UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

COORDINACIÓN DE FORMACIÓN BÁSICA
COORDINACIÓN DE FORMACIÓN PROFESINAL Y VINCULACIÓN UNIVERSITARIA
PROGRAMA DE UNIDADES DE APRENDIZAJE POR COMPETENCIAS

I. DATOS DE IDENTIFICACIÓN

1. Unidad Académica: **Facultad de ciencias**

2. Programa (s) de estudio: (Técnico, Licenciatura) **Licenciatura en Ciencias Computacionales** 3. Vigencia del plan: **2008-1**

4. Nombre de la Unidad de Aprendizaje**:** **Ingeniería de Software** 5. Clave: 9833

6. HC: **2** HL **4** HT: HPC: HCL: CR: **8**

7. Ciclo Escolar: 8. Etapa de formación a la que pertenece: **Disciplinaria**

9. Carácter de la Unidad de Aprendizaje: **Obligatoria X** Optativa

10. Requisitos para cursar la Unidad de Aprendizaje: **Se recomienda tomar las materias obligatorias de Metodología de la Programación, Programación Orientada a Objetos, Introducción a la Programación. Es deseable un buen dominio del idioma inglés.**

Formuló: Dr. Alberto Leopoldo Morán y Solares

VoBo. Marcelo Rodríguez Meraz
Cargo: Subdirector

II. PROPÓSITO GENERAL DE LA UNIDAD DE APRENDIZAJE

Al término del la unidad de aprendizaje el estudiante comprenderá las nociones fundamentales de los métodos, técnicas y herramientas actuales que encierra la Ingeniería de Software, así como las diferentes actividades que se realizaran para la producción de software de calidad. Considerando los aspectos relacionados con la Calidad y la Gestión de Configuraciones en el desarrollo de software y las implicaciones sociales como el costo de las fallas y las responsabilidades profesionales. Además, el estudiante será capaz de aplicar estos conocimientos para resolver situaciones y/o problemas reales a través de soluciones de software.

La unidad de aprendizaje de Ingeniería de Software es una unidad obligatoria y pertenece a la etapa disciplinaria. Las unidades de aprendizaje consecuentes relacionadas con ésta son Metodología de la Programación, Programación Orientada a Objetos e Introducción a la Programación (todas obligatorias), y otras unidades optativas.

III. COMPETENCIA (S) DE LA UNIDAD DE APRENDIZAJE

Analizar, diseñar, e implementar productos de software con calidad como solución a un problema real, eligiendo las normas de calidad, métodos y tecnologías que permitan especificar, diseñar y generar el código, comprometiéndose con su equipo trabajo.

IV. EVIDENCIA (S) DE DESEMPEÑO

Se desarrollará y presentará un producto de software terminado resultado de un proyecto final. Además de exposición oral y escrita de artefactos generados durante la fase de desarrollo. Se aplicarán exámenes de la teoría presentada.

V. DESARROLLO POR UNIDADES

Competencia:
Comprender la importancia del desarrollo de software y la ingeniería del software, la terminología del área, así como los dominios de aplicación de la misma en las organizaciones.

Contenido temático **Duración: 8 horas**

1. **Introducción a la Ingeniería de Software (IS)**
   1.1. ¿Qué es la Ingeniería de software?
   1.2. Historia de la IS
   1.3. Naturaleza y cualidades del software
   1.4. Administración de proyectos
   1.5. Plan de administración de proyecto
   1.6. Herramientas y documentación en IS

Competencia:
Conocer y comprender los conceptos generales y los procesos formales que se utilizan para el desarrollo de software, así como los principales tipos y modelos de desarrollo y administración del proceso, para su posterior aplicación en el análisis y diseño de software.

Contenido temático **Duración: 12 horas**

2. Procesos formales de desarrollo de Software

2.1. Proceso de Software
2.2. Modelo de ciclo de vida del software
2.3. Conceptos de Modelado Orientado a Objetos y Patrones
2.4. Diagramas de UML
2.5. Modelado del Proceso de Software con UML
2.6. Modos y mecanismos de comunicación del equipo de desarrollo
2.7. Modelos de referencia

Competencia:
Comprender y comprender los procesos de elicitación y administración de requerimientos, para poder realizar de manera exitosa del proceso de especificación de software de forma profesional y congruente con los procesos de la organización que demanda los servicios.

