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 Asignatura: **Metodología de la Programación**	5. Clave: **9828**
6. HC: 2 HL**:**2 HT:2 HE: 2 CR: 8	
7. Ciclo escolar: **2009-1**	8. Etapa de formación a la que pertenece: **Disciplinaria**
8. Carácter de la Asignatura: **X Obligatoria**: Optativa:	
10. Requisitos para cursar la asignatura: **Programación orientada a objetos, Estructura de datos y algoritmos**	
Actualizó: **Dra. Mónica Elizabeth Tentori Espinosa**	**Vo. Bo. Biol. Marcelo Rodríguez Meraz**
Fecha: **Junio de 2009**	Cargo: **Subdirector**

II. PROPÓSITO GENERAL DE LA UNIDAD DE APRENDIZAJE

Introducir al estudiante en el desarrollo de sistemas complejos que requiere la coordinación, comunicación y el trabajo en equipo a través de una metodología. Deberá ser capaz de modelar no solos sistemas de software sino otro tipo de sistemas reales de la empresa, siempre utilizando los conceptos de la orientación a objetos (OO). Establecer un acoplamiento explícito de los conceptos y los *elementos* ejecutables y administrar los problemas típicos de los sistemas complejos de misión crítica.

III. COMPETENCIA (S) DE LA UNIDAD DE APRENDIZAJE

Desarrollar un proyecto de desarrollo de software a mediana escala colaborando con un grupo de personas interdisciplinario, valorando las opiniones de todos los miembros del equipo de desarrollo.

IV. EVIDENCIA DE DESEMPEÑO

Exposiciones grupales de los avances del desarrollo de software.
Reporte de cada etapa del proceso de desarrollo de software.
Un producto de software funcional sobre máquinas reales.
Reporte final de la documentación del proyecto

V. DESARROLLO POR UNIDADES

Nombre de la unidad:
Desarrollo de software
Competencia:
Distinguir el proceso general de desarrollo de software y analizar el enfoque orientado a objetos.

Contenido temático:

1. Logistica del curso, logística de los proyectos y logística de evaluación
2. Principios del desarrollo de software
3. el proceso y el producto
4. Proceso de desarrollo de software.
    a. Administración.
    b. Desarrollo (Análisis, diseño, implementación, pruebas, mantenimiento).
    c. Control (Documentación, calidad, validación y verificación, configuración).
5. Modelos de desarrollo de software.
6. Proceso de desarrollo de software.
    a. Introducción a orientación a objetos.
    b. Clases y objetos.
    c. Modelado.
    d. Notación UML.
    e. Ventajas de orientación a objetos.
    f. Características de orientación a objetos.

Duración: 12 horas

Nombre de la unidad:
Métricas y técnicas del desarrollo de software
Competencia:
Establecer medidas de calidad para la elaboración del software.

Contenido temático:

1. Principios del desarrollo de software
2. Calidad del proceso y del producto.
3. Documentación.
    3.1. Estándares.
4. Administración de configuración y control de cambios.
    4.1. Administración de versiones.
5. Validación y verificación.
    5.1. Revisiones técnicas.

Duración: 8 horas

Nombre de la unidad:
Fase de análisis
Competencia:
Interpretar problemáticas existentes desarrollando documentación y modelado de dicha información
.

Contenido temático:

1. Aspectos de análisis.
2. Análisis del dominio.
3. El proceso de análisis.
   3.1. Modelado funcional (Casos de uso).
   3.2. Modelado estático (modelo de clases).
   3.3. Modelado dinámico (Diagrama de interacción).
   3.4. Diagramas de estado.
4. Revisiones de análisis.

Duración: 18 horas

Nombre de la unidad:
Fase de diseño
Competencia:
Construir modelos de diseño para sistemas de software basados en la documentación de análisis disponible
.

Contenido temático:

1. Aspectos de diseño.
2. Proceso de Diseño del sistema.
   2.1. Particionar el sistema en subsistemas.
   2.2. Concurrencia entre subsistemas.
   2.3. Administración de subsistemas.
   2.4. Interfaces de usuario.
   2.5. Administración de datos.
   2.6. Administración de recursos.
   2.7. Comunicación entre subsistemas
3. Proceso de Diseño de Objetos.
   3.1. Descripción de objetos.
   3.2. Diseño de algoritmos y estructuras de datos.
4. Patrones de diseño.
5. Revisiones de diseño.

Duración: 8 horas

Nombre de la unidad:
Fase de implementación y pruebas
Competencia:
II. Traducir el diseño de un sistema de software en un lenguaje de programación evaluando el estado del sistema resultante.

Contenido temático:	**Duración:** 18 horas
1. Implementación. 1.1. Evaluación de lenguajes. 1.2. El modelo de clases. 1.3. Características de orientación a objetos. 1.4. Estrategias de implementación. 2. Pruebas. 2.1. Estrategias de pruebas. 2.2. Procedimientos de pruebas 2.3. Diseño de casos de pruebas. 2.4. Presentación de resultados.	

