

INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE

F.Javier Zarazaga Soria
Javier Nogueras Iso
Ingeniería del Software I
febrero 2.008



Departamento de Informática e Ingeniería en Sistemas

Índice

- Introducción y conceptos básicos
 - El ciclo de vida
 - Herramientas CASE



INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE

Introducción y Conceptos Básicos

F.Javier Zarazaga Soria
Javier Nogueras Iso
Ingeniería del Software I



Intr. y conceptos básicos. Índice

- Historia del software
- Características especiales del software
- Crisis del software e Ingeniería del software
- Mitos del software
- La industria del software
- Sistemas de información
- El analista de sistemas
- Estandarización del desarrollo del software
- Introducción a Métrica 2



Breve sinopsis de la historia del SW (I)

□ 1^a generación del software

- ❖ ??? - 1965
 - ❖ Hardware de propósito general
 - ❖ Software como algo añadido
 - ❖ Desarrollo a medida
 - ❖ Ninguna planificación
 - ❖ Orientación por lotes

□ 2^a generación del software

- ❖ 1965 - 1975
 - ❖ Sistemas multiusuario
 - ❖ Interactividad (Tiempo Real)
 - ❖ Almacenamiento y bases de datos
 - ❖ La industria del software
 - ❖ Software de gran volumen
 - ❖ Mantenimiento



Breve sinopsis de la historia del SW (II)

- 3^a generación del software
 - ❖ 1975 - 1990
 - ❖ Microprocesadores, PCs y sistemas distribuidos
 - ❖ Hardware de bajo coste
 - ❖ Industria planetaria

 - 4^a generación del software
 - ❖ 1990 - ????
 - ❖ Tecnologías Orientadas a Objeto
 - ❖ Interfaces gráficas de usuario
 - ❖ Sistemas expertos
 - ❖ Proceso paralelo
 - ❖ Tecnologías de componentes
 - ❖ COTS (Commercial Off-The-Shelf)
 - ❖ Internet y Servicios Web



Terminología habitual en el software



□ Bug:

- ❖ Chinche, bicho, microbio
 - ❖ Fastidiar, molestar

□ Patch:

- ## ❖ Parche, remiendo, zurcido

□ Desfase de presupuesto

- ## ❖ Costes por encima de lo previsto

☐ Retrasos en entregas

- ## ❖ No cumplimiento de plazos

Mantenimiento

- ❖ Rehacer la aplicación añadiendo nuevas posibilidades y mejorando las existentes

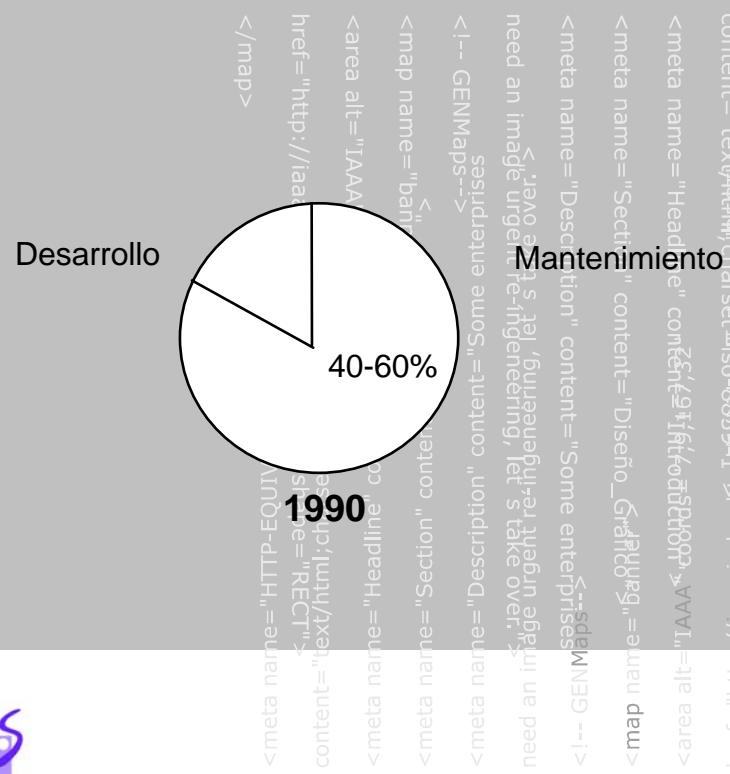
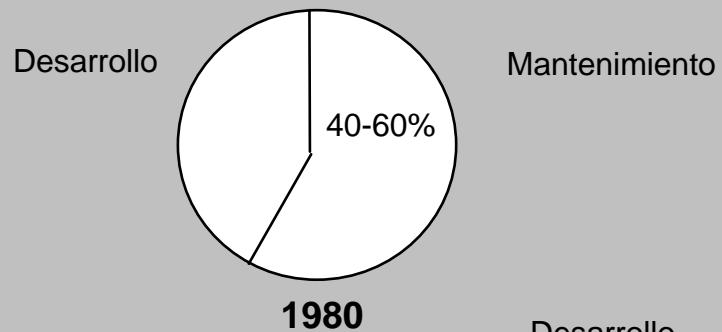
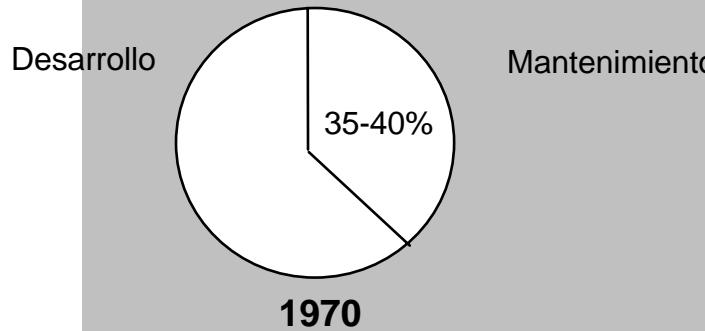


Críticas a las aplicaciones Software

- Retrasos no previstos
- Desbordamiento de costes
- Software no acorde con los requisitos
- Errores en los programas
- Sensibilidad a los errores humanos y a las averías físicas
- Dificultad de puesta en marcha
- Dificultad de evolución
- Mantenimiento ruinoso



Coste de Mantenimiento del Software



La complejidad del Software (I)

“La complejidad del software es una propiedad esencial, no una propiedad accidental”

La complejidad del Software (II)

- Motivos que llevan a que el software sea complejo
 - ❖ Complejidad del dominio del problema
 - Imagen que del dominio del problema tiene el cliente
 - Imagen que del dominio del problema tiene el desarrollador
 - El dominio del problema en si
 - ❖ La dificultad de la gestión del proceso de desarrollo
 - ❖ La flexibilidad del desarrollo software
 - Necesidad de grandes labores de abstracción
 - Falta de estándares
 - ❖ Problemas en la caracterización del comportamiento de sistemas discretos
 - Gran volumen de variables
 - Interacciones entre las mismas

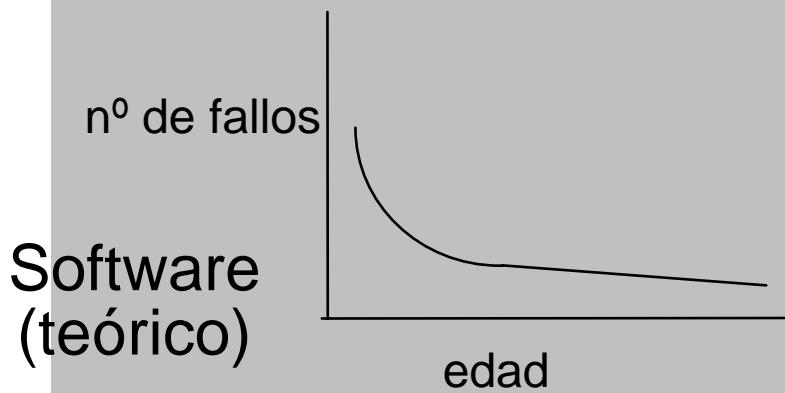
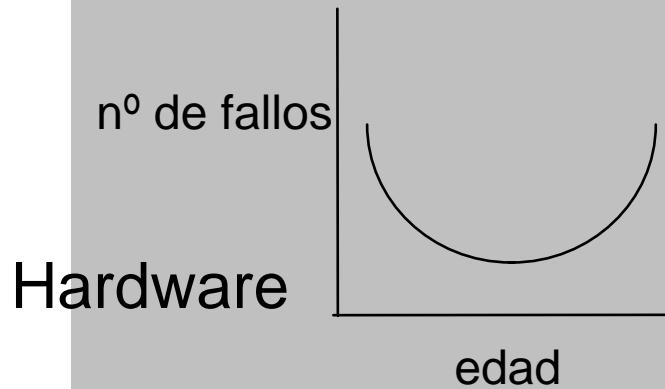


Peculiaridades del Software

- El producto software es enteramente conceptual.
- No tiene propiedades físicas como peso, color o voltaje, y, en consecuencia no está sujeto a leyes físicas o eléctricas.
- Su naturaleza conceptual crea una distancia intelectual entre el software y el problema que el software resuelve.
- Difícil para una persona que entiende el problema entender el sistema software que lo resuelve.
- Para probar es necesario disponer de un sistema físico.
- El mantenimiento no es sólo una substitución de componentes.



Fallos en Hardware vs fallos en Software



El software se degrada con el tiempo

cambios

Características de un buen software

- Corrección.
- Completitud.
- Concisión.
- Robustez
- Fiabilidad.
- Eficiencia.
- Integridad.
- Facilidad de uso.
- Facilidad de mantenimiento.
- Facilidad de traza.
- Generalidad.
- Modularidad.
- Flexibilidad.
- Facilidad de prueba.
- Portabilidad.
- Facilidad de reuso.
- Interoperabilidad.
- Facilidad de auditoría.
- Exactitud y precisión de cálculos.
- Consistencia.
- Estandarización de los datos.
- Independencia del Hardware.
- Legibilidad.



Origen de la Ingeniería del Software

- ❑ Aunque no hay consenso, el origen del término se atribuye a dos conferencias organizadas por la OTAN en 1967 y 1968. Ambas conferencias fueron convocadas para tratar la llamada crisis del software.
 - ❑ La llamada crisis del software es todavía hoy un problema no resuelto.

Crisis del software (I)

- Primera Fase. Los albores (1945-1955)
 - ❖ Programar no es una tarea diferenciada del diseño de una máquina.
 - ❖ Uso de lenguaje máquina y ensamblador.
- Segunda Fase. El florecimiento (1955-1965)
 - ❖ Aparecen multitud de lenguajes.
 - ❖ Era posible hacer de todo.
- Tercera Fase. La crisis (1965-1970)
 - ❖ Desarrollo inacabable de grandes programas.
 - ❖ Ineficiencia, errores, coste impredecible.
 - ❖ Nada es posible.



Crisis del software (II)

- Cuarta Fase. Innovación conceptual (1970-1980)
 - ❖ Fundamentos de programación.
 - ❖ Verificación de programas.
 - ❖ Metodologías de diseño.
 - Quinta Fase. El diseño es el problema (1980-????)
 - ❖ Entornos de programación.
 - ❖ Especificación formal.
 - ❖ Programación automática.



