INGENIERIA DE SOFTWARE II

Año 2010

Carrera: ***Licenciatura en Informática***
Plan 2003-2007
Licenciatura en Sistemas
Plan 2003-2007

Año: **3°**
Duración: ***Semestral***
Profesor: ***Lic. Patricia Pesado***
Lic. Marcos Boracchia
C.C. Silvia Esponda
Hs. semanales: **6 hs.**

OBJETIVOS GENERALES:

Continuar con los temas desarrollados en Ingeniería de Software 1, a partir del diseño de sistemas de software. Introducir los conceptos de re-ingeniería e ingeniería inversa. Estudiar los temas de gestión, planificación y evaluación de proyectos de software, incluyendo el análisis de riesgo. El alumno deberá desarrollar sistemas concretos utilizando las metodologías/herramientas estudiadas.

CONTENIDOS MINIMOS:

- Diseño e Implementación.
- Verificación y validación.
- Mantenimiento.
- Interacción hombre-máquina.
- Reingeniería e ingeniería inversa.
- Gestión de proyectos. Planificación. Métricas.
- Estimaciones. Análisis y gestión del riesgo.
- Conceptos de Auditoría y Peritaje

Programa

1. Gestión de Proyectos

- Conceptos. El problema de las 4 “P” (personal, producto, proceso, proyecto). Actividades de gestión, planificación del proyecto, hitos y entregas. El plan de proyecto.

- Métricas y Estimaciones.

- Clasificación de las métricas. Métricas del proceso y del proyecto. Métricas orientadas al tamaño, a la función, a casos de uso. Recopilación, cálculo y evaluación de métricas.

- Estimación de proyectos. Técnicas de descomposición. Modelos empíricos (COCOMO). Decisión de desarrollar-comprar.

- Planificación Temporal: calendarización del proyecto, distribución del esfuerzo, redes de tareas, seguimiento de la planificación. Métodos PERT, Gantt.

- Planificación Organizativa: del equipo y del proyecto.

- Gestión del Riesgo: identificación de riesgos, proyección, impacto, reducción, supervisión y gestión. Planes de contingencia. El plan de RSGR.

- Gestión de la configuración del software: Línea base, gestión del cambio, control de versiones, auditoría.

- Nociones de Sistemas Colaborativos.

2. Diseño

- Conceptos. Abstracción, arquitectura, patrones, modularidad, ocultamiento de la información, independencia funcional, cohesión, acoplamiento, refinamiento.

- El modelo de diseño: diseño de datos, diseño arquitectónico, diseño de interfaz, diseño al nivel de componentes.

- Diseño Arquitectónico.

  - Organización del sistema: modelo de repositorio, modelo cliente-servidor, modelo de capas. Arquitecturas de Sistemas Distribuidos: multiprocesador, c-s, objetos distribuidos, interorganizacional (peer-to-peer, sistemas orientados a servicios).

  - Descomposición modular: orientada a objetos, orientada a flujos de funciones.

  - Control: centralizado, dirigido por eventos.

- Diseño de interfaces de usuario: interacción del usuario, presentación de la información, análisis del usuario, prototipo de la interfaz, evaluación de la interfaz.

- Diseño a nivel de componentes: notaciones gráficas, notaciones tabulares, lenguajes de diseño.

- Características de un bien diseño. Técnicas para la mejora del diseño. Evaluación y validación del diseño. Documentando el diseño.

3. Implementación

- Estándares de programación y procedimientos
- Pautas para la programación
- Documentación

4. Verificación y Validación

- Técnicas de Prueba
  - Pruebas de Caja blanca: camino básico, bucles.
  - Pruebas de Caja negra: partición equivalente, análisis de valores límites.
- Estrategias de Prueba
  - Defectos y fallas. Planificación. Diseño de casos de prueba. Resultados. Documentación de las pruebas. Automatización.
  - Pruebas de unidad (arquitecturas convencionales y arquitecturas orientadas a objetos)
  - Pruebas de integración (arquitecturas convencionales y arquitecturas orientadas a objetos)
  - Pruebas de validación: alfa y beta.
  - Pruebas del sistema: de recuperación, de seguridad, de resistencia, de desempeño.
  - Pruebas de regresión.
  - La depuración: proceso, estrategia, corrección del error.

5. Entrega

- Entrenamiento
- Documentación

6. Mantenimiento

- Evolución del software. Tipos de mantenimiento: correctivo, adaptativo, perfectivo, preventivo.
- Sistemas heredados.
- Métricas, técnicas y herramientas para el mantenimiento.
- Rejuvenecimiento del software: redocumentación, reestructuración, ingeniería inversa, reingeniería.

7. Auditoría y Peritaje

- Conceptos
- Objetivos
- Planeamiento de Auditoría

Metodología de enseñanza

La materia se dicta mediante clases teóricas, explicaciones de práctica y desarrollo de trabajos prácticos.
Las clases teóricas consisten en una exposición y desarrollo de conceptos centrales.
Las explicaciones de práctica apuntan a brindar las herramientas necesarias para la posterior realización de los ejercicios.
Las actividades prácticas comprenden el desarrollo de un proyecto de software definido por la asignatura, contemplando todas las etapas de desarrollo.

Propuesta de evaluación

La aprobación de la cursada consiste en un coloquio en el cuál se defiende el trabajo realizado en las clases prácticas. Los alumnos se estructuran en grupos.
La aprobación de la asignatura consiste en un examen final escrito.
Como instancias previas de evaluación los alumnos pueden rendir una serie de pruebas teóricas, las cuáles en el caso de ser aprobadas reducen la cantidad de temas tomados en el examen final.
Todos los alumnos tienen un seguimiento continuo utilizando WebUNLP.

Bibliografía

- ✓ Ingeniería de Software. 7ma Edición. Ian Sommerville. Pearson - Addison Wesley. 2005

- ✓ Ingeniería de Software. Teoría y Práctica. Shari Pfleeger. Pearson Education. 2002.
- ✓ Ingeniería de Software. Un enfoque práctico. Roger Pressman. McGraw Hill. 2006.
- ✓ Auditoria en Informatica 2da Edicion. Jose Antonio Echenique Garcia. Mc Graw Hill