1. El producto

Índice

Referencias

- Pressman, R.S. *Ingeniería del Software. Un Enfoque Práctico. Sexta Edición*. McGraw-Hill, 2005
- Sommerville, I. *Ingeniería del Software*. 7ª edición. Addison-Wesley, 2005

El producto
Introducción

- Al comienzo de la informática el hardware era más importante que el software
- El software era un añadido al hardware
- Con el tiempo cambia y se llega a un equilibrio

Evolución del software
Etapas

- Primera etapa (1950-1965 aprox.)
  - Orientación por lotes (batch).
  - Distribución limitada.
  - Software a medida.

Evolución del software
Etapas

- Segunda etapa (1965-1975 aprox.)
  - Sistema multiusuario.
  - Sistemas en tiempo real.
  - Sistemas de Bases de Datos (BD).

  } Sistemas Interactivos*

  - Productos software independientes del hardware → mantenimiento del software con versiones.

* HCI: *Human Computer Interaction (http://www.hcibib.org/hci-sites/)*

Evolución del software
Etapas

- Tercera etapa (1975-1985 aprox.)
  - Sistemas distribuidos.
  - Hardware de bajo coste (microprocesador).
  - Impacto en el consumo.
- Cuarta etapa (1985-2000 aprox.)
  - Sistemas personales potentes.
  - Tecnologías orientadas a objetos.
  - Redes de computadoras.
  - Computación en paralelo.

Evolución del software
Etapas

- ¿Quinta etapa (desde 2000 aprox.)?
  - Omnipresencia de la Web.
  - Reutilización de información.
  - Componentes software reutilizables.
  - Integración de sistemas
  - Informática móvil

Evolución del software Problemas

- Problemas persistentes en la evolución:
  - El software nunca explota las posibilidades plenas del hardware.
  - El desarrollo del software no es tan rápido como su demanda.
  - Sociedad dependiente de las computadoras → necesitamos software *fiable*.
  - Los programas no son *escalables* ni *mantenibles* por culpa de diseños pobres y recursos inadecuados.

Evolución del software Perspectiva industrial

- En los comienzos:
  - Hardware de propósito general.
  - Software ligado al hardware, sin venta independiente.
  - Software a medida de la organización.
  - Baja movilidad de programadores → falta de documentación sistematizada.
  - Esto no implica mal software, simplemente software *poco mantenible.*
  - Proyectos centrados en el hardware y su coste
  - Uso de ingeniería hardware, pero no software

Evolución del software Perspectiva industrial

- Hoy en día, el software es el factor principal en el presupuesto.
  - Software con alto tiempo de desarrollo, incluso fuera de plazo → Costes elevados.
  - Software entregado a clientes con errores (defectos).
  - Falta de relación entre el trabajo realizado y la escritura total del software.

Discusión

- ¿Hay *truco* en el razonamiento de porque el software ha desbancado al hardware en el coste de los proyectos informáticos?

Evolución del software
Fábrica

- La fábrica de software que envejece
  - Metáfora de Pressman respecto a la industria del metal en EEUU en los años 50-60.
- Situación:
  - Sistemas que nadie entiende y son difícilmente modificables.
  - Solución: abandonar parches y construir nuevo software.
  - Evitar el: "si funciona para que cambiarlo".

Software
Definición

- Definición:

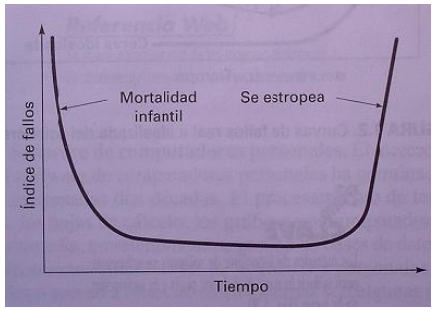
1. Instrucciones (programas de computadora) que cuando se ejecutan proporcionan la función y el rendimiento deseados.
2. Contenedores de datos que permiten a los programas manipular adecuadamente la información.
3. Documentos que describen la construcción y uso de programas

Software
Características

- El software es una entidad lógica → características:
  - 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.

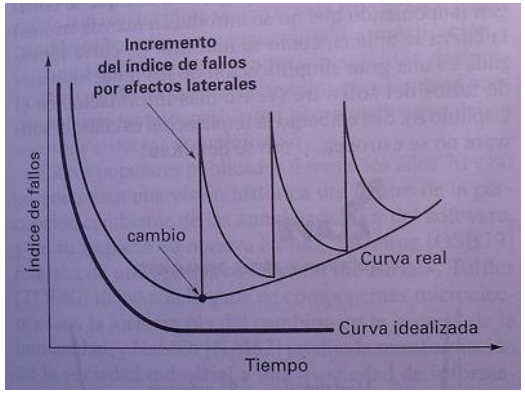
Software
Características

- Curvas de fallos



Curva de fallos del hardware (bañera)

Software
Características



Curvas de fallos software

Software
Características

- *Reparación* del software
  - El software deteriorado *no se puede* reparar (¿revisar miles de líneas de código?).
  - Muchas veces las *reparaciones dañan más al software*.
  - El software debe estar bien diseñado para facilitar su *evolución*.

Software
Bien diseñado

- Software bien diseñado
  - *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

Software
Bien diseñado

- Atributos del software bien diseñado
  - *Mantenible*. Software capaz de evolucionar según las necesidades de cambio de los clientes.
  - *Seguro*. Software robusto que no produce daños incluso bajo un fallo del sistema.
  - *Eficiente*. Software que no desperdicia los recursos del sistema (e.g. memoria o ciclos de reloj).
  - *Amistoso*. Software con buena interfaz y documentación.
  - Atributos *en tensión* → su importancia depende del sistema.

Software Aplicaciones

- Podemos clasificar las aplicaciones software en base al *contenido* y *determinismo* de la información que procesan
  - *Contenido*: tipo y forma de la información de Entrada/Salida (E/S) (e.g. procesador texto vs. control avión).
  - *Determinismo*. Predecibilidad del orden y del tiempo de la llegada de los datos (e.g. predicción del tiempo vs. sistema operativo).

Software Aplicaciones

- Clasificación:
  - Software de sistemas.
    - Programas escritos para servir a otros programas.
    - Compiladores, Sistemas Operativos (SOs), etc.
    - Características:
      - Fuerte interacción con el hardware de computadora.
      - Múltiples usuarios.
      - Operación concurrente.
      - Compartición de recursos.
      - Estructuras de datos complejas.

Software Aplicaciones

- Software de tiempo real
  - Mide, analiza y controla sucesos del mundo real conforme ocurren.
  - Control de aviones, etc.
  - Componentes:
    - *Adquisición de datos*. Recolecta y da formato a la información recibida del entorno externo.
    - *Análisis*. Transforma la información según lo requiere la aplicación.
    - *Control/salida*. Responde al entorno externo.
    - *Monitorización*. Coordina a los demás componentes para obtener una respuesta en *tiempo real* (de 1 milisegundo a 1 minuto).

Software Aplicaciones

- Software de gestión.
  - Procesamiento de información comercial, accediendo a Bases de datos (BDs) que contienen dicha información.
  - Gestión de nóminas, control de almacén, etc.
- Software de ingeniería y científico.
  - Algoritmos numéricos.
  - Programas CAD, predicción metereológica, etc.

Software
Aplicaciones

– Software empotrado.
  - Controla productos y sistemas de mercados industriales y de consumo.
  - Control de fábricas, etc.
  - Reside en ROM.

– Software de computadoras personales.
  - Se venden en grandes almacenes.
  - Procesadores de texto, hojas de cálculo, etc.

Software
Aplicaciones

– Software de inteligencia artificial.
  - Algoritmos no numéricos para resolver problemas complejos para los que no son adecuados el cálculo o análisis directo.
  - Sistemas expertos, reconocimiento de patrones (voz, imágenes, etc.), etc.

Mitos del software

- Mitos surgidos durante los primeros años del desarrollo del software.
- Son actitudes erróneas que causan serios problemas tanto a gestores como a técnicos.
- Pueden *afectar* a:
  – Gestores.
  – Clientes.
  – Programadores.

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?

*CASE: Computer-Aided Software Engineering

Mitos del software

– *Mito*: Si fallamos en la planificación, podemos añadir más programadores y adelantar el tiempo perdido (concepto de *horda mongoliana*).

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.

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.

Mitos del software

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

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 cincuenta y el setenta por ciento 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.

Mitos del software

– *Mito*: Lo último que se entrega al terminar el proyecto es el programa funcionando.
Realidad: Software = programas + datos + documentos.

– *Mito*: La IS obliga a la creación de una documentación voluminosa e innecesaria volviendo el proceso más lento.
Realidad: La IS se sustenta sobre una aproximación de calidad, y lo que consigue son proyectos mantenibles acabados a tiempo.

Conclusiones

- Software: producto.
- Software: rápida evolución.
- Constante en la evolución: problemas.
- IS: solución a estos problemas.
- Cuidado con los mitos.