¿Qué es la Ingeniería del Software?

□ EI IEEE define:

- ❖ Ingeniería es la aplicación de un método sistemático, estructurado y cuantificable a estructuras, máquinas, productos, sistemas o procesos.
 - ❖ Ingeniería del software es la aplicación de un método sistemático, estructurado y cuantificable al desarrollo, operación y mantenimiento de software.

□ Bauer, 1972

- ❖ La IS es el establecimiento y uso de sólidos principios de ingeniería y buenas prácticas de gestión, así como la evolución de herramientas y métodos aplicables y su uso cuando sea apropiado para obtener, dentro de las limitaciones de recursos existentes, software que sea de alta calidad en un sentido explícitamente definido.

Comparación con otras ingenierías

- ❑ Ingeniería mecánica como buscar un gato negro en una habitación iluminada.
 - ❑ Ingeniería química como buscar un gato negro en una habitación oscura.
 - ❑ Ingeniería software como buscar un gato negro en una habitación oscura donde no hay ningún gato.
 - ❑ Ingeniería de sistemas como buscar un gato negro en una habitación oscura donde no hay gato y alguien dice !!!lo encontré!!!.

Desarrollo de Software sin Ingeniería

Especificación de Requerimientos

?

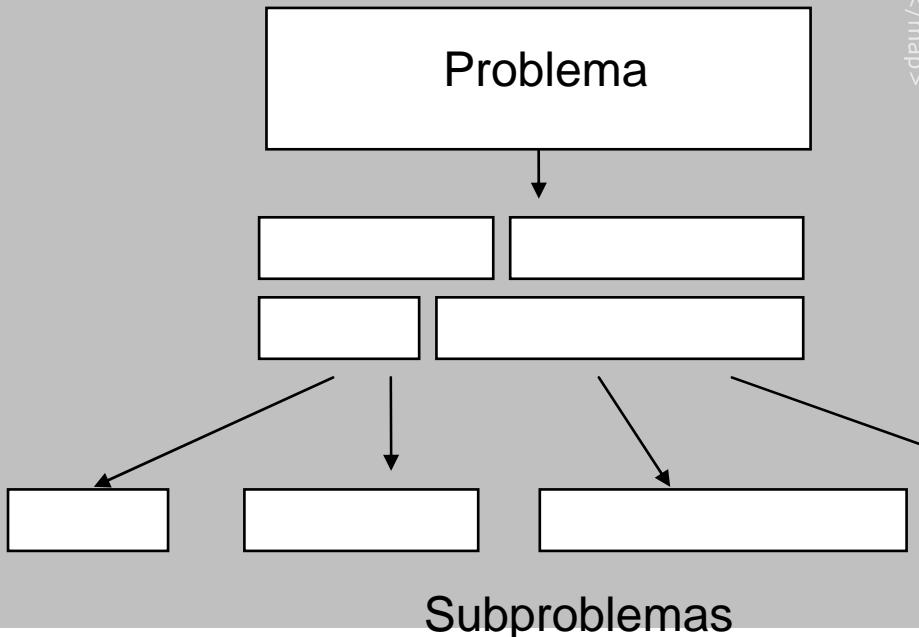
Producto Software

- Existencia de un vacío de información entre la especificación del producto software y su entrega.
- El producto surge espontáneamente después de un largo periodo de oscuridad
- ¿Qué pasa si alguien se va a mitad de un proyecto? ¿y si se va todo un equipo?



Ingeniería del SW - Análisis de problemas

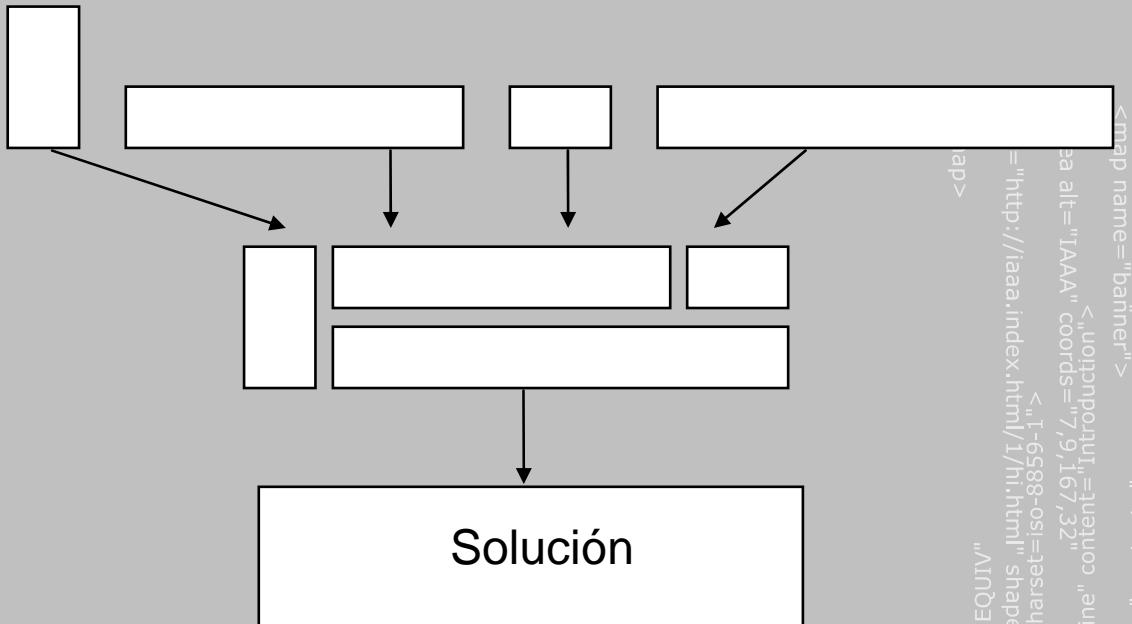
- Actividad dirigida a la solución de problemas
 - Análisis:
 - Comprender la naturaleza del problema
(descomponerlo)



Ingeniería del SW - Síntesis de problemas

□ Síntesis:

- ❖ Unir las soluciones parciales componiendo una estructura de mayor tamaño



Soluciones
parciales

<meta name="HTTP-EQUIV"

22 >



Objetivos de la Ingeniería del Software

La ingeniería del software persigue la producción de sistemas de calidad a bajo coste y a tiempo

Sistemas de calidad

La calidad de un sistema viene definida por el cumplimiento de los objetivos establecidos para el sistema

Bajo coste

El coste de un sistema debe incluir tanto el coste de desarrollo como el de mantenimiento

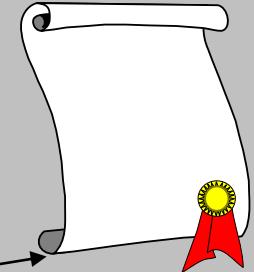
A tiempo

En unos plazos preestablecidos y que vienen garantizados por el establecimiento de una secuencia de actividades a llevar a cabo

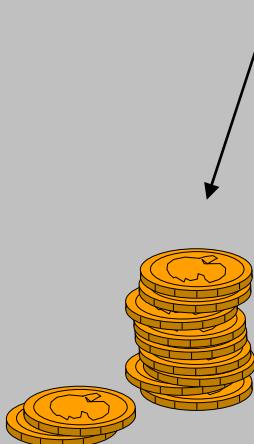


Definición de Ingeniería del Software

Técnicas, Metodologías y Herramientas

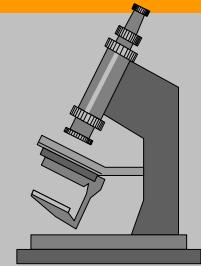


que ayudan a la producción de un software de alta calidad, con un determinado presupuesto y antes de una determinada fecha



1992						
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Ingeniería del Software vs Informática



Computer Scientist

- ❖ Proves theorems about algorithms, designs languages, defines knowledge representation schemes
 - ❖ Has infinite time...

□ Engineer

- ❖ Develops a solution for an application-specific problem for a client
 - ❖ Uses computers & languages, tools, techniques and methods

Software Engineer

- ❖ Works in multiple application domains
 - ❖ Has only 3 months...
 - ...while changes occurs in requirements and available technology



Ingeniería del Software - Terminología

□ Técnica (Método):

Procedimiento formal para obtener resultados utilizando alguna notación bien especificada

□ Metodología:

Colección de métodos aplicados a lo largo del ciclo de vida del software y unificados mediante alguna aproximación filosófica genérica

Herramienta

Instrumento, o sistema automatizado, utilizado para poner en práctica un método

Métodos o Técnicas

Modelado del comportamiento haciendo uso de los diagramas de estado

Entrevistas

Modelado de datos mediante el uso del modelo Entidad-Relación (o Entidad Asociación)

Técnicas matriciales

Modelado de sistemas mediante los diagramas de clases propuestos por BOOCH

delado de sistemas liante los diagramas clases propuestos por BOOCH



Metodologías (I)

MERISE

Metodología pública francesa

SSADM

Metodología pública británica

MÉTRICA 2

Metodología pública española

METHOD 1

Metodología de
Andersen Consulting

SUMMIT-D

Metodología de
Coopers & Lybrand



Metodologías (II)

