

Fundamentos de Ingeniería del Software



Capítulo 1. Introducción

Capítulo 1. Introducción

Estructura



1. Motivación.
2. El software.
3. Factores de calidad del software.
4. Problemas en el desarrollo de software.
5. La ingeniería del software.
6. Visión general del proceso de ingeniería del software.
7. Responsabilidad ética y profesional en ingeniería del software.

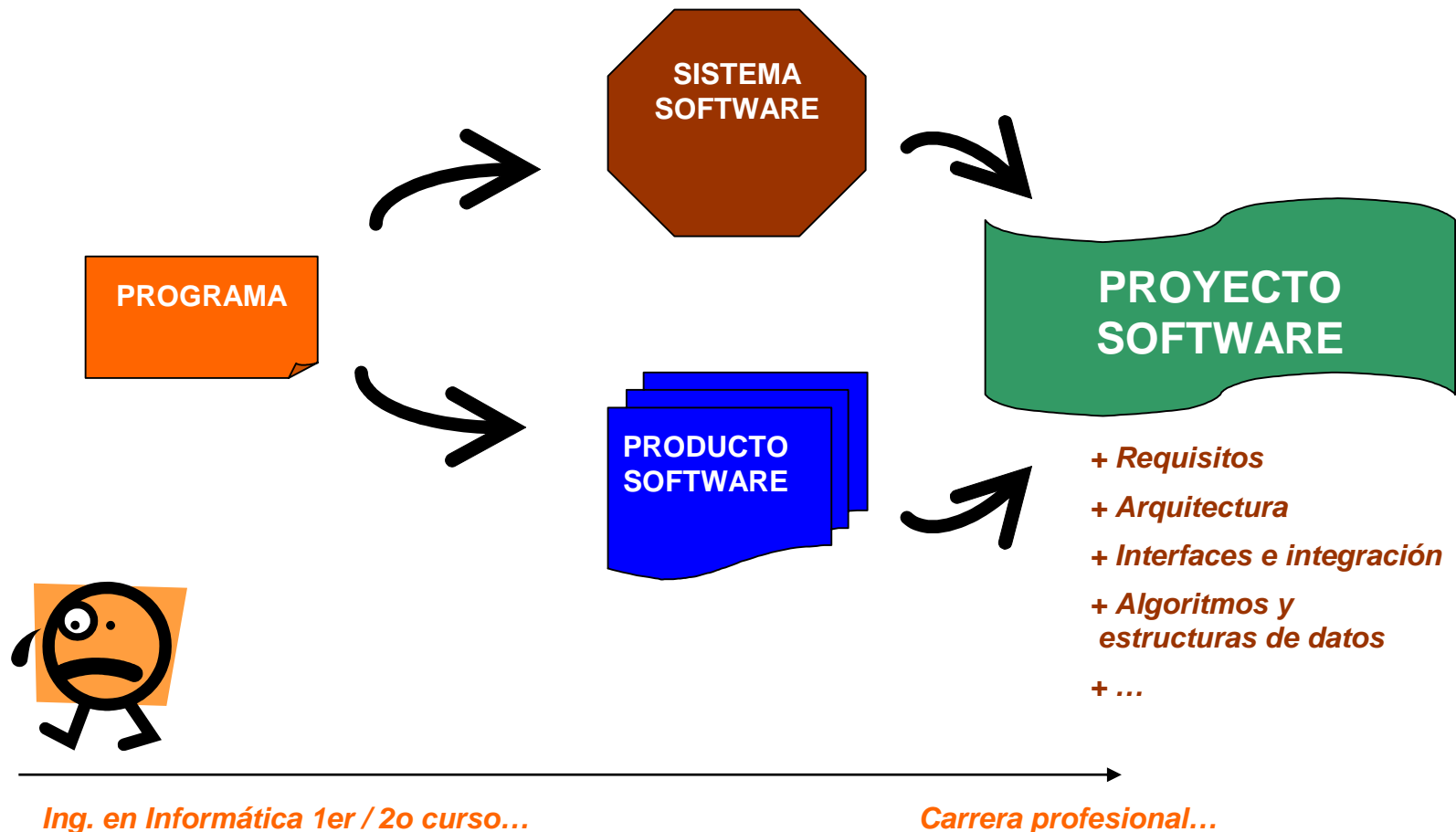
Capítulo 1. Introducción

Bibliografía



- (Pressman 2006), (Pressman 2001) o (Pressman 1998)
 - Cap.1 y Cap. 2 (aptdo. 2.1)

1. Motivación. Hacia un proyecto software...



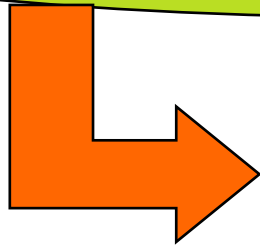
Motivación. Un par de preguntas...



- Con las herramientas actuales,
¿la programación en sí es un reto?
- Como futuro ingeniero/a de software,
¿qué crees que te debería distinguir en el
mercado laboral?

Motivación. Necesaria aproximación disciplinada...

Información = Ppal. activo de las empresas



desarrollo de SI \Leftarrow fuertes presiones
calidad,
productividad

~~¿Desarrollo artesanal suficiente?~~

\Rightarrow Disciplina de ingeniería

(sw. fiable, económico y eficiente)

Gestión de calidad

*Métodos (técnicas,
procesos, herramientas)*

Gestión de proyectos

Motivación. Un símil con la industria de la construcción...

INDUSTRIA de la **CONSTRUCCION**

PEQUEÑOS PROYECTOS

(pintar dormitorio de los niños)

1 día x 1 hombre (autodidacta)

GRANDES PROYECTOS

(El Nido, Pekín)

Varios años x

*Contratistas,
constructores,
arquitectos,
delineantes,
obreros,
albañiles,
auditores,
aficionados al arte*

...

INDUSTRIA del **SOFTWARE**

(pequeño programa)

1 día x 1 hombre (autodidacta)

(Gran proyecto sw)

Varios años x

*Contratistas,
factoría software,
ingenieros software,
analistas,
operadores,
programadores,
auditores,
usuarios*

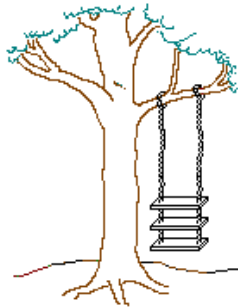
...

- Los proyectos más pequeños (de uso personal) se ‘parecen’ a los pequeños programas:
puede desarrollarlo el propio interesado, en un tiempo mínimo.
- Los proyectos más grandes se ‘parecen’ a los grandes proyectos software:
 - ⇒ *gran cantidad de personal y usuarios,*
 - ⇒ *personas distintas desarrollan, usan y mantienen,*
 - ⇒ *importancia fundamental: tareas relacionadas con aspectos administrativos, de planificación, estimación y control.*

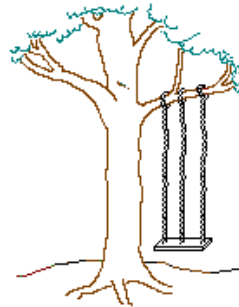
⇒ **“Planos” en la industria de la construcción: bien establecidos**

⇒ **¿tenemos “planos” en la industria del software?**

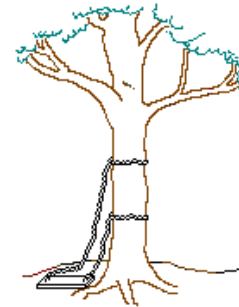
Motivación. Comunicación compleja en el desarrollo...



