



UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES
PROGRAMA DE INGENIERIA DE SISTEMAS

ASIGNATURA:	INGENIERIA DEL SOFTWARE II
CODIGO:	SIS602
MODALIDAD:	PRESENCIAL TEORICO
INTENSIDAD:	4 HORAS TEORICAS / SEMANA.
PREREQUISITOS:	INGENIERIA DEL SOFTWARE I Y LABORATORIO DE INGENIERIA DEL SOFTWARE I,
CO-REQUISITOS:	LABORATORIO DE INGENIERIA DEL SOFTWARE II
AREA:	INGENIERIA APLICADA
CREDITOS:	3

OBJETIVO GENERAL

Proporcionar al estudiante el espacio de conocimiento, conceptualización e integración para que desarrolle habilidades ingenieriles avanzadas relativas a la construcción de software orientado a objetos de calidad, dentro del marco de trabajo de un proceso de desarrollo guiado por casos de uso, iterativo e incremental y centrado en la arquitectura, abordando y dando solución a los problemas típicos que se presentan en el desarrollo de software.

OBJETIVOS ESPECIFICOS

1. Construir software orientado a objetos basado en los modelos de análisis y diseño realizados en notación UML y patrones de diseño existentes.
2. Aplicar principios fundamentados en buenas prácticas para el diseño de Frameworks orientados a objetos.
3. Aplicar diversas estrategias de prueba para la validación de calidad del software orientado a objetos.

METODOLOGIA

1. El alumno adquirirá los conocimientos básicos a través de clases magistrales.
2. El alumno desarrollará talleres en clase que le ayudarán a llevar a la práctica los conocimientos teóricos adquiridos.
3. El alumno deberá profundizar sus conocimientos con temas complementarios y trabajos de investigación.

CONTENIDO

CAPÍTULO 1. MAPEO DE LOS DISEÑOS A CÓDIGO ORIENTADO A OBJETOS

- 1. Implementando los métodos de funcionalidad mínima en las Clases.**
 - a. Métodos Básicos: Constructor, Destructor, Get, Set y Compare.
 - b. Métodos Asociativos: Associate_unObj() y unAssociate_unObj().
- 2. Técnicas para el Mapeo a Código del Diagrama de Clases.**
 - a. Mapeando los Atributos Básicos y Asociativos.
 - b. Mapeando Operaciones Básicas, Asociativas y del Dominio.
- 3. Técnicas para el Mapeo a Código de los Diagramas de Colaboración.**
 - a. Asignando Operaciones a las clases a partir de los mensajes.
 - b. Escribiendo el código de las operaciones a partir de los mensajes.
- 4. Estándares de Documentación para Código Orientado a Objetos.**

CAPITULO 2. Arquitecturas de Software

- 1. Definición**
- 2. Diseño de Arquitecturas**
 - a. Ciclo de Influencias
 - b. Lenguajes de Descripción de Arquitecturas
 - c. Captura de requisitos Arquitecturales: QAW (Quality Attribute Workshop)
 - d. Diseño de la Arquitectura: ADD (Attribute Driven Design)
 - e. Patrones Arquitectónicos
3. Evaluación de Arquitecturas
4. Reconstrucción de Arquitecturas de Software
5. Arquitecturas Reflexivas

CAPITULO 3. DISEÑO DE UN FRAMEWORK DE PERSISTENCIA

- 1. Introducción**
 - a. El Problema: Objetos Persistentes y sus Mecanismos de Almacenamiento.
 - b. La Solución: Esquema de Persistencia, Requerimientos e Ideas Básicas.
- 2. Patrones Asociados al Diseño del Framework.**
 - a. Patrón Representación de Objetos como Tablas.
 - b. Patrón Identificador de Objeto.
 - c. Patrón Intermediario (Broker) de Bases de Datos.
 - d. Patrón Método Plantilla y la Materialización de Objetos.
 - e. Patrón Administración del Caché para administración de Transacciones.
 - f. Patrón Agente (Proxy) Virtual y Puente (Referencias Inteligentes).
 - g. Patrón Factory Method para conexión entre Proxies y Brokers.
 - h. Patrón Instanciación de Objetos Complejos para Materialización por Demanda.
- 3. El caché y el control de las transacciones.**
- 4. Cuestiones sin Resolver.**

CAPITULO 4. PATRONES DE DISEÑO ORIENTADO A OBJETOS

1. Motivación

- Que es un patrón de diseño?
- Como se especifica un patrón de diseño?
- Como resuelven los patrones los problemas de diseño?
- Como seleccionar un patrón de diseño?
- Como usar un patrón de diseño?

2. Patrones GRASP

- Creador, Controlador, Alta Cohesión, Bajo Acoplamiento, Experto.
- Polimorfismo, Fabricación Pura, Indirección, No Hables con Extraños.

3. Patrones GOF

a. Patrones de Creación

- Abstract Factory, Builder, Factory Method, Prototype, Singleton.
- Discusión sobre los patrones de Creación.

b. Patrones Estructurales

- Adapter, Bridge, Composite, Decorator, Facade, FlyWeight,
- Discusión sobre los patrones Estructurales.

c. Patrones de Comportamiento

- Chain, Command, Interpreter, Iterator, Mediator, Memento, Observer.
- State, Strategy, Template Method, Visitor.
- Discusión sobre los patrones de Comportamiento.

CAPITULO 5. PRUEBAS DE SOFTWARE

1. Motivación: El Porque de las Pruebas?

2. Fase de Pruebas.

- Actividades.
- Artefactos.
- Trabajadores.

3. Estrategias de Pruebas

- Pruebas de Integridad de Datos.
- Pruebas de Funcionalidad.
- Pruebas de la Interfaz de Usuario.
- Pruebas de Desempeño.
- Pruebas de Carga.
- Pruebas de Stress.
- Pruebas de Volumen.
- Pruebas de Seguridad y Control de Acceso.
- Pruebas de Caída y Recuperación.
- Pruebas de Configuración.
- Pruebas de Instalación.

EVALUACIONES

Se realizarán tres (3) evaluaciones de la siguiente forma:

NUMERO	%	COMPONENTES
Primer Parcial	35%	Parcial Escrito 70%
		Quices, Talleres 30%
Segundo Parcial	35%	Parcial Escrito 70%
		Quices, Talleres 30%
Tercer Parcial	30%	Parcial Escrito 70%
		Quices, Talleres 30%

Los laboratorios en grupo serán evaluados individualmente y deben estar debidamente documentados.

BIBLIOGRAFÍA

- Pressman, Roger. Ingeniería del Software, Un Enfoque Práctico. McGraw-Hill. 1998.
- Craig, Larman. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Prentice Hall. 1999.
- UML Ed. Addison Wesley
- <http://www.devx.com/>
- <http://www.sei.cmu.edu/>
- Ivar Jacobson, Grady Booch y James, Rumbaugh. The Unified Software Development Process. Rational Software Corporation. Addison Wesley, 1999. ISBN:0-201-57169-2. 463 Págs.

LECTURAS COMPLEMENTARIAS

- M. Fowler, Diagramas de clase: conceptos avanzados, Addison Wesley Longman, *UML gota a gota*, (México: Estado de México, 1999).
- M. Fowler, Los casos de uso, Addison Wesley Longman, *UML gota a gota*, (México: Estado de México, 1999).
- Watts S. Managing the Software Process. Humphrey. Addison-Wesley. 1989.
- Jones, Capers. Applied Software Measurement: Assuring Productivity and Quality. McGraw-Hill. 1991.
- Watts S. Humphrey. Introduction to the Team Software Process.