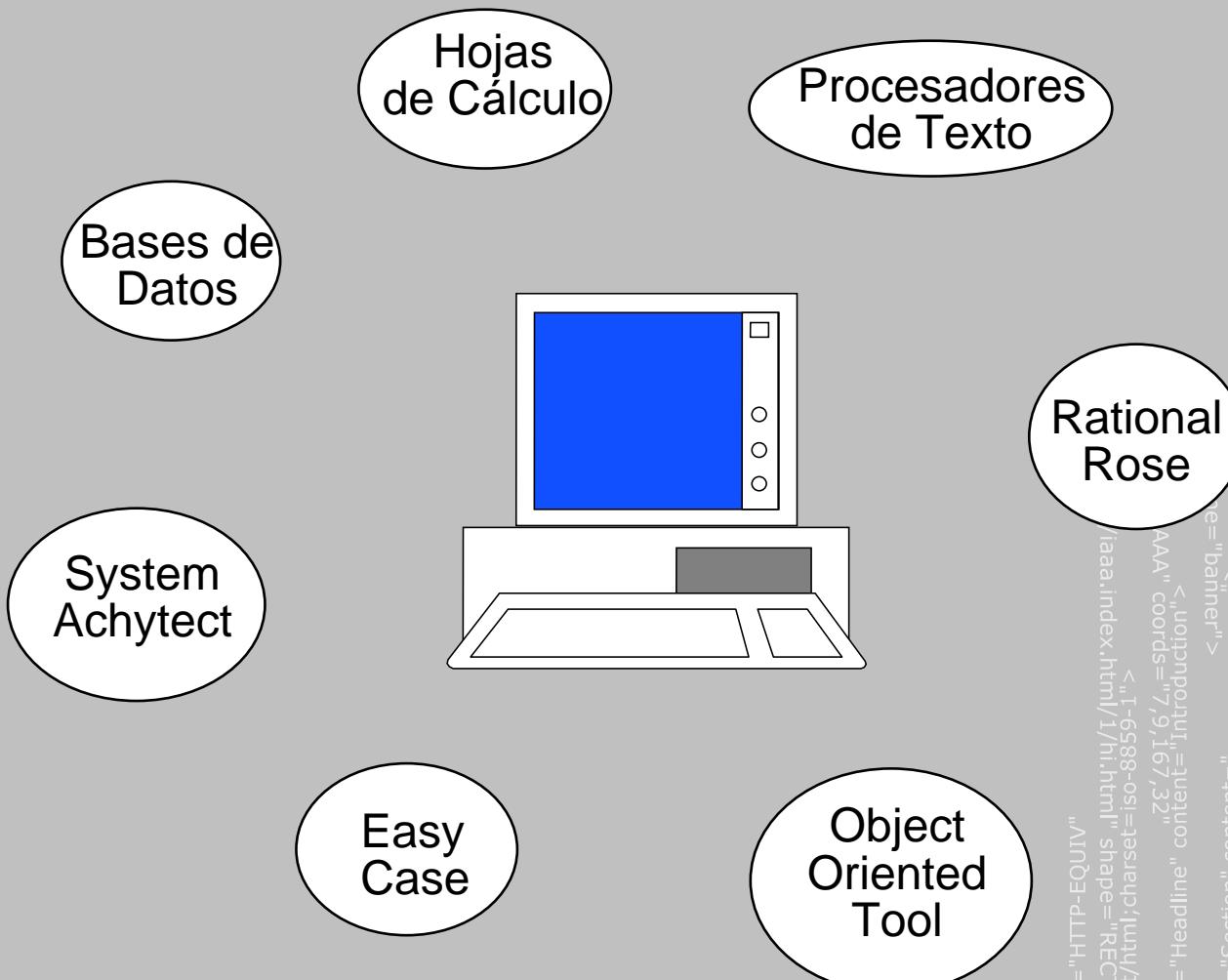
□ Metodologías estructuradas

- ❖ Basadas en técnicas estructuradas
 - ❖ Algunos ejemplos:
 - Gane-Sarson
 - Yourdon/DeMarco
 - Métrica 2

Metodologías Orientadas a Objeto

- ❖ Basadas en técnicas orientadas a objeto
 - ❖ Algunos ejemplos:
 - OMT
 - Booch
 - Métrica 3

Herramientas



Mitos del SW. El gestor

- Ya hay estándares y procedimientos. ¿No tiene ya bastante el equipo?
- Tenemos las herramientas de desarrollo más avanzadas porque compramos los ordenadores más modernos.
- Si fallamos en la planificación, añadimos más gente y sanseacabó.



Mitos del SW. El cliente

- ❑ Una declaración inicial de objetivos es suficiente para empezar a escribir programas. Ya detallaremos más adelante.
 - ❑ Los requisitos del proyecto cambian continuamente, pero los cambios pueden acomodarse fácilmente porque el software es flexible.
 - ❑ La labor del desarrollador consiste ÚNICAMENTE en escribir líneas de código fuente.

Mitos del SW. El desarrollador

- ❑ Una vez que escribimos el programa y hacemos que funcione, se acabó el trabajo.
 - ❑ No podemos comprobar la calidad hasta que se ejecute el programa.
 - ❑ Lo que se entrega al terminar el proyecto es el programa funcionando

La industria del Software

El software es una industria,...



... no es un arte.

Importancia de la Ingeniería del Software

- La economía de todos los países desarrollados depende del software, representando cada vez un mayor porcentaje de su PIB.
 - Cada vez son más los sistemas controlados por software.
 - Los costes del software llegan, en ocasiones, a dominar los costes de todo el sistema.

Tipos de productos software

- Genéricos. Sistemas autónomos desarrollados por una organización y puestos en el mercado para cualquier cliente.
 - Software a medida (especializado). Sistemas pensados para un cliente específico y desarrollados específicamente por alguna organización contratada a tal efecto.
 - Sistemas genéricos adaptados a clientes específicos.
 - La mayor parte del volumen de facturación es de software genérico (Microsoft), sin embargo el mayor número de ingenieros se encuentran trabajando en software a medida o adaptando software genérico (SAP R3, Meta4).

Sistemas de Información (I)

Sistema

- ❖ Conjunto de componentes que interaccionan entre sí para lograr un objetivo común. Ejemplos: el sistema nervioso, el sistema económico.
 - ❖ Un sistema puede estar compuesto por varios niveles de sistemas, también denominados subsistemas.

□ Sistemas de Información

- ❖ Es un sistema cuyo objetivo principal es el procesamiento de información.
 - ❖ La finalidad de los sistemas de información es procesar entradas, mantener archivos de datos relacionados con la organización en la que están enmarcados, y producir información.
 - ❖ Los principales componentes de un sistema de información son las personas, los equipos y los procedimientos.

Sistemas de Información (II)

Análisis

- ❖ Descomposición de un sistema para conocer su naturaleza, funciones, relaciones y requerimientos.

□ Análisis de Sistemas

- ❖ Proceso de clasificación e interpretación de hechos, diagnóstico de problemas y empleo de la información para recomendar mejoras al sistema.

Diseño de Sistemas

- ❖ Proceso de planificar, reemplazar o complementar un sistema organizacional existente.
 - ❖ El Análisis especifica lo **QUÉ** el sistema debe hacer.
 - ❖ El Diseño establece **CÓMO** alcanzar el objetivo.



Cambios generados por los SI (I)

- Trabajo inteligente
 - ❖ El ordenador se ocupa de las labores repetitivas y rutinarias.
- Fusión global de empresas
 - ❖ La transformación de información, no de bienes, permite a la empresa diversificar su área de negocio.
- Idea e Información
 - ❖ Antes la base del negocio era el uso del dinero y los recursos tangibles.
 - ❖ Ahora son las ideas y la información.
- Usuarios: trabajadores de la información
 - ❖ La industria de la manufactura representa el 28% de los sueldos, el resto se alojan en empresas de servicios y manejo de información.



Cambios generados por los SI (II)

- Soporte tecnológico al crecimiento de la empresa
 - ❖ Hace 20 años una empresa de distribución generaba 500 facturas año utilizando máquinas de escribir. Actualmente genera 50.000 con ayuda de ordenadores.
 - Dependencia legal de los SI
 - ❖ Una caída en la red de comunicaciones forzó la publicación en BOE de una extensión de plazo para presentación de declaración de impuestos

Los participantes en el juego de los SI (I)

□ El usuario

- ❖ heterogéneos
- ❖ visión muy sesgada de las problemáticas

□ El administrador

- ❖ de usuarios - supervisor de los usuarios
- ❖ de informática - gestor de proyectos
- ❖ general - de la organización, financieros.
Planificación estratégica.

□ El cliente

□ El personal de operaciones

- ❖ instalaciones
- ❖ copias de seguridad
- ❖ mantenimiento de infraestructura



Los participantes en el juego de los SI (II)

- ❑ El auditor
 - ❖ control de calidad y consultor de proceso
 - ❑ El analista de sistemas
 - ❖ es la persona cuyo trabajo consiste en centralizar el desarrollo del sistema en su conjunto
 - ❑ El diseñador
 - ❖ es la persona que partiendo del análisis de un problema planteamiento de una solución al mismo libre de trab tecnológicas, construye el diseño de dicha solución enmarcandolo dentro de un contexto tecnológico par
 - ❑ El programador
 - ❖ es la persona encargada de codificar el diseño de la solución
 - ❑ Analogía con el mundo de la construcción
 - ❖ Analista - Arquitecto, Diseñador - Aparejador, Programador - Albañil

El analista de sistemas

Labores del analista de sistemas

- ❖ es arqueólogo y escribano
 - ❖ es innovador
 - ❖ es mediador

□ Separación de papeles

- ❖ Analista, Analista/Diseñador, Analista/Diseñador/Programador
 - ❖ Evolución: Programador ... Analista

Más terminología (I)

□ Especificación de requerimientos

- ❖ Es el proceso de recopilación de las características con las que deberá contar el software a desarrollar.
 - ❖ Términos relacionados:
 - Especificación del sistema.
 - Requisitos de usuario.

□ Codificación / Implementación

- ❖ Traducción de las especificaciones de diseño a un lenguaje de programación determinado.

□ Prueba / Test

- ## ❖ Verificación del funcionamiento requerido del software

Riesgo

- ❖ Problema que puede surgir en un proyecto afectando a su correcto desarrollo.
 - ❖ Los riesgos deberían estar identificados y se debería determinar estrategias que permitan mitigarlos.

Ejemplos de riesgos

- ❑ Tecnología desconocida (ejemplos: lenguaje de programación no conocido, nuevo hardware, etc.).
 - ❑ Poca experiencia del equipo de desarrollo.
 - ❑ Problema mal descrito y/o modificaciones en la descripción del problema.
 - ❑ Retraso en algún determinado proveedor.
 - ❑ Problemas en módulos que sean cuellos de botella en el desarrollo.

Más terminología (II)

❑ Integración

- ❖ Unión de los diferentes componentes del software

❑ Mantenimiento

- ❖ Modificaciones que se realizan sobre el software una vez que este se ha entregado al cliente. El motivo de estas modificaciones puede ser la corrección de defectos detectados en su funcionamiento, o la ampliación o modificación de la funcionalidad a petición del cliente.

❑ Contrato o acuerdo del proyecto

- ❖ Acuerdo entre desarrollador y cliente en el que se especifica **QUÉ** es lo que se va a entregar como resultado del proyecto, **CUÁNTO** dinero se va a pagar por ello y **CUÁNDO** se realizará esta entrega. Debe estar firmado por las dos partes.

❑ Transferencia al cliente

- ❖ Proceso por el cual se entrega al cliente el resultado del proyecto y éste comprueba que el resultado es el que se había acordado.



