UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

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

I. DATOS DE IDENTIFICACIÓN

1. Unidad Académica: Facultad de Ingeniería, Mexicali., Facultad de Ingeniería, Arquitectura y Diseño, Ensenada; y Facultad de Ciencias Químicas e Ingeniería, Tijuana;

2. Programa Educativo: Ingeniero en Computación

3. Plan de Estudios: 2020-1

4. Nombre de la Unidad de Aprendizaje: Ingeniería de Software

5. Clave: 36297

6. HC: 01 **HL:** 00 **HT:** 03 **HPC:** 00 **HCL:** 00 **HE:** 01 **CR:** 05

7. Etapa de Formación a la que Pertenece: Terminal

8. Carácter de la Unidad de Aprendizaje: Obligatoria

9. Requisitos para Cursar la Unidad de Aprendizaje: Ninguno

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
REGISTRADO
17 OCT 2019
REGISTRADO
COORDINACIÓN GENERAL DE FORMACIÓN BÁSICA

Equipo de diseño de PUA

J. Reyes Juárez Ramírez
José Martín Olguín Espinoza

Vo.Bo. de Subdirectores de Unidades Académicas
Alejandro Mungaray Moctezuma
Humberto Cervantes de Ávila
Rocío Alejandra Chávez Santoscoy

Fecha: 17 de octubre de 2019

II. PROPÓSITO DE LA UNIDAD DE APRENDIZAJE

La finalidad de la asignatura Ingeniería de Software es promover un enfoque sistemático para la creación de sistemas de información mediante el uso de técnicas de planeación de proyectos, costeo, manejo de grupos, trabajo en equipo, y medición de la calidad del producto a desarrollar; así como el uso de herramientas de modelado y programación siguiendo patrones de diseño que permitan lograr sistemas mantenibles y de alta usabilidad.

Su utilidad radica en que le permite al estudiante implementar sistemas informáticos bajo un enfoque estructurado e ingenieril, promoviendo el aseguramiento de la calidad del producto y el proceso de desarrollo. Para lo cual le habilita en la utilización del lenguaje UML para el modelado de un sistema de información, un lenguaje de alto nivel para implementar dicho sistema, frameworks de desarrollos, y herramientas de edición de texto para su documentación.

La unidad de aprendizaje se imparte en la terminal, es de carácter obligatorio; pertenece al área de conocimiento Ingeniería Aplicada.

III. COMPETENCIA DE LA UNIDAD DE APRENDIZAJE

Implementar sistemas de software que satisfagan las necesidades de una organización y las expectativas de los usuarios finales, utilizando técnicas de análisis, modelado e implementación, tecnologías de vanguardia y las especificaciones de requerimientos y de diseño, para lograr aplicaciones escalables, mantenibles y de alto nivel de usabilidad con un enfoque sistemático, con actitud crítica, reflexiva y propositiva.

IV. EVIDENCIA(S) DE DESEMPEÑO

Elabora y entrega los reportes de prácticas de especificación de requerimientos, análisis y diseño de un sistema de software para una organización.
Desarrolla proyecto de un sistema prototipo que incluya códigos fuentes y ejecutables, manual de usuario, plan de pruebas y reporte de pruebas de un sistema de software para una organización.

V. DESARROLLO POR UNIDADES

UNIDAD I. Fases del proceso de desarrollo de software

Competencia:
Analizar las diferentes fases de desarrollo de software, mediante la identificación de las actividades y productos resultantes de cada una de ellas, así como las relaciones entre estas, para visualizar el proceso de creación de sistemas de software en forma sistemática y organizada en apego a los principios, alcances y orientación de cada una de las fases, con una actitud crítica y reflexiva.

Contenido: **Duración:** 3 horas

1.1. Análisis de requerimientos
- 1.1.1. Tipos de requerimientos.
- 1.1.2. Técnicas de recolección de requerimientos.
- 1.1.3. Validación de requerimientos.
- 1.1.4. Administración de los requerimientos.

1.2. Diseño
- 1.2.1. Niveles de diseño: alto nivel, bajo nivel.
- 1.2.2. Arquitectura lógica.
- 1.2.3. Arquitectura física.
- 1.2.4. Diseño de la GUI.

1.3. Implementación
- 1.3.1. Implementación apegada a los requerimientos.
- 1.3.2. Administración de la configuración del software.

1.4. Pruebas
- 1.4.1. Tipos de pruebas
- 1.4.2. Planes de pruebas.
- 1.4.3. Ejecución de pruebas

1.5. Mantenimiento
- 1.5.1. Tipos de mantenimiento.
- 1.5.2. Planes y guías de mantenimiento.

UNIDAD II. Modelos tradicionales y metodologías de desarrollo de software

Competencia:
Analizar las fases y estructura de procesos de los modelos y metodologías de desarrollo de software, mediante la identificación de sus elementos esenciales tales como los ciclos, actividades, conjunto de roles, entregables e interacción con el cliente, para visualizar sus ventajas y desventajas, así como sus escenarios de aplicación, con una actitud crítica y reflexiva.

