UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

COORDINACIÓN DE FORMACIÓN BÁSICA
COORDINACIÓN DE FORMACIÓN PROFESIONAL Y VINCULACIÓN
PROGRAMA DE UNIDAD DE APRENDIZAJE

I. DATOS DE IDENTIFICACIÓN

1. Unidad Académica: Facultad de Ciencias

2. Programa (s) de estudio: Lic. en Ciencias Computacionales, Lic. en Matemáticas Aplicadas

3. Vigencia del plan:

4. Nombre de la Unidad de aprendizaje: Metodología de la Programación 5. Clave:

6. HC: 2 HL 3 HT 0 HPC ____ HCL ____ HE 2 CR 7

7. Etapa de formación a la que pertenece: Disciplinaria.

8. Carácter de la Unidad de aprendizaje: Obligatoria ___X___ Optativa _____

9. Requisitos para cursar la unidad de aprendizaje:

Formuló: Alberto Leopoldo Morán y Solares

Fecha: Agosto de 2016

Vo.Bo. Dr. Juan Crisóstomo Tapia Mercado

Cargo: Director

II. PROPÓSITO GENERAL DE LA UNIDAD DE APRENDIZAJE

La unidad de aprendizaje de Metodología de la programación se encuentra ubicada en la etapa disciplinaria con carácter obligatorio para la Lic. en Ciencias Computacionales y optativa para la Lic. en Matemáticas Aplicadas. Tiene como recomendación la acreditación de las asignaturas de Introducción a la Programación y Programación Orientada a Objetos.

Es una unidad de aprendizaje teórico-práctico y consta de 5 unidades. En la unidad I se realiza una introducción general al proceso de desarrollo de software. En la unidad II se presentan métricas y técnicas del desarrollo de software, mientras que en las unidades III, IV y V se presentan aspectos de una metodología de desarrollo de software, incluyendo análisis, diseño, programación y pruebas.

La finalidad de la unidad de aprendizaje es que el estudiante comprenda y aprenda a aplicar los métodos, técnicas y herramientas de una metodología de desarrollo de software.

Su importancia reside en que permite comprender y aplicar un conjunto de métodos, técnicas y herramientas para la producción de software de calidad a una escala pequeña o mediana para un cliente real.

Es deseable un buen dominio del idioma inglés.

III. COMPETENCIA DE LA UNIDAD DE APRENDIZAJE

Desarrollar un proyecto de software de pequeña-mediana escala de manera colaborativa utilizando una metodología de desarrollo de software, para resolver un problema que satisfaga las necesidades de un cliente real, con actitud creativa y responsable.

IV. EVIDENCIA (S) DE DESEMPEÑO

Entrega portafolio que dé cuenta de la planeación, análisis, diseño, construcción y pruebas del proyecto de software, que integra minutas de reuniones de avance periódicas, documentación técnica de cada etapa del proceso de desarrollo, un producto de software funcional acorde a los requerimientos y la documentación final técnica y de usuario del proyecto.

V. DESARROLLO POR UNIDADES

Competencia:
Explicar conceptos, ámbito, terminología y modelos del desarrollo de software, así como aspectos del control del mismo proceso mediante la revisión de la literatura especializada metodologías de desarrollo, para contextualizar la importancia de esta actividad en las organizaciones, con una actitud objetiva y crítica.

Contenido **Duración 4 horas**

Parte I Fundamentos

1. Desarrollo de software

- 1.1. Principios del desarrollo de software
- 1.2. El proceso y el producto
- 1.3. Proceso de desarrollo de software.
  - 1.3.1. Desarrollo (Análisis, diseño, implementación, pruebas, mantenimiento).
  - 1.3.2. Control (Documentación, calidad, validación y verificación, configuración).
- 1.4. Modelos de desarrollo de software.

Competencia:
Describir métricas y técnicas que se utilizan para el desarrollo de software, a través de una revisión de la literatura especializada en metodologías de desarrollo, gestión de calidad y control de configuraciones para su posterior aplicación en el desarrollo de software, con una actitud disciplinada y, objetiva.