Más terminología (III)

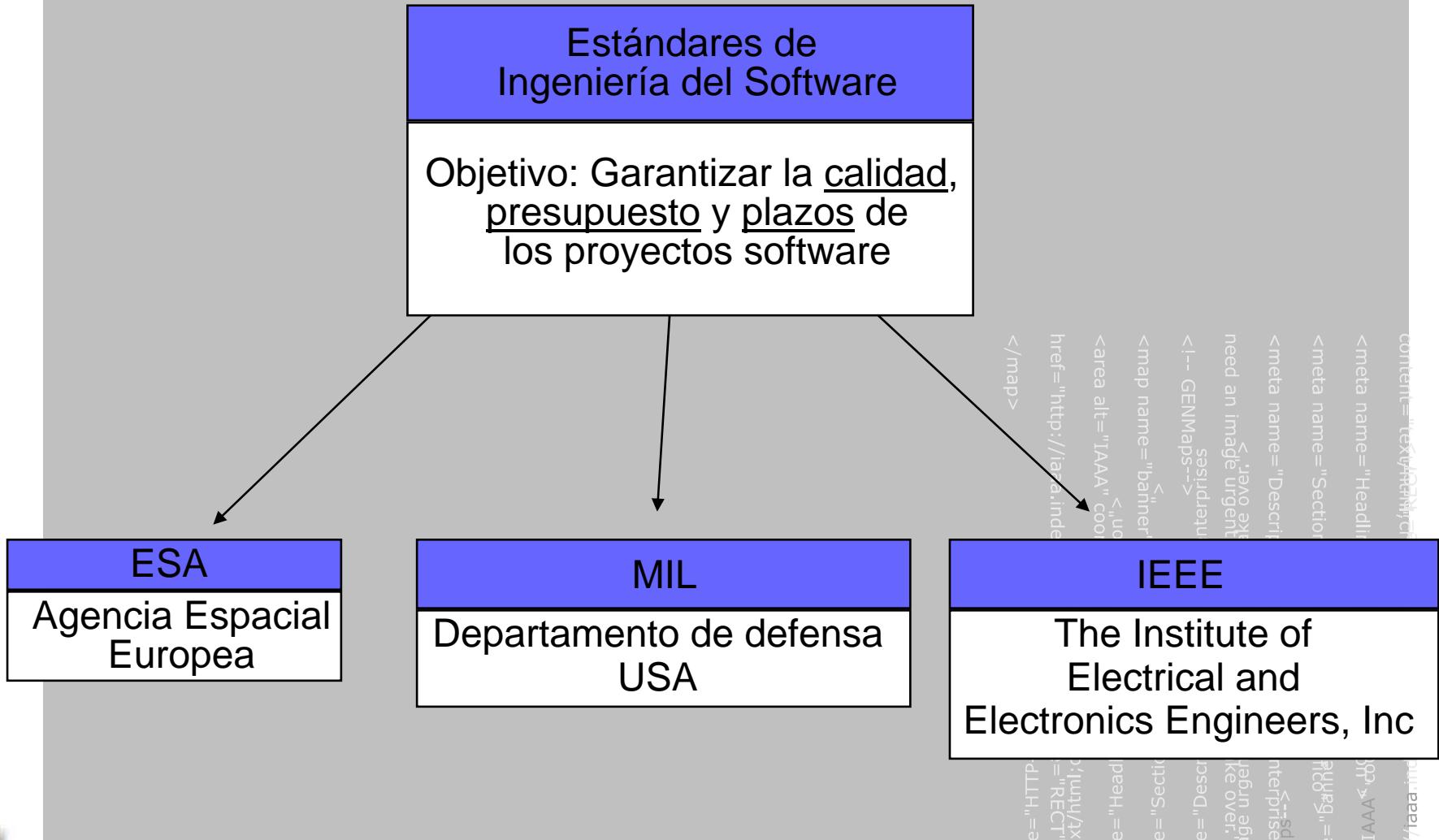
□ Acoplamiento

- ❖ Medio de evaluar la relación entre elementos de un sistema.
 - ❖ Es la medida del grado de interdependencia entre los elementos de un sistema.

□ Cohesión

- ❖ Indica el grado de conexión funcional entre elementos.
 - ❖ Es una medida de la fuerza de la relación funcional entre elementos de un sistema.

Estandarización del desarrollo del SW



IEEE - Estándares para Ingeniería del SW

- ❑ IEEE 1058 - Standard for Software Project Management Plans
 - ❑ IEEE 828 - Standard for Software Configuration Management Plans
 - ❑ Otros estándares de interés:
 - ❖ IEEE 610 - Glossary of Software Engineering Terminology
 - ❖ IEEE 830 - Guide for Software Requirements Specification
 - ❖ IEEE 1002 - Standard Taxonomy for Software Engineering Standards
 - ❖ IEEE 1074 - Standard for Developing Software Life Cycle Processes
 - ❖ IEEE 1028- Standard for Software Reviews and Audits
 - ❑ Los estándares son revisados y actualizados cada 5 años
 - ❑ Reconocidos y aceptados por gran número de organismos y empresas

MIL DOD - STD - 2167A

- Establece los requerimientos del Departamento de Defensa USA para el ciclo de vida del software
- Actividades de desarrollo del software
 - ❖ Análisis y Diseño de requerimientos del sistema
 - ❖ Análisis de los requerimientos software
 - ❖ Diseño preliminar
 - ❖ Diseño detallado
 - ❖ Codificación y testeo de unidades
 - ❖ Integración y tests de CSC (Computer Software Component, se compone de CSC's y unidades)
 - ❖ Test de CSCI (Computer Software Configuration Item)
 - ❖ Integración del sistema y tests



Estándares de la ESA

Conjunto de estándares definidos por la Agencia Espacial Europea para la especificación, desarrollo y mantenimiento del software

Software Engineering Standards

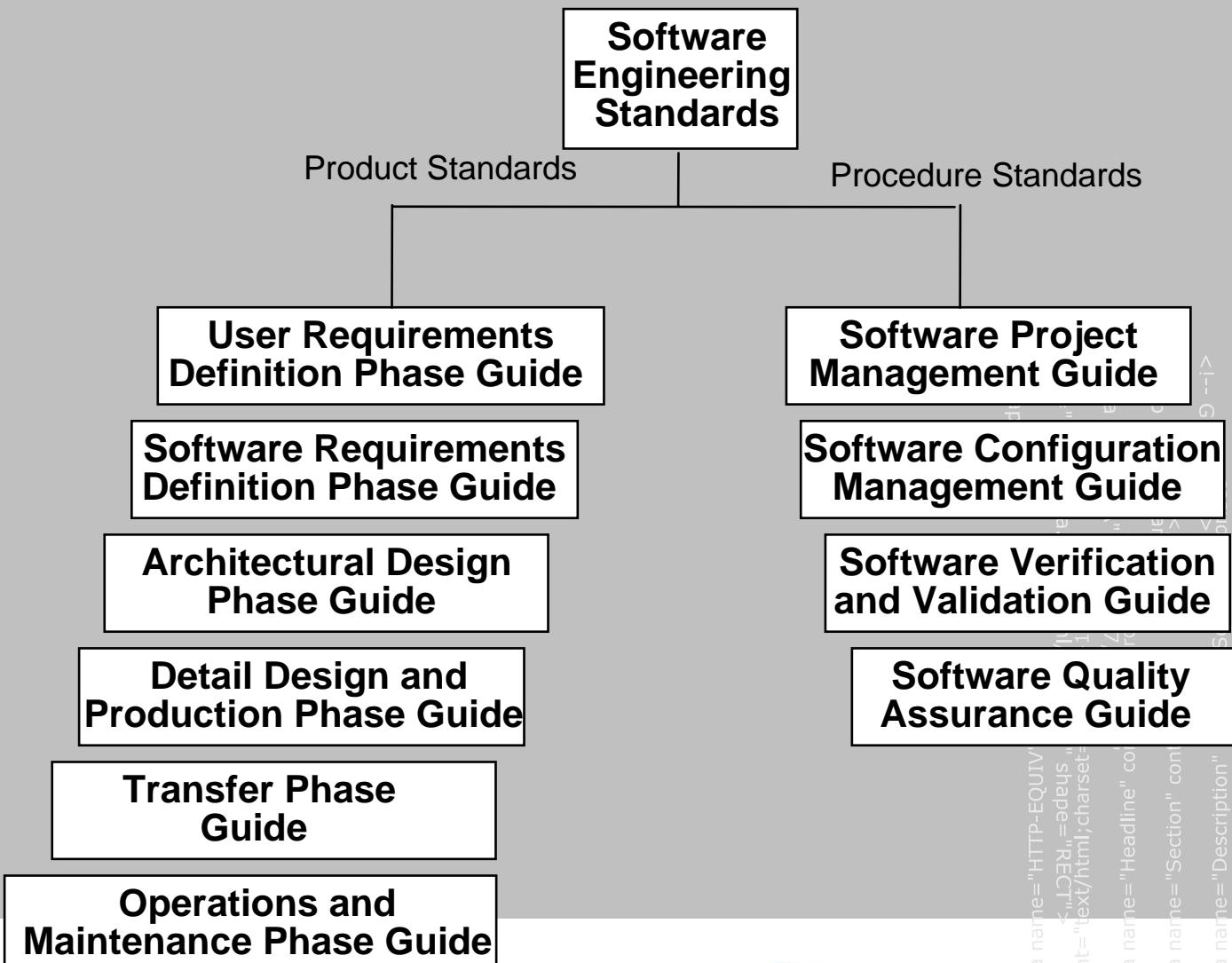
Definen las obligaciones y recomendaciones prácticas para la especificación, desarrollo y mantenimiento del software

Software Engineering Guides

Discuten su aplicación práctica en mayor nivel de detalle, y describen los métodos y herramientas para llevarlos a cabo. También contienen información para la redacción de los documentos requeridos por los estándares



Estructura de los estándares y guías ESA



Otros estándares

- ISO (International Organization for Standardization)
 - ❖ OSI (Open System Interconnection)
 - ❖ EDI (Electronic Data Interchange)
 - ❖ ISO 9001 (Estandar para el desarrollo de software)
- CCITT (International Consultative Committee on Telephony and Telegraphy)
 - ❖ Compuesto por compañías telefónicas de todo el mundo
 - ❖ X.25 estd. de comunicaciones para servicios del nivel de red
- ANSI (American National Standard Institute)
 - ❖ Representante oficial de USA en ISO y CCITT
 - ❖ ASCII
- OMG (Object Management Group): Comité que busca la armonización de los fabricantes en dos líneas, la terminología en la orientación a objeto, y la estandarización de interfaces
 - ❖ CORBA (Common Object Request Broker Architecture)



Introducción Métrica 2.1 (I)

□ Metodología de Planificación y Desarrollo de Sistemas Informáticos

Promovida por el Consejo Superior de Informática (órgano encargado de elaborar y desarrollar la política informática del Gobierno)

□ Objetivos:

Crear un entorno que permita al equipo de trabajo construir

Sistemas, que:

- Den **SOLUCIONES** a los objetivos considerados prioritarios en la **Administración**.
- Se desarrolle **CUANDO EL USUARIO LOS NECESITE** y
DE ACUERDO A LOS PRESUPUESTOS Y DURACIÓN ESTIMADOS
- Se **MANTENGAN FÁCILMENTE** para soportar los cambios futuros en la organización



Introducción Métrica 2.1 (II)

- Métrica versión 2 ofrece un marco de trabajo en el que se define:
 - ❖ Una estructura de proyecto que sirva de guía al equipo de trabajo e involucre a los usuarios en su desarrollo y en sus puntos decisivos.
 - ❖ Un conjunto de productos finales a desarrollar.
 - ❖ Un conjunto de técnicas para obtener los productos finales.
 - ❖ Las diferentes responsabilidades y funciones de los miembros del equipo de proyecto y de los usuarios.