Contenido: **Duración:** 3 horas

2.1. Modelos tradicionales
- 2.1.1. Cascada.
- 2.1.2. Espiral.
- 2.1.3. Iterativo.

2.2. Metodologías ágiles
- 2.2.1. Manifiesto de las metodologías ágiles.
- 2.2.2. Programación extrema.
- 2.2.3. Método de Desarrollo de Sistemas Dinámicos (DSDM, por sus siglas en inglés).
- 2.2.4. SCRUM.

2.3. ESSENCE
- 2.3.1. El kernel.
- 2.3.2. Métodos y Composición de tareas.
- 2.3.3. Elementos básicos de la Ingeniería de Software desde ESSENCE.
- 2.3.4. Los esfuerzos: Equipo, trabajo, forma de trabajo.
- 2.3.5. Lenguaje de la Ingeniería de Software desde ESSENCE: Alphas, estado de un Alpha, productos de trabajo.
- 2.3.6. Controles y técnicas de seguimiento en ESSENCE.

UNIDAD III. Modelos de procesos para el desarrollo de software

Competencia:
Analizar las fases y estructura de los modelos de procesos, mediante la identificación de sus elementos esenciales tales como las prácticas genéricas y específicas, actividades, conjunto de roles y entregables, para visualizar sus ventajas y desventajas, así como los escenarios donde es más conveniente utilizarlos, con actitud crítica, reflexiva y con respeto.

Contenido: **Duración:** 3 horas

3.1. PSP
- 3.1.1. Los principios de PSP.
- 3.1.2. Estructura de PSP.
- 3.1.3. Creación y seguimiento de planes de proyecto.
- 3.1.4. Planeación y seguimiento de la calidad del software.
- 3.1.5. El manejo de datos de medición en PSP.
- 3.1.6. Factibilidad de la aplicación de PSP.

3.2. TSP
- 3.2.1. Los principios de TSP.
- 3.2.2. Estructura de TSP.
- 3.2.3. Integración del equipo de trabajo.
- 3.2.4. Roles del equipo de trabajo.
- 3.2.5. Liderazgo de un equipo de trabajo.
- 3.2.6. Control de procesos: manejo de estadísticas.
- 3.2.7. Creación y seguimiento de planes de proyecto.
- 3.2.8. Planeación y seguimiento de la calidad del software.

3.3. CMMI
- 3.3.1. Los orígenes y principios de CMM.
- 3.3.2. Áreas de procesos en CMMI.
- 3.3.3. Representaciones de CMMI.
- 3.3.4. Niveles de madurez en CMMI.
- 3.3.5. Ventajas y desventajas de la aplicación de CMMI.

3.4. MoProSoft
- 3.4.1. Los orígenes y principios de MoProSoft.
- 3.4.2. Las capas de procesos de MoProSoft.
- 3.4.3. Los nueve procesos en MoProSoft.
- 3.4.4. El proceso de evaluación de la madurez: EvalProSoft.
- 3.4.5. Ventajas y desventajas de la aplicación de MoProSoft.

UNIDAD IV. Los aspectos sociales de la Ingeniería de Software

Competencia:
Analizar los aspectos sociales y los factores humanos que involucra la Ingeniería de Software, identificando los diferentes roles y sus funciones con base en las recomendaciones de metodologías específicas, con el fin de lograr equipos de trabajo eficientes y auto-organizados, con una actitud crítica, de manera responsable y con respeto.

Contenido: **Duración:** 3 horas

4.1 Manejo de roles
- 4.1.1. Principales roles en el desarrollo de software: Administrador del Proyecto, Analista, Diseñador, Programador, Administrador de la configuración, Documentador.
- 4.1.2. Roles complementarios: Ingeniero de requerimientos, Ingeniero de Calidad, Ingeniero de pruebas, “Tester”, Ingeniero de procesos, Ingeniero de Verificación y Validación.

4.2. El equipo de trabajo
- 4.2.1. Integración del equipo de trabajo.
- 4.2.2. El rol del líder.
- 4.2.3. Manejo de personalidades difíciles: clientes/usuarios complicados; individuos conflictivos, manejo de conflictos.
- 4.2.4. Conjunto de roles propuestos por metodologías específicas: SCRUM.

