TEMA 1: Introducción a la Ingeniería del Software

SOFTWARE

DEFINICIÓN: programas (líneas de código) + datos (estructuras de datos) + documentación

desarrollo + mantenimiento

CARACTERÍSTICAS

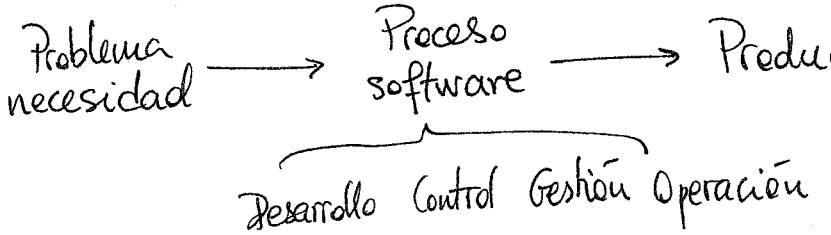
- Elemento lógico
- Desarrollado, no fabricado
- Se deteriora
- No hay "piezas de repuesto"
- Se construye a medida

CRISIS DEL SOFTWARE

- Causas
  - Hardware más potente
  - Mayor demanda
  - Falta de metodologías
  - Uso inadecuado de recursos
  - Sistemas complejos
- Consecuencias
  - → baja productivi
  - → baja calidad
- Síntomas
  - Baja productividad de los desarrollo en comparación con la demanda
  - Los sistemas no responden a las expectativas de lo usuarios
  - Los programas fallan (fiabilidad) a menudo.
  - Baja calidad
  - Costes difíciles de predecir
  - Mantenimiento costoso y complejo
  - No hay aprovechamiento de recursos (eficiencia)
- Solución → aplicar Ingeniería del Software en la construcción de

PROYECTO, PRODUCTO Y PROCESO

- Trabajo operativo: efectuar permanentemente actividades que generan un mismo producto o proveen un servicio repetitivo.
- Proyecto: esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único.



INGENIERÍA DEL SOFTWARE

DEF1: Establecimiento y uso de principios de ingeniería robustos, orientados a obtener económicamente software que sea fiable y funcione eficientemente sobre máquinas reales.

DEF2: Aproximación sistemática al desarrollo operación y mantenimiento del software.

OBJETIVOS

- Conseguir un producto fiable, de alta calidad y de bajo coste.
- Conducir un proceso de desarrollo y mantenimiento Software de manera eficiente y con éxito.

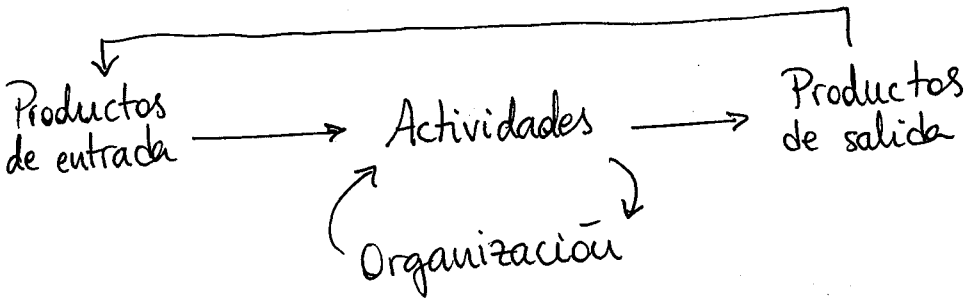
ACTIVIDADES DE DESARROLLO

Análisis → Diseño → Codificación → Pruebas → Entrega → Mant

Además: actividades de control, de gestión y de operación.

TEMA 2: Metodologías, ciclos de vida y proceso software

PROCESO SOFTWARE:



(*) Comprobar modelo de proceso soft. de IEEE

METODOLOGÍAS Y CICLOS DE VIDA

Metodología es el conjunto de métodos que se utilizan en una actividad con el fin de normalizarla y optimizarla. Determina los pasos a seguir y cómo seguirlos.
Aplicada a la Ing. Software define qué, cómo y cuándo hacer durante todo el desarrollo y mantenimiento de un proyecto.