Contenido temático **Duración: 12 horas**

3. Análisis de requerimientos y especificación

3.1. Requerimientos vs especificación
3.2. Conceptos de la obtención de requerimientos
3.3. Actividades para la obtención de requerimientos
3.4. Administración de requerimientos

Competencia:
Comprender y aplicar los diferentes conceptos, mecanismos y herramientas para poder realizar un análisis orientado a objetos de la problemática objetivo de una manera profesional y ética.

Contenido temático **Duración: 16 horas**

4. Análisis Orientado a Objetos (AOO)

4.1. Refinamiento del modelo de casos de uso
4.2. Conceptos de análisis
4.3. Actividades de AOO
4.4. Administración del análisis

Competencia:
Comprender y aplicar los conceptos, mecanismos y herramientas del diseño orientado a objetos para poder hacer un corrimiento del dominio de la aplicación hacia el dominio de la solución por software, de manera profesional y objetiva, y optimizando las soluciones de acuerdo a las necesidades y requerimientos de la organización que demanda el servicio.

Contenido temático **Duración: 16 horas**

5. Diseño Orientado a Objetos (DOO)
- 5.1. Conceptos del DOO
- 5.2. Actividades del DOO para obtener el modelo completo
- 5.3. Administración del diseño del sistema
- 5.4. Modelo de diseño de objetos
- 5.5. Diseño de patrones
- 5.6. Programación Orientada a Objeto (POO)

Competencia:
Comprender y aplicar los conceptos métodos y herramientas de las pruebas orientadas a objetos para verificar y validar los productos de software resultantes del proceso de desarrollo de software orientado a objetos de una manera clara, precisa y profesional.

Contenido temático **Duración: 16 horas**

6. Pruebas Orientada a Objetos y Control de Calidad
- 6.1. Conceptos de Control de Calidad
- 6.2. Pruebas de los modelos AOO y DOO
- 6.3. Estrategias de de Pruebas Orientadas a Objetos (POO)
- 6.4. Diseño de casos de POO
- 6.5. Métodos de pruebas aplicables al nivel de clases
- 6.6. Diseño de caso de pruebas inter-clases.

Competencia:
Conocer y comprender las métricas, técnicas y métodos de administración de la calidad del software para aplicarlos en los procesos de desarrollo de software de calidad, con el fin de obtener productos de software libres de defectos, eficientes y efectivos, y acordes a las necesidades y especificaciones de la organización demandante de los servicios.

Contenido temático **Duración: 16 horas**

7. Métricas de la calidad y administración de configuraciones

7.1. Modelos de referencia para la calidad en la IS
7.2. Administración de Configuración de Software
7.3. Control de Versiones
7.4. Control de cambios
7.5. Auditoria de la configuración
7.6. Informe de estado

IV. ESTRUCTURA DE LAS PRÁCTICAS

No. de Práctica	Competencia(s)	Descripción	Material de Apoyo	Duración

VII. METODOLOGÍA DE TRABAJO

Investigación

La investigación será empleada en los trabajos extra-clase que se pedirán al estudiante sobre temas de actualidad o sobre temas que se verán posteriormente en clase. El propósito de estos trabajos es que el estudiante aprenda hacer investigación en medios electrónicos (Internet), libros, y revistas sobre temas del área. Las fuentes serán tanto en el idioma inglés como español para fomentar la enseñanza del idioma extranjero. Los reportes deberán contener las referencias que se utilizaron para la realización del trabajo y debe contar imprescindiblemente una conclusión personal acerca de la investigación. El maestro debe enfatizar a los estudiantes que los reportes escritos sean claros y bien redactados, recalcándoles también las faltas de ortografía.

