UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
COORDINACIÓN DE FORMACIÓN BÁSICA
COORDINACIÓN DE FORMACIÓN PROFESIONAL Y VINCULACIÓN UNIVERSITARIA
PROGRAMA DE UNIDAD DE APRENDIZAJE

I. DATOS DE IDENTIFICACIÓN

1. Unidad Académica: Facultad de Ciencias
2. Programa (s) de estudio: Licenciado en Ciencias Computacionales 3. Vigencia del plan:
4. Nombre de la Unidad de aprendizaje: Ingeniería de Software 5. Clave:
6. HC: 2 HL 3 HT 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: Metodología de la Programación

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 Ingeniería de Software se encuentra ubicada en la etapa disciplinaria con carácter obligatorio. Su área de conocimiento es Programación e Ingeniería de Software, tiene como requisito haber acreditado la asignatura de Metodología de la Programación.

La finalidad de la unidad de aprendizaje es que el estudiante comprenda y aprenda a aplicar los métodos, técnicas y herramientas actuales de la ingeniería de software, así como las diferentes actividades que se realizan para la producción de software de calidad.

Su importancia reside en que permite comprender y aplicar un conjunto de métodos, técnicas y herramientas actuales de la Ingeniería de Software, así como diferentes actividades que se realizan para la producción de software de calidad a una escala mediana y grande en el desarrollo de software para un cliente real. Asimismo, permite asociar aspectos de control de calidad, gestión de configuración, y gestión de proyecto en el desarrollo de software.

III. COMPETENCIA DE LA UNIDAD DE APRENDIZAJE

Desarrollar un proyecto de software de mediana-grande escala de manera colaborativa utilizando técnicas y metodologías de la ingeniería de software, para resolver un problema que satisfaga las necesidades de un cliente real, con actitud disciplinada y creativa.

IV. EVIDENCIA (S) DE DESEMPEÑO

Portafolio que dé cuenta de los procesos de administración, desarrollo y control de calidad del proyecto de software, que integra minutas de reuniones de avance periódicas, documentación técnica por fase y disciplina 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 los conceptos, ámbito y terminología de la ingeniería del software, así como sus dominios de aplicación mediante la revisión de literatura especializada en ingeniería de software, para establecer su importancia en el desarrollo de software en las organizaciones, con una actitud objetiva y crítica.

Contenido **Duración 3 horas**

Parte I Fundamentos

1. Introducción a la Ingeniería de Software (IS)

- 1.1. ¿Qué es el software?
- 1.2. Naturaleza y cualidades del software
- 1.3. Software de Calidad
- 1.4. ¿Qué es la Ingeniería de software?
- 1.5. Historia de la IS
- 1.6. Ámbitos de aplicación en las organizaciones

Competencia:
Describir los procesos formales que se utilizan para el desarrollo de software, así como los principales tipos y modelos del proceso de desarrollo, a través de una revisión de la literatura especializada en metodologías para su posterior aplicación en el desarrollo de software, con una actitud disciplinada y objetiva.

Contenido **Duración 4 horas**

2. Procesos formales de desarrollo de Software

- 2.1. Proceso de Software
- 2.2. Modelo de ciclo de vida del software
- 2.3. Metodologías predictivas
- 2.4. Metodologías ágiles
- 2.5. Estudio de casos
  - 2.5.1. Proceso Unificado de Desarrollo (RUP)
    - 2.5.1.1. Fases
    - 2.5.1.2. Disciplinas fundamentales y de apoyo
    - 2.5.1.3. Artefactos
    - 2.5.1.4. Roles
  - 2.5.2. Programación Extrema (XP)
  - 2.5.3. Scrum

Competencia:
Detallar aspectos fundamentales de la gestión de un proyecto, mediante el uso de métodos y técnicas de gestión de proyectos, para especificar, planear y dar seguimiento a su proyecto de software, con una actitud disciplinada y crítica.