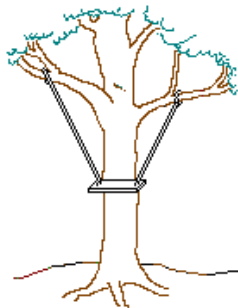
1. Lo que el director desea.



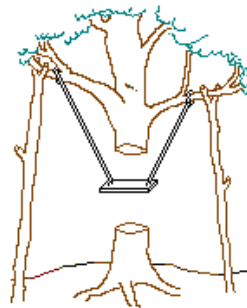
2. Como lo define el director de proyecto.



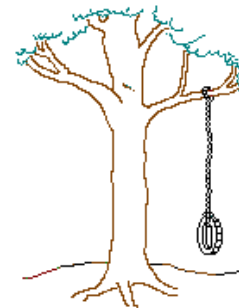
3. Como se diseña el Sistema.



4. Como lo desarrolla el programador.



5. Como se ha realizado la instalación.



6. Lo que el usuario quería.

Origen: desconocido
(finales años sesenta)

⇒ Problemas de comunicación con el cliente

⇒ Problemas de comunicación en el equipo

2. *El software*



- “**Programas** de computador, **procedimientos**, y, posiblemente, la **documentación** asociada y los **datos** pertenecientes a las operaciones de un sistema de computación”
 - **Instrucciones** que, cuando se ejecutan, proporcionan la funcionalidad deseada.
 - **Estructuras de datos**, que facilitan a las instrucciones manipular adecuadamente la información.
 - **Base de datos**, que registra la información que maneja el sistema.
 - **Documentos**, que describen el desarrollo, uso, instalación y mantenimiento de los programas.
- Incluye: **entrenamiento, soporte al consumidor e instalación.**

Características del software (frente al hardware)



- Más difícil de medir, validar, verificar:
 - Elemento lógico, no físico.
 - Desarrollado, no 'fabricado'.
- No se 'estropea', ¡pero se deteriora!
deterioro por 'cambios'
- Mayoritariamente 'cerrado':
 - tradicionalmente, usado todo o nada
 - tradicionalmente, poco ensamblaje de componentes

Perspectiva histórica del desarrollo de software

■ Década 1950-60:

- "Software como un añadido"
- Aplicaciones sencillas
- Desarrollo artesanal, a medida
- Lenguajes de bajo nivel

■ Década 1960-70:

- Software como producto
- Primeras aplicaciones complejas
- Década lenguajes y compilación
- "Crisis del software"

■ Década 1970-80:

- Programación estructurada
- Modelo relacional
- Primeras etapas Ingeniería del Software
- Primeros métodos estructurados
- Modelado de datos

■ Década 1980-90:

- Programación OO
- 4GLs
- C/S
- Tecnología de SGBDs, SOs
- Métodos estructurados
- Primeros métodos OO
- Tecnología CASE (1ª generación)

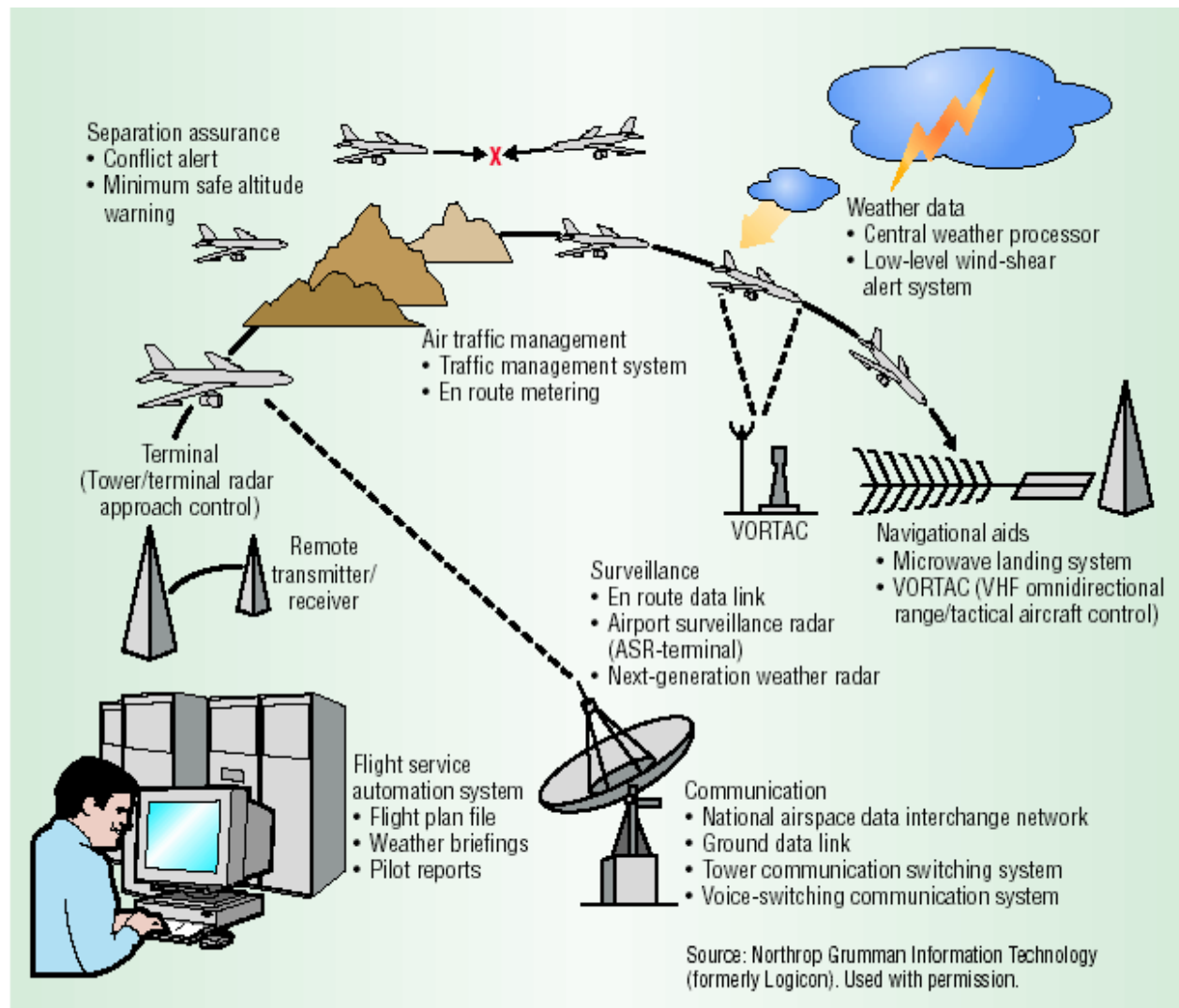
■ Década 1990-00

- Generalización POO
- Programación visual
- Tecnología de componentes
- Interoperabilidad (CORBA)
- Nuevas plataformas (Java, .NET)
- Análisis/Diseño OO
 - Fin "Guerra de los métodos"
 - UML (Unified Modeling Language, 1997)
- Patrones
- Tecnología CASE (2ª generación)
- Popularización de Internet

■ Década 2000-10

- Generalización comercio electrónico
- Web 2.0
- Desarrollo web
- Seguridad
- Arquitecturas basadas en servicios (SOA)
- Métodos ágiles
- GSD: Global Software Development
- Desarrollo open source
- MDE: Model-Driven Engineering

Actualmente, con frecuencia, el sw es la parte más compleja



3. Factores de calidad del sw. (internos y externos) *(Bell 2000)*

■ Correcto

- Se ajusta a las especificaciones dadas por el usuario.