Ciclo de vida es el conjunto de fases por las que pasa el sistema que se está desarrollando desde que nace la idea inicial hasta que el software es retirado.
Debe determinar el orden de las fases del proceso, establecer los criterios de transición y definir entradas y salidas de cada fase

Elementos de una metodología
- tareas a realizar en cada fase
- E/S de cada fase y documentos
- procedimientos: apoyo a la realización de cada tare
- criterios de evaluación del proceso y del producto

TIPOS DE METODOLOGÍA

METODOLOGÍAS TRADICIONALES

- CASCADA
- REFINAMIENTO POR PASOS
- INCREMENTAL Y/O ITERATIV

- **CASCADA** → cascada simple / cascada con refinamiento

CASCADA CLÁSICO (SIMPLE)

- el proceso software es una sucesión de etapas que producen productos intermedios
- las fases continuan hasta que el objetivo se cumpl
- se deben hacer todas las fases y en orden.

CASCADA CON REFINAMIENTO

- los productos de las diferentes etapas se van refinando y mejorando (pequeñas mejoras a productos de fases ya acabadas)

- **INCREMENTAL**

Se analiza el problema y se subdivide en subtareas (subproblemas) que se van desarrollando en cada iteración. La división de requisitos se realiza al comienzo del proyecto. El producto sw se desarrolla por partes. En cada incremento se agrega más funcionalidad al sistema, y se integran.

- **ITERATIVO**

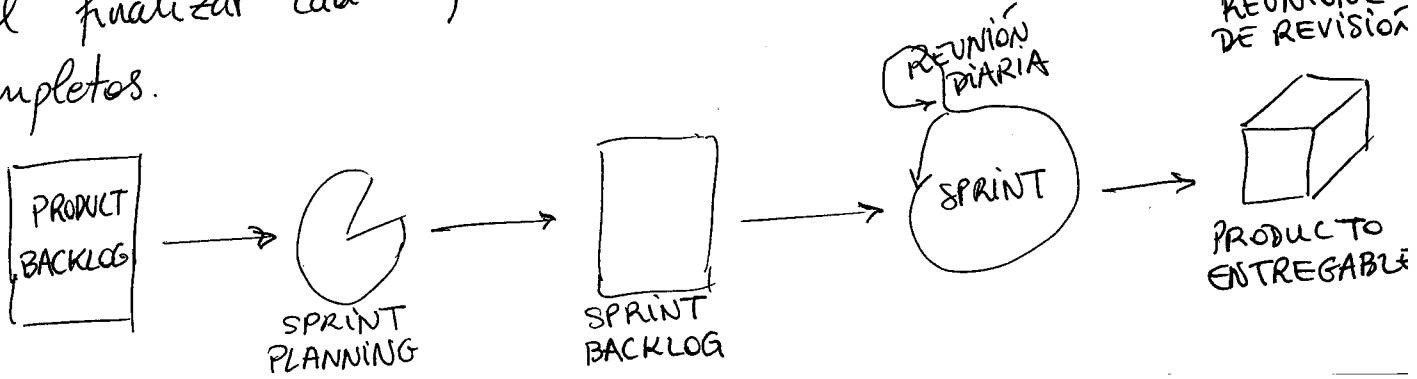
Refinamiento por pasos pero, normalmente, se iteran todas la fases cada vez. En cada ciclo se revise y mojara la calidad Permite construir una implementación parcial del sistema global y posteriormente ir aumentando la funcionalidad del sistema (incorporar nuevos requisitos). El producto final de cada iteración será usado en el entorno operativo.
Produce un sistema operacional rápidamente.

METODOLOGÍAS ÁGILES: proporcionan una visión más ágil y dinámica frente a las metodologías tradicionales, que pueden resultar más pesadas según el contexto del proyecto.

