un sistema de gestión de notas:
 - Capaz de gestionar 100 alumnos y hasta 10 exámenes y 5 prácticas por alumno
 - Seguridad de acceso a las notas basada en clave de 128 bits

Categorías requisitos no funcionales



Estándar ERS: IEEE std. 830

- Introducción
 - Objetivo, ámbito, definiciones y siglas, referencias
- Descripción general
 - Visión general de producto, funciones, usuarios, limitaciones generales, supuestos y dependencias
- Requisitos específicos numerados
 - Funcionales
 - Interfaz
 - Rendimiento, restricciones de diseño, calidad, otros
- Apéndice
 - No obligatorio, por ejemplo: ejemplos de formato de entrada/salida

Técnicas de especificación

- **Formales:**

- **Basadas en lógica formal:** ej., notación Z
 - $usados \cap libres = \emptyset$

- **Semiformales:**

- **Lenguaje natural claro y preciso**
- **Puede acompañarse de:**
 - **Gráficos** (diagrama de flujo de datos, diagrama de estados,...)
 - **Texto** basado en gramáticas (pseudocódigo de proceso, BNF,...)
 - **Plantillas** (descripción de datos,...)
 - **Matrices:** para comprobar más que definir, por ejemplo Matriz de trazabilidad (requisito/quien lo genera, quien lo aprueba)

Validación de requisitos

- **Objetivo:**

- **Comprobar que los requisitos definidos en la ERS son correctos.**

- **Los parámetros a validar en los requisitos son :**

- **Validez:** Todos los usuarios involucrados conocen y están de acuerdo con los requisitos definidos.
- **Consistencia:** No debe haber contradicciones entre unos requisitos y otros.
- **Compleitud:** Deben estar todos los requisitos. Esto es imposible en un desarrollo iterativo, pero, al menos, deben estar disponibles todos los requisitos de la iteración en curso.
- **Realismo:** Se pueden implementar con la tecnología actual.
- **Verificabilidad:** Tiene que existir alguna forma de comprobar que cada requisito se cumple.

Captura de requisitos con casos de uso

Ingeniería de software

Casos-1

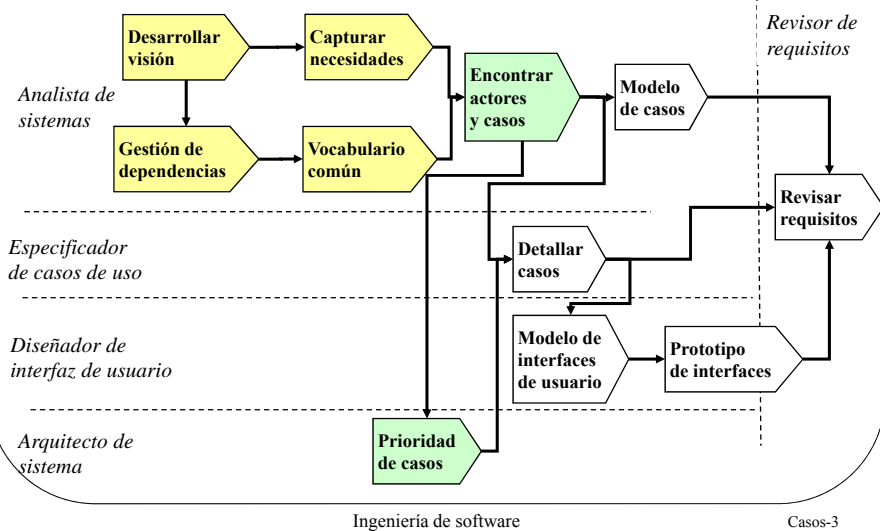
Determinación de requisitos

- Técnicas de recogida de requisitos
Entrevistas, cuestionarios, observación, etc.
- Apoyo en casos de uso para representar requisitos
- Requisitos:
 - Funcionales (qué debe hacer el sistema)
 - No funcionales (atributos como fiabilidad, seguridad, etc.)
- Tipos de funciones:
 - Evidente: debe realizarse y es visible para el usuario
Calcula el total de la venta
 - Oculta: no visible para usuario (cuidado para detectar)
Reduce la cantidad de inventario cuando se realiza una venta
 - Prioridades: posibles opcionales y superfluas
Enviar SMS al comprador

Ingeniería de software

Casos-2

Captura de requisitos apoyada en CU



Visión y dependencias

- **Visión:**
 - **Enunciado abreviado:**
 - Habitual: recogida de información de directivos/responsables
 - Principales características:
 - Requiere **labor posterior de detalle y análisis** con usuarios finales
- **Dependencias:**
 - Con otros proyectos, sistemas, etc.
 - Ejemplo:
 - Se consultan **datos de usuarios existentes en otro sistema**

Necesidades y vocabulario

- **Técnicas de captura de requisitos:**
 - Entrevistas, JAD, observación, cuestionarios, estudio, etc.
 - Recoger información sobre detalles de funciones, datos, requisitos no funcionales, etc.
- **Vocabulario común:**
 - Términos del dominio del usuario
 - Ejemplos:
 - Ejemplar: una copia de una obra registrada en la biblioteca
 - Préstamo: registro de un préstamo de un ejemplar a un usuario

Ingeniería de software

Casos-5

Identificar funciones

- Lista de funciones solicitadas
 - Actor 1:
 - Función 1.1: Prioridad, E, S, Función
 - Función 1.2: Prioridad, E, S, Función
 - Actor 2:
 - Función 2.1: Prioridad, E, S, Función
- Lista de requisitos no funcionales (características de las anteriores)
 - Seguridad, tiempo de respuesta, volumen de datos, facilidad de uso, etc.
 - Medibles: no “respuesta rápida” sino “menor de 2 sg.”

Ingeniería de software

Casos-6

Identificar actores y casos

- **Identificar actores**
 - ¿Quién usa el sistema? ¿quién inicia cada función? ¿quién aporta o recibe datos del sistema?
 - ¿Qué otros sistemas interactúan obteniendo o proporcionando información? ¿ocurre algo en un momento preestablecido?
- **Identificar casos de uso: analizar cada actor**
 - Los actores inician los casos (sólo en ciertas ocasiones comienzan desde el sistema)
 - ¿Qué funciones querrá el actor realizar en el sistema? ¿qué información quiere obtener?
 - Si el sistema almacena información, ¿qué actores la crean, consultan, actualizan o borran? ¿El sistema notifica cambios en su estado?
 - ¿Hay acontecimientos externos que el sistema debe conocer? ¿Qué actor informa al sistema de dichos acontecimientos?