■ Fiable

- Capacidad de ofrecer los mismos resultados bajo las mismas condiciones.

■ No Erróneo

- No existe diferencia entre los valores reales y los calculados

■ Eficiente

- Utilización óptima de los recursos de la máquina.

■ Robusto

- No poseer un comportamiento catastrófico ante situaciones excepcionales (tolerante a fallos).

■ Portable

- Capaz de integrarse en entornos distintos con el mínimo esfuerzo.

■ Adaptable (extensible)

- Modificar alguna función sin que afecte a sus actividades.


■ Inteligible

- Diseño claro, bien estructurado y documentado.

■ Reutilizable

- El software puede ser usado con facilidad en nuevos desarrollos.

Factores de calidad del software (II) (Sommerville 2004)



- Facilidad de mantenimiento
- Confiabilidad
 - *Fiabilidad*
 - *Seguridad*
 - *Protección*
- Eficiencia
- Facilidad de uso

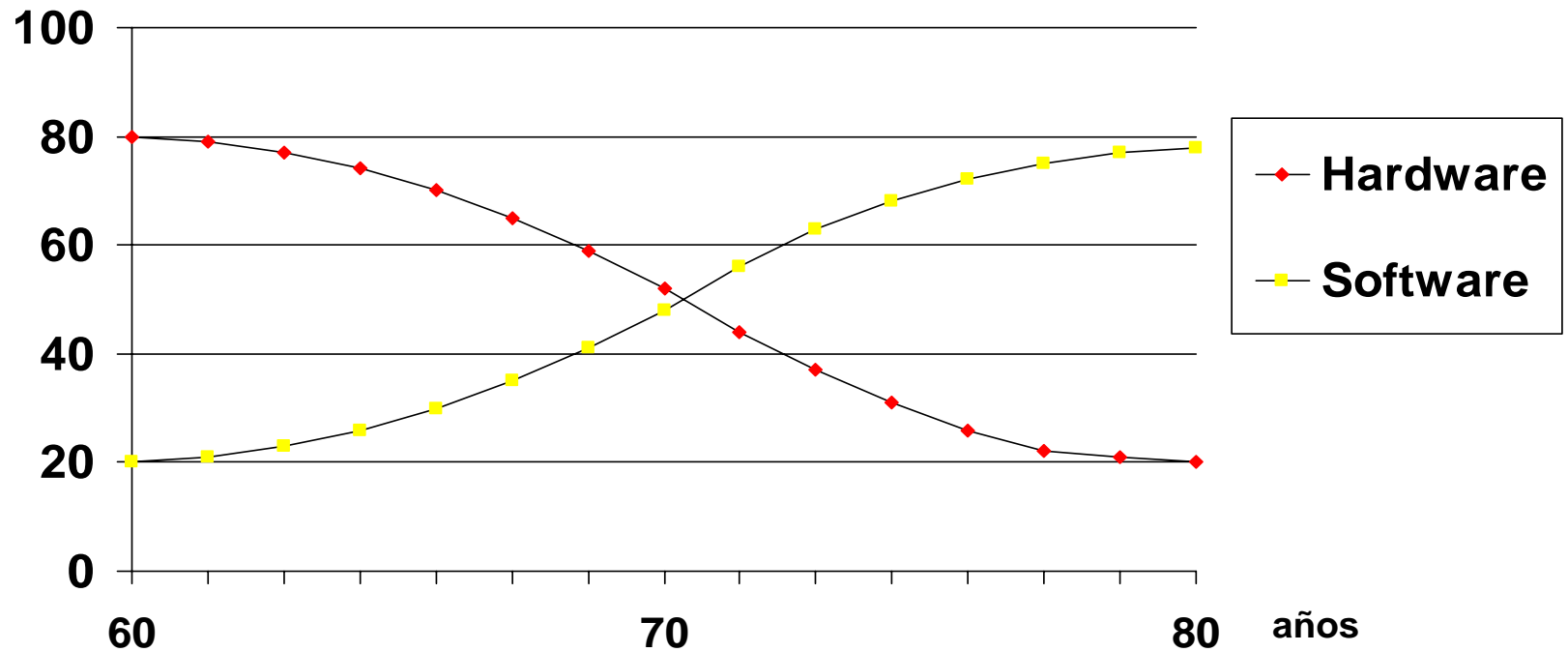
4. Problemas en el desarrollo de software



- Con el avance del hardware, necesidad de aplicaciones más complejas
 - ⇒ Se produjo un cambio en la relación entre el coste hardware/software
- Son problemas “tradicionales”:
 - Incapacidad para estimar tiempo, coste y esfuerzo para el desarrollo de un producto software.
 - Falta de calidad del producto software.

Relación coste hw./sw.