Contenido **Duración 4 horas**

2. Métricas y técnicas del desarrollo de software
   - 2.1. Principios del desarrollo de software
   - 2.2. Calidad del proceso y del producto.
   - 2.3. Documentación.
     - 2.3.1. Estándares.
   - 2.4. Administración de configuración y control de cambios.
     - 2.4.1. Administración de versiones.
   - 2.5. Validación y verificación.
     - 2.5.1. Revisiones técnicas.

Competencia:

Realizar modelos de análisis (funcional, estático, dinámico y de máquinas de estado) del problema propuesto mediante el uso de las herramientas y técnicas de análisis orientado a objetos para proponer una aplicación de software que resuelva la problemática especificada por el cliente de forma profesional y congruente con los procesos de la entidad que demanda los servicios, con una actitud disciplinada, creativa y crítica.

Contenido **Duración 8 horas**

3. Fase de Análisis

- 3.1. Aspectos de análisis.
- 3.2. Análisis del dominio.
- 3.3. El proceso de análisis.
  - 3.3.1. Modelado funcional (Casos de uso).
  - 3.3.2. Modelado estático (Modelo de clases).
  - 3.3.3. Modelado dinámico (Diagrama de interacción).
  - 3.3.4. Diagramas de estado.
- 3.4. Revisiones de análisis.

Competencia:
Concebir modelos de diseño de alto y bajo nivel de posibles soluciones de software mediante el uso de las herramientas y técnicas de diseño orientado a objetos para proponer de manera exitosa un de software que resuelva la problemática especificada por el cliente de forma profesional y congruente con los procesos de la entidad que demanda los servicios, con una actitud disciplinada y creativa.

Contenido **Duración 8 horas**

4. Fase de Diseño

- 4.1. Aspectos de diseño.
- 4.2. Proceso de Diseño del sistema.
  - 4.2.1. Particionar el sistema en subsistemas.
  - 4.2.2. Concurrencia entre subsistemas.
  - 4.2.3. Administración de subsistemas.
  - 4.2.4. Interfaces de usuario.
  - 4.2.5. Administración de datos.
  - 4.2.6. Administración de recursos.
  - 4.2.7. Comunicación entre subsistemas
- 4.3. Proceso de Diseño de Objetos.
  - 4.3.1. Descripción de objetos.
  - 4.3.2. Diseño de algoritmos y estructuras de datos.
- 4.4. Patrones de diseño.
- 4.5. Revisiones de diseño.

Competencia:
Desarrollar aplicaciones de software de escala pequeña o mediana mediante el uso de las herramientas y técnicas de construcción y prueba del software para resolver el problema propuesto por el cliente de manera satisfactoria con una actitud disciplinada, creativa y responsable.

Contenido **Duración 8 horas**

5. Fase de Implementación y Pruebas

5.1. Implementación.
- 5.1.1. Evaluación de lenguajes.
- 5.1.2. El modelo de clases.
- 5.1.3. Características de orientación a objetos.
- 5.1.4. Estrategias de implementación.

5.2. 2. Fundamentos de Pruebas.
- 5.2.1. Estrategias de pruebas.
- 5.2.2. Procedimientos de pruebas
- 5.2.3. Diseño de casos de pruebas.
- 5.2.4. Presentación de resultados.