Ingeniería de software

Casos-7

Estructurar y modelar casos

- **Estructurar casos (diagrama)**
 - Factorizar comportamiento común (include)
 - Factorizar variantes (extends)
- **Descripción completa de cada caso (modelado)**
 - Descripción detallada de las características y el comportamiento del caso

Ingeniería de software

Casos-8

Casos de uso (I)

- Ayuda para determinar y especificar requisitos
- Enfoque:
 - Sistema como caja negra (funciones)
 - Comportamiento, no implementación
 - Terminología de usuarios
- Concepto de caso de uso:
 - Manera específica de usar el sistema
 - Interacción típica entre un usuario y el sistema
 - Representa requisitos funcionales, pero no exactamente

Ingeniería de software

Casos-9

Casos de uso (II)

- Definición oficial:
 - Conjunto de secuencias de acciones, con variantes, que ejecuta un sistema para producir resultados valiosos para un actor
- Aplicación:
 - Determinación de requisitos funcionales
 - Refinamiento a lo largo del desarrollo
 - Sistema completo o subsistemas
- Representación gráfica: diagramas de casos de uso
- Representación textual: modelado de casos de uso

Ingeniería de software

Casos-10

Casos de uso (III)

- Lista de casos:
 - Para rápida comprensión de principales procesos globales
 - Descripción breve de un proceso: 1/2 frases
- Tipos de casos (por prioridad)
 - Primario: proceso común importante (*comprar producto*)
 - Secundario: procesos menores o raros (*surtir nuevo producto*)
 - Opcional: proceso que puede no abordarse
- Cada iteración debe planear qué casos va a incluir

Ingeniería de software

Casos-11

Casos de uso (IV)

- Comportamiento:
 - Descripción completa:
 - Objetivo, cómo se inicia, flujo de mensajes entre actor y caso, flujos alternativos, cómo termina
 - Flujo de acontecimientos textual:
 - Texto estructurado, formal y pseudocódigo (plantilla)
 - Descripción clara y coherente, para validación de cliente

Retirada de efectivo (cajero)

Actor: 1. El cliente introduce tarjeta
3. Introduce número en teclado
5. Elige opción retirada

Sistema: 2. Pide número personal
4. Muestra menú de opciones
6. Presenta opciones de retirada...

Ingeniería de software

Casos-12

Elementos del diagrama: actores

- **Actor:**
 - Conjunto coherente de roles que juegan personas/sistemas relacionados con los casos, al interactuar
 - Categorías orientativas:
 - principal: usuario
 - secundario: administrador
 - hardware externo: periféricos, etc.
 - otros sistemas y aplicaciones
 - Distinguir iniciador y participantes

Elementos del diagrama: casos de uso

- **Nombre: verbo + nombre:** “introducir pedido”
- **Qué hace** el sistema (visión desde el actor)

Elementos del diagrama: relaciones (I)

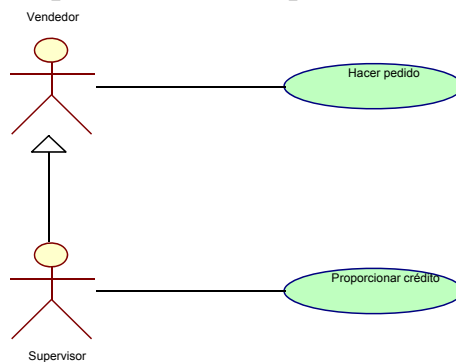
- **Inclusión** (*include* o *uses*): incorporar comportamiento de otro caso en el lugar especificado
 - ModificarPedido: *Include* <Validar usuario>.
- **Extensión** (*extends*): comportamiento opcional de caso o subflujo sólo ejecutado en ciertas condiciones
 - Hacer pedido: Introducir nº de pedido (*Punto de Extensión*: si no recuerda nº, <buscar por empresa>). *Include* Validar usuario...

Ingeniería de software

Casos-15

Elementos del diagrama: relaciones (II)

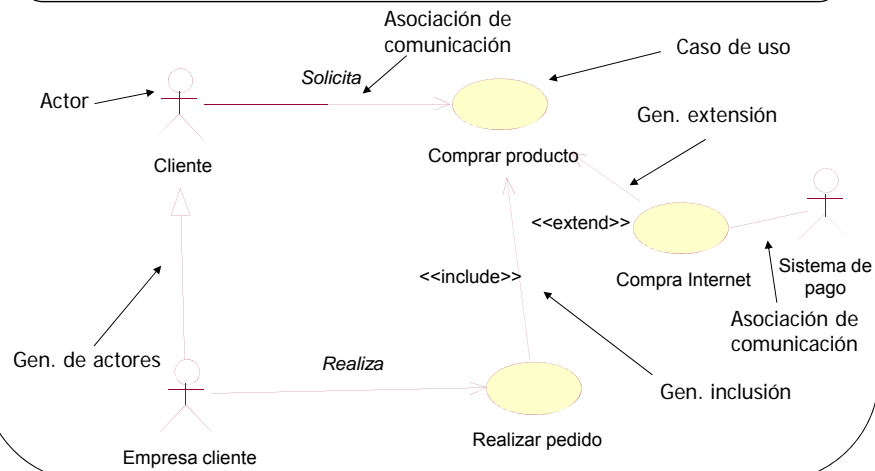
- **Relaciones entre actores**: **Generalización** (Supervisor hereda de Vendedor, puede hacer lo que hace el Vendedor)



Ingeniería de software

Casos-16

Notación gráfica

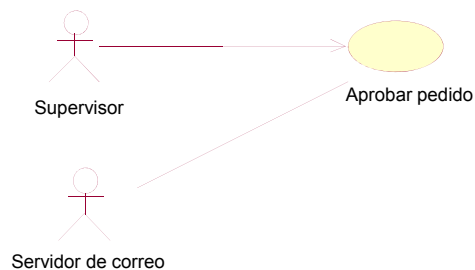


Ingeniería de software

Casos-17

Aclaraciones de notación (I)

- Comunicación actor-caso:
 - **Flecha:** supone que el actor activa el caso
 - **Sin flecha:** el actor interviene, no lo activa



Ingeniería de software

Casos-18

Aclaraciones de notación (II)

- Vinculación de un CU a otros diagramas:
 - **Vínculo invisible** (normal en **CASE**) a:
 - Diagrama de actividad
 - Diagrama de colaboración de diseño
 - Vista del modelo conceptual de clases
 - Diagrama de secuencia de interacción con el sistema

Ingeniería de software

Casos-19

Modelado de casos

- Flujo de acontecimientos:
 - **Objetivo del caso:** ¿qué se quiere conseguir?
 - ¿Cómo se inicia? ¿Qué actor lanza la ejecución?
 - El **flujo entre actores y caso de uso:**
 - ¿qué mensajes o acontecimientos se intercambian?
 - ¿qué debería describir el flujo principal?
 - Flujos alternativos: alternativas de ejecución
 - Según condiciones o excepciones
 - ¿Cómo acaba el caso?
 - Aportando algún tipo de valor al actor
- Se utiliza una plantilla

Ingeniería de software

Casos-20

Ejemplo flujo de acontecimientos

Validar usuario

• Flujo principal

- *Cajero: Comienza cuando el sistema pide el PIN al cliente. El cliente puede introducir el PIN por teclado. El cliente acepta con ENTER. El sistema comprueba entonces que el PIN es válido, el sistema acepta la entrada y acaba el caso.*

• Flujo excepcional/alternativo

- *El cliente puede cancelar una transacción en cualquier momento con CANCELAR, reiniciando así el caso. No se hace ningún cambio a la cuenta de cliente.*

• Flujo excepcional/alternativo

- *El cliente puede borrar el PIN en cualquier momento antes de introducirlo y volver a teclear uno nuevo.*

Ingeniería de software

Casos-21

Plantilla de casos (I)

- **Objetivo:** Describe cómo el usuario puede consultar el saldo de su cuenta

- **Flujo principal de acontecimientos:**

1. El usuario introduce tarjeta	2. El sistema solicita el PIN
3. El usuario teclea el PIN y pulsa <i>ACEPTAR</i>	4. El sistema valida el usuario y muestra el menú principal
5. El usuario pulsa la opción CONSULTAR SALDO	6. El sistema muestra en pantalla el saldo de cuenta y solicita que pulse SEGUIR
7. El usuario pulsa SEGUIR	8. El sistema le pregunta al usuario si desea otra operación
Etc.	