Porcentaje del coste
total del sistema

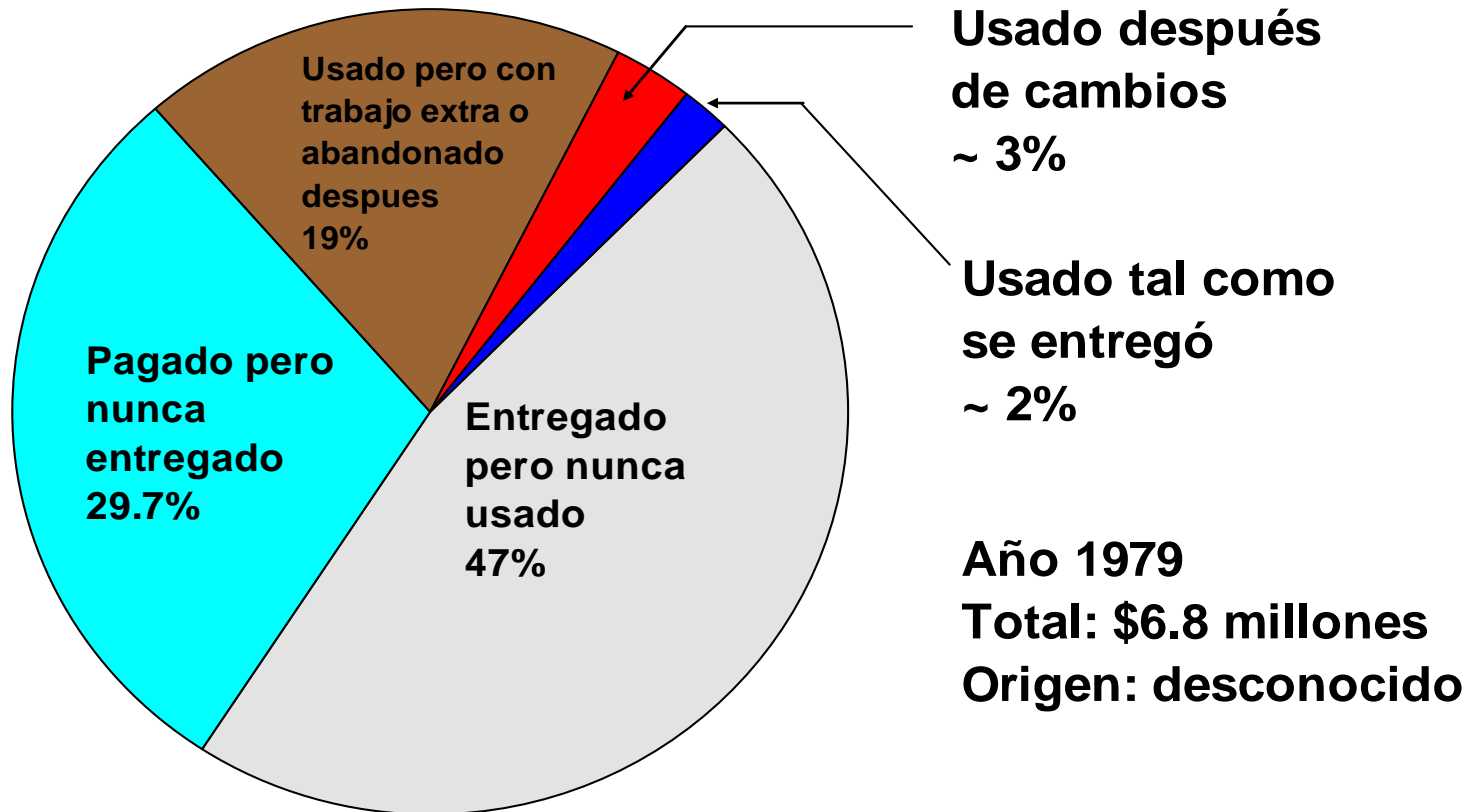


Problemas en el desarrollo de software (II) (Pressman)



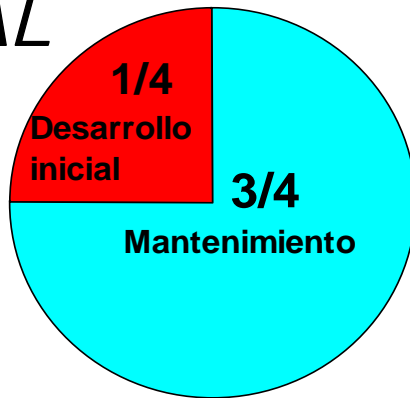
- *¿Por qué lleva tanto tiempo terminar los programas?*
- *¿Por qué es tan elevado su coste?*
- *¿Por qué no podemos encontrar todos los errores antes de entregar el software a nuestros clientes?*
- *¿Por qué nos resulta difícil constatar el progreso conforme se desarrolla el sw.?*

INVERSION EN DESARROLLO DE SISTEMAS SOFTWARE

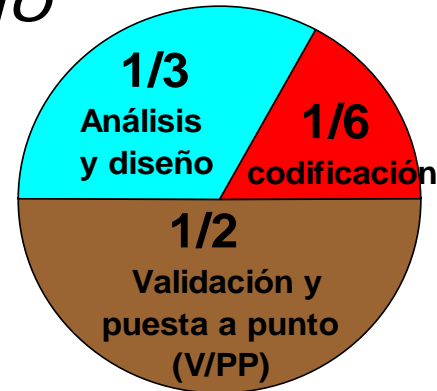


Coste del software

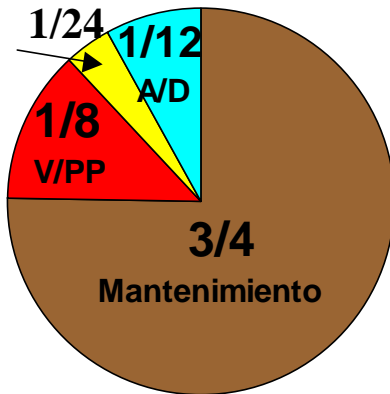
TOTAL



Desarrollo inicial



Codif.



TOTAL

VALIDACIÓN + PP + MANT. = $\frac{7}{8}$ (88%)

CODIFICACIÓN = $\frac{1}{24}$ (4%)

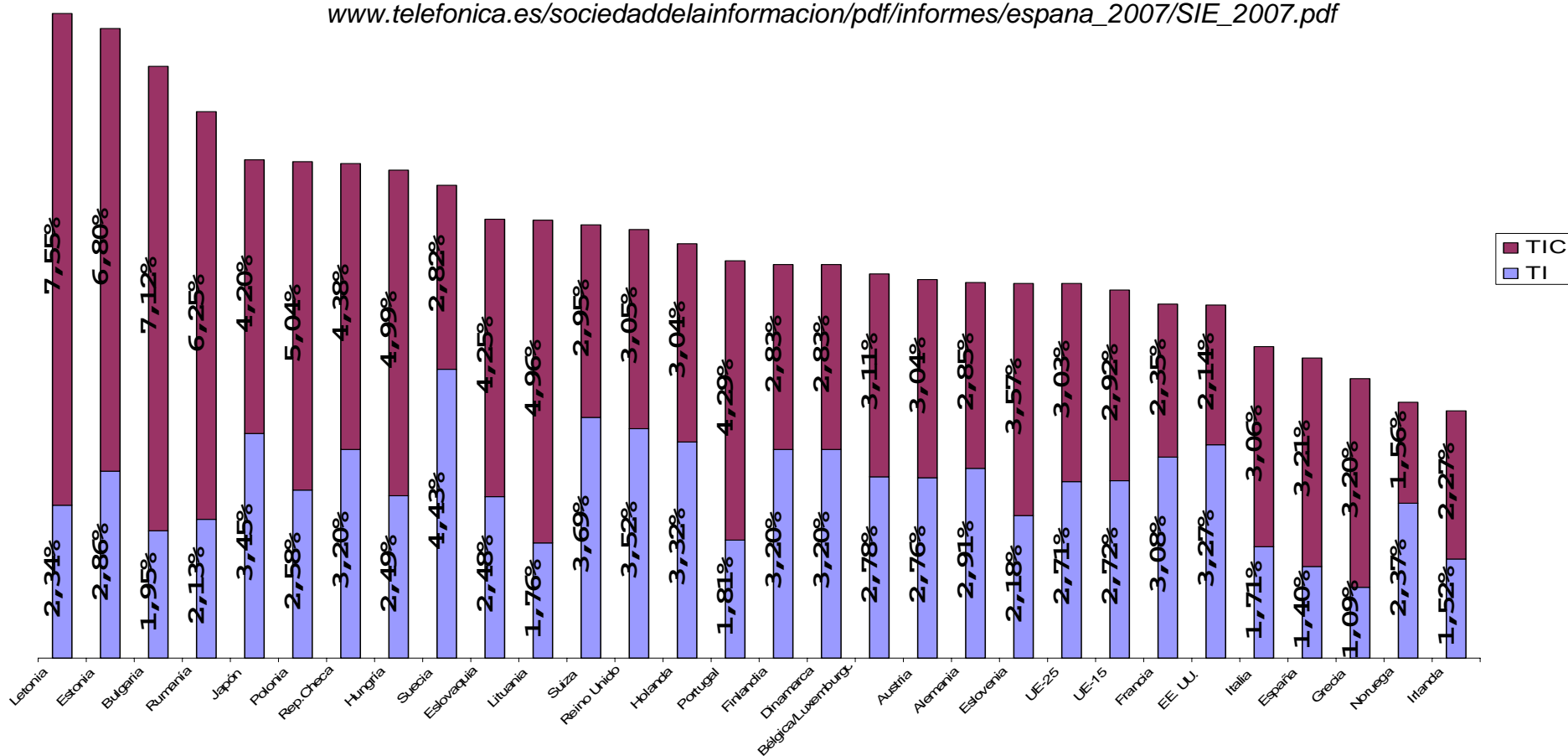
ANÁLISIS + DISEÑO = $\frac{1}{12}$ (8%)

