

Introducción a la Ingeniería del Software

Ingeniería del Software

Juan Pavón Mestras
Dep. Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense Madrid



Ingeniería del Software

¿Qué es la Ingeniería del Software?

¿En qué se diferencia un **Programador** de un **Ingeniero de Software**?

¿Cuál es la diferencia entre un Ingeniero de Software y un Ingeniero de Sistemas?

¿Qué diferencia la *Ingeniería* del Software de la *Ciencia* de la Computación?

¿Qué es el software?

¿Qué es un proceso de software?

¿Qué es la arquitectura del software?

Mitos del software

- *Te explico un poco como va y ya concretaremos luego*
- Es fácil modificar el software
- *Como es complejo, el software puede fallar*
- Una vez que el programa funciona, hemos terminado
- *Hasta que empieza a funcionar no sabré si está bien*
- Al cliente basta con darle un código que funcione
- El programa no falla, es el cliente que no sabe utilizarlo
- Con pruebas y verificación formal se pueden eliminar todos los errores
- *Cuanto más voluminosa sea la documentación de un producto, mejor será*
- Siempre podemos añadir más programadores
- *Si una característica de la aplicación no es necesaria para el 80% de los usuarios, al 20% restante realmente no le hará falta*
- Si un error ha sobrevivido a dos revisiones, no es un error, sino comportamiento normal del sistema

Desastres causados por fallos del software

- Explosión del Ariane 5, 1996
 - Motivo: conversión de datos de un número demasiado grande
- Pérdida del *Mars Climate Observer*, 1999
 - Motivo: mezcla de kilos y libras. El satélite acabó pegándose en Marte
- Airbus 320 derribado por un misil lanzado desde el USS Vincennes durante la guerra Irán-Irak, 1988
 - Fallo en el software de reconocimiento de patrones, que confundió a un avión civil con un F-14 iraní: 290 pasajeros muertos
- Muertes de pacientes de cáncer por sobredosis de radiación del equipo Therac-25, 1986
 - Fallo de control de condiciones de carrera
- Redondeo en la conversión del Euro a DM
 - 1 EURO = 1.95583 DM
→ 0.01 DM = 0.01 Euro y 0.01 Euro = 0.02 DM
- Virus y gusanos

¿Qué es el software?

■ Pressman:

1. Instrucciones (programas de computadora) que cuando se ejecutan proporcionan la función y el rendimiento deseados
2. Estructuras de datos que permiten a los programas manipular adecuadamente la información, y
3. Documentos que describen la construcción y uso de programas

■ Sommerville:

- Programas de ordenador y documentación asociada
- Los productos de software pueden ser
 - Genéricos: desarrollados para clientes muy diversos
 - Hecho a medida: para un cliente particular de acuerdo a su especificación

¿Qué es la Ingeniería del Software?

■ La Ingeniería de Software (IS) es

- una disciplina de ingeniería
 - Aplicación de teorías, métodos, herramientas para hacer cosas que funcionen:
 - Software que sea fiable y trabaje en máquinas reales
 - Teniendo en cuenta restricciones financieras, organizacionales y técnicas
- que comprende todos los aspectos de la producción de software
 - Desde la especificación inicial al mantenimiento del sistema
 - Administración y gestión del proceso de producción
 - Principios y metodologías para desarrollo y mantenimiento de sistemas de software

■ IEEE 610-12 (*Software Engineering*)

- Aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación (funcionamiento) y mantenimiento del software

¿Qué es la Ingeniería del Software?

- *La IS es aplicar el sentido común al desarrollo de sistemas software* [Navarro, UCM]

¿Qué es el sentido común?

- *Planificar* antes de desarrollar
 - *Diseñar* antes de programar
 - *Reutilizar* diseños que funcionan y son mantenibles
- *... utilizando las herramientas apropiadas* [Pavón, UCM]

Herramientas CASE

- *Computer-Aided Software Engineering (CASE)*
 - Software que facilita la realización de actividades del proceso de desarrollo de software
 - Edición de diagramas
 - Comprobar la consistencia de los diagramas
 - Generación de documentación
 - Seguimiento de actividades del proyecto
- *Upper-CASE*
 - Herramientas que ayudan en las actividades de captura de requisitos, análisis y diseño
- *Lower-CASE*
 - Herramientas para la programación, depuración y pruebas