4.3. Aspectos humanos y de motivación
- 4.3.1. Interacción en los grupos de trabajo.
- 4.3.2. Aspectos de comunicación y socialización.
- 4.3.3. Factores de la motivación: Autorrealización, reconocimiento, pertenencia, seguridad, fisiológicos.
- 4.3.4. Códigos de ética y de la práctica profesional en la Ingeniería de Software: ACM, IEEE.
- 4.3.5. Gestión del conocimiento humano.
- 4.3.6. La Internet y social media como apoyo a la Ingeniería de Software.

UNIDAD V. Gestión de proyectos de software

Competencia:
Formular un plan de un proyecto de software, basado en el análisis de factibilidad, la estimación de costos y de riesgos, esto con el fin de asegurar desarrollos en tiempo, en costo y que satisfagan las necesidades del cliente, con una actitud crítica, reflexiva y con responsabilidad en la gestión de recursos y el manejo de riesgos.

Contenido: **Duración:** 4 horas

5.1. Análisis de factibilidad
- 5.1.1. Desarrollo de una propuesta de solución.
- 5.1.2. Factibilidad técnica.
- 5.1.3. Factibilidad económica.
- 5.1.4. Factibilidad operativa.

5.2. Planeación de proyectos de software.
- 5.2.1. Planificación de tareas.
- 5.2.2. Técnica de la ruta crítica.
- 5.2.3. Integración de calendario de trabajo.
- 5.2.4. Integración del plan de proyecto.

5.3. Manejo de riesgos.
- 5.3.1. Tipos de riesgos.
- 5.3.2. Identificación de riesgos: magnitud, impacto.
- 5.3.3. Estimación de riesgos: análisis cualitativo, análisis cuantitativo.
- 5.3.4. Estrategias de anulación, estrategias de mitigación.
- 5.3.5. Plan de manejo de riesgos.

5.4. Técnicas de estimación de costos
- 5.4.1. Aspectos a considerar: esfuerzo, tiempo, costo.
- 5.4.2. Factores significativos: tamaño del producto, capacidad del equipo de desarrollo.
- 5.4.3. Determinación del tamaño: Líneas de código, puntos de función.
- 5.4.4. Determinación de complejidad en metodologías ágiles: Puntos de Historia.
- 5.4.5. El modelo por puntos de función y el método de COCOMO.

5.5. Técnicas de seguimiento y control de proyectos.
- 5.5.1. Los factores críticos del éxito de los proyectos.
- 5.5.2. Evaluación del avance del proyecto: comparación de datos.
- 5.5.3. Análisis del presupuesto del proyecto: estado actual.
- 5.5.4. Herramientas tecnológicas de control y seguimiento de proyectos.
- 5.5.5. Técnicas de seguimiento en ESSENCE: Los Alfas, el juego de cartas

VI. ESTRUCTURA DE LAS PRÁCTICAS DE TALLER				
No. de Práctica	**Competencia**	**Descripción**	**Material de Apoyo**	**Duración**
UNIDAD I				
1	Analizar la fase de requerimientos dentro del proceso de desarrollo de software, mediante la identificación de la importancia, su contenido y las distintas formas de expresar los requerimientos, para comprender dicho proceso de forma independiente a los modelos/metodologías existentes, con actitud crítica, reflexiva y responsabilidad.	1. El docente explica qué es un requerimiento dentro del contexto de desarrollo de software. 2. El docente explica la importancia que tienen los requerimientos en cualquiera de los modelos y metodologías de desarrollo. 3. El docente explica las diferentes formas de expresar los requerimientos. 4. El docente propociona la descripción de la práctica a realizar. 5. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 6. El alumno identifica las características de los requerimientos en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 7. El alumno identifica las fuentes de generación de requerimientos y a los interesados en los requerimientos. 8. El alumno identifica las formas y lenguajes de expresar los requerimientos. 9. El alumno identifica el ciclo de administración de los	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas. Equipo: Computadora, Conexión a Internet. Herramientas software: Editor de texto.	3 horas

		requerimientos. 10. El alumno prepara un reporte con la explicación de cada uno de los elementos de los requerimientos, con base a las indicaciones de la práctica.		
2	Analizar la fase de diseño dentro del proceso de desarrollo de software, mediante la identificación de la importancia, su contenido y las distintas formas de expresar el diseño, para comprender dicho proceso de forma independiente a los modelos/metodologías existentes, con actitud crítica, reflexiva y con responsabilidad.	1. El docente explica conceptos generales del diseño dentro del contexto de desarrollo de software: alto nivel, bajo nivel. 2. El docente explica la importancia que tienen el diseño en cualquiera de los modelos y metodologías de desarrollo. 3. El docente explica las diferentes formas de expresar el diseño. 4. El docente propociona la descripción de la práctica a realizar. 5. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 6. El alumno identifica las características del diseño en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 7. El alumno identifica las entradas para la fase de diseño (los “requerimientos”) y a los involucrados en la realización del diseño. 8. El alumno identifica los niveles de diseño: alto nivel, bajo nivel. 9. El alumno identifica las formas y lenguajes de expresar el diseño	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas. Equipo: Computadora, Conexión a Internet. Herramientas software: Editor de texto.	3 horas