VI. ESTRUCTURAS DE LAS PRÁCTICAS DE LABORATORIO				
No. de práctica	**Competencia(s)**	**Descripción**	**Material de apoyo**	**Duración**
1	Distinguir el proceso general de desarrollo de software y analizar el enfoque orientado a objetos.	Desarrollar una investigación sobre el proceso de desarrollo de software y un modelo para la aplicación del mismo. Se deben obtener las ventajas y desventajas del modelo así como la viabilidad de orientación a objetos	Libros Computadora	Desarrollo (1 semana) Discusión (1 hora)
2	Distinguir el proceso general de desarrollo de software y analizar el enfoque orientado a objetos.	Desarrollar en grupo (3 personas) un manual sobre UML el cual sea uniforme para cada equipo involucrado. Esto proporciona homogeneidad entre los desarrolladores	Internet	Desarrollo (1 semana) Análisis (3 días) Actualización (2 días)
3	Establecer medidas de calidad para la elaboración del software.	Se establecen equipos de trabajo para el semestre completo, estos equipos deben oscilar entre 4 y 6 personas. Cada equipo debe desarrollar y establecer sus propias políticas de calidad, indicando sus estándares de documentación y del proceso a utilizar durante el desarrollo del software.	Libros Revistas Internet	Investigación (1 semana) Desarrollo (1semana) Revisiones periódicas
4	Interpretar problemáticas existentes desarrollando documentación y modelado de dicha información	Cada equipo de desarrollo debe desarrollar un documento donde se documente y se modele un problema. Indicando como afecta el desarrollo del producto en cuestión. El problema será planteado por una persona ajena al grupo de estudio, la cual pueda ser objetiva cuando visualice los resultados. El profesor funge como asesor de los grupos.	Libros Internet Revistas Computadora	Investigación (1 semana) Análisis (1 semana). Desarrollo (2 semanas) Presentación (30 min.) Corrección (1 semana)
5	Construir modelos de diseño para sistemas de software basados en la documentación de análisis disponible	Cada equipo de desarrollo debe construir los modelos de diseño para el sistema que este trabajando. Estos modelos deben ser elaborados basados en el lenguaje UML cubriendo los estándares de calidad establecidos previamente	Libros Internet Computadora	Investigación (1 semana) Desarrollo (2 semanas) Presentación (30 min.) Corrección (1

				semana).)
6	Traducir el diseño de un sistema de software en un lenguaje de programación evaluando el estado del sistema resultante	Los equipos de trabajo se organizan para la implementación del sistema, basados en los conceptos vistos en clase. Cada vez que se obtiene un prototipo, se debe presentar un reporte (de pruebas) del estado del sistema en desarrollo.	Libros Internet	Investigación (1 semana) Desarrollo (2 semana) Revisiones periódicas Presentación (15 min.)

VII. METODOLOGÍA DE TRABAJO

La asignatura consta de dos partes:
a) Fundamentos teórico-prácticos.
En este apartado se expone el contenido temático indicado para la asignatura, mediante la impartición de clases, donde se presentan los fundamentos teóricos junto con la realización de ejercicios prácticos.
b) Ejercicios de aplicación.
Los alumnos realizarán, en grupos reducidos, ejercicios de aplicación, que consistirán básicamente en el desarrollo de software.
Estos ejercicios de aplicación se comenzarán al inicio del curso y deberán ser entregados (con su respectiva presentación formal) de la siguiente manera:

- Documentación de análisis después de 5 semanas de clases.
- Documentación de diseño después de 9 semanas de clases.
- Código del sistema funcional después de 12 semanas de clases.
- Documentación de pruebas después de 14 semanas de clases.
- Sistema completo al final del semestre

Tareas, lecturas y exposición
La crítica de artículos de relevancia científica 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 estas tareas es el fomentar en el estudiante el hábito de la lectura y crítica sobre trabajos de relevancia internacional. Los estudiantes deberán de preparar un tema relevante a la materia y exponerla en clase. El propósito de estos trabajos es que el estudiante aprenda hacer investigación por su cuenta sobre temas del área. Las fuentes serán tanto en el idioma inglés como español. 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.

Exámenes
El maestro deberá aplicar al menos 1 examen 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, crucigramas o mapas mentales.

Proyecto final de cursos
El estudiante deberá crear un sistema utilizando tecnología ubicua o móvil. El propósito de esta tarea es permitir al estudiante aplicar todo lo aprendido en clase.

VIII. CRITERIOS DE EVALUACIÓN

La evaluación total del curso consistirá de un examen teórico, tareas, críticas de lecturas, proyecto final con un demo y reporte escrito, así como, una exposición oral con un reporte escrito.

Los porcentajes de evaluación son los siguientes:

- Exámenes 20%
- Tareas, exposición y prácticas 30%
- Proyecto final 50%

Criterios de acreditación:

- Entregar las tareas, lecturas y prácticas en tiempo y forma
- Cumplir con la presentación final en tiempo y forma
- Cumplir con la exposición formal en tiempo y forma

Criterios de evaluación:

- Realizar las tareas y lecturas de forma individual
- Colaborar en el desarrollo de las prácticas, proyecto final y exposición con su equipo de trabajo
- Reportar el porcentaje de lo trabajado de manera individual para todos los trabajos en equipo

IX. BIBLIOGRAFÍA

- Pressman, Roger S. 2002. "*Software Engineering: A practitioner's approach*". Quinta edición.
- Larman, Craig. 1999. “UML y patrones : introducción al análisis y diseño orientado a objetos”. Prentice-Hall.
- Muller, Pierre-Alain. ”Modelado de objetos con UML”. GESTIÓN 2000.
- Coad, P; Yourdon, E. “*Object Oriented Analysis”.* Yourdon Press.
- Rumbaugh, James, Blaha, Michael, Premerlani, William,
- Eddy, Frederick, Lorensen, William. 1996. "*Modelado y Diseño Orientados a Objetos. Metodología OMT*". Prentice Hall.
- G. Booch. 1996. “Análisis y diseño orientado a objetos con aplicaciones”. Addison-Wesley/Díaz de Santos.
- Diller, Z. “An Introduction to Formal Methods”. John Wiley & Sons Ltd.
- Hayes, *Specification Case Studies*, Prentice-Hall. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen. “*Object-Oriented Modeling and Design”.* Prentice-Hall.