- La **Ingeniería de Sistemas** se refiere a todos los aspectos del desarrollo de sistemas basados en computadora, tanto del hardware como del software y los procesos de diseño y distribución de sistemas
 - La Ingeniería del Software es solo parte de este proceso
 - Los ingenieros de sistemas se encargan de especificar el sistema, definir su arquitectura, integrar sus partes
 - Están menos relacionados con la ingeniería de los componentes del sistema (HW y SW)
- Al ser el software muchas veces la parte más importante del sistema, las técnicas de ingeniería del software se aplican en el proceso de ingeniería de sistemas

Ingeniería de Software y Ciencia de la Computación

- La Ciencia de la Computación se refiere a las teorías y los fundamentos subyacentes en los sistemas de computación
- La Ingeniería del Software trata los problemas prácticos del desarrollo de software
- Con las teorías de la ciencia de la computación no es suficiente para desarrollar software (al menos cuando el sistema tiene suficiente envergadura)

Relevancia de la IS

- Las economías de TODOS los países desarrollados dependen en gran medida del software
- Cada vez más sistemas son controlados por software
 - Comunicaciones
 - Seguridad
 - Administración
 - Fábricas
 - Comercio
 - Agricultura
 - Etc.
- El gasto en la Ingeniería de Software, representa un alto porcentaje del PIB de los países desarrollados

Costes del software

- Los gastos del software dominan sobre los de sistema
 - El HW es cada vez más barato
 - Cuesta más el software que hay en un PC que el PC
- Cuesta más mantener el software que desarrollarlo
 - En sistemas con una larga vida, los costes de mantenimiento llegan a multiplicar varias veces los costes de desarrollo
- Lo más caro en un proyecto son los desarrolladores
- La IS trata de mejorar el coste del desarrollo de software

¿Cuáles son los costes de la IS?

- Coste del software
 - Gastos de desarrollo
 - Gastos de mantenimiento y evolución
- El coste varía dependiendo de:
 - Tipo de sistema que se desarrolle y los requisitos de atributos del sistema como eficiencia y fiabilidad
 - Modelo de desarrollo
- Generalmente, para el desarrollo del software
 - 60% en desarrollo
 - 40% en pruebas
- En software hecho a medida los gastos de evolución suelen ser mayores que los de desarrollo
 - En software genérico muchas veces no se considera la evolución sino que cada nueva versión se trata como un nuevo producto (razones mercantiles)

Retos de la IS

- Sistemas heredados (*legacy systems*)
 - Mantenimiento, actualización, integración
- Heterogeneidad (SW y HW) de sistemas distribuidos
 - Integración y evolución
- Tiempos de desarrollo cada vez más cortos
 - Y con menos recursos
 - Proyectos web: 3 meses–3 personas–3 kilos
- Modas
 - Métodos, lenguajes, ...
- Cultura de ingeniería
- Formalidad
 - Existe una gran demanda de que exista formalidad en el proceso de desarrollo de software

Personas

Producto

Proceso

Producto: el software

- El software cada día es más caro, más grande, menos eficiente, menos robusto, ...
 - El hardware cada día es más barato, más pequeño, más eficiente, más robusto, ...
- El software hoy:
 - Distribuido: web, nube, ...
 - Multi-componente
 - Multi-plataforma
 - Ubicuidad
 - Interfaces multi-modales
 - Múltiples versiones
 - Inteligente

Características del software

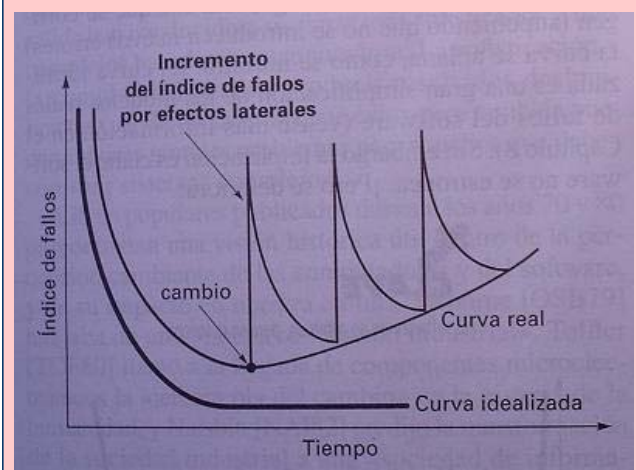
- El software se **desarrolla**, no se fabrica
 - Los costes se centran en ingeniería, no en fabricación
 - Los proyectos software no se pueden gestionar como procesos de fabricación
- El software no se estropea
- Pero hay que **mantenerlo**