- Características:
  - máxima prioridad es la satisfacción del cliente a través de entregas continuas y evaluable
  - cambios de requisitos posibles en cualquier fase
  - entregas frecuentes (2 semanas - 2 meses)
  - interesados y desarrolladores trabajan juntos
  - los equipos deben reflexionar cómo ser más eficaces y mejorar la calidad.

Más utilizadas: eXtreme Programming, KANBAN, SCRUM.

- SCRUM
  - Se basa en sprints: iteracciones de desarrollo de corta duración (2-4 semanas).
  - Durante cada sprint el equipo crea un incremento software
  - El product backlog recoge un conjunto de requisitos de alto nivel priorizados, que definen el trabajo a realizar.
  - El sprint backlog recoge los elementos que forman parte del sprint actual, decididos en el sprint planning donde el product owner identifica los elementos del product backlog que quiere ver completados.
  - Al finalizar cada sprint se entregan productos funcionales completos.



- Roles principales SCRUM
  - PRODUCT OWNER: representa la voz del cliente. Conoce las prioridades del proyecto, organiza y dirige las tareas. Escribe las historias de usuario y las pone en el PB.
  - SCRUM MASTER: asegura el seguimiento de la metodología. No es el líder del equipo.
  - SCRUM TEAM: responsables de implementar las funcionalidades. Descentralizado y auto-dirigido.

- Roles secundarios
  - STAKEHOLDERS: interesados en el negocio: clientes, vendedores, proveedores. Participan en las revisiones del sprint.
  - OTROS: usuarios finales y expertos del negocio.

METODOLOGÍA MÉTRICA: Desarrollada por el ministerio para la administración pública.
- Surge para paliar la problemática de los Dtos. de I que no utilizan ninguna metodología.
- Escasa o nula documentación: mantenimiento difícil.
- Falta de comunicación con el cliente.

DESARROLLO CENTRADO EN EL USUARIO: Filosofía de diseño q tiene como objetivo la creación de productos que resuelvan necesidades concretas. El usuario juega un papel importante antes, durante y después.
Objetivo: Asegurar la usabilidad del producto final
Fases → Conocer a fondo a los usuarios
→ Diseñar un producto que resuelva sus necesidades
→ Poner a prueba lo diseñado (test de usuario)

TEMA 3 : Actividades tempranas en el desarrollo Software

DEFINICIÓN DE CICLO DE VIDA

→ Definir fases, orden, E/S, características proyecto

DEFINICIÓN DE ESTÁNDARES, MÉTODOS, TÉCNICAS Y HERRAMIENTAS

DEFINICIÓN DE OBJETIVOS, ENTRAS/SALIDAS Y FUNCIONES BÁSICAS

DEFINICIÓN DE CRITERIOS DE CALIDAD Y VALIDACIÓN

De acuerdo con los requisitos, resultados esperados, plan de validación, plan de calidad y atributos de calidad.

DEFINICIÓN DE HITOS

→ productos de cada hito
→ qué, quién, cuándo y cómo se va a evaluar.
→ se consigue un hito cuando se ha revisado la calidad

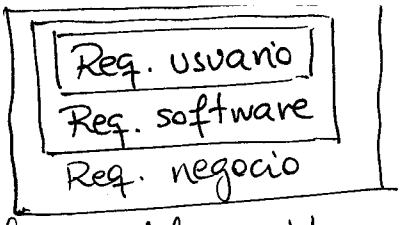
CREACIÓN DE EQUIPO DE DESARROLLO INICIAL

→ El personal es el factor más importante
→ Definir jefe del proyecto
→ Definir miembros del equipo y papeles.

DEFINICIÓN DE MECANISMOS DE SEGUIMIENTO Y CONTROL

→ Cumplimiento de tiempos
→ Evaluación de objetivos
→ Coordinación del equipo de desarrollo
→ Revisiones

TEMA 4: Análisis y diseño



PARTE 1: ANÁLISIS

DEF: El análisis de requisitos es el análisis del problema y especificación completa del comportamiento externo que se espera del sistema software que se va a construir, así como de los flujos de información y control.