Exposición oral

El alumno debe ser capaz de desenvolverse oralmente al exponer un tema o al establecer una discusión sobre una temática en particular del curso. El maestro debe involucrar a los estudiantes en la exposición oral ya sea de una publicación reciente o de un tema particular el alumno haya tenido el tiempo necesario para investigarlo.

Prácticas de Laboratorio

Llevar a la práctica los conocimientos teóricos vistos en clase es el mejor método de enseñanza-aprendizaje, por eso es importante que el estudiante desarrolle habilidades que le permitan resolver problemas reales en el área de sistemas colaborativos.

Exámenes de conocimientos

El maestro deberá aplicar al menos 2 exámenes de conocimientos durante el curso, de tal manera que refuercen los conocimientos aprendidos durante la clase. Los exámenes podrán ser de varios tipos, tales como: de preguntas abiertas, opción múltiple o mapas mentales.

Proyecto final

Parte central de la unidad de aprendizaje es la elaboración de un producto de software en el que todos los alumnos participan como un equipo de desarrollo profesional. Los estudiantes asumen roles en este equipo (ingeniero de requerimientos, arquitecto, diseñador, administrador de proyecto, etc.) durante la duración del curso y el objetivo es producir un producto de calidad. Los trabajos se realizarán, según se requiera, en forma individual o en forma grupal. La participación del maestro en la aplicación de está metodología es de mediador. El proyecto será expuesto oralmente al final del semestre.

VIII. CRITERIOS DE EVALUACIÓN

La evaluación general del curso consistirá de exámenes teóricos, tareas-reportes, prácticas de laboratorio y una exposición oral con un reporte escrito.

Los porcentajes de evaluación serán los siguientes:

Exámenes	10%
Tareas/prácticas	10%
Proceso de Desarrollo (Proyecto)	40%
Producto terminado (Proyecto)	40%
Total	100%

Criterio de acreditación

- Resolver al menos 2 exámenes parciales en tiempo y forma.
- Las tareas y las prácticas serán en modalidades individuales y/o de grupo.
- Deberán ser al menos 6 prácticas y tareas extra-clase por semestre
- Cumplir con las prácticas y tareas extra-clase en tiempo y forma.
- Cumplir con el proyecto final, su presentación oral y reporte escrito en tiempo y forma.

Criterio de evaluación

- Las tareas, prácticas y exámenes serán resueltos en clases posteriores a la entrega para que el estudiante conozca inmediatamente la solución propuesta en cada uno de los trabajos o exámenes.
- En el caso de la exposición final por equipo, la evaluación se dividirá en dos: reporte escrito y exposición, en el primer caso la calificación será por equipo y los puntos a evaluar serán, contenido, claridad y forma, así como ortografía y redacción; para la exposición oral ésta se calificará de manera individual y los puntos a evaluar serán, dominio del tema, claridad y estructura. Los alumnos puede ayudarse en la exposición mediante apoyos visuales tales como proyector de transparencias, acetatos u medios multimedia.

El proyecto final será por equipos y constara de la aplicación funcionando, un reporte escrito y la exposición oral. La exposición oral se evaluara individualmente, el reporte escrito y la aplicación funcionando se calificará por equipo.

IX. BIBLIOGRAFÍA

Básica

Pressman, R., Ingeniería del software: un enfoque práctico. S.5ª ed. McGraw Hill. 2002, ISBN 844813214-9.

Bernd Bruegge and Allen H. Dutoit, Object-Oriented Software Engineering, Prentice Hall Inc., 2000

Rumbaugh, J., et al., Object-Oriented Modeling and Design, Prentice Hall Inc., 1991

Notas del curso, disponibles en el sitio del curso.

Artículos diversos disponibles en Internet.

Complementaria

G. Booch, J.Rumbaugh, I.Jacabson, The Unified Modeling Language User Guide, Addison Wesley, 1999.

P. Stevens, R. Pooley, Using UML Software Engineering with Objects and componets, Addison Wesley, 1999.

Sommerville, Software Engineering, Fifth Edition, Addison Wesley, 1995.

R. Fairley, Ingeniería del Software, Mc.Graw-Hill.