Ingeniería de software

Casos-22

Plantilla de casos (II)

- **Flujos excepcionales/alternativos:**

- En 3, si el usuario pulsa CANCELAR, se interrumpe el proceso sin efectuar ninguna acción y se cierra la sesión.
- En 4, si el PIN no es correcto, se pide al usuario que vuelva a introducirlo.

- **Precondiciones y poscondiciones**

Caso de uso	Precondiciones	Postcondiciones
Registrar usuario		Usuario registrado
Registrar artículo	Usuario registrado como Vendedor	Artículo registrado
Pujar	Usuario registrado como Comprador Artículo registrado y no adjudicado	Artículo asociado a la puja

- Precondiciones. Condiciones no sometidas a prueba durante el CU, pero básicas para su éxito, normalmente probadas en un CU ejecutado anteriormente.
- Poscondiciones. Hechos relevantes (envío de mensajes, impresión de un documento, etc.) y cambios en el modelo conceptual:
 - Creación y eliminación de ejemplares
 - Modificación de atributos
 - Asociaciones formadas o canceladas

Ingeniería de software

Casos-23

Plantilla de casos (III)

- **Diagramas complementarios** (diagrama de actividad o de estados): normalmente diagrama de actividad
- **Referencia a requisitos funcionales y no funcionales** propios del caso
- **Descripción de interfaz:** pantallas, navegación, etc.

Ingeniería de software

Casos-24

Interfaz

- **Diseño basado en el usuario:**
 - Parte del proceso de extracción de requisitos. Mismas técnicas: prototipos, storyboard, entrevistas, etc.
- **Estándares:**
 - ISO/IEC 11581: Usage and appropriateness of icons in the user interface.
 - ISO 13407: Designing user interfaces with humans in mind.
 - ISO/IEC 14754: Defines the basic gesture commands.
 - ISO 14915: Recommendations for multimedia controls and navigation.
- **Usabilidad y Accesibilidad:**
 - Uso equiparable: Útil a personas con diversas capacidades.
 - Uso flexible: Que ofrezca posibilidades de elección en los métodos de uso.
 - Simple e intuitivo: Fácil de entender y que proporcione avisos eficaces.
 - Información perceptible: Que destaque lo importante y ofrezca distintos formatos.
 - Tolerancia a error: Que minimice los riesgos de acciones involuntarias.
 - Mínimo esfuerzo: Que evite acciones repetitivas y uso de la memoria.

Ingeniería de software

Casos-25


Diagrama de actividades

- **Muestra flujo de actividades**
 - **Actividad:** ejecución no elemental en curso
 - **Actividades producen acciones**
 - **Acción:** elemento atómico ejecutable que produce cambios de estado en sistema o devuelve un valor
 - Llamadas a otras operaciones
 - Envío de señales
 - Creación o destrucción de objetos
 - Cálculos simples

Ingeniería de software

Casos-26

Elementos (I)

- Estado de acción:
 - No se pueden descomponer, son atómicos
- Estado de actividad:
 - Pueden descomponerse en otros diagramas de actividad
- Estado inicial:
 - Marca el punto de inicio del flujo de ejecución
- Estado final: 
 - Marca el punto final del flujo de ejecución

Ingeniería de software

Casos-27

Elementos (II)

- Transiciones:
 - Paso inmediato al siguiente estado
 - La transición guía el flujo entre estados
- Bifurcaciones:
 - Posibilidad de caminos alternativos
 - Situar expresiones booleanas en las transiciones de salida
 - Por ejemplo: [Dato no disponible], [valor = 0], etc.
 - Permite lograr un efecto de iteración
 - Simular estructura de bucle

Ingeniería de software

Casos-28

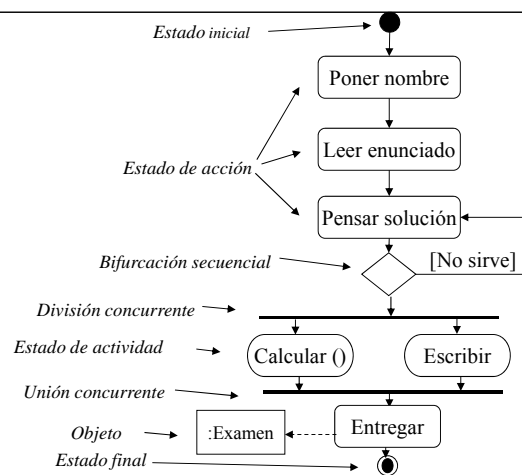
Elementos(III)

- **Unión y división**
 - Posibilitan la concurrencia de acciones y flujos
 - Representados por **barras de sincronización**
- **Objeto (no para casos de uso):**
 - Conectados a actividad o acción que lo crea, destruye o modifica

Ingeniería de software

Casos-29

Ejemplo Diagrama de actividades



Ingeniería de software

Casos-30

Restricciones

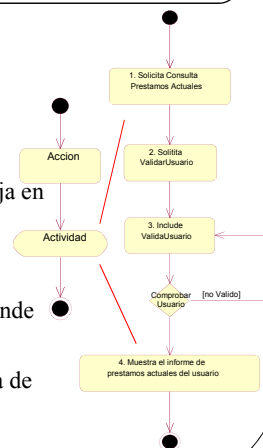
- Puede haber **cero o más estados finales** (por ejemplo, un proceso continuo no tendrá estado final)
- Un **estado inicial no puede ser destino de una transición**
- Toda **actividad tiene al menos un flujo de entrada y otro de salida**
- Una **decisión** tiene un **flujo de entrada y dos o más de salida**
- Las **condiciones** de todos los flujos de salida de una decisión deben ser **disjuntas y completas**
- Todo **flujo de salida de una decisión debe estar etiquetado con una condición**
- Una **fusión** tiene dos o más flujos de entrada y un flujo de salida

