Ingeniería del Software

Curso 2007-2008

German Rigau

german.rigau@ehu.es

Máster Sistemas Empotrados

Índice

Objetivos

- Mostrar las técnicas básicas para planificar, gestionar y desarrollar productos de software complejos (Proyectos Informáticos, Sistemas de Información) de gran tamaño.
- Dominar el proceso y las herramientas de análisis, diseño, implementación y pruebas de software orientado a objetos (PUD, UML).
- Aplicación práctica a un problema real.

- Un mensaje básico: en cada ámbito, una buena organización es necesaria si queremos producir software de calidad (eficaz, eficiente, robusto, etc.) rápidamente.

Temario (1)

- Introducción a la Informática
- Introducción a la programación
- Programación avanzada

Temario (2)

- Planificación, Gestión y Desarrollo de proyectos
- Análisis y diseño de Sistemas de Información
- Implementación y pruebas

Bibliografía (1)

- De Euclides a Java. Historia de los algortimos y de los lenguajes de programación. Ricardo Peña. 2006.

- Aprenda (Java, C, C++, ...) como si estuviera en primero. ...

- Object oriented software construction (2nd edition), B. Meyer. Prentice Hall, 2000.
- Concepts in programming languages, J. Mitchell. Cambridge University Press, 2001.
- Estructuras de datos en Java, M. Weiss. Prentice Hall, 2000.
- Bundle of Algorithms in CPP, Parts 1-5: Fundamentals, Data Structures, Sorting, Searching, and Graph Algorithms (3rd Edition), R. Sedgewick Addison-Wesley, 2001.
- Introduction to Algorithms, T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein Second Edition, MIT Press, 2001.

Bibliografía (2)

- *Applying UML and patterns.* Larman C. Prentice hall. 2001
- *Practical Software Engineering.* Maciaszek & Liong. Addison Wesley. 2004
- *Ingeniería de Software orientada a objetos con UML, Java e Internet*. Alfredo Weitzenfeld. Thomson, 2005

- *Especificación de Sistemas Software en UML*. Costal D., Ribera M., Teniente E., Edicions UPC. 2003
- *Diseño de sistemas software en UML*. Gómez C., Mayol E., Olivé A. Teniente E., Edicions UPC. 2003
- *El Proceso Unificado de Desarrollo de Software,* Jacobson, Booch, Rumbaugh, Addison-Wesley 1999
- *Ingeniería del software: Un enfoque práctico,* Pressman, R. S., 5a Edición, McGraw-Hill, 2001.
- *Using UML,* Pooley, R. and Stevens, P., Addison-Wesley 1999.
- *Design Patterns,* Gamma, E., Helm, R., Johnson, R. and Vlissides, J., Addison-Wesley, 1995.
- *Object-Oriented Analysis and Design with Applications,* Booch, G., Addison-Wesley, 1994.

La nota

- Laboratorio (L)
- Práctica (P)
- E = 0,7 * L + 0,3 * P

El Proyecto

- El proyecto lo realiza un equipo de 2 personas
- Tiene un peso del 30% de la nota final.

- El proyecto consiste en planificar, analizar, diseñar, construir, probar y entregar un producto software.
- EL objetivo de la práctica es enseñar qué puede ir mal en el desarrollo de un proyecto informático (de la manera más real!).

La motivación

- El desarrollo de software frecuentemente va mal. Mal significa:
  - TARDE (nunca se cumplen los plazos)
  - CARO (por encima del presupuesto)
  - INEFICACES (no consiguen lo que se pretendía)
  - causan STRESS (al desarrollador y al cliente!)
- Se ha estimado que el 15% de los proyectos informáticos se anulan antes de terminarse. Este número crece al 25% de los proyectos que requieren más de 25 años/persona.

La motivación

- Por qué desarrollar software es tan difícil?
- El software es:
  - siempre nuevo (si no queda obsoleto)
  - cada vez más complejo (cómo se gestiona la complejidad?)
  - difícil de controlar y verificar (poco robusto)
  - desarrollado básicamente de forma artesanal (la mayor parte del coste del desarrollo de un SI es en personal!)
  - desarrollado básicamente a medida (casuística)
  - Entre el 60 - 85% de los costes en software se invierten en *modificaciones*

Las soluciones

- Lenguajes de alto nivel
- Programación estructurada
- Diseño modular
- Métodos formales
- Tipos Abstractos de Datos
- Programación Orientada a objetos
- Programación lógica, funcional, etc.
- Lenguajes de 4 generación
- Entornos visuales de desarrollo, ...

Buenas prácticas

- No hay varita mágica (una solución única)
- Nuestro mejor aliado será analizar y comprender la diferencia entre buenos y malos proyectos, descubrir cuáles son las buenas prácticas y adoptarlas.
- En otras palabras, debemos aplicar el “sentido común”:
  - Estar atentos al proceso de desarrollo
  - Adoptar estándares para la documentación, codificación, etc.
  - Aprender de los errores (preferiblemente de otra gente!)

Importancia de la Ingeniería del Software

- Ariane Flight 501 Failure Report
  - El fallo ocurrió el 4 de Junio de 1996
  - Section 3.2 Cause of the failure
    - “The failure of the Ariane 501 was caused by the complete loss of guidance and attitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift- off). This loss of information was due to specification and design errors in the software of the inertial reference system”.
  - EL vuelo 501 de Ariane falló por un error sistemático de software, no por mala suerte.