Def. de requisitos →
- para usuario: condiciones o capacidade necesarias para que el usuario pueda resolver un problema o alcanzar un objetivo.
- para equipo de desarrollo: condiciones o capacidades que debe reunir un sistema para satisfacer un contrato, estándar o cualquier otro documento impuesto formalmente.

REQUISITOS DE USUARIO Y SOFTWARE

- Requisitos de usuario: declaraciones en lenguaje natural de las funciones o acciones que los distintos usuarios pueden realizar con la aplicación y bajo qué restricciones.
  Forman el documento de requisitos de usuario (DRU)
- Requisitos software: especifican de una manera completa, consistente y detallada qué debe hacer y cómo debe comportarse el software para cumplir con los objetivos de la aplicación.
  Sirve a los desarrolladores de base para diseñar el sistema.
  Se recogen en la especificación de requisitos Software (ERS)
  Debe responder a la pregunta "¿qué características necesita cumplir el sistema software para permitir alcanzar los requisitos expuestos en el DRU?"

Principios del análisis

- Se debe comprender el problema y su entorno
- Los requisitos han de determinarse siguiendo una aproximación descendente, primero globalmente y luego al detalle.
- Se debe separar el qué del cómo.
- La especificación de requisitos debe poder ser ampliable.

Tareas:

1. Educción de requisitos
2. Análisis de usuarios y tareas.
3. Análisis del problema y de los requisitos.
4. Modelización
5. Validación

TIPOS DE REQUISITOS

- FUNCIONALES: Acciones fundamentales que tienen que tener lugar en la ejecución del software.
  Son acciones elementales necesarias para el correcto comportamiento de nuestro sistema.
- NO FUNCIONALES: Representan características o cualidades generales que se esperan del software para conseguir su propósito.
  - De interfaz
  - Usabilidad
  - Operacionales
  - Soporte
  - Seguridad
  - Rendimiento
  - Mantenibilidad
  - Fiabilidad
  - Recursos
  - Comportamiento
  - Disponibilidad
  - Legales

ANÁLISIS DE REQUISITOS

¿Qué? – Proceso a través del cual se determina qué requisitos son aceptables y se definen cuáles van a formar parte del producto.

¿Cómo? –

Evaluación de viabilidad técnica y económica
Valoración de riesgos
Clasificación de requisitos en categorías

REPRESENTACIÓN: proceso de registrar los requisitos de una o más formas y de especificar aquellos requisitos todavía no educidos.

¿Cómo? lenguaje natural, catálogo de requisitos, lenguaje formal, modelos, diagramas, maquetas.

- Técnicas de representación de requisitos
  - análisis estructurado ⇉ orientadas a datos / orientadas a funciones / orientadas a estados
  - análisis orientado a objetos → diagramas UML → casos de uso / de iteración
  - maquetas
  - lenguaje formal

El cliente no suele tener una idea clara de lo que quiere, no sabe explicarlo bien. Para ello, los desarrolladores usan maquetas o prototipos.

VALIDACIÓN DE REQUISITOS

Los requisitos deben ser
- completos
- no ambiguos
- relevantes
- trazables
- correctos y consistentes

Véase DOCUMENTO FINAL: ESPECIFICACIÓN DE REQUISITOS SOFTWARE.

PARTE 2: DISEÑO

Cambiar la atención del qué al cómo

DEF: Es el proceso de definición de la arquitectura, componentes, módulos, interfaces, procedimientos de prueba y datos de un sistema software para satisfacer unos requisitos especificados.

NIVELES DE DISEÑO

- Diseño de la arquitectura: proceso de definición de la colección de componentes del sistema y sus interfaces.
  Objeto: determinar el marco de referencia que guiará la construcción del sistema.
- Diseño detallado: proceso de descripción detallada de la lógica de cada uno de los módulos, de las estructuras de datos que utilizan y de los flujos de control.
  Objeto: describir el sistema con el grado de detalle suficiente y necesario para su posterior implementación.

