NOMBRE DE LA INSTITUCIÓN	Universidad de Sonora
DIVISIÓN ACADÉMICA	División Ciencias Exactas y Naturales
DEPARTAMENTO QUE IMPARTE LA MATERIA	Departamento de Matemáticas
LICENCIATURAS USUARIAS	Ciencias de la Computación
NOMBRE DE LA MATERIA	**Análisis y Diseño Orientado a Objetos**
CLAVE	**9460**
EJE FORMATIVO	Profesional
REQUISITOS	Ingeniería de Software II, Bases de Datos I
CARÁCTER	Obligatoria
VALOR EN CRÉDITOS	8 (3 hr. teoría/2 hr. laboratorio)

Introducción

La invención de lenguajes orientados a objetos y el desarrollo de metodologías y técnicas con este mismo enfoque promueven la evolución de toda un área dentro de la ingeniería de software. La meta del paradigma orientado a objetos es crear software reduciendo tiempos de entrega, facilitar la incorporación del cambio en un producto de software y reutilizar componentes de una biblioteca conformada por éstos.

En la actualidad, la mayoría de los lenguajes de programación de alto nivel que se utilizan en la construcción de nuevas aplicaciones son orientados a objetos o basados en objetos (por ejemplo: C++, Visual C++, Delphi, y Java). Los ambientes de desarrollo de software y los generadores de interfaces hombre-máquina tienen como principio la orientación a objetos para desarrollar nuevos programas. Además, desde finales de la década de los 80, han evolucionado metodologías y técnicas que utilizan el enfoque orientado a objetos para analizar el dominio del problema y el dominio de solución. Con este enfoque se logra la simplificación de técnicas de notación y definiciones de herramientas de modelado y diagramación, ya que se aplican los mismos principios teóricos a todas las etapas del ciclo vital del desarrollo de un sistema o producto de software.

En este curso el alumno comprenderá los fundamentos teóricos que sustentan al paradigma orientado a objetos y podrá aplicarlos en el desarrollo de un producto de software. Además, aprenderá a utilizar el lenguaje de modelado unificado para construir distintas vistas del producto, según la fase del proceso en que se encuentre trabajando. Además, el alumno requiere conocer y desarrollar las habilidades de comunicación y coordinación que le permitan integrarse y colaborar eficazmente en un equipo de desarrolladores de software utilizando técnicas orientadas a objetos.

Objetivo General del Curso

El alumno será capaz de comprender los conceptos que fundamentan el paradigma orientado a objetos y podrá aplicarlos en un proyecto de desarrollo de software para un cliente real, en donde el alumno seleccione y adapte, junto con sus compañeros, las técnicas y notación orientada a objetos para especificar el dominio del problema y el dominio de solución, de acuerdo al nivel de calidad requerido, tiempo especificado, y funcionalidad solicitada por el cliente y entregue un producto de software que satisfaga los criterios establecidos.

Objetivos Específicos del Curso

- Comprender la naturaleza del software que se construye en la actualidad y la problemática asociada a éste.
- Comprender los elementos que fundamentan el paradigma orientado a objetos y cómo se pueden reconocer en un lenguaje de programación y en la construcción de una aplicación de software.
- Establecer con claridad el ámbito del software en el proyecto.

- Realizar la planificación del proyecto fundamentada en las necesidades de procesamiento de información de un cliente real.
- Adaptar el proceso de desarrollo de software de acuerdo a la cantidad de desarrolladores de software, recursos disponibles y complejidad del producto solicitado por el cliente.
- Establecer y aplicar los lineamientos de aseguramiento de la calidad a cada uno de los productos generados durante el proceso de desarrollo del software.
- Aplicar técnicas orientadas a objetos para obtener cada una de las vistas de la arquitectura del sistema de software.
- Verificar y validar la calidad del producto entregado al cliente.
- Organizar y distribuir las tareas entre los integrantes del equipo de desarrolladores.
- Establecer criterios específicos para evaluar el avance el proyecto y favorecer la comunicación y coordinación entre todos los miembros del equipo.

Contenido

1. **Panorama general**
    - 1.1. Complejidad inherente al software.
    - 1.2. Estructura de los sistemas complejos
    - 1.3. Diseño de sistemas complejos
    - 1.4. Elementos del modelo de objetos

2. **Modelado de objetos**
    - 2.1. Importancia del modelado.
    - 2.2. Principios de modelado.
    - 2.3. Modelado orientado a objetos.
    - 2.4. Introducción al UML.

