UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Departamento Ingeniería en Sistemas de Información

ASIGNATURA:	TECNICAS AVANZADAS DE PROGRAMACIÓN
DEPARTAMENTO:	ING. EN SIST. DE INFORMACION
AREA:	ELECTIVA
BLOQUE	TECNOLOGÍAS APLICADAS

MODALIDAD:	Cuatrimestral
HORAS SEM.:	4 horas
HORAS/AÑO:	64 horas
HORAS RELOJ	48
NIVEL:	3º
AÑO DE DICTADO:	Plan 2008

Objetivos

- Seleccionar herramientas conceptuales y estrategias de programación orientadas a objetos, a partir de la identificación de sus ventajas y desventajas en el contexto específico de utilización para resolver problemas de desarrollo de software de complejidad creciente
- Acercar a los alumnos a la complejidad real de los sistemas, donde el software construido debe adaptarse a las interacciones con un entorno que presente restricciones o que no maneje las mismas abstracciones conceptuales del software en construcción.
- Combinar, extender y/o modificar las herramientas conocidas para su adaptación a problemas donde la implementación “de libro” no resulte adecuada.
- Comprender como las tecnologías de implementación afectan al proceso de construcción e incorporar herramientas que permitan mantener una estrategia conceptual en ese contexto.
- Incorporar buenas prácticas de desarrollo que simplifiquen la construcción de software comercial de alta complejidad.
- Adquirir una visión técnica de más alto nivel para, desde un rol de arquitecto o gerente comprender las prácticas en las que se basan los proyectos desarrollados y/o sistemas utilizados en el área de su incumbencia.
- Vincular la programación con las demás áreas de incumbencia del profesional de sistemas para poder basar sus decisiones gerenciales en las cuestiones técnicas subyacentes.

Contenidos Mínimos (Programa Sintético).

- Conceptos avanzados de programación orientada a objetos.
- Técnicas de desarrollo iterativo. Introducción a metodologías ágiles.

- Relación Tecnología/Diseño. Herramientas de testing y manejo de errores.
- Herramientas y buenas prácticas para el modelado de problemas complejos. Criterios para la toma de decisiones entre varias soluciones posibles.
- Definición de interfaces entre los módulos de un sistema

Contenidos Analíticos:

UNIDAD 1: Revisión de conceptos OO

Polimorfismo tipado. Comparación con polimorfismo no tipado. Contratos fuertes y débiles. Interface vs. Clase abstracta. Binding estático y dinámico. Modelado con clases/instancias. Separación de los momentos de instanciación y uso de objetos. Definición e implementación de interfaces entre componentes funcionales.

UNIDAD 2: Herramientas tecnológicas

Implementando un diseño. Vinculación diseño-código. Codificación de casos de prueba. Herramientas de testeo unitario. Introducción a TDD. Manejo de errores. Domain Driven Design.

UNIDAD 3: Patrones y buenas prácticas

Implementación y adaptación de patrones de diseño. Patrones GoF. Double dispatch. Type object. Wrappers. Inversion of Control. Framework vs. biblioteca. Best practices: “Once and only once”, “Tell, don’t ask”, “Program to an interface not to an implementation”, “Favor object composition over class inheritance”, “Make it work, make it right, make it fast”.

UNIDAD 4: Aspectos metodológicos

Elección e implementación de un modelo de ciclo de vida de desarrollo de software. Prácticas de diseño y programación iterativas. Técnicas de refactorización. Cualidades del software. Software configuration management.

UNIDAD 5: Extensiones al paradigma: declaratividad y metaprogramación

Elementos declarativos en la construcción de software. Metadatos. XML. Annotations. Introducción a la programación reflexiva. Modelo y metamodelo.

Bibliografía.

- Design Patterns: Elements of Reusable Object-Oriented Software – Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. – (1995)
- The Design Patterns Smalltalk Companion – Sherman Alpert, Kyle Brown, Bobby Woolf. (1998)
- Design Patterns in Java - Steven John Metsker, William C.Wake. (2006)
- Object Design: Roles, Responsibilities, and Collaborations – Rebecca Wirfs-Brock, Alan McKean. (2002)
- Extreme Programming Explained: Embrace Change – Kent Beck. (2000)
- The pragmatic programmer: from journeyman to master – Andrew Hunt, David Thomas. (1999)

Correlativas

Para Cursar:

Cursadas:

- Análisis de Sistemas

Aprobada

- Paradigmas de Programación

Para rendir:

Aprobadas:

- Análisis de Sistemas