En España

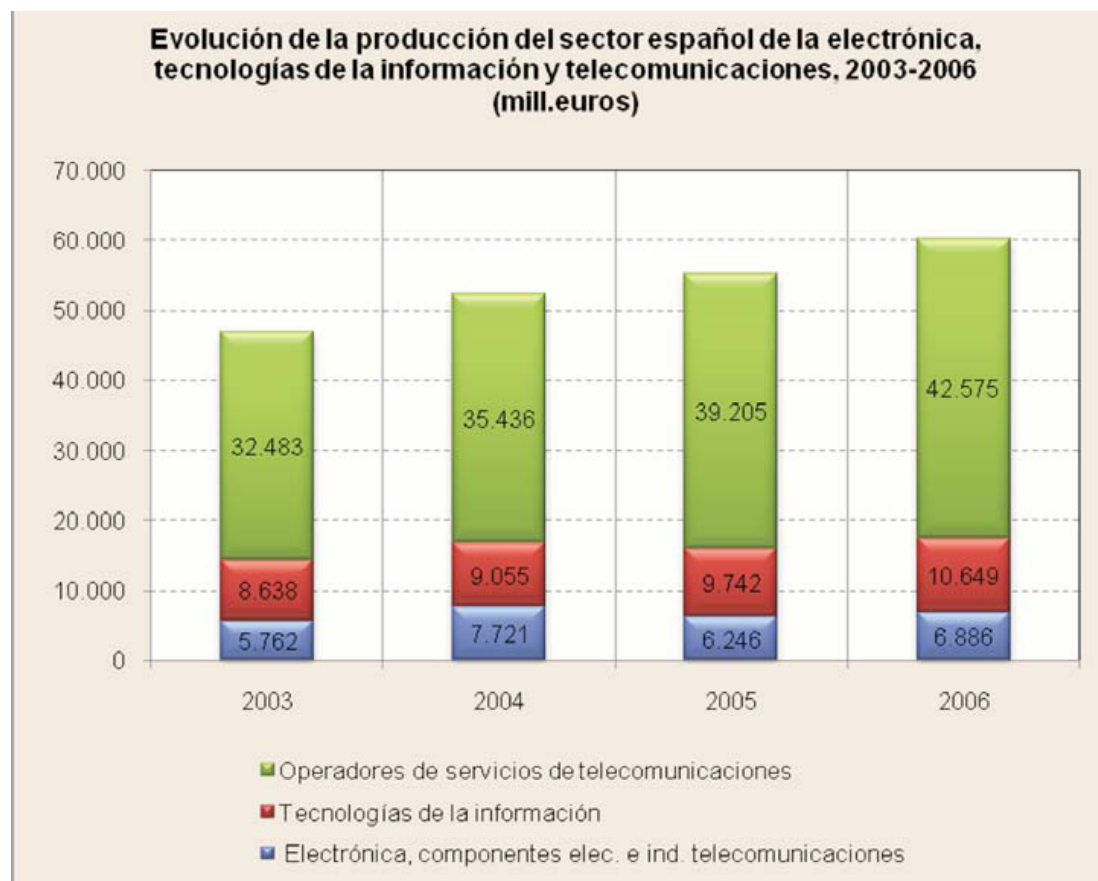
GASTO EN TECNOLOGÍAS DE LA INFORMACIÓN COMO PORCENTAJE DEL PIB (MUNDO)

Fuente: La Sociedad de la Información en España 2007. Fundación Telefónica. pág. 237

www.telefonica.es/sociedaddelainformacion/pdf/informes/espana_2007/SIE_2007.pdf



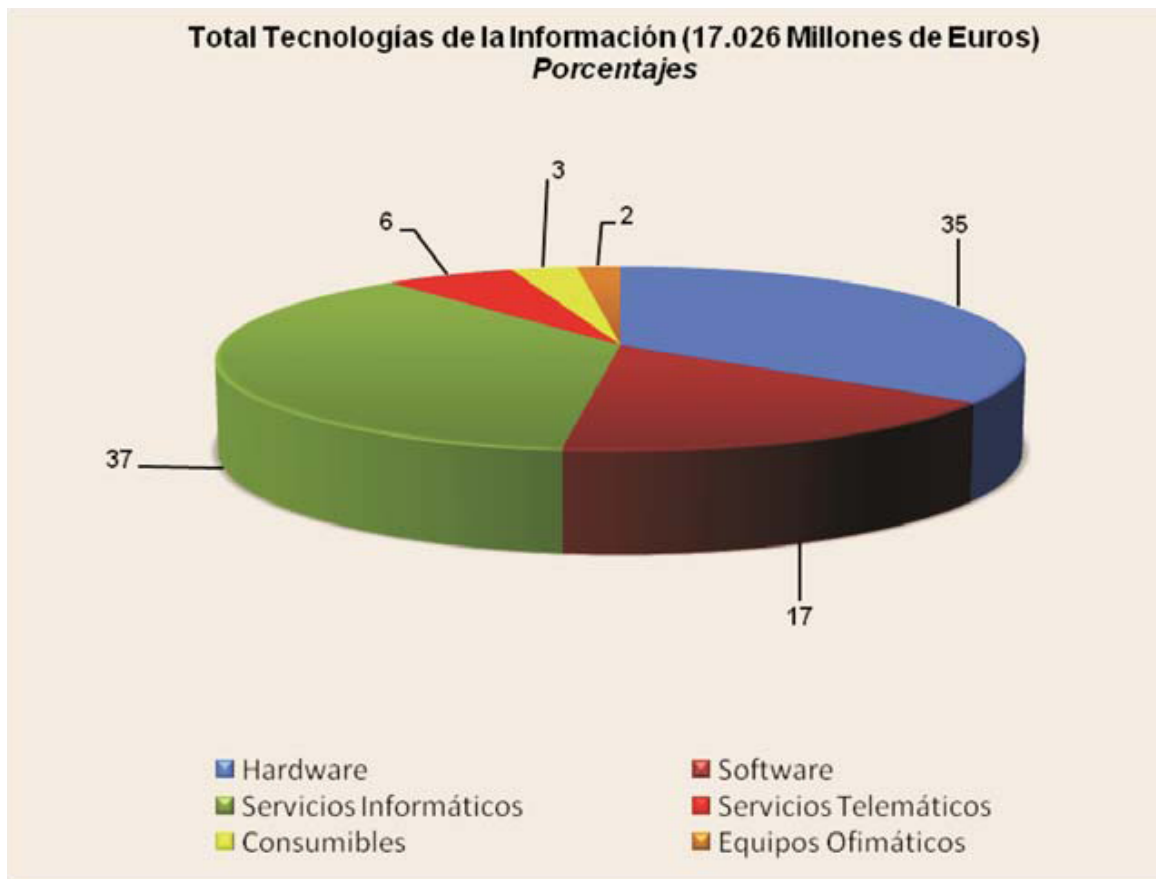
En España (II)



Fuente: AETIC (Asociación de Empresas de Electrónica, Tecnologías de la Información y Telecomunicaciones de España)

www.aetic.es

En España (III)



- Software de aplicación a medida.
- Poco software de base, paquetes integrados.
- Vigente la situación general.