3. **Proceso de desarrollo de software**
    - 3.1. Modelos de desarrollo de software.
    - 3.2. Proceso unificado de desarrollo de software.
    - 3.3. Proyecto de desarrollo de software.

4. **Análisis y diseño orientado a objetos**
    - 4.1. Captura de requisitos.
    - 4.2. Modelado del dominio.
        - 4.2.1. Casos de uso.
        - 4.2.2. Clases.
    - 4.3. Modelado de diseño
        - 4.3.1. Clases de diseño.
        - 4.3.2. Subsistemas
        - 4.3.3. Interfaces.
        - 4.3.4. Arquitectura.
5. **Implementación y pruebas**
    - 5.1. Papel de la implementación.
    - 5.2. Modelo de la implementación.
    - 5.3. Modelo de pruebas.
6. **Metodologías orientadas a objetos**
    - 6.1. Object Modeling Technique (OMT).
    - 6.2. Booch.
    - 6.3. Jacobson (OOSE).
    - 6.4. Shlaer/Mellor (OOSAA/OOL).
    - 6.5. Coad-Yourdon (OOA/OOD).
    - 6.6. ICONIX.
    - 6.7. Criterios de selección.
    - 6.8. Herramientas CASE
7. **Aplicaciones**

Estrategias Didácticas

- Propiciar la vinculación del alumno con el sector productivo al solicitarle que construya un sistema de software que sea utilizado por un cliente real.
- Promover la colaboración de los alumnos al permitirle la elección de distintas responsabilidades de trabajo, de acuerdo a las diversas tareas involucradas en el proyecto.
- Promover la participación oral y escrita de todos los alumnos al discutir el ámbito de los proyectos que aborden y especificar los requisitos de usuario en un documento.
- Implementar diversas técnicas de revisión, formales e informales, para evaluar el avance del proyecto y la calidad de los productos generados.
- Promover la investigación bibliográfica y en línea de información necesaria para comprender el dominio del problema abordado en el proyecto por el alumno así como de aspectos técnicos relacionado con la construcción del sistema de software a entregar.

Estrategias de Evaluación

Para la evaluación de los estudiantes, el instructor tomará en cuenta:

- Resultados de los exámenes parciales aplicados (de acuerdo a la teoría abordada, el maestro decidirá la cantidad de éstos).
- La calidad de los productos generados en el proceso de desarrollo de software y que se entreguen de acuerdo al plan de desarrollo establecido.
- participación individual y colectiva en las revisiones.
- Proyecto completo terminado y aprobado por el cliente.

Los porcentajes de cada uno de los indicadores serán acordados al inicio del semestre.

Bibliografía

La bibliografía básica del área de ingeniería del software es la siguiente:

- Sommerville, Ian. Ingeniería de software. 6ta. ed. Pearson Educación, México, 2002.
- Pressman, Roger. Ingeniería del software: Un enfoque práctico. 5ta. ed. McGraw Hill, Madrid, 2002.
- Jacobson, I., Booch, G. y Rumbaugh, J. El proceso unificado de desarrollo de software. Pearson Educación, Madrid, 2000.

Bibliografía específica a temas del enfoque orientado a objetos:

- Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Prentice Hall, México, 1999.
- Fowler, Martin. UML gota a gota. Addison Wesley Longman de México, México, 1999.
- Sintes, Anthony. Aprendiendo programación orientada a objetos en 21 lecciones avanzadas. Pearson Educación, México, 2002.
- Jacobson, I., Booch, G. y Rumbaugh, J. El proceso unificado de desarrollo de software. Pearson Educación, Madrid, 2000.
- Booch, G. Análisis y diseño orientado a objetos con aplicaciones. 2da ed. Addison-Wesley Iberoamericana, México, 1996.
- Rosenberg, Doug y Scott, Kendall. Use Case Driven Obejct Modeling with UML: A practical approach. Addison Wesley, Boston, 1999.
- Rosenberg, Doug y Scott, Kendall. Appying Use Case Driven Object Modeling with UML: An annotated e-commerce Example. Addison Wesley, Boston, 2001.
- Booch, Grady, Rumbaugh, J. y Jacobson, I. The Unified Modeling Language User Guide. Addison Wesley, 1998.

Además, se sugiere que se consulten revistas especializadas del área:

- ACM Journals.
- IEEE Journals.

Perfil académico deseable del maestro

Se recomienda que el profesor tenga las siguientes características:

- Formación sólida en ciencias de la computación o área afín y se haya especializado en ingeniería de software.
- Tenga conocimientos del proceso unificado y del lenguaje de modelado unificado.