VI. ESTRUCTURA DE LAS PRÁCTICAS				
No. de Práctica	**Competencia(s)**	**Descripción**	**Material de Apoyo**	**Duración**
1	Realizar modelos de análisis (funcional, estático, dinámico y de máquinas de estado) del problema propuesto a través del uso de técnicas y herramientas de análisis orientado a objetos con el fin de establecer el alcance técnico del problema y de una posible solución vía software con una actitud disciplinada, creativa, objetiva, responsable y crítica..	Realizar un ejercicio de análisis orientado a objetos para su proyecto de desarrollo. Se deben especificar los requerimientos funcionales y no funcionales, y versiones de análisis de los modelos estático, dinámico y funcional.	Pizarrón, plumones y borrador. Computadora, Software de diagramación, Software de modelado UML, procesador de texto y de presentaciones	16 horas
2	Elaborar y refinar los modelos estático, dinámico, funcional y de despliegue de su proyecto a través del uso de técnicas y herramientas de diseño orientado a objetos con el fin de proporcionar una solución de alto y bajo nivel al problema establecido por el cliente con una actitud disciplinada, creativa, objetiva, responsable y crítica..	Realizar un ejercicio de diseño orientado a objetos para su proyecto de software. Se debe refinar la especificación de los modelos estático, dinámico y funcional para la solución propuesta al problema planteado por el cliente. Los artefactos a generar incluyen las versiones refinadas de los diagramas de casos de uso y sus descripciones, del modelo de clases y objetos, diagramas de secuencia, diagramas de actividad, diagramas de comunicación, diagramas de máquinas de estado, diagramas de componentes y diagramas de despliegue. Se recomienda que la	Pizarrón, plumones y borrador. Computadora, Software de diagramación, Software de modelado UML, procesador de texto y de presentaciones	18 horas

		solución propuesta incluya al menos 1 patrón de diseño.		
3	Desarrollar una versión ejecutable y validada de la aplicación propuesta mediante el uso de técnicas y herramientas de construcción y validación de software con el fin de establecer que es una solución válida y que satisface los requerimientos del cliente con una actitud disciplinada, creativa, objetiva, responsable y crítica..	Realizar un ejercicio de implementación y pruebas para su proyecto de software. Se deben generar versiones ejecutables de módulos, subsistemas y sistemas. Asimismo, se debe generar un plan de pruebas de diferentes tipos y niveles y se debe entregar una bitácora del proceso de pruebas. Se debe generar la documentación técnica y de usuario del sistema.	Pizarrón, plumones y borrador. Computadora, Software de modelado UML, Ambiente integrado de desarrollo, procesador de texto y de presentaciones	14 horas

VII. METODOLOGÍA DE TRABAJO

Trabajo de investigación documental
La investigación documental será empleada en los trabajos extra-clase que se pedirán al estudiante sobre temas de actualidad. El propósito de estos trabajos es que el estudiante aprenda a 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 el aprendizaje del idioma extranjero. Los reportes deberán contener las referencias que se utilizaron para la realización del trabajo y deben contar imprescindiblemente con una conclusión personal acerca de la investigación. El maestro debe enfatizar a los estudiantes que los reportes escritos sean claros, estén bien redactados, y no incluyan faltas de ortografía.

Exposición oral
El estudiante debe ser capaz de desenvolverse oralmente al exponer un tema o al establecer una discusión sobre una temática en particular de la unidad de aprendizaje. El maestro debe involucrar a los estudiantes en la exposición oral de publicaciones recientes y de temas particulares del interés de los mismos, quienes deberán investigarlo y desarrollarlo.

Prácticas de Laboratorio
Llevar a la práctica los conocimientos teóricos vistos en clase es uno de los mejores métodos de enseñanza-aprendizaje, por eso es importante que el estudiante desarrolle habilidades que le permitan resolver problemas reales.

Exámenes de conocimientos
El maestro deberá aplicar al menos 2 exámenes de conocimientos como parte de la evaluación de la unidad de aprendizaje, de tal manera que refuercen los conocimientos aprendidos durante la clase. Los exámenes podrán ser de varios tipos, incluyendo: 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 el desarrollo del proyecto y el objetivo es producir un producto de calidad. Los trabajos se realizarán en forma grupal. La participación del maestro en la aplicación de está metodología es de facilitador. El proyecto se evaluará tanto de manera oral como escrita al final del semestre (ver criterios de evaluación).

VIII. CRITERIOS DE EVALUACIÓN

La evaluación general de la unidad de aprendizaje consistirá en dos exámenes teóricos, reportes de tareas y prácticas de laboratorio y la presentación oral de las evidencias del proyecto de desarrollo de software final.