Características del software

Curva de fallos del hardware



Curva de fallos del software



Características del software

- Reparación del software
 - El software deteriorado no se puede reparar
 - ¿revisar miles de líneas de código?
 - ¿cambia una versión con el tiempo?
 - Muchas veces las reparaciones dañan más al software
 - El software debe estar *bien diseñado* para facilitar su evolución

Software bien diseñado

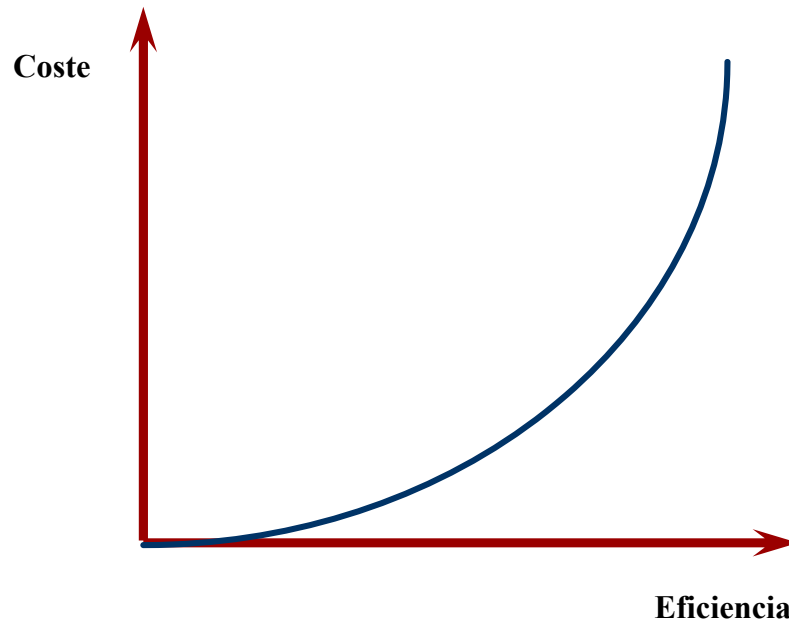
- Atributos del software bien diseñado
 - Mantenible
 - Capaz de evolucionar según las necesidades de cambio de los clientes
 - Seguro
 - Robusto, que no produce daños incluso bajo un fallo del sistema
 - Eficiente
 - No desperdicia los recursos del sistema (memoria, procesador, disco)
 - Amigoso
 - Buena interfaz
 - Bien documentado

Atributos en tensión: su importancia depende del sistema y del entorno en el que será utilizado

El coste tiende a ser alto si se exige un alto nivel de alguna característica

Características del software

- Coste de la eficiencia del software



Software bien diseñado

- Componentes software
 - Ingeniería: creación y mantenimiento de una serie de componentes estándar con el fin de no reinventar la rueda
 - Software bien diseñado debe favorecer la reutilización de código
 - Las tecnologías OO y de componentes software reutilizables favorecen dicha reutilización

Distintos tipos de software

- Software de **sistemas**
 - Programas escritos para servir a otros programas
 - Compiladores, Sistemas Operativos (SSOO), etc.
- Software de **gestión**
 - Proceso de información comercial, accediendo a bases de datos que contienen dicha información
 - Gestión de nóminas, control de almacén, etc.
- Software de **PC**
 - Procesadores de texto, hojas de cálculo, navegadores, etc.
- Software de **ingeniería y científico**
 - Algoritmos numéricos
 - Programas CAD, simuladores, etc.
- Software **empotrado** (*embedded systems*)
 - Controla productos y sistemas de mercados industriales y de consumo
 - Reside en ROM
 - Relacionado con el tiempo real
- Software de **inteligencia artificial**
 - Problemas complejos
 - Sistemas expertos
 - Reconocimiento de patrones (voz, imágenes, etc.)
 - Agentes software
- Software en **móviles**
 - Recursos limitados

De vuelta con los mitos del software

- Mitos del gestor
 - Mito: Tenemos un manual de desarrollo de software, ¿qué más necesitamos?
 - Realidad: ¿Se entiende? ¿Se utiliza? ¿El personal tiene práctica en su aplicación?
 - Mito: Disponemos de las herramientas de desarrollo más avanzadas, ya que compramos siempre los mejores equipos
 - Realidad: ¿Se invierte en herramientas CASE? ¿Y en entornos de desarrollo? ¿Se forma al personal en el uso de estas herramientas?
 - Mito: Si fallamos en la planificación, podemos añadir más personal y adelantar el tiempo perdido
 - Realidad: En el proceso de software añadir gente puede retrasar más el proyecto. La gente debe añadirse de forma planificada y ordenada. Además si sacamos a gente de otros proyectos, en último término retrasaremos otros proyectos

De vuelta con los mitos del software

- Mitos del cliente
 - Mito: Una declaración general de objetivos es suficiente para comenzar a escribir los programas, y podemos dar los detalles más adelante
 - Realidad: Una mala definición inicial conlleva trabajo inútil
 - Mito: Los requisitos del proyecto cambian continuamente, pero los cambios pueden acomodarse fácilmente porque el software es flexible
 - Realidad: Es cierto que los requisitos cambian, pero el impacto del cambio varía en función del momento en que se introduzcan los cambios

De vuelta con los mitos del software

| Impacto del cambio | |
|--------------------|------------------|
| Momento | Coste del cambio |
| Definición | 1x |
| Desarrollo | 1,5-6x |
| Después entrega | 60-100x |

De vuelta con los mitos del software

- Mitos de los desarrolladores
 - Mito: Una vez que escribamos el programa y hagamos que funcione, nuestro trabajo ha terminado
 - Realidad: Entre el 50% y el 70% de todo el esfuerzo dedicado a un programa se realiza después de que se entregue al cliente por primera vez
 - Mito: Hasta que no tenga el programa ejecutándose, no tengo forma de medir su calidad
 - Realidad: Revisiones Técnicas Formales durante el desarrollo de software.
 - Mito: Lo último que se entrega al terminar el proyecto es el programa funcionando
 - Realidad: Software = programas + datos + documentos

Responsabilidad y ética profesional

- Confidencialidad
 - De los demás empleados y de los clientes
- Competencia
 - Reconocer los límites y capacidades para aceptar un trabajo
- Derechos de propiedad intelectual
 - Patentes, copyright
 - Trabajo de otros colegas
- Mal uso de los sistemas
 - Juegos, virus, pirateo

Responsabilidad y ética profesional

- Código ético de ACM/IEEE
 - Principios que deben guiar el comportamiento y decisiones de ingenieros software profesionales (incluyendo gestores, estudiantes y profesores)
 1. Actuar en bien del interés público
 2. Actuar en el mejor interés del cliente y el empleador, siendo consistente con el interés público
 3. Asegurar que los productos y modificaciones reúnen los mejores estándares profesionales posibles
 4. Mantener la integridad e independencia en el juicio profesional
 5. Suscribir y promocionar un comportamiento ético en la gestión y mantenimiento del desarrollo de software
 6. Colaborar en el avance de la integridad y la reputación de la profesión siendo consistente con el interés público
 7. Ser justo y ayudar a los colegas
 8. A lo largo de la vida, reciclarse en la práctica de la profesión y promocionar un comportamiento ético en la práctica de la profesión

Responsabilidad y ética profesional

- Dilemas en el ejercicio de la profesión
 - Desacuerdo con los principios y política de los superiores
 - El empleador actúa de manera no ética y libera un sistema crítico de seguridad sin haber acabado las pruebas del sistema
 - Participación en el desarrollo de sistemas con fines contrarios a la propia moral

Conclusiones

- El desarrollo y mantenimiento de software:
Personas, Producto, Proceso
- El software siempre evoluciona
- La IS trata de abordar estas cuestiones de forma rigurosa

¡ Cuidado con los mitos !

Bibliografía

- R. Pressman: Ingeniería del Software - Un enfoque práctico, 7ª edición. McGraw-Hill, 2010
- I. Sommerville: Ingeniería del Software, 7ª edición. Addison Wesley, 2006
- Frederick P. Brooks, Jr., The Mythical Man-Month, Addison Wesley, 1975 (Anniversary edition 1995)