□ Métrica versión 3

- ❖ Basada en técnicas Orientadas a Objetos
- ❖ Ya está operativa



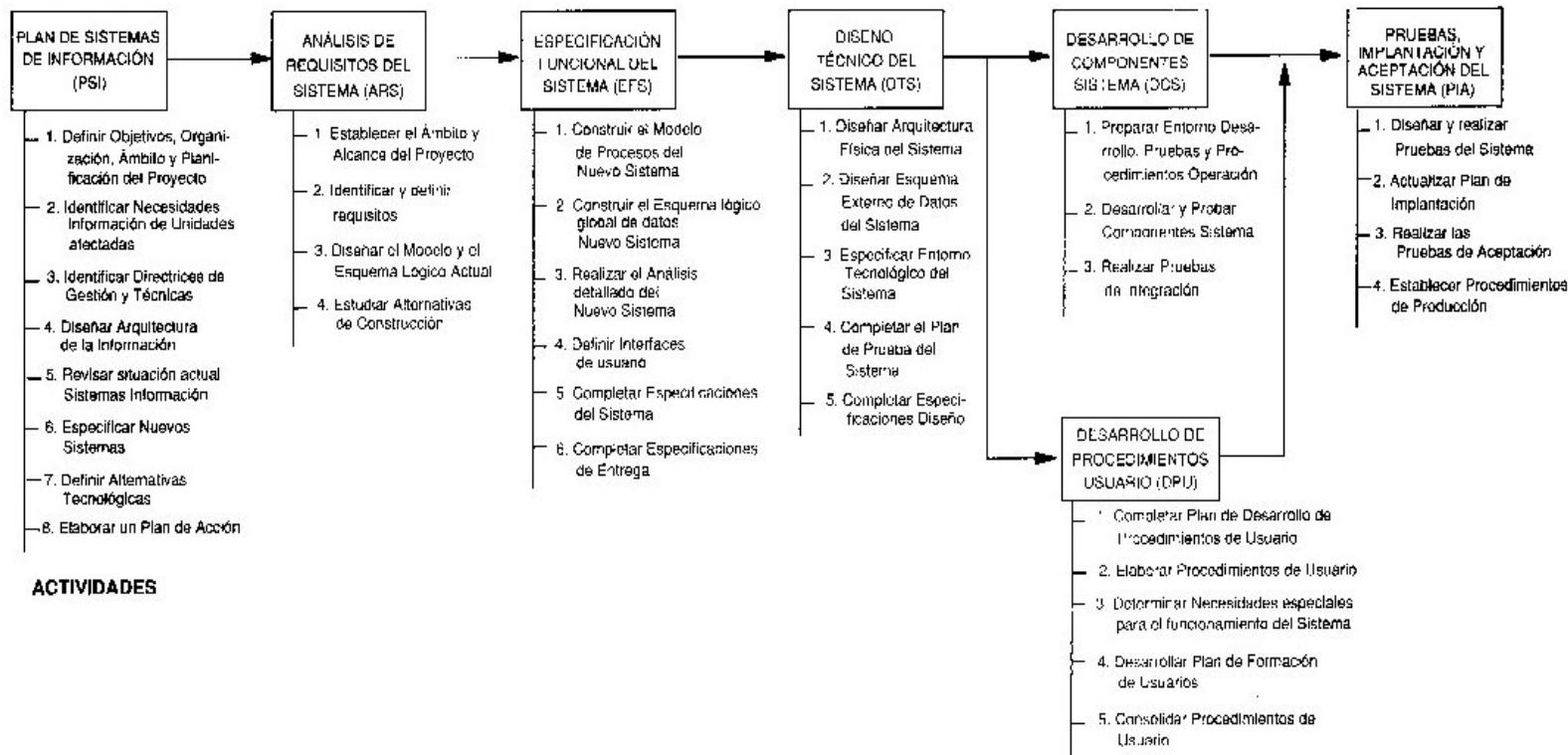
Visión general de Métrica 2.1

VISIÓN GENERAL DE LA METODOLOGÍA MÉTRICA 2.1

FASES

PLAN DE SISTEMAS DE INFORMACIÓN	ANÁLISIS DE SISTEMAS	DISEÑO DE SISTEMAS	CONSTRUCCIÓN DE SISTEMAS	IMPLANTACIÓN DE SISTEMAS
---------------------------------	----------------------	--------------------	--------------------------	--------------------------

MÓDULOS



ACTIVIDADES



INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE

El Ciclo de Vida

F.Javier Zarazaga Soria
Javier Nogueras Iso
Ingeniería del Software I



Ciclo de vida del software. Índice

- Introducción
- Fases, Tareas y Actividades
- Ciclo de Vida de un producto Software

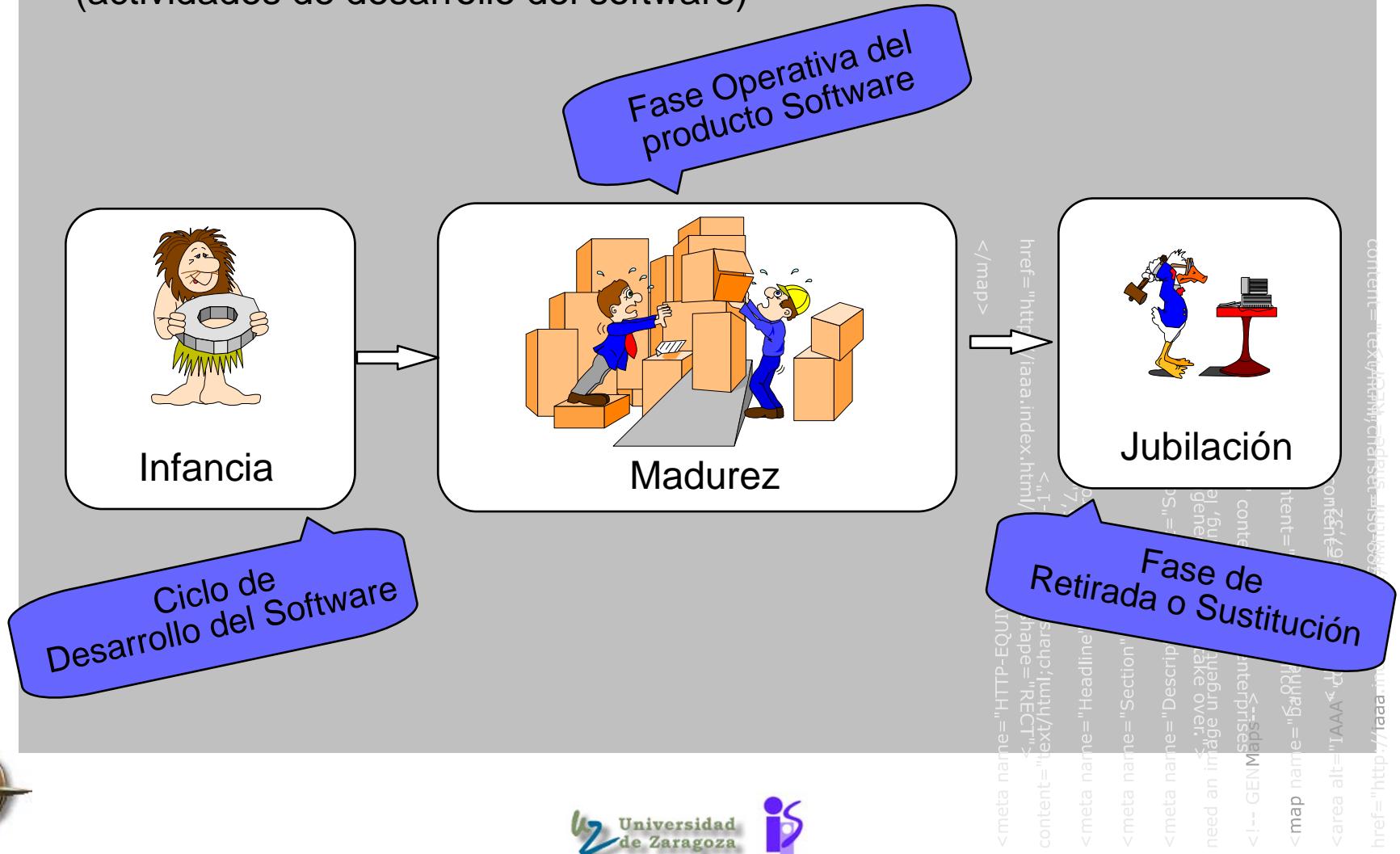
- ❖ Ciclo de Vida en Cascada Clásico
- ❖ Ciclo de Vida en V
- ❖ Ciclo de Vida en Cascada Mejorado
- ❖ Ciclo de Vida con Prototipado
- ❖ Modelo Incremental
- ❖ Ciclo de Vida en Espiral

- Estándares para el desarrollo de Ciclos de Vida



Introducción al ciclo de vida

El desarrollo del software atraviesa una sucesión de estados (actividades de desarrollo del software)



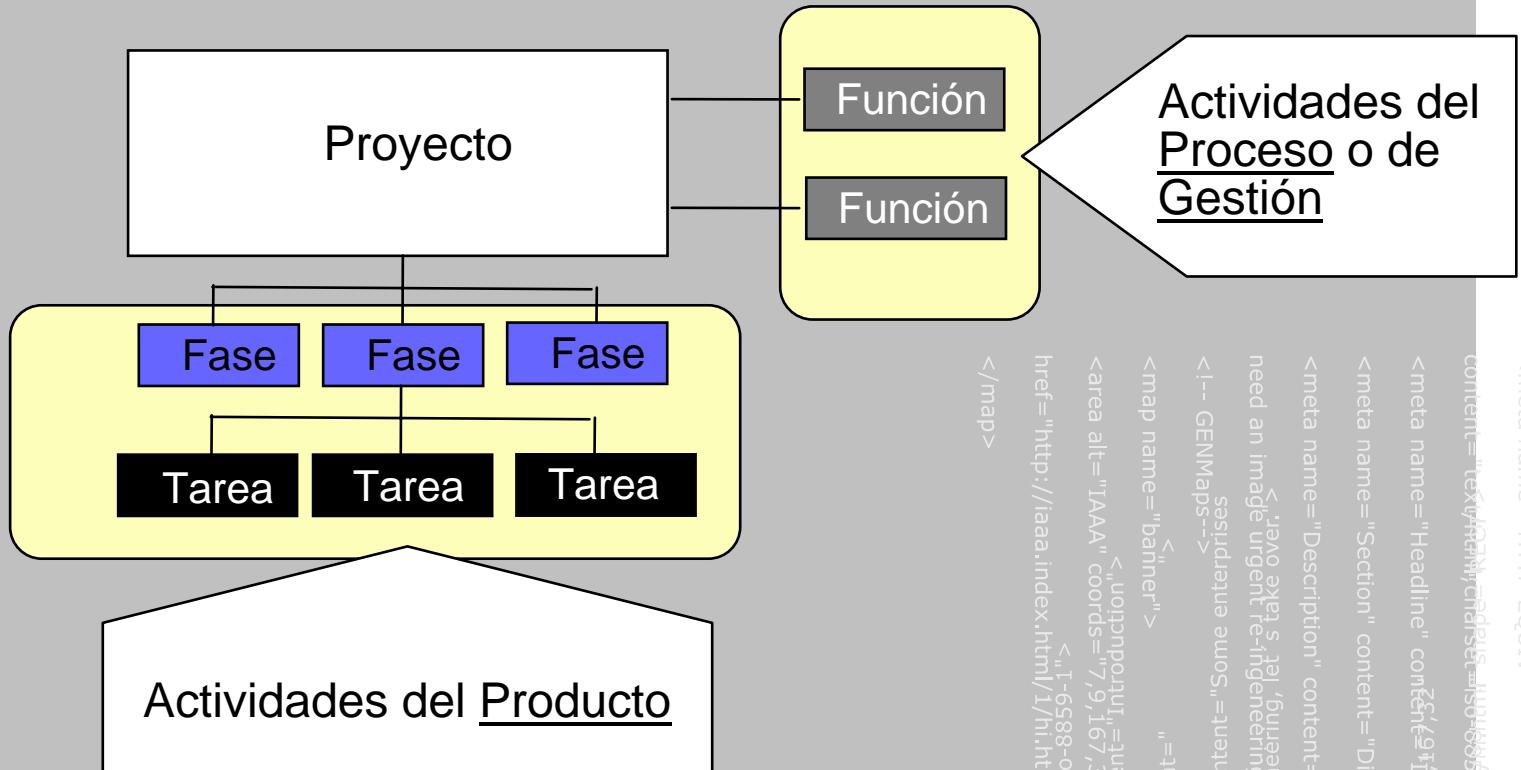
Ciclo de vida. Terminología I

Ciclo de Vida de un Producto Software

Sucesión de pasos a través de los cuales el producto software va progresando. Estos pasos abarcan desde el planteamiento del problema a resolver mediante el producto software, hasta la retirada de dicho producto una vez que ha finalizado su vida operativa