Los porcentajes de evaluación serán los siguientes:
Exámenes 15%
Reportes de tareas y prácticas 15%
Proceso de Desarrollo (Proyecto) 35%
Producto terminado (Proyecto) 35%
Total 100%

Criterio de acreditación
Resolver los dos exámenes parciales en tiempo y forma.
Las tareas y las prácticas serán en modalidades individuales y/o de grupo. Los reportes deberán contener una introducción, planteamiento de pregunta/problema, metodología, desarrollo, discusión, conclusión y referencias.
Deberán ser al menos 4 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 la solución propuesta en cada uno de los trabajos o exámenes.
Se recomienda que la elaboración del proyecto final se realice durante las últimas 4 ó 5 semanas de la unidad de aprendizaje. El proyecto final, incluirá: (1) Minutas de reuniones de ingeniería periódicas (semanal o quincenal), (2) Plan de trabajo y reporte de actividades, (semanal o quincenal), (3) Documentación técnica por fase (requerimientos, análisis y diseño, implementación y pruebas), (4) Producto de software funcional acorde a los requerimientos, (5) Documentación final técnica y de usuario del proyecto.
En todos los reportes y documentos escritos se evaluará la calidad del contenido, claridad y forma, así como ortografía y redacción. En las presentaciones orales se evaluará el dominio del tema, claridad y estructura.
Los estudiantes pueden ayudarse en la exposición mediante apoyos visuales tales como cañón proyector y presentación powerpoint.
"Para la acreditación de la unidad de aprendizaje se atenderá al Estatuto Escolar Vigente, artículos 70-71, el estudiante deberá contar un mínimo de 80% de asistencias en el periodo. Tener un mínimo aprobatorio de 60 en su calificación final".

IX. BIBLIOGRAFÍA

Básica

- Fontela, Carlos, UML: modelado de software para profesionales, 1a ed., 2011.
- Pressman, Roger S., Software Engineering: A practioner's approach, 8th ed., 2014.
- Sánchez Alonso, S., Sicilia Urbán, M.A., Rodríguez García, D., Ingeniería del Software: Un enfoque desde la guía SWEBOK, 1a. Ed., Alfaomega Grupo Editor, S.A. de C.V., ISBN: 978-607-707-420-5, 2012.
- Sommerville, Ian, Software Engineering, 10th ed., Pearson, 2015.
- Biblioteca Digital ACM, artículos varios en Ingeniería de Software y temas relacionados de Ciencias Computacionales (Disponible en http://dl.acm.org a través de la biblioteca UABC, y en la Cimarred de los campi) – en Inglés.
- Bibliotea Digital IEEE Xplore, artículos varios en Ingeniería de Software y temas relacionados de Ciencias Computacionales (Disponible en http://ieeexplore.ieee.org/Xplore/home.jsp a través de biblioteca UABC, y la Cimarred de los campi) – en Inglés.

Complementaria

- Booch, Grady, El lenguaje unificado de modelado: guía del usuario, 2a ed., 2006 (clásico).
- Braude, Eric, Ingeniería de software: una perspectiva orientada a objetos, 2a ed., Alfa-Omega 2005 (clásico).
- Gamma, Erich, Patrones de diseño: elementos de software orientado a objetos reutilizable, 2003 (clásico).
- Jacobson, Ivar, El proceso unificado de desarrollo de software, 2000 (clásico).
- Rumbaugh, J., El lenguaje unificado de modelado: manual de referencia, 2a ed., 2007 (clásico).
- Rumbaugh, James, Modelado y diseño orientado a objetos, Edición actualizada, 1997 (clásico).
- Schach, Stephen R., Análisis y diseño orientado a objetos con UML y el proceso unificado, 1a ed., 2005 (clásico).
- Weitzenfeld, Alfredo, Ingeniería de software orientada a objetos con UML, Java e Internet, 2005 (clásico).

X. PERFIL DOCENTE

Preferentemente con título de licenciatura de Ciencias Computacionales, área afín, y experiencia probada en el uso de metodologías de desarrollo de software y en docencia.