Fuente: AETIC

Año: 2007

www.aetic.es

Algunas causas



- Naturaleza “no física” de la programación.
- El sw es la parte más ‘maleable’ del sistema
- Problemas de comunicación con los clientes.
- Problemas derivados de la intervención de grupos.
- Problemas de gestión.

Planificaciones optimistas, plantillas poco cualificadas...

- Poco esfuerzo en análisis y diseño.
- Difusión limitada de las nuevas técnicas, métodos y herramientas.

Algunas soluciones...

¡No existe bala de plata! (Brooks 87)

**métodos de análisis,
diseño, prueba...**

**Modelado del
negocio**

**ingeniería de
requisitos**

reutilización

Prototipado

POO

INGENIERÍA DEL SOFTWARE

métodos formales

herramientas CASE:


*editores dirigidos por la sintaxis,
entornos integrados de desarrollo,
herramientas para la gestión de
proyectos,
herramientas de prototipado, etc.*

herramientas 4ª gen.

*lenguajes no
procedimentales para
consulta a BD's
generadores de pantallas,
generadores de código,
generadores de informes*

Algunas soluciones...

¡No existe bala de plata! (II)



Aproximaciones realmente prometedoras
(se dirigen a la complejidad “esencial” en el desarrollo de software):

- Comprar en lugar de construir.
- Refinamiento de requisitos y prototipado rápido.
- Desarrollo incremental –“hacer crecer” el software, no construirlo–.
- Contar con grandes diseñadores.

Más información: “*No Silver Bullet - Essence and Accidents of Software Engineering*”
Brooks, F., IEEE Computer, 20, 4, April 1987 (http://en.wikipedia.org/wiki/No_Silver_Bullet)

5. *La Ingeniería del Software*



- Término que aparece en 1968
- La producción de programas debe abordarse como una ingeniería más.
- (Boehm) La Ingeniería del Software es la aplicación práctica y sistemática del conocimiento científico a:
 - la producción de programas correctos, que se desarrollan a tiempo y dentro de las estimaciones de presupuesto,
 - y a la correspondiente documentación para desarrollarlos, usarlos y mantenerlos.
- La Ingeniería del Software se fundamenta en técnicas relacionadas con:
 - ciencia de la computación, programación, ingeniería, administración, matemáticas, economía, etc.
- Forma parte de la “Ingeniería de Sistemas”

Más definiciones de ISW...



- “La ISW es el establecimiento y uso de principios sólidos de ingeniería, orientados a obtener software económico que sea fiable y trabaje de manera eficiente en máquinas reales” (Fritz Bauer).
- “ISW: (1) La aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, la operación y el mantenimiento del software; es decir, la aplicación de la ingeniería al software; (2) El estudio de enfoques como en (1)” (Glosario Estándar de Términos de Ingeniería del Software de IEEE, 1998).
- “Una disciplina que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza” (Sommerville 2002).

Situación actual de la ISW

(Sommerville 2004)



- En los últimos 20 años,
 - Los cambios en hardware han sido enormes.
 - Aparentemente, los cambios en software también:
 - P.ej., las grandes infraestructuras –energía, comunicaciones, transporte- descansan sobre sistemas muy complejos y en general muy fiables.
 - P.ej., auge de Internet y aplicaciones relacionadas
 - Se dispone de una enorme variedad de tecnologías (p.ej. J2EE, .NET, EJB, SAP, BPEL4WS, SOAP, CBSE) para construir aplicaciones –como las aplicaciones web- que pueden ser desplegadas mucho más rápidamente que en el pasado.
 - Sin embargo, más allá de la tecnología, si miramos los procesos de ingeniería del software, desgraciadamente muchas cosas permanecen igual.

Situación actual de la ISW (II)

(Sommerville 2004)



- El modelo en cascada sigue siendo utilizado por más del 40% de las empresas (IEEE Software, Dic. 2003), a pesar de que sus serios problemas fueron identificados hace 20 años.
- La prueba es la técnica de validación predominante, a pesar de que otras técnicas, como la inspección de programas, han sido usados más eficientemente desde los años 70.
- Las herramientas CASE son todavía simplemente editores de diagramas con algunas funcionalidades de chequeo y generación de código.
- Todavía muchos proyectos terminan tarde, exceden el presupuesto o no entregan el software que esperaban los clientes.

Situación actual de la ISW (III)



- En muchas áreas sigue sin existir un conjunto de estándares que se use ampliamente.
- No existen suficientes datos - guía (estadísticas).
- A diferencia de otras ingenierías, se carece de una base formal.
- En definitiva,
 - ⇒ La disciplina no es todavía madura
 - ⇒ Necesario mayor esfuerzo en educación en ISW

Situación actual de la ISW (IV)

(Sommerville 2004)



- Pero hay también **aproximaciones prometedoras**. Por ejemplo:
 - Se ha establecido **UML (Lenguaje Unificado de Modelado)** como una notación estándar de análisis y diseño OO.
 - Aparecen **métodos ágiles** como *Extreme Programming*.
 - *SWEBOK (Guide to the Software Engineering Body of Knowledge)* (2001).
 - Algunas universidades han comenzado a ofrecer un título en isw.
 - Comités CSAB (*Computer Science Accreditation Board*) y ABET (*Accreditation Board for Engineering and Technology*).
 - CMMI (*Capability Maturity Model Integration*) del SEI (*Software Engineering Institute*) y la familia de estándares ISO 9000 son usados para valorar la capacidad de una organización de isw.
 - En EE UU, el Colegio de Ingenieros Profesionales de Texas (*Texas Board of Professional Engineers*) ha comenzado a *licenciar* ingenieros del software.
 - ACM e IEEE-CS han desarrollado y adoptado conjuntamente un Código de Ética para Profesionales en Ingeniería del Software.

Situación actual de la ISW (V)



- Resumiendo, tres problemas esenciales en los comienzos del siglo XXI (Sommerville 2004):
 - El reto de lo heredado.
 - El reto de la heterogeneidad.
 - El reto de la entrega.
- Hoy día,
 - Existe un consenso en la importancia de la ISW.
 - Se ha avanzado mucho, pero queda mucho por hacer.
 - Muchos autores comienzan a renegar de la vigencia de la “crisis del software”, aunque la disciplina todavía no es madura.

Algunos principios de la ISW



■ **Abstracción**

- Permite parcelar la complejidad. Por ello se olvidan aspectos irrelevantes del sistema y se potencian los fundamentales.