		en los diferentes niveles. También identifica las herramientas para realizar el diseño. 10. El alumno prepara un reporte con la explicación de cada uno de los elementos del diseño, con base a las indicaciones de la práctica.		
3	Analizar la fase de pruebas dentro del proceso de desarrollo de software, mediante la identificación de la importancia, los distintitos tipos y su expresión en planes de pruebas, para comprender dicho proceso de forma independiente a los modelos/metodologías existentes, con actitud crítica, reflexiva y responsabilidad.	1. El docente explica conceptos generales de las pruebas dentro del contexto de desarrollo de software. 2. El docente explica la importancia que tienen las pruebas de software en cualquiera de los modelos y metodologías de desarrollo. 3. El docente enuncia las diferentes tipos de pruebas. 4. El docente explica qué es un plan de pruebas. 5. El docente propociona la descripción de la práctica a realizar. 6. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 7. El alumno identifica las características de las pruebas en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 8. El alumno identifica las entradas para la fase de pruebas (los “requerimientos”) y a los involucrados en la realización de	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas. Equipo: Computadora, Conexión a Internet Herramientas software: Editor de texto.	3 horas

		las pruebas. 9. El alumno identifica los tipos y niveles de pruebas. 10. El alumno identifica las formas y lenguajes de expresar los planes de prueba y los reportes de realización 11. El alumno prepara un reporte con la explicación de cada uno de los elementos de las pruebas, con base a las indicaciones de la práctica.		
UNIDAD II				
4	Analizar los modelos tradicionales de desarrollo de sofware, para valorar su utilidad, aplicabilidad y eficiencia en proyectos reales, identificarndo las fases, roles, entradas, entregables y controles de calidad, haciendo una comparación uno a uno entre los distintos modelos, con actitud crítica, reflexiva y respeto.	1. El docente enuncia los modelos tradicionales de desarrollo de software. 2. El docente enuncia las características principales de los modelos tradicionales. 3. El docente propociona la descripción de la práctica a realizar. (Ver Manual de Prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 5. El alumno identifica las características principales de los distintos modelos tradicionales en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno realiza una comparación entre los los distintos modelos tradicionales. 7. El alumno prepara un reporte	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas. Equipo: Computadora, Conexión a Internet Herramientas software: Editor de texto.	2 horas

		con la comparación entre los distintos modelos tradicionales.		
5	Analizar las diferentes metodologías ágiles de desarrollo de sofware, para valorar su utilidad, aplicabilidad y eficiencia en proyectos reales, identificarndo las fases, roles, entradas, entregables y controles de calidad, mediante una comparación uno a uno entre las distintas metodologías, con actitud crítica, reflexiva y respeto.	1. El docente enuncia las metodologías ágiles de desarrollo de software. 2. El docente enuncia las características principales de las metodologías ágiles. 3. El docente propociona la descripción de la práctica a realizar. (Ver Manual de Prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 5. El alumno identifica las características principales de las metodologías ágiles en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno realiza una comparación entre las metodologías ágiles. 7. El alumno prepara un reporte con la comparación entre los los distintos modelos tradicionales.	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas. Equipo: Computadora, Conexión a Internet Herramientas software: Editor de texto.	2 horas
6	Analizar los diferentes lenguajes y técnicas de especificación de los métodos y prácticas de desarrollo de software, para valorar sus capacidades y seleccionar el más adecuado para expresar las tareas a realizar, las competencias de los roles y los entregables, mediante una comparación de sus componentes,	1. El docente enuncia los diferentes lenguajes y técnicas de especificación de los métodos y prácticas de desarrollo de software. 2. El docente enuncia las características principales de los diferentes lenguajes y técnicas de especificación de los métodos y prácticas de desarrollo de		2 horas

	sintaxis de expresión y forma de comunicarlos, con actitud crítica, reflexiva y respeto.	software, enfatizando en ESSENCE. 3. El docente propociona la descripción de la práctica a realizar. (Ver Manual de Prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 5. El alumno identifica las características principales de los diferentes lenguajes y técnicas de especificación de los métodos y prácticas de desarrollo de software en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno realiza una comparación entre los diferentes lenguajes y técnicas de especificación de los métodos y prácticas de desarrollo de software. 7. El alumno prepara un reporte con la comparación entre los diferentes lenguajes y técnicas de especificación de los métodos y prácticas de desarrollo de software.		
UNