UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

INGENIERIA DE SOFTWARE III

Carrera: ***Licenciatura en Sistemas***
Planes 2003 y 2007

Año 2010

Año: **4°**
Duración: ***Semestral***
Profesor: **Dr*. Gregorio Perichinsky***
Hs. semanales: **6 hs.**

OBJETIVOS GENERALES:

Completar el ciclo de temas básicos de la Ingeniería de Software dados en los dos cursos anteriores, con el estudio de aspectos de evaluación y seguimiento de proyectos, control de calidad de sistemas de software y auditoría de los mismos.
Desarrollar el estudio de casos concretos.

CONTENIDOS MINIMOS:

- Calidad de software.
- Auditoría y peritaje de sistemas.
- Costeo.
- Seguimiento y evaluación de proyectos.

Programa

Administración de Proyectos
Factores críticos en la conducción de proyectos. Planificación. El concepto de Ciclo de Vida. Modelos de Cascada, Espiral, Prototipación, Especificación Funcional, Transformativo, Basado en Conocimiento, Análisis de Dominio. Mapeo (Mapping). Ambiente del Proyecto. El equipo de proyecto. Organización. Los equipos de proyecto. Formación, criterios y perfiles. Compromisos. Revisión. Puntos de revisión. Concepto de hito. Medición. Factores a medir, unidades de medida. Valor de la medición. Historia. Estimación. Puntos Funcionales. Líneas de código. Modelos de costo. Conceptos.

Modelos de Sistemas y su relación con el ciclo de vida del software.
Monitoreo y Control. Riesgos y Planificación. Paradigma de partición por eventos. Métodos y herramientas de análisis, diseño, codificación y prueba. Análisis Esencial. Modelo de análisis. Paradigma de objetos. Ingeniería de Software. Modelos.

El proceso de desarrollo de software.
Pre-desarrollo. Guiar el estudio de alternativas. Planificar las transiciones. Refinar ideas y requerimientos. Ingeniería de requerimientos. Pre - factibilidad y factibilidad.

Características del producto y del proceso. Procesos de desarrollo. Requerimientos. Diseño. Implementación. Post – desarrollo. Instalación. Operación y Soporte. Mantenimiento. Retiro.

Garantía de calidad del software (SQA). Actividades del SQA.

Control de calidad. Costo de calidad. Costos de prevención. Planificación de la calidad. Revisiones técnicas normales. Equipo de prueba. Formación. Costos de evaluación. Costos de fallos. Revisión. Reparación. Análisis de las modalidades de fallos. Costos de fallos externos. Revolución de situación de productos. Soporte de línea de ayuda. Trabajo de garantía. Técnicas de detección de errores. Dinámica. Prueba (Testing). Estática de las Revisiones. Identificar defectos. Relación con la variación entre los resultados obtenidos y los esperados. El rol del SQA y aspectos de la prueba (testing) y el presupuesto. Revisiones técnicas formales que se aplican durante cada paso de la ingeniería del soft. Estrategia de prueba. Control de la documentación del soft. Procedimiento de ajuste a los estándares. Mecanismos de medida y de información.

Recorridos e Inspecciones de software.

Establecimiento de un plan SQA para un proyecto Evaluaciones a realizar. Auditorias y revisiones y realizar. Estándares que se pueden aplicar al proyecto. Procedimientos para información y seguimiento de errores. Documentos producidos para el grupo SQA. Realimentación de información proporcionada al equipo de proyecto del software. Revisiones técnicas formales (RTF). Descubrir errores en la función. Verificar que el software bajo su revisión alcanza sus requisitos. PFR: Factibilidad del producto. SRR: Requisitos del soft. PDR: Diseño preliminar. CDR: Diseño crítico. ATR: Prueba de aceptación. PRR:Revisión final. Enfoques formales a la SQA.

Control de Calidad.

Calidad Total. Factores que determinan la calidad del software. Evolución Histórica. Situación Actual. Ventajas. Problemas. Certificaciones. Procesos Integrales. Verificación y Validación. (V&V). Composición del Software. Directrices de Calidad. Asegurar la Calidad y la Confiabilidad del Software. (SQA). Documentación del desarrollo. Diccionario de Recursos de Información. Repositorios. Modelo de Capacidad y Madurez (CMM). Niveles. Otros Modelos CMM. CMMI. Críticas y relación con las Normas ISO. Conclusiones. SPICE. Situación en el mundo. Proceso de entrenamiento. Responsabilidades. Revisiones. Prueba. Conceptos. Estructura. Áreas Claves. Claves prácticas.

Nivel Inicial.
Nivel repetible.
Nivel definido.
Nivel de manejo (Managed).
Nivel óptimo.

Metodología de enseñanza

Las clases teóricas se desarrollan a través de la explicación de los contenidos mínimos. En ambas instancias, tanto teórica como práctica, los alumnos cumplen con las etapas epistemológicas del conocimiento científico, primero con la detección y planteo de problemas a partir de las hipótesis desarrolladas en la teoría, análisis documental y bibliográfico para determinar el marco teórico, procedimientos deductivos a partir del análisis documental, consecuencias contrastables en la "práctica", modelo experimental y experiencia con validación y verificación del qué y el cómo para determinar errores y falsedades y realizar un back tracking. De ser necesario plantear una nueva hipótesis en un modelo iterativo.
Esto se materializa a través de una monografía realizada por los alumnos en donde esté el problema, el marco teórico y las conclusiones. Puede ser individual o grupal y las correcciones son sucesivas e interactivas.

Propuesta de evaluación

La evaluación de la cursada se realiza a través del desarrollo de proyectos individuales y/o grupales, con revisiones en la práctica, y se realizan dos parciales de desarrollo conceptual.
Para la aprobación final de la materia deben entregar y defender el trabajo final que comenzaron a desarrollar durante la cursada, en donde toman tópicos de la teoría para armar el marco teórico con el que van a resolver el problema planteado.

Bibliografía:

- Boehm, Barry W., Software Engineering Economics, Prentice Hall, 1981.
- Brooks, Frederick P., The Mythical Man-Month, Addison Wesley, l995.
- Date, C. J., Introducción a los Sistemas de Bases de Datos l986.
- De Marco, Tom, Controlling Software Projects: Management, Measurement, Prentice Hall, l982.
- Flavin, Matt, Fundamental Concepts of Information Modelling, l981.
- Humphrey, Watts S., Managing the Software Process. Addison-Wesley Massachusetts, 1989.
- Ivar Jacobson Object-Oriented Software Engineering. Addison-Wesley. 1994.
- Mac Menamin & Palmer, Essential System Analysis, 1984.
- Meyer, Bertrand, Object-oriented software construction, Prentice Hall, 1988.
- Myers Glenford J., The Art of Software Testing, Wiley, 1979.
- Page-Jones, Meilir, The Practical Guide To Structured Desing, 1988.
- Managing code inspection information, Jack Barnard, ART Price AT&T Bell Laboratories, 1992

- Software Inspection. An industry Best Practice, Wheeler, DA, Brykezyski, B, Meeson, RN, IEEE Computer Society Press, 1996
- Jones, T.C., Programming Productivity, McGraw-Hill, 1986
- "Implementating Software Inspections", Notas del curso, IBM Systems Sciences Institute, IBM Corporation, 1981
- Handbook of walkthroughs, Inspections and Technical Reviews, 3° edición, Freeman, D.P., Weimberg, G.M., Dorset House,1990
- Software Requirements, Alan M. Davis, Prentice Hall, 1993
- El Lenguaje Unificado de Modelado, G. Booch, J.Rumbaugh, I.Jacobson – Addison Wesley, 1999
- Ingeniería de Software, un enfoque práctico 5° edición – Roger S.. Pressman – Mc Graw ,Hill, 1998
- Software Testing Technics, Van Nostrand Reinholds, 1990.
- Fundamental of software Enginnering, Ghezzi et al., Prentice Hall, 1992.
- Gonzalo Cuevas Agustín: Una Guía del CMM. Para Comprender el Modelo de Madurez de Capacidad del Software. Traducción del Inglés "A Guide to the CMM" de Kenneth M. Dymond. 1998.
- Constantine, L. and Lockwood, L. "Software for use". A Practical Guide to the Models and Methods of Usage - Centred Design. Addison - Wesley. 1999.
- Ghezzi, C., Jazayeri, M., Mandrioli, D. "Fundamentals of Software Engineering". Prentice Hall. 1991.
- Grimaldi, Ralph P. “Matemáticas discreta y combinatoria. Introducción y aplicaciones”. Addison-Wesley Iberoamericana, 1989.
- IEEE STD 1074-1991 Customer Request IEEE Standard fo developing Software Life Cycles Process, Identify Ideas or Needs, Preliminary Statement of Need, Feasibility Studies, Statement of need.
- Jacobson, Ivar; Booch, Grady; Rumbaugh, James. “The Unified Software Development Process”. Addison-Wesley ISE. 1999.
- Pfleeger, S. "Software Engineering: Theory and Practice". Prentice Hall. 1998.
- Sommerville, I. "Software Engineering". Addison - Wesley. 1996.
- Thomas, I. & Nejmeh, B. A., 1992, Definitions of Tool Integration for Environments, IEEE Software, 9, 2 (Mar), 29-35.
- Fenton, N.E., Pfleeger, Sh.L. “Software Metrics”. PWS Publishing Company. 1997.
- Kitchenham,B., Pickard,L., Pfleeger, S.L. “Case studies for method and tool evaluation”. IEEE Software, 12(4) pp 52-62. 1995.

- SEI - Software Engineering Institute
- ESI - European Software Institute
- System Security Engineering CMM
- Finkelstein's Capability Immaturity Model paper (PDF file)
- Tom Schorsch's Capability Immaturity Model study

- Desarrollo, notas y apuntes de la Teoría de la cátedra.