■ **Encapsulamiento u Ocultación de la información**

- Esconder todos los detalles que no afecten a otros módulos, definiendo interfaces estrictos que sirvan de interacción entre los distintos modelos.

■ **Modularidad**

- Sirve para parcelar la solución en módulos independientes con fuerte cohesión interna.

■ **Localización**

- Deben estar agrupados todos aquellos elementos que están afectados por un mismo hecho.

■ **Uniformidad**

- Todos los módulos deben tener una notación similar.

■ **Compleitud**

- Deben estar desarrollados todos los aspectos del sistema.

■ **Validación y Verificación**

- El producto final debe ser fácilmente validable y verificable:
 - ¿Estamos desarrollando el programa correcto?
 - ¿Estamos desarrollando correctamente el programa?

6. *Visión general del proceso de ingeniería del software*



- Con independencia del área de aplicación, tamaño o complejidad del proyecto, el desarrollo de cualquier sistema se encontrará al menos en uno de los siguientes 'procesos' genéricos:
 - *Definición ~ análisis (del sistema, del sw.)*
 - *Desarrollo ~ diseño, codificación y prueba*
 - *Mantenimiento*

Definición



- *¿Qué debe hacer el sistema?*
 - funcionalidad del sistema
 - información que ha de manejar
 - necesidades de rendimiento
 - restricciones de diseño
 - interfaces del sistema con los usuarios y con otros sistemas
 - criterios de validación
- Documentos de requisitos del sistema (SyRS, *System Requirements Specification*) (en su caso) y del software (SRS, *Software Requirements Specification*)

Desarrollo



■ *¿Cómo construir el sistema?*

- Se diseñan las estructuras de datos y los programas
 - cómo se caracterizan las interfaces,
 - cómo realizar el paso del diseño al lenguaje de programación,
 - cómo ha de realizarse la prueba,
- se escriben y documentan los programas,
- y se prueba el software construido.

Mantenimiento

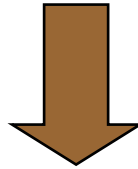


- Comienza una vez construido el sistema, cuando se pone en explotación.
- Se centra en el ***cambio***.
- El software es sometido a reparaciones y modificaciones cada vez que se detecta un fallo o se necesita cubrir una nueva necesidad de los usuarios.
- En esta fase recae el mayor porcentaje del coste de un sistema.

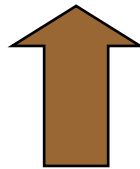
Mantenimiento (II)



Un buen sistema no es sólo un conjunto de programas que funcionan.



Debe ser ***fácil de mantener***



Documentación esencial
(CASE, *Computer Assisted
Software Engineering*)

Tipos de mantenimiento



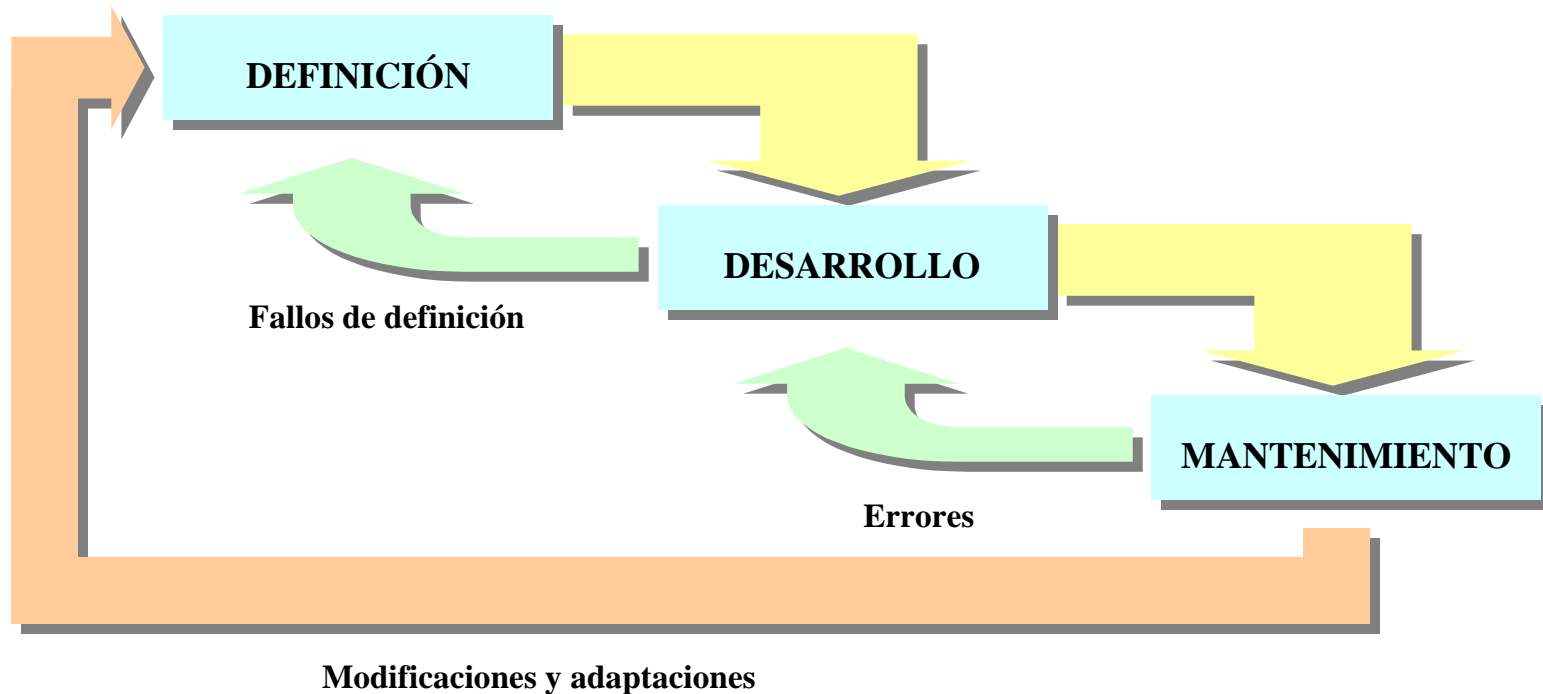
- ***Correctivo***: un programa no realiza correctamente la aplicación para la que ha sido diseñado, y, por tanto, debe ser modificado
- ***Perfectivo***: modificaciones a los programas para conseguir mayor adecuación a los requisitos, mayor eficiencia, o simplemente recoger nuevas funcionalidades no expresadas en la fase de definición del sistema

Tipos de mantenimiento (II)