Ingeniería de software

Casos-31

Utilidad: ayuda al caso de uso

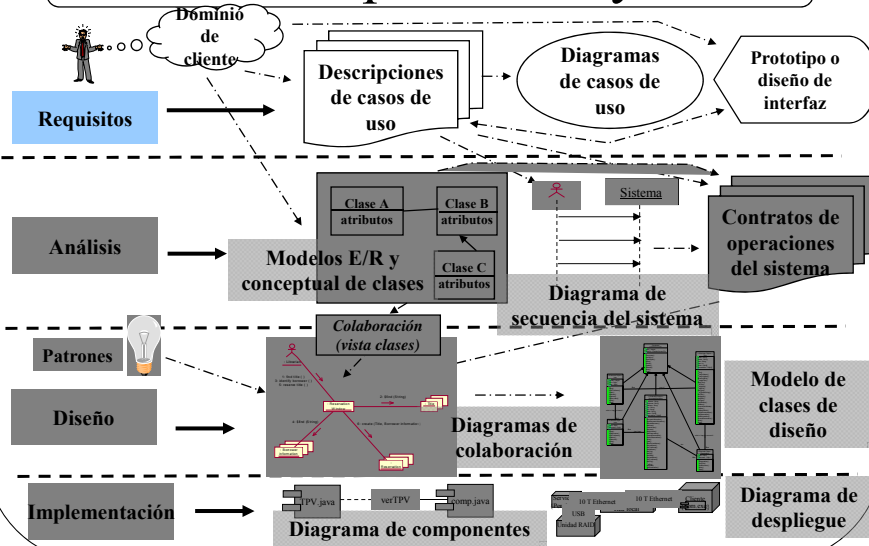
- Ayuda gráfica para entender interacción
- Reflejar en el diagrama de actividad:
 - Los pasos del caso de uso
 - Basado en flujo principal y reflejando los flujos alternativos
 - Cada paso (celda de tabla del flujo del CU) se refleja en una acción
 - Si el paso supone un *include* o *extends*:
 - Dibujamos una actividad cuyo subdiagrama es el diagrama de actividad del caso incluido o que extiende
 - Unión y división:
 - Acciones paralelas representadas en la misma celda de la tabla del flujo del CU
 - Estado inicial y final:
 - Coincide con inicio y final en descripción de caso



Ingeniería de software

Casos-32

Conexión rápida fases y UML



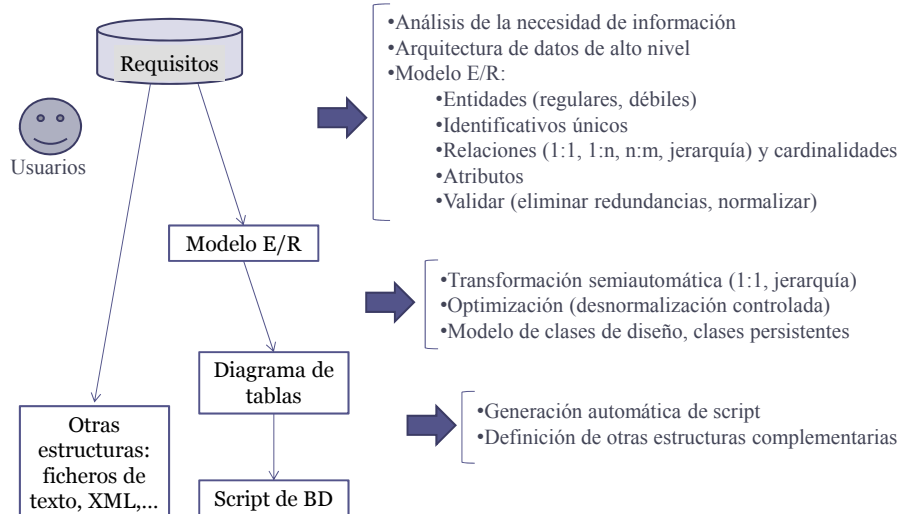
Ingeniería de software

Casos-33

Modelo de datos

Ingeniería del Software Avanzada
Técnicas de análisis y diseño

Proceso



Modelo E/R

- **Entidades**
 - Objeto sobre el que se desea almacenar información
 - Todas las ocurrencias de una entidad tienen los mismos atributos (información que las caracteriza) y se distinguen entre ellas:
 - Identificador único (conjunto mínimo de atributos que las identifica)
 - Débiles, su existencia depende de otra (posible dependencia en identificación)
- **Relaciones**
 - Correspondencia entre entidades
 - Cardinalidad:
 - Diferentes tipos de relación indican cardinalidad máxima
 - Importancia como reglas de negocio la cardinalidad mínima
 - n:m con posibilidad de atributos
 - Jerarquía (total, disjunta, no disjunta)

Modelo E/R – Explicación adicional

Débiles: Ciudad/Código Postal, si se elimina una ciudad, desaparecen sus códigos postales.

Débiles con dependencia en identificación: Líneas de pedido. Si se elimina un pedido desaparecen sus líneas de pedido. Las líneas de pedido se llaman con el código del pedido más un secuencial.

Jerarquía total: todos los empleados son administrativos o profesores. **Disjunta:** no hay empleados que sean administrativos y profesores a la vez. **No disjunta:** un empleado puede ser comercial y también directivo.

Redundancia de relaciones: mirar triángulos. Un coche está matriculado en una ciudad y solo en una y una persona está empadronada en una ciudad y solo en una. Si un coche tiene un solo propietario y debe estar matriculado en la ciudad de su propietario, entonces la relación coche/ciudad sobra. Si un coche puede tener más de un propietario o matricularse en una ciudad distinta a la de su propietario, entonces no sobra.

Diagrama de tablas

- **Transformación**
 - Entidad, tabla; Identificador único, clave; Atributo, campo; Relaciones, claves externas o tablas
 - Decisiones de diseño:
 - Relaciones de jerarquía: unir subtipos a supertipo, sin uniones, unir supertipo a subtipos
 - 1:1, extender claves externas en una u otra dirección (o en las dos)
- **Optimización**
 - Objetivo: reestructurar el modelo físico de datos para mejorar la eficiencia del sistema
 - Desnormalización controlada del modelo físico para reducir o simplificar el número de accesos a la base de datos:
 - Elementos redundantes (atributos, relaciones y tablas)
 - Elementos calculados (atributos)
- **Consistencia**
 - Entre el modelo de clases (clases persistentes) y las tablas, sus campos y sus relaciones

Diagrama de tablas - Explicación adicional

Paso E/R a Tablas - Jerarquía: tabla empleados (una tabla con un campo tipo que indique el tipo de empleado), tablas profesores y administrativos (dos tablas), tablas profesores, administrativos y empleados (esta última con los campos comunes a los dos). Depende de la cantidad de información común y propia, de las relaciones con otras tablas (si son desde el supertipo o desde algún subtipo), de la forma normal de acceso, de la privacidad de la información,...

Paso E/R a Tablas - 1:1: empleado/despacho. Lo habitual es usar la cardinalidad mínima (todos los despachos son de alguien pero hay muchos empleados sin despacho, luego la clave externa se pone en despacho, se evita que sea null). Se puede poner clave externa en las dos, mejor accesos pero cuidado con información redundante.

Desnormalización: incluir la edad, además de la fecha de nacimiento. Incluir una tabla de empleados/cliente, con las horas trabajadas por cada empleado en cada cliente, además de las tablas empleado/proyecto (con las horas trabajadas de cada empleado en cada proyecto) y proyecto/cliente.