Ciclo de vida. Terminología II



Ciclo de vida. Terminología III

Función

Actividad o conjunto de actividades que abarcan la duración del proyecto

Ejemplos:

Gestión de Configuraciones
Gestión del Proyecto

En IEEE-1074 -> Procesos Integrales

Fase

Mayor unidad de trabajo

Finalizan con los hitos más importantes del proyecto

Ejemplos:
usuario

Definición de los Requerimientos de
Diseño Arquitectural

Tarea

Menor unidad de trabajo sujeta a gestión

Suficientemente pequeña para una adecuada planificación y seguimiento

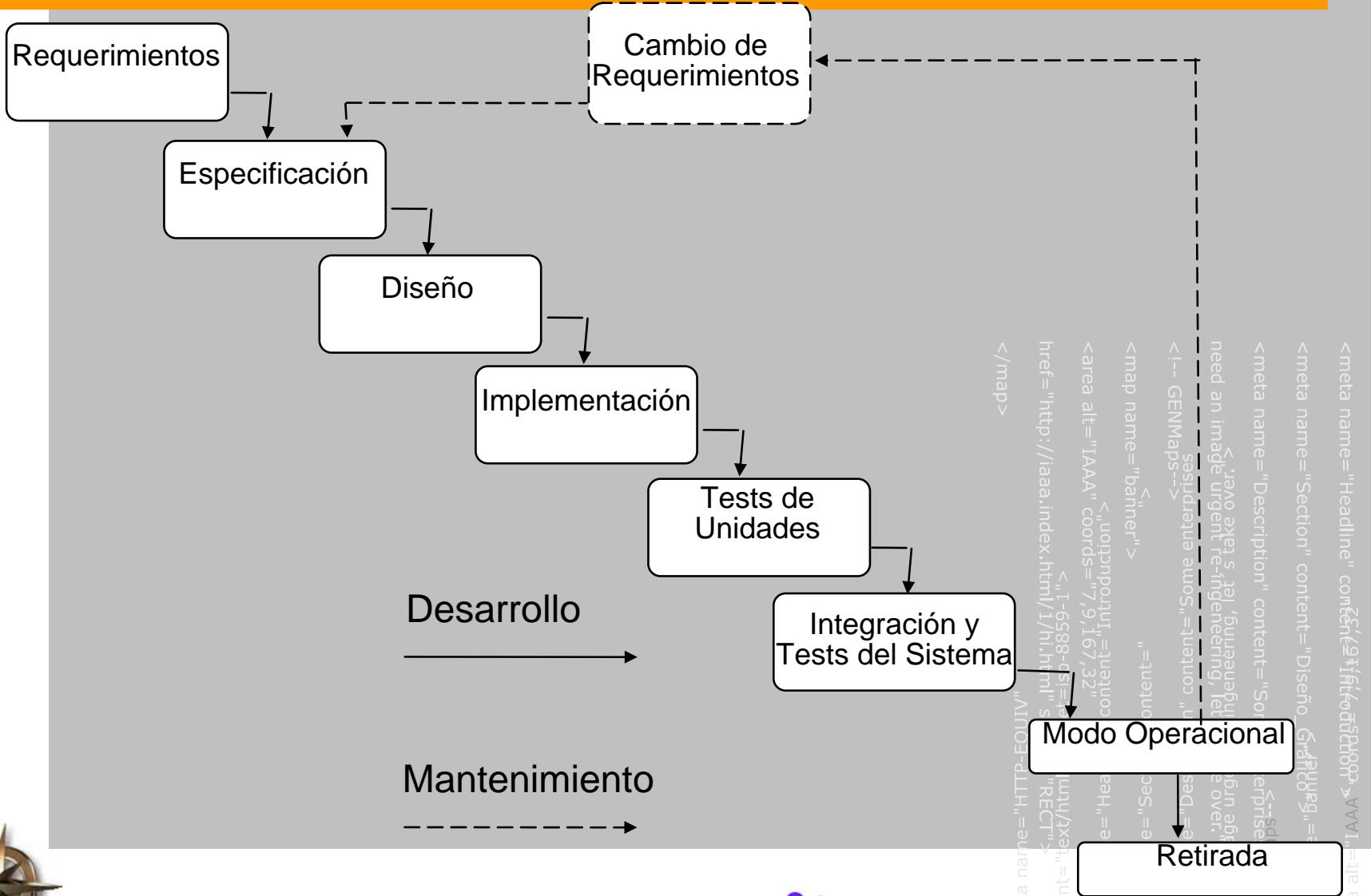
Suficientemente grande para permitir micro gestión

Ejemplos:

Escribir el manual de usuario
Test de subsistemas



Ciclo de Vida en Cascada Clásico (I)



Ciclo de Vida en Cascada Clásico (II)

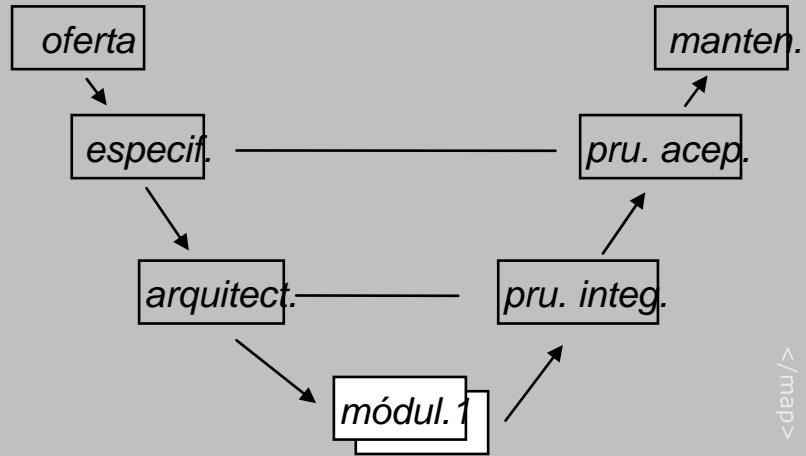
Ventajas:

- ❖ Fuerza a una aproximación disciplinada fruto de su surgimiento a partir del ciclo convencional de una ingeniería
 - ❖ Sencillo de implantar y gestionar (es el preferido por los gestores de proyectos)

□ Problemática:

- ❖ Es linear y el desarrollo del software no lo es
 - ❖ El cliente debe tener paciencia (no ve resultados hasta final)
 - ❖ Diferencias en la interpretación de los requerimientos entre cliente y desarrollador
 - ❖ Los proyectos raramente siguen el flujo secuencial que propone el modelo (imposibilidad de actividades en paralelo)
 - ❖ Es difícil para el cliente establecer al principio todos los requisitos de forma explícita

Ciclo de Vida en V



□ Ventajas:

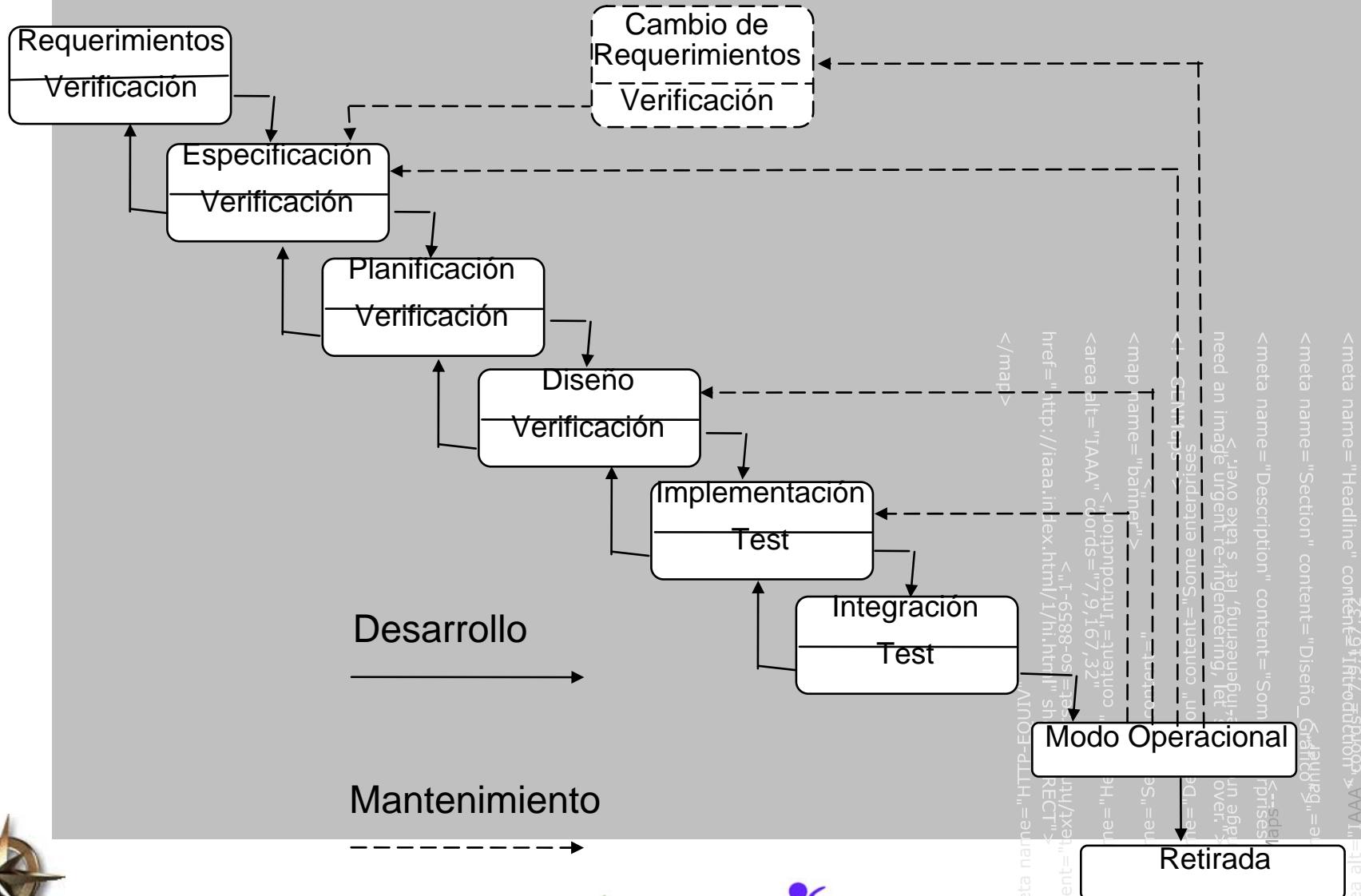
- ❖ Supone una mejora respecto al modelo en cascada, eliminando parte de la secuencialidad