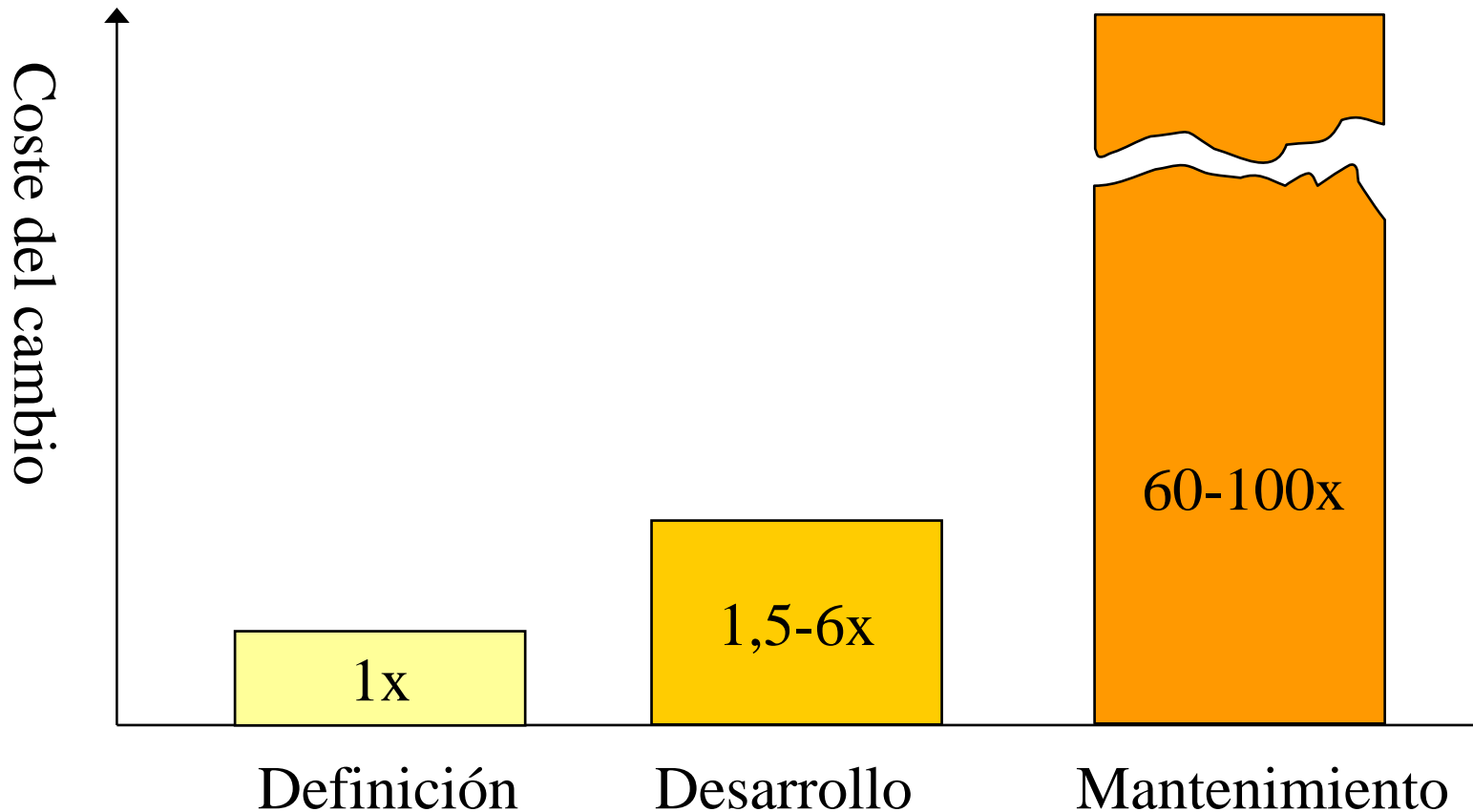
- ***Adaptativo***: Adaptar los programas para acomodarlos a los cambios de su entorno externo (modificaciones en la legislación, CPU, SO, las reglas de negocio, etc.)
- ***Preventivo***: El software se deteriora con los cambios, y este tipo de mantenimiento hace cambios en los programas para que se puedan corregir, adaptar y mejorar más fácilmente (*reingeniería del software*)

Visión general del proceso de ISW (II)



En la práctica, no es secuencial: ha de ser **iterativo e incremental** ('procesos', no 'fases')

Impacto del cambio (efecto bola de nieve) (Pressman)



7. Responsabilidad ética y profesional en ISW (Sommerville 2004)



- Está cobrando más interés en los últimos años.
- Los ingenieros de software tienen responsabilidades frente a la profesión y la sociedad.
- Su responsabilidad no es exclusivamente técnica.
- Deben comportarse de forma ética y moralmente responsable si quieren ser respetados como profesionales.
- Un ingeniero de software no debería comportarse de manera deshonesta o de una forma que perjudique a la profesión.

Responsabilidad ética y profesional en ISW (II)



- Hay áreas donde el concepto de *conducta aceptable* no está limitada por la ley sino por la noción más tenue de *responsabilidad profesional*.
- Algunas de ellas son:
 - *Confidencialidad*. Se debería respetar la confidencialidad de la empresa empleadora o de los clientes independientemente de que se haya firmado un contrato formal de confidencialidad.
 - *Competencia*. El ingeniero de software no debería aceptar conscientemente trabajo que esté fuera de su competencia.

Responsabilidad ética y profesional en ISW (III)



- *Derechos de la propiedad intelectual.* El ingeniero de software debería conocer las leyes que gobiernan la propiedad intelectual, como patentes y derechos de autor, y debería proteger la propiedad intelectual de clientes y empleadores.
- *Mal uso del ordenador.* El ingeniero de software no debería usar sus conocimientos técnicos para utilizar de forma incorrecta los ordenadores de otras personas. El mal uso va desde lo relativamente trivial (usar el ordenador de la empresa para jugar, p.ej.) hasta lo extremadamente serio (diseminación de virus).

Responsabilidad ética y profesional en ISW (IV)

- ACM (*Association for Computer Machinery*) e IEEE (*Institute of Electrical and Electronic Engineers*) publicaron en 1999 un código conjunto de ética y conducta profesional, que establece los estándares de conducta esperados de sus miembros.

http://www.acm.org/serving/se/code_s.html

- Es preciso aceptar dicho código para poder ser miembro de estas organizaciones.
 - ⇒ *El Octavo Principio establece como obligación el aprendizaje continuo a través de toda la vida profesional.*

Responsabilidad ética y profesional en ISW (V)



VERSIÓN CORTA (http://www.acm.org/serving/se/code_s.html)

Los ingenieros de software deberán comprometerse a convertir el análisis, especificación, diseño, implementación, pruebas y mantenimiento de software en una profesión respetada y benéfica. De acuerdo a su compromiso con la salud, seguridad y bienestar social, los ingenieros de software deberán sujetarse a los ocho principios siguientes:

- **Sociedad.** Los ingenieros de software actuarán en forma congruente con el interés social.
- **Cliente y empresario.** Los ingenieros de software actuarán de manera que se concilien los mejores intereses de sus clientes y empresarios, congruentemente con el interés social.
- **Producto.** Los ingenieros de software asegurarán que sus productos y modificaciones correspondientes cumplen los estándares profesionales más altos posibles.
- **Juicio.** Los ingenieros de software mantendrán integridad e independencia en su juicio profesional.
- **Administración.** Los ingenieros de software gerentes y líderes promoverán y se suscribirán a un enfoque ético en la administración del desarrollo y mantenimiento de software.
- **Profesión.** Los ingenieros de software incrementarán la integridad y reputación de la profesión congruentemente con el interés social.
- **Colegas.** Los ingenieros de software apoyarán y serán justos con sus colegas.
- **Personal.** Los ingenieros de software participarán toda su vida en el aprendizaje relacionado con la práctica de su profesión y promoverán un enfoque ético en la práctica de la profesión.