Estructura física

- La arquitectura del sistema de alto nivel determina la arquitectura de datos (bases de datos relacionales, ficheros, etc.)
- En fase de **diseño** se **determina la tecnología y herramientas** (ficheros xml, postgresql, etc.)
- Las **herramientas CASE** permiten **obtener automáticamente el script de definición de base de datos** (según la tecnología) a **partir del diagrama de tablas**
- Otras opciones de estructuras de datos (fichero de texto, xml, etc.) deben definirse y generarse

Modelo conceptual de clases

Modelo de clases -1

Modelo conceptual de clases

- Tres perspectivas o niveles para el modelado de clases:
 - **Conceptual**: conceptos del dominio del problema
 - **Especificación**: estructura de software
 - Coincide con el diagrama de clases de diseño en la asignatura
 - **Implementación**: implementación real
- Modelo conceptual
 - Diag. de estructura estática de la información (E/R)
 - Punto de partida para el diagrama de clases de diseño
 - **No es descripción de diseño** (p.ej., clases Java y C++)

Modelo de clases -2

Clases (I)

- Clases:
 - Bloque básico de sistemas OO
- Clase:
 - Descripción de conjunto de objetos que comparten atributos, operaciones, relaciones y semántica
- Nombre:
 - Simple: expresión (nombre: “cliente”)
 - Camino: paquete :: expresión
 - Extraído del vocabulario del problema

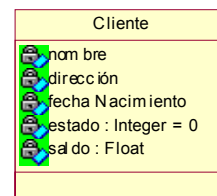
Cliente

TPV::cliente

Modelo de clases -3

Clases (II)

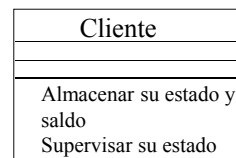
- Atributo:
 - Propiedad de una clase identificada con un nombre, compartida por todos los objetos
 - Describe rango de valores: tipo
 - Una clase puede tener 0, 1 o varios atributos
 - Especificación:
 - Nombre: si hay varias palabras, iniciales en mayúscula
 - Tipo: primitivo o definido
 - Valor inicial
 - Otras características



Modelo de clases -4

Clases (III)

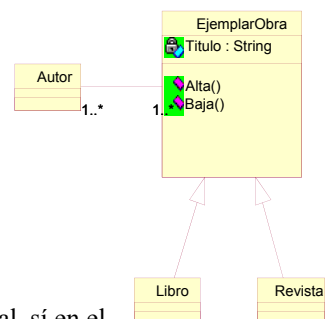
- **Responsabilidades:**
 - Contrato u obligación de una clase
 - Los atributos y las operaciones son el medio para cumplir las responsabilidades de la clase
 - Una clase puede tener cualquier número, incluso ninguna, y no demasiadas
 - Se deducirán de análisis
 - **Especificación**
 - Texto libre



Modelo de clases -5

Relaciones entre clases

- En general, muy pocas clases están aisladas
- Tres tipos de relaciones:
 - **Asociaciones:**
 - Relaciones estructurales
 - **Generalizaciones:**
 - Abstracciones/especializaciones
 - **También dependencias:**
 - Pero no aparecen en modelo conceptual, sí en el diagrama de clases de diseño



Modelo de clases -6

Asociación (I)

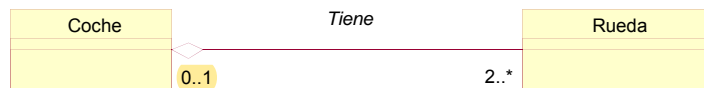
- **Asociación:** relación estructural
 - Objetos de una clase conectados a los de otra
 - Puede ser reflexiva
 - Normalmente binaria: se admiten n-arias
- **Especificación**
 - Nombre (y flecha de lectura)
 - Rol de cada clase (cómo se presenta la clase a la/s otra/s)
 - Multiplicidad, mínima y máxima: 0, 1, * (y nº exacto o lista de valores).



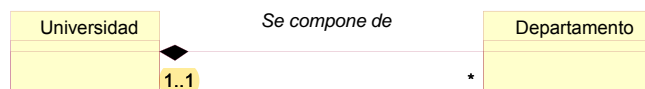
Modelo de clases -7

Asociación (II)

- **Agregación:** relación “todo/parte” o “tiene un”
 - Asociación entre iguales (existencia independiente)
 - Sólo distingue el todo de la parte, no liga la existencia del todo y sus partes



- **Composición:** agregación fuerte
 - La clase “parte” solo puede pertenecer a un “todo” y siempre debe pertenecer a él (cardinalidad 1..1)
 - Las “partes” viven y mueren con el “todo”, borrado en cascada



Modelo de clases -8

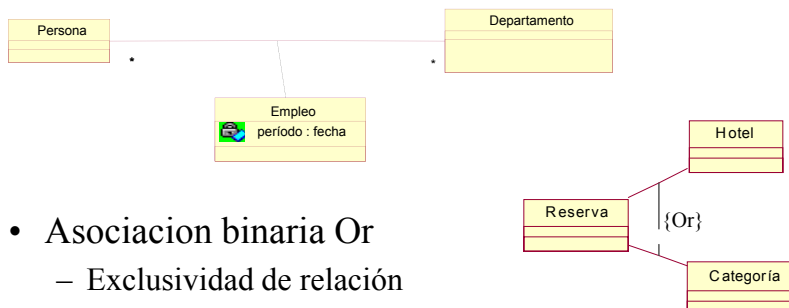
Asociación (III)

- Interpretación práctica:
 - Una relación de asociación con una semántica del tipo ‘está formado por’ pasa a ser de agregación
 - Una relación de agregación con cardinalidad 1:1 en la parte del ‘todo’ pasa a ser de composición
- Asociaciones n-arias
 - Se representa con un rombo
 - En binaria y en n-aria puede haber clase de relación

Modelo de clases -9

Asociación (IV)

- Clase de asociación:
 - Sirve para añadir atributos, operaciones, etc.

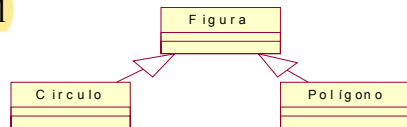


- Asociación binaria Or
 - Exclusividad de relación

Modelo de clases -10

Generalización (I)

- Generalización:
 - Relación entre un elemento general y un caso específico
 - Relación “es un tipo de”
 - El hijo hereda atributos y operaciones
 - UML permite herencia simple y múltiple
 - Puede tener nombre (es raro)
 - Se permiten ciertas restricciones: solapada/disjunta, total/parcial



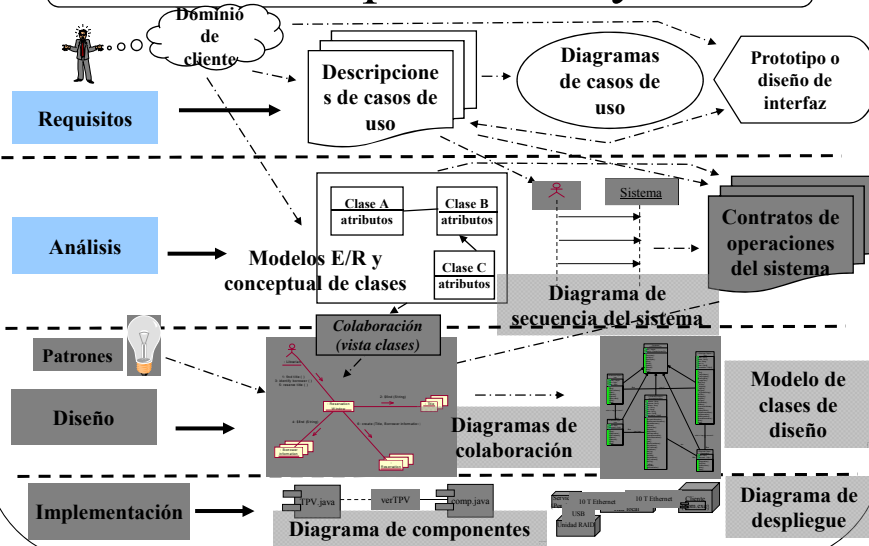
Modelo de clases -11

Generalización (II)

- Relaciones de herencia:
 - Buscar responsabilidades y atributos comunes
 - Elevar/factorizar lo común a clase general
 - Especificar lo concreto a la clase específica
- Niveles:
 - Posibilidad de más de uno
 - Polimorfismo de operaciones: implementación concretas en niveles inferiores de operaciones definidas en niveles superiores

Modelo de clases -12

Conexión rápida fases y UML



Modelo de clases -13

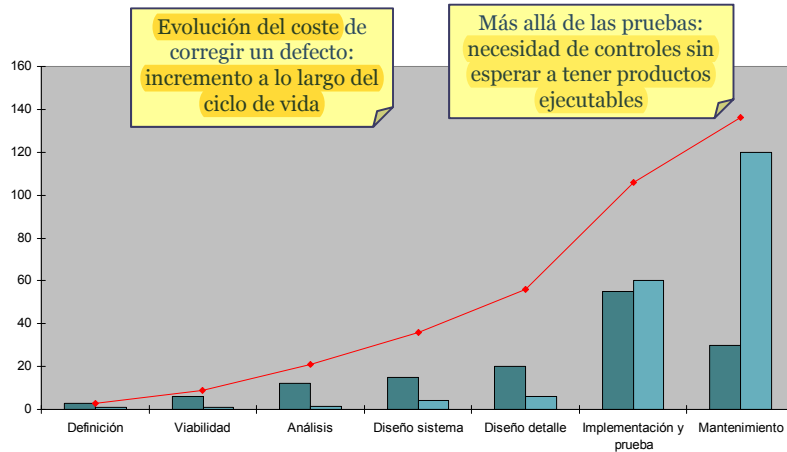
Revisiones del software

Ingeniería del Software Avanzada
Revisiones

V y V

- **Concepto:**
 - Conjunto de procedimientos, técnicas y herramientas
 - Uso paralelo al desarrollo de software
 - Asegurar que el producto satisface necesidades
- **Boehm:**
 - **Verificar:** ¿construir correctamente el producto?
Comprobar calidad en el ciclo de vida
 - **Validar:** ¿construir el producto adecuado? Comprobar si satisface los requisitos

Evolución de costes de corrección



Diferentes técnicas

- Tres aspectos que diferencian a las diferentes técnicas de revisión entre sí:
 - Sobre qué se realiza la revisión (producto o proyecto).
 - Objetivos perseguidos (aunque genéricamente sean la búsqueda de defectos).
 - El proceso o mecánica de realización.
- En cuanto a su mecánica:
 - Revisiones informales: no hay procedimientos definidos, se realiza de la forma más flexible posible.
 - Revisiones semi-formales: se definen unos procedimientos mínimos a seguir.
 - Revisiones formales: se define completamente el proceso, los participantes y sus funciones, los documentos, etc.

Clasificación de técnicas

- Revisiones de **proyecto**
 - Revisiones de gestión
- Revisiones de **producto**
 - Revisiones técnicas
 - Inspecciones
 - Walkthroughs
 - Pruebas
 - Otras
- Auditorías de software

Dar **recomendaciones para actividad** de gestión según estado de producto
Control de proyecto
Cambio de dirección de proyecto

Evaluación más o menos formal del software: identifica desviaciones
Satisfacción de especificaciones
Conformidad con planes, estándares, guías aplicables a cada producto

Revisión independiente
Evaluación objetiva de prod. y proy.
Conformidad o implantación: estándares, procedimientos, etc. contractuales o normativos

Comparativa

	Mecánica	Objeto a probar	Errores	Resultado
Revisión de gestión	Formal / Equipo cualificado	Proyecto	De gestión y planificación	Lista de errores y recomendaciones, no soluciones
Revisión técnica	Formal / Equipo cualificado	Análisis, diseño y pruebas	Funcionales, técnicos y normativos	Lista de errores
Inspección	Formal / Equipo con autor y otros	Análisis, diseño, construcción y pruebas	Funcionales, técnicos y normativos	Lista de errores (soluciones en Tercera hora, opcional) y seguimiento
Walkthrough	Cualquiera / Equipo con autor y otros	Análisis, diseño, construcción y pruebas	Funcionales, técnicos, normativos y estilo	Lista de errores y soluciones
Prueba	Cualquiera / Autor y otros mismo nivel	Construcción	Funcionales y técnicos	Lista de errores
Auditoría de software	Formal / Auditor	Proyecto, análisis, diseño, construcción y pruebas	Funcionales, normativos y estilo	Lista de errores y recomendaciones, no soluciones

Otras verificaciones y validaciones

- Cada proyecto y organización decidirá las que necesita
- Ejemplos de otras técnicas complementarias:
 - **Análisis de algoritmos.** Verificar funcionalidad y consumo de recursos en tiempo de ejecución.
 - **Análisis de simulación.** Evaluación del rendimiento para planificar la capacidad de un sistema.
 - **Audidores de código.** Examinar código fuente y controlar cumplimiento de estándares y prácticas de programación.
 - **Generadores de referencias cruzadas.** Control basado en nombres de variables, procedimientos, etiquetas, etc.
 - **Analizadores de flujo de control.** Determinar secuencias incorrectas en la ejecución del flujo de control de un programa.
 - **Comprobación de interfaces.** Analizar consistencia y compleción y analizar usabilidad y accesibilidad.
 - **Análisis de requisitos.** Errores sintácticos, inconsistencias lógicas o ambigüedades en entradas, salidas, procesos y datos.