Contenido **Duración 5 horas**

3. Fundamentos de gestión de proyectos de software
- 3.1. Visión general de la gestión de proyectos
- 3.2. Definición del proyecto
  - 3.2.1. Panorámica del proyecto y condiciones de satisfacción
  - 3.2.2. Enunciado de definición del proyecto
- 3.3. Planeación y seguimiento del proyecto
  - 3.3.1. Actividades y tareas
    - 3.3.1.1. Estructura de Descomposición de Trabajo (Work Breakdown Structure)
  - 3.3.2. Estimación de duración de tareas y esfuerzo de trabajo
  - 3.3.3. Estimación de recursos humanos asociados al proyecto
  - 3.3.4. Estimación de costos
  - 3.3.5. Calendarización del proyecto
    - 3.3.5.1. Diagramas de Gantt

Competencia:
Especificar los requerimientos de su proyecto de software, mediante el uso de los conceptos y actividades propios de la ingeniería de requerimientos para establecer las necesidades prioritarias del cliente, con una actitud creativa y objetiva.

Contenido **Duración 4 horas**

4. Análisis de requerimientos y especificación
- 4.1. Requerimientos vs especificación
- 4.2. Conceptos, actividades, actores y características de los requerimientos
- 4.3. Tipos de requerimientos
- 4.4. Actividades para la obtención de requerimientos
- 4.5. Notación
  - 4.5.1. Casos de uso
  - 4.5.2. Modelos Entidad-Relación
  - 4.5.3. Diagramas de clases y objetos
  - 4.5.4. Diagramas CRC

Competencia:
Diseñar los modelos estático, dinámico, funcional y de despliegue mediante el uso de las herramientas y técnicas para la especificación y concepción de un producto de software que resuelva la problemática especificada por el cliente de forma profesional y congruente con los procesos de la organización que demanda los servicios, con una actitud disciplinada y creativa.

Contenido **Duración 7 horas**

5. Análisis y Diseño Orientado a Objetos (ADOO)
- 5.1. Conceptos fundamentales de análisis y diseño
- 5.2. Actividades de ADOO
- 5.3. Modelo de análisis y diseño orientado a objetos
- 5.4. Notación
  - 5.4.1. Refinando los casos de uso y el diagrama de clases y objetos
  - 5.4.2. Diagramas de secuencia, actividad y comunicación
  - 5.4.3. Diagramas de máquinas de estado
  - 5.4.4. Diagramas de componentes y de despliegue
- 5.5. Patrones de Diseño

Competencia:
Desarrollar una aplicación de software mediante el uso de las herramientas y técnicas para construcción y validación del software para resolver el problema propuesto por el cliente conforme a los procesos de su organización, con una actitud responsable y crítica.

Contenido **Duración 5 horas**

6. Construcción y Pruebas Orientada a Objetos
- 6.1. Del modelado al lenguaje de programación
- 6.2. Reutilización de código
- 6.3. Principios fundamentales de la construcción de software
  - 6.3.1. Minimizar complejidad
  - 6.3.2. Anticipar los cambios
  - 6.3.3. Construir para verificar
  - 6.3.4. Utilización de estándares
- 6.4. El porqué de las pruebas
- 6.5. Conceptos fundamentales
- 6.6. Técnicas de Pruebas
  - 6.6.1. Pruebas de caja blanca
  - 6.6.2. Pruebas de caja negra
- 6.7. Niveles de prueba (objeto, objetivo)
- 6.8. El proceso de prueba
- 6.9. Estudio de caso
  - 6.9.1. JUnit

Competencia:
Detallar aspectos fundamentales del control de calidad y de la gestión de configuraciones, a través del uso de técnicas, herramientas y estándares de control y gestión para incrementar la calidad del proceso y del producto en su proyecto de software, con una actitud objetiva y crítica.