MÉTODOS DE DISEÑO

- DISEÑO ESTRUCTURADO: considera que el sistema es un conjunto de módulos, cada uno con una función bien definida, organizados de forma jerárquica.
- DISEÑO ORIENTADO A OBJETOS: considera que el sistema es una colección de objetos que interactúan a través de mensajes. Cada objeto tiene su propio estado y operaciones

ANÁLISIS Y DISEÑO ORIENT. A OBJETOS CON UML

Diagramas estructurales
- D. de clases
- D. de objetos
- D. de componentes
- D. de despliegue
- D. de paquetes

Diagramas de comportamiento
- D. Casos de Uso
- D. de secuencia
- D. de comunicación
- D. de estados
- D. de actividad

MÉTRICAS DE DISEÑO

- Cohesión: es una medida de la relación (funcional) de los elementos de un módulo.
- Acoplamiento: es una medida de interconexión entre los módulos de un programa.

DOCUMENTO FINAL: DISEÑO DEL SOFTWARE

TEMA 5 : Codificación, pruebas, entrega y finalización

DEF. CODIFICACIÓN: traducir las especificaciones de diseño en un lenguaje de programación.

DEF. ERROR SOFTWARE: el software no hace lo que el usuario espera que haga, acordado previamente en la especif. requisitos.

DEF. PRUEBA: proceso de ejecutar un software con el fin de encontrar errores.

VERIFICACIÓN Y VALIDACIÓN

Verificación: comprueba el funcionamiento del software, que implemente correctamente una función específica.

Validación: comprueba si los requisitos de usuario se cumplen y los resultados obtenidos son los previstos.

1) PRODUCT BACKLOG: conjunto de todas las historias de usuar donde cada historia es un requisito del cliente

2) SPRINT PLANNING: se selecciona el conjunto de historias de usuario a realizar en el sprint.

3) SPRINT BACKLOG: se escriben y se descomponen en tareas las historias de usuarios a realizar en este sprint.

4) FORMATO TARJETA: se pasan a este formato las historias y se colocan en el tablero

5) DAILY MEETINGS: acta informal (½ pág) ← actualización tablero de tareas → plasmar estado final tablero

7) FIN SPRINT: entrega de dos documentos

SPRINT REVIEW
↓
acta de la sprint review

→ doc. de análisis: diagramas de casos de uso, especificación de uso de al menos tres de los requisitos principales.

→ doc. de diseño. diagr. de clases diagr. de secuencia relacionados con los casos de uso anteriores.

Documentos de reflexión

RECORDAR
→ tablero de tareas
→ diagrama de quemado

APLICACIÓN COMETELOOTOO : sistema de envío comida a domicilio.

4 perfiles de usuario: propietarios de restaurantes
repartidores ~~autónomos~~
clientes del servicio
administrador único

~~Registro~~: [...] subrayado escritorio

EJERCICIOS TEMA 2

1. Cascada iterativa → pequeñas iteraciones para aplicar ligeros cambios

2. Cascada (clásico-tradicional)

3. Scrum puro → las empresas tienen que estar muy bien coordinadas, además se realizan en equipos pequeños y los requisitos están sin definir

4. Cascada iterativa con prototipo
   - Sin modificar la funcionalidad se implementan cambios.
   - tienen clara la funcionalidad pero no la interfaz ni el servidor

   No sería SCRUM porque una vez que tenemos la funcionalidad, esta no es movida/cambiada.

5. Dos posibilidades → centrada en el usuario
   → modelo en cascada (clásica) con maqueta (o prototipo

6. Centrada en el usuario porque:
   - tiene que valer para cualquier ciudadano
   - la usabilidad es muy importante

   Y una segunda fase (después de centrar los requisitos) de desarrollo con una metodología métrica.

7. Incremental

   No es iterativo porque los equipos van desarrollando cada uno por su lado y al final lo juntan y lo evalúan.