□ Problemas:

- ❖ Prácticamente los mismos que los del modelo en cascada



Ciclo de Vida en Cascada Mejorado (I)



Ciclo de Vida en Cascada Mejorado (II)

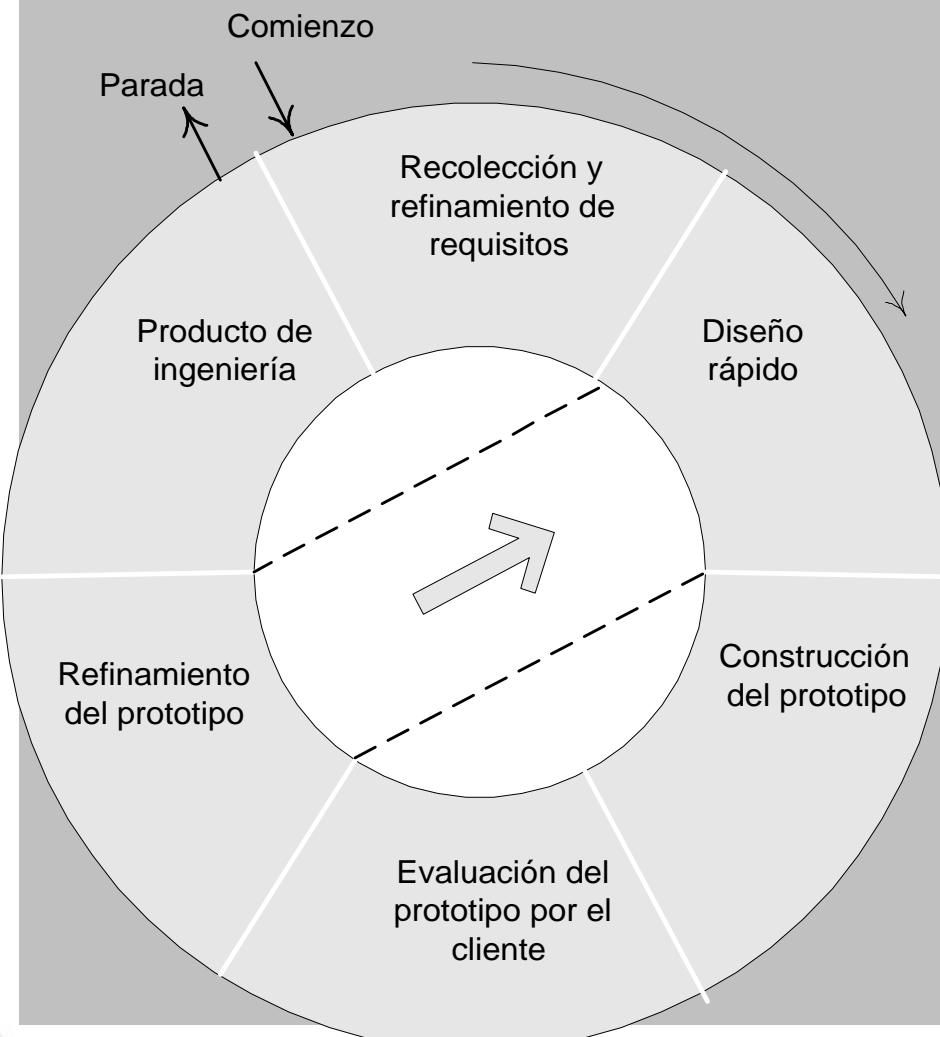
□ Mejoras:

- ❖ Estipula la documentación que debe acompañar cada fase y los requerimientos que todos los productos de cada una de las fases deben cumplir
 - ❖ En los hitos que finalizan cada fase se realiza la aprobación de todos los productos de esa fase, incluyendo los documentos
 - ❖ El testing es inherente a cada una de las fases del modelo y no se trata como una fase más del mismo a ser realizada una vez que se ha construido el producto
 - ❖ Todos los cambios que se produzcan en las operaciones de mantenimiento deben ser reflejados en los documentos de las fases a las que afecten

□ Problemática:

- ❖ Modelo guiado por la documentación (los documentos son los hilos conductores del modelo). Esto puede producir diferencias en la interpretación de los mismos entre cliente y desarrollador

Ciclo de Vida con Prototipado (I)



Objetivo:

Obtener rápidamente un prototipo de la aplicación que permitan al cliente interactuar con ella con el fin de detectar deficiencias en las especificaciones
Construimos un rápido *boceto* de la aplicación

Ciclo de Vida con Prototipado (II)

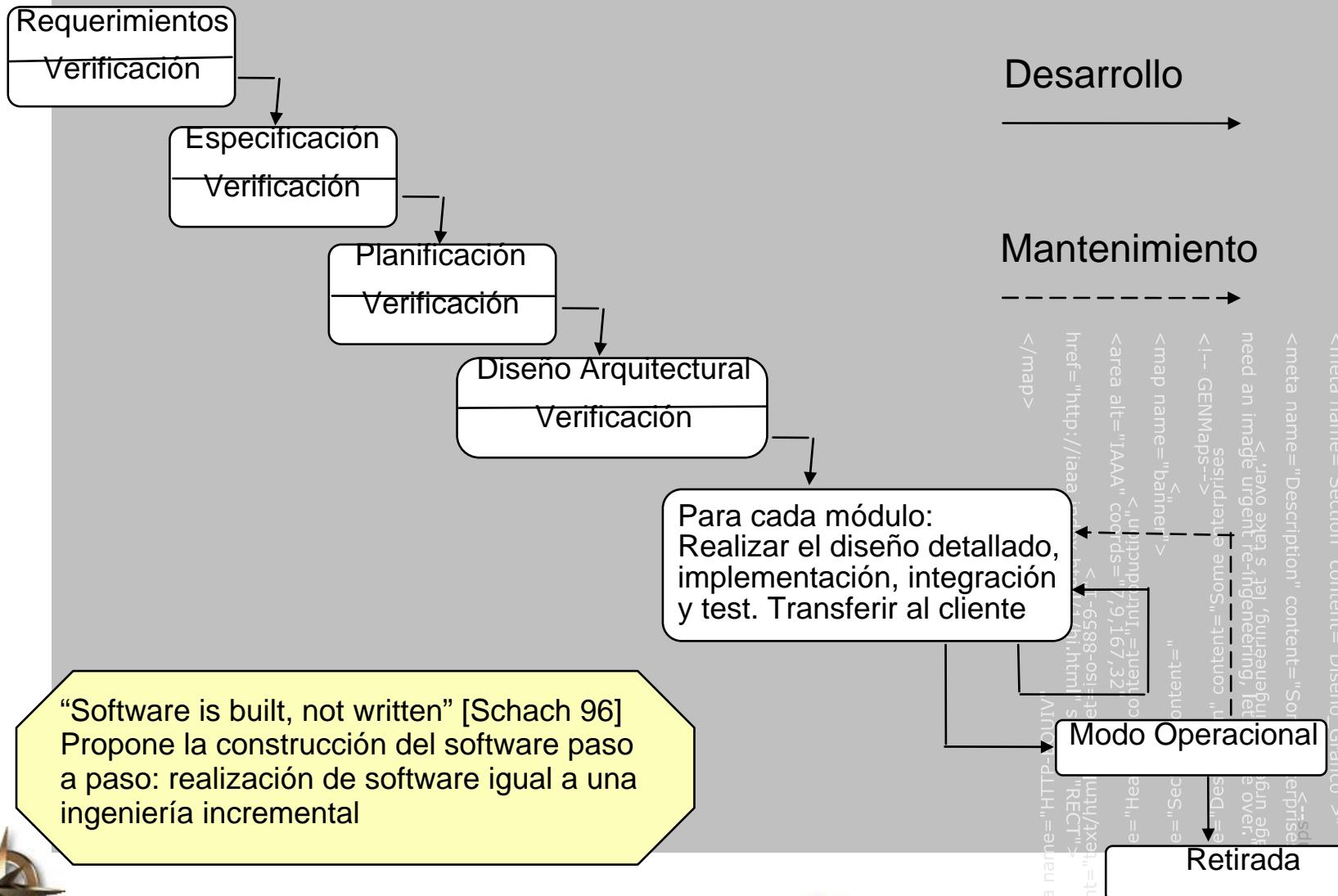
Ventajas

- ❖ Mejora la comunicación analista - cliente
 - ❖ Mejor identificación de los requerimientos del cliente
 - ❖ Satisface la curiosidad del cliente (en seguida puede ver cosas)

□ Problemas:

- ❖ Identificación del prototipo con el producto final
 - ❖ Prototipo no reutilizable
 - ❖ Posibles asunciones técnicas inapropiadas con el fin de prototipar más rápidamente

Modelo Incremental (I)



“Software is built, not written” [Schach 96]
Propone la construcción del software paso a paso: realización de software igual a una ingeniería incremental



Modelo Incremental (II)

□ Ventajas:

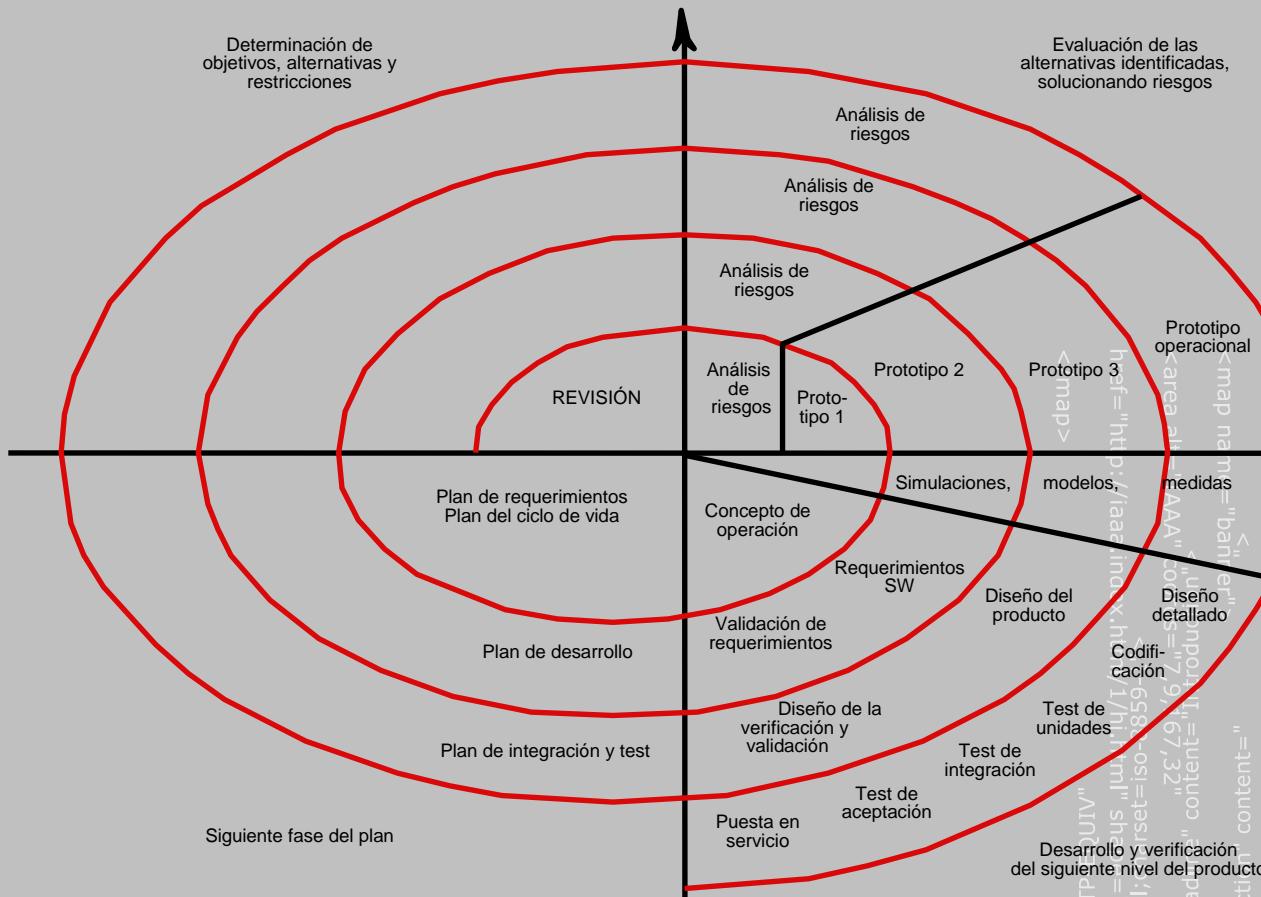
- ❖ Satisface la curiosidad del cliente
 - ❖ El cliente puede ir "viendo" la aplicación real, no un prototipo
 - ❖ Muy ligado al proceso de sustitución de un sistema por otro
 - ❖ Posibilidad de detener el proyecto sin perder todo lo realizado
 - ❖ Mayor flexibilidad ante cambios en los requerimientos durante el desarrollo

□ Problemática:

- ❖ Cada nuevo módulo se debe integrar en el sistema sin afectar a lo que ya está funcionando (datos, interfaces,...)
 - ❖ El diseño y desarrollo de los diferentes módulos debe ser coherente entre sí
 - ❖ Gran tendencia a la degeneración del control del proyecto ante la imposibilidad de verlo como un todo



El Ciclo de Vida en Espiral (I)



Propuesto por Boehm, 1988



El Ciclo de Vida en Espiral (II)

- ❑ Identificación de Riesgos
 - ❑ Asignar prioridades a los riesgos
 - ❑ Desarrollar una serie de prototipos para los riesgos identificados comenzando con el más prioritario
 - ❑ Hacer uso del modelo en cascada para cada fase de desarrollo
 - ❑ Si los riesgos han sido resueltos satisfactoriamente, evaluar los resultados de la fase y planificar la siguiente
 - ❑ Si alguno de los riesgos no puede ser resuelto, abandonar el proyecto inmediatamente



El Ciclo de Vida en Espiral (III)

□ Ventajas:

- ❖ Inclusión de análisis de riesgos a lo largo del proceso
 - ❖ Desarrollo de software = proceso evolutivo
 - ❖ Uso de prototipos, uso de simulaciones
 - ❖ El cliente ve evolucionar el proyecto
 - ❖ Tratamiento del mantenimiento al mismo nivel que el desarrollo (se trata de otra espiral más)

□ Problemas:

- ❖ Difícil de convencer al cliente de su **utilidad** (hay que realizar muchos tests y esto cuesta dinero)
 - ❖ Implementación muy compleja
 - ❖ Necesita gran habilidad en la **valoración de riesgos**
 - ❖ Método muy nuevo

IEEE - Std 1074 - 1991

- Std 1074-1991
 - ❖ Standard for Developing Software Life Cycle Processes
 - Alcance
 - ❖ Indica un conjunto de actividades que constituyen los procesos que son obligatorios para el desarrollo y mantenimiento del software, tanto por si sólo, como formando parte de un sistema
 - ❖ No incluye actividades para el desarrollo del hardware, realización de compras, ...
 - ❖ No recomienda un ciclo de vida específico
 - Aplicabilidad
 - ❖ Proyectos de desarrollo y mantenimiento de software
 - ❖ Descomponer proyectos grandes en subproyectos, aplicar el estándar a cada uno de ellos y al conjunto

Estándar ESA para el Ciclo de Vida (I)

□ Producto

PHASES \ ITEMS	UR	UR/R	SR	SR/R	AD	AD/R	DD	DD/R	TR	OM
MAJOR ACTIVITIES	<ul style="list-style-type: none"> • determination of operational environment • identification of user requirements 		<ul style="list-style-type: none"> • construction of logical model • identification of software requirements 		<ul style="list-style-type: none"> • construction of physical model • definition of major components 		<ul style="list-style-type: none"> • module design • coding • unit tests • integration tests • system tests 		<ul style="list-style-type: none"> • installation • provisional acceptance tests 	<ul style="list-style-type: none"> • final acceptance tests • operations • maintenance of code and documentation
DELIVERABLE ITEMS	User Requirements Document arrow implies under change control	URD	Software Requirements Document	SRD	Architectural Design Document	ADD	Detailed Design Document DDD Code SUM Software User Manual		Software Transfer Document STD	Project History Document PHD
REVIEWS		■ technical reviews	■ ■ walkthroughs inspections	■ technical reviews	■ ■ walkthroughs inspections	■ technical reviews	■ ■ walkthroughs inspections	■ technical reviews		
MAJOR MILESTONES		URD approved	SRD approved	ADD approved	code/DDD/SUM approved	STD delivered	PHD delivered		provisional acceptance	final acceptance

Estándar ESA para el Ciclo de Vida (II)

□ Proceso

ACTIVITY PLAN	USER REQUIREMENTS REVIEW		SOFTWARE REQUIREMENTS DEFINITION		ARCHITECTURAL DESIGN		DETAILED DESIGN AND PRODUCTION	
	Activity	Output	Activity	Output	Activity	Output	Activity	Output
SOFTWARE PROJECT MANAGEMENT	Estimate project cost Plan SR phase WBS & staffing Outline plan for whole project	SPMP/SR	Estimate project cost to 30% accuracy Plan AD phase WBS & staffing	SPMP/AD	Estimate project cost to 10% accuracy Plan DD phase WBS & Staffing	SPMP/DD	Detail DD phase WBS Plan TR phase WBS & Staffing	SPMP/DD updates SPMP/TR
SOFTWARE CONFIGURATION MANAGEMENT	Define SR phase procedures for: - documents - CASE tool products - prototype code	SCMP/SR	Define AD phase procedures for: - documents - CASE tool products - prototype code	SCMP/AD	Define DD phase procedures for: - documents - CASE tool products - deliverable code	SCMP/DD	Define operational environment procedures for: - documents - deliverable code	SCMP/TR
SOFTWARE VERIFICATION AND VALIDATION	Define SR phase review and traceability procedures Plan acceptance tests	SVVP/SR	Define AD phase review and traceability procedures Plan system tests	SVVP/AD	Define DD phase review and traceability procedures Plan integration tests	SVVP/DD	Define acceptance tests Define system tests Define integration tests Plan and define unit tests	SVVP/AT updates SVVP/ST updates SVVP/IT updates SVVP/UT
SOFTWARE QUALITY ASSURANCE	Plan SR phase monitoring activities Outline plan for whole project	SQAP/SR	Plan AD phase monitoring activities	SQAP/AD	Plan DD phase monitoring activities	SQAP/DD	Plan TR phase monitoring activities	SQAP/TR

INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE

Herramientas y Tecnología CASE

F.Javier Zarazaga Soria
Javier Nogueras Iso
Ingeniería del Software I



Herramientas CASE. Índice

- Introducción a las herramientas CASE
- Historia de las Herramientas CASE
- Tipos de herramientas CASE



Introducción a las herramientas CASE

- ❑ CASE: Computer Assisted Software Engineering
 - ❑ Herramienta CASE
 - ❖ Herramienta de software que automatiza tareas particulares del ciclo de vida del software.
 - Dibujado de diagramas.
 - Prototipado de la interfaz de usuario.
 - Utilidades para almacenar y obtener informes así como consultar toda la información de desarrollo con la que se cuenta.
 - Verificación y corrección de especificaciones
 - Generación de código.

Historia de las herramientas CASE

- A principios de los 80
 - ❖ dibujado de diagramas
 - ❖ elaboración de documentación
 - A mediados de los 80
 - ❖ comprobación automática de errores basada en especificación de reglas y lenguajes formales
 - ❖ repositorios o diccionarios para almacenar la información del sistema
 - Finales 80, principios 90
 - ❖ generación automática de código partiendo de la especificación del diseño
 - Actualmente
 - ❖ ingeniería inversa o reingeniería
 - ❖ reutilización



Tipos de herramientas CASE

Herramientas de

- ❖ Ingeniería de la Información.
 - Manejan datos de negocio, sus relaciones y la forma en que fluyen entre distintas áreas de la organización.
 - ❖ planificación y administración de proyectos.
 - estimación de esfuerzos y coste de desarrollo del software.
 - planificación de proyectos.
 - ❖ análisis de riesgos.
 - ❖ control de calidad.
 - ❖ análisis y diseño.
 - ❖ prototipado y simulación.
 - ❖ diseño y desarrollo de interfaces de usuario.
 - ❖ programación.
 - ❖ ingeniería inversa.

□ Entornos integrados.



```
<meta name="HTTP-EQUIV"  
content="text/html; charset=iso-8859-1">  
<meta name="Headline" content="Introduction">  
<meta name="Section" content="Diseño_Gráfico">  
<meta name="Description" content="Some enterprises  
need an image urgent re-ingeneering, let's take over.">  
<!-- GENMaps-->  
<map name="banner">  
<area alt="AAA" coords="7,9,167,32"  
href="http://iaa.inkx.html/1/hi.html" shape="RECT">  
</map>
```