Pruebas - Definiciones

- **Pruebas**
 - Ejecución de software con datos controlados (casos) con el fin de descubrir defectos
- **Caso de prueba**
 - Conjunto específico de entrada, procedimientos y salida esperada para una situación de operación
 - Buen caso de prueba, gran probabilidad de detectar defectos no encontrados antes
- **Éxito**
 - Descubrir un defecto no detectado
- Dijkstra: Las pruebas no pueden asegurar la ausencia de defectos, sólo demostrar que (los que localizamos) existen en el software

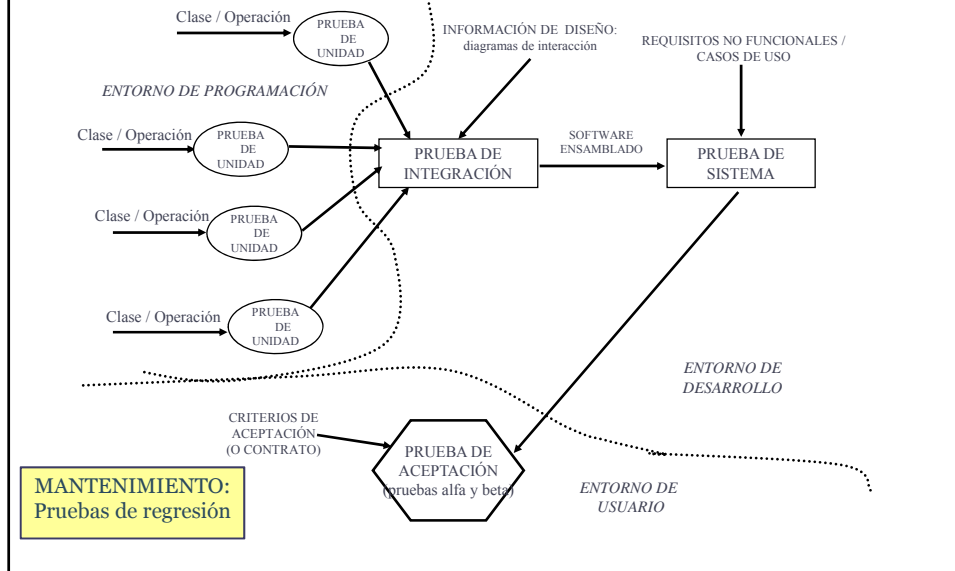
Pruebas - Recomendaciones generales

- 😊 ¿El autor prueba su software?
- 😊 Cada caso debe definir en detalle la salida esperada
- 😊 Inspeccionar con detalle la salida obtenida
- 😊 Incluir tanto entradas no válidas e inesperadas como válidas y esperadas
- 😊 Comprobar si el software:
 - No hace lo que debe hacer (fallo de funciones)
 - Hace lo que supone no debe hacer (efectos secundarios)
- 😊 Evitar utilizar casos desechables (control y economía)
- 😊 No planear suponiendo ausencia de defectos
- 😊 Estudios:
 - Probabilidad de nuevos defectos es proporcional al número de defectos ya encontrados

Pruebas - Importancia práctica

- El diseño de casos es totalmente dependiente de una buena especificación
 - Muchas veces, el trabajo de pruebas supone hacer el trabajo que no hicieron los analistas
- En cuanto tenemos una buena especificación se pueden diseñar los casos de prueba
 - En especial, si contamos con casos de uso
 - No se diseña justo antes de probar
- Eficiencia y limitación de recursos
 - No buscar pruebas perfectas: equilibrio riesgo-coste
- Como última fase, las pruebas sufren los retrasos de todas las fases de desarrollo
 - La planificación de pruebas debe contemplar plazos generosos


Pruebas - Niveles



Plan de pruebas

- **Documento** que contiene **tipos de pruebas a realizar**. Para cada tipo de prueba:
 - **Quien** prueba: individual o equipo (número de personas), autor u otros (definir nivel), varias personas o equipos prueban lo mismo o solo una prueba
 - **Cuando** y **plazo estimado de duración**
 - **Con que métodos** (diseño de pruebas) y herramientas
 - **En que entorno** (desarrollo, pre-producción, específico para pruebas, producción)
 - **Que documentación** se proporciona al iniciar la prueba y se debe generar al finalizarla

Diseño de pruebas

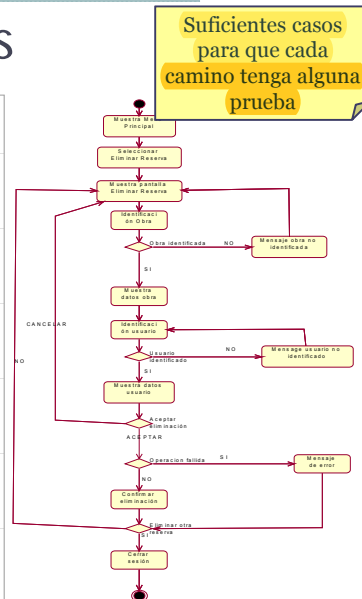
- Tres enfoques:
 - Funcional o caja negra 
 - Particiones de equivalencia
 - Valores límite
 - Conjetura de errores
 - Estructural o caja transparente
 - Grafo de flujo
 - Complejidad ciclomática de McCabe
 - Caminos de prueba ($a-n+2$)
 - Criterios de cobertura (de sentencias, de decisiones,...)
 - Aleatoria

Prueba de sistema a partir de casos de uso

- Generar casos de prueba:
 - Caminos/escenarios de ejecución de caso
 - Diagrama de secuencia del sistema, de actividad, de interacción o de estados
 - Recogen combinaciones y opciones no válidas (escenarios alternativos)
 - Sobre cada escenario, diseño de caja negra:
 - Clases de equivalencia y límites
 - Determinar los datos de entrada/salida de los casos de prueba
 - No olvidar efectos secundarios (acciones no visibles al usuario)

Determinar escenarios

Bibliotecario	Sistema
	1.- Muestra por pantalla el menú principal
2.- Selecciona la opción <i>Eliminar Reserva</i>	3.- Muestra la pantalla de <i>Eliminar Reserva</i>
4.- Introduce el <i>idObra</i> <extends Buscar Título >	5.- Muestra los datos de la obra y pide el DNI del usuario.
6.- Introduce el DNI <extends Buscar usuario >	7.- Muestra datos del usuario y confirmar eliminar reserva
8.- Pulsa <i>ACEPTAR</i>	9.- Mensaje de confirmación y si eliminar otra reserva.
10.- Pulsa <i>NO</i>	11.- Cierra sesión y muestra menú principal



Particiones de equivalencia

Añadir valores límite y casos especiales de conjetura de errores

Condición o restricción de entrada	Clases válidas	Clases no válidas
Código de área	(1) $200 \leq \text{código} \leq 999$	(2) código < 200 (3) código > 999 (4) no es número
Texto de la operación	(5) 6 caracteres	(6) < 6 caracteres (7) > 6 caracteres
Operación	(8) ingreso (9) transferencia (10) pago recibo (11) retirada efectivo	(12) ninguna orden válida

Casos válidos:

- 300 Nómina Ingreso (1)(5)(8)
- 400 Bilbao Transferencia (1)(5)(9)
- 500 I.T.V. Pago recibo (1)(5)(10)
- 600 Cheque Retirada efectivo (1)(5)(11)

Casos no válidos:

- 180 Nómina Ingreso (2)(5)(8)
- 1032 XXXXXX Ingreso (3) (5) (8)
- ZV YYYYYY Ingreso (4) (5) (8)
- 350 ↵ Transferencia (1)(6) (8)
- 450 Regalos Transferencia (1)(7) (8)
- 550 IRPF97 Saldo (1)(5)(12)

Inspección

- **Objetivos:** enfocada en descubrir defectos
 - No sobre cómo corregir, añadidos o mejoras
- **Equipos de inspección**
 - Pequeños grupos de compañeros de trabajo (de 3 a 6): 4-5, lo más común.
 - Autor y compañeros que desarrollan productos relacionados
 - Uso de listas de comprobación
 - Duración de reuniones: máximo 2 horas
- **Roles**
 - Autor no puede ser moderador o lector (presenta el producto)
 - Los moderadores capacitados son esenciales: asegurar preparación, ritmo de reunión, enfoque en defectos y evitar ataques a autor
- **Productos**
 - Aplicable a muchos productos: especificaciones, diseño, código, pruebas, ..
 - Tamaño habitual: 10-20 páginas o 200-250 LOC
- **Salidas**
 - Lista de defectos e informe de revisión: qué, quién, nº y gravedad de defectos
 - Tasa de inspección (velocidad en páginas o LOC por hora)

Inspección - Fases

Etap	Responsable	Actividades
Planificación	Moderador	Producto cumple requisitos de entrada Conseguir participantes adecuados Fijar lugar y momento adecuados
Vista general (opcional)	Equipo al completo	Formar a participantes Asigna papeles
Preparación	Individual	Cada participante estudia el producto, prepara su papel y anota defectos
Reunión	Equipo al completo	Encontrar defectos (sin discutir posibles soluciones)

Inspección -Tercera hora (opcional)

Etapa	Responsable	Actividades
Tercera hora	Equipo al completo	Exponer ideas suprimidas en la reunión sobre mejoras o soluciones
Corrección	Autor o a quien se asigne	Corregir defectos encontrados
Seguimiento	Moderador	Verificar la corrección de todos los defectos
Análisis causal	Equipo al completo o depto. de calidad	Analizar estadísticas de reuniones (detectar causas y mejoras en los procesos, métodos desarrollo, etc.)

Inspección - Criterios de terminación

- La inspección acaba cuando:
 - Todos los defectos detectados se han resuelto
 - Los resultados de la inspección se han pasado a los informes
- El moderador:
 - Verifica ambos criterios antes de declarar terminada y completa la inspección
- Cada proyecto debería desarrollar criterios propios según las necesidades del mismo y del entorno

Inspecciones - Listas de comprobación

- Series de preguntas o comprobaciones para examinar el producto
- Proporciona definición clara de la tarea a los participantes
- Se construyen a base de acumular experiencias en revisiones
 - Asimilar las estadísticas de defectos
- Extensión:
 - Una sola página con 20-25 preguntas o ítems

Walkthroughs

- **Walkthrough:** recorrido del producto
- Revisión de cualquier producto por un grupo de nivel similar al que lo desarrolló
 - En general, del equipo de desarrollo
- **Objetivo:** el autor explica el producto (presenta el enfoque de diseño y programación e informa a compañeros del progreso de trabajo) mientras los demás se centran en:
 - **Encontrar defectos:**
 - Errores, omisiones, contradicciones, debilidades, mejoras
 - Estudio de la técnica y el estilo de desarrollo
 - Búsqueda de alternativas y soluciones

Walkthroughs - Tipos

- Según el grado de organización y estructuración:
 - **Formales:**
 - Preparación larga, autor gasta tiempo en presentar, realimentación alta y lenta (semanas, buen trabajo)
 - **Informales**
 - Preparación muy baja, realimentación rápida y baja
 - **Semiformales**
 - Lo más recomendable: ventajas de ambos
- En ciclo de vida:
 - Informales en primeras fases y formales en las finales

Walkthroughs - Roles

- **Autor**
- **Moderador** (puede ser el propio autor)
- Posibles miembros del equipo:
 - **Encargado de mantenimiento:**
 - Prever futuro de mantenimiento
 - Producto autoexplicativo, mantenimiento sin el autor, etc.
 - **Supervisor de estándares**
 - Conformidad respecto de normativa y estándares
 - No literal sino el espíritu y teniendo en cuenta el entorno
 - **Representante de usuario**
 - Ajuste a las necesidades, especialmente en especificaciones
 - **Otros revisores**, normalmente del mismo nivel que el autor
 - Opinión general sobre corrección y calidad
 - Posibles externos para visión más distanciada

Walkthrough - Fases

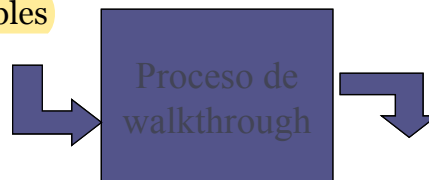
Depende de su grado de formalidad

- Planificación, incluye lista de comprobación
- Reunión de visión general
- Preparación individual
- Reunión de trabajo:
 - Reconocimiento de objetivos:
 - Presentador: para guiar la reunión
 - Presentación del producto (si procede)
 - Breve, si ha habido suficiente preparación
 - Críticas y comentarios
 - Incluyendo walkthroughs anteriores
 - Calificación final:
 - Aceptado, aceptado con modificaciones, nuevo walkthrough.

Walkthrough - Documentación

Depende de su grado de formalidad

- Elemento de software
- Lista de objetivos
- Estándares aplicables
- Especificaciones



- Lista de elementos de acción: defectos y para cada uno grado de gravedad (indicar si es crítico para aceptación) y acciones (para corregir y/o mejorar)
- Informe del *walkthrough*: qué, quién, objetivos y grado de alcance, conclusión y propuestas para posterior seguimiento

Auditorías

- **Objetivo:**
 - Confirmar el cumplimiento de producto y/o proyecto para asegurar el cumplimiento de:
 - Estándares, Guías, Especificaciones, Procedimientos, Ejecución de proyecto y productividad
- **Revisión independiente y muy disciplinada**
 - No participativa: visión del auditor
 - Figura de autoridad: auditor
- **Permite evaluar:**
 - Elementos de software
 - Procesos
 - Proyectos
 - Planes de calidad

Auditorías - ¿Cuándo realizarlas?

- **Punto singular del proyecto**
 - Seguimiento de planes
 - Fecha, criterio existente, etc.
- **Demanda de grupo externo:**
 - Agencia reguladora, usuarios o clientes, etc.
 - Requisitos contractuales o normativos
- **A petición de la organización:**
 - Jefe de proyecto
 - Departamento de SQA

Auditorías - Fases

- **Planificación**
 - Objetivos de auditoría y Alcance
- **Visión general**
 - Definición del proceso de auditoría, calendario (pueden durar meses) y acuerdos de colaboración con la empresa auditada
- **Preparación**
 - Conocer la organización a auditar y preparar un plan
- **Examen**
 - Recogida de datos (documentación y entrevistas)
- **Análisis**
 - Determinar la salud del proyecto o producto usando métricas
- **Informe de resultados**
 - Defectos, descubrimientos y recomendaciones