Contenido **Duración 4 horas**

7. Control de Calidad y Gestión de configuraciones

7.1. Conceptos y objetivos de la calidad de Software
7.2. Calidad del producto
7.3. Calidad del proceso
7.4. Estándares y modelos de calidad de software
7.5. Conceptos y objetivos del control de configuración
7.6. Actividades de la configuración del software
7.7. Control de Versiones
7.8. Control de cambios
7.9. Técnicas y herramientas

VI. ESTRUCTURA DE LAS PRÁCTICAS				
No. de Práctica	**Competencia(s)**	**Descripción**	**Material de Apoyo**	**Duración**
1	Especificar y planear un proyecto de desarrollo de software mediante técnicas y herramientas de gestión de proyectos que facilite el desarrollo del mismo en los tiempos adecuados, con los recursos humanos establecidos y en los costos establecidos, con una actitud disciplinada, creativa, objetiva, responsable y crítica.	Realizar un ejercicio de especificación y planeación de su proyecto de desarrollo de software. Se debe generar el enunciado de panorámica del proyecto, el enunciado de definición del proyecto, la estructura de descomposición de trabajo, la estimación de duración del trabajo, de esfuerzo, de recursos humanos asociados al proyecto y de costos, y el diagrama de Gantt.	Pizarrón, plumones y borrador. Computadora, Software de diagramación, Software de gestión de proyectos, procesador de texto y de presentaciones	10 horas
2	Especificar los requerimientos y realizar el modelado de casos de uso y de clases del proyecto a través del uso de técnicas y herramientas de gestión de requerimientos y modelado 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 y especificación de requerimientos para su proyecto de desarrollo. Se deben especificar los requerimientos funcionales y no funcionales, y según sea el caso, el modelo de casos de uso y las descripciones breves y detalladas, el modelo entidad-relación, los diagramas CRC y los diagramas de clases y objetos.	Pizarrón, plumones y borrador. Computadora, Software de diagramación, Software de modelado UML, procesador de texto y de presentaciones	8 horas
3	Diseñar los modelos estático, dinámico, funcional y de despliegue de su proyecto a través del uso de técnicas y herramientas de análisis y diseño orientado a objetos con el fin	Realizar un ejercicio de análisis y diseño orientado a objetos para su proyecto de software. Se deben especificar los modelos estático, dinámico y funcional para la solución	Pizarrón, plumones y borrador. Computadora, Software de	18 horas

	de proporcionar una solución de alto y bajo nivel al problema establecido por el cliente, con una actitud disciplinada, creativa, objetiva, responsable y crítica.	propuesta al problema planteado por el cliente. Los artefactos a generar incluyen versión refinada de los diagramas de casos de uso y sus descripciones, versión refinada del modelo de Entidad-Relación, versión refinada 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 solución propuesta incluya al menos 2 patrones de diseño.	diagramación, Software de modelado UML, procesador de texto y de presentaciones	
4	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.	Pizarrón, plumones y borrador. Computadora, Software de modelado UML, Ambiente integrado de desarrollo, procesador de texto y de presentaciones	12 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 durante el desarrollo 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 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). Se recomienda iniciar el proyecto a partir de la semana 4 ó 5 de la unidad de aprendizaje.

VIII. CRITERIOS DE EVALUACIÓN

La evaluación general de la unidad de aprendizaje consistirá en exámenes teóricos, reportes de tareas y prácticas de laboratorio y la presentación oral y portafolio de 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 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.

Entregables del proyecto, incluyendo: (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 (concepción, elaboración, construcción y transición) y disciplina del proceso de desarrollo (modelado de negocio, requerimientos, análisis y diseño, gestión de proyecto, etc.), (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 evaluarán 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, por lo que 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 campus) – 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 campus) – 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, 1a ed., 2003 (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

Profesionista en Ciencias Computacionales o área afín, y experiencia probada en la aplicación de la ingeniería de software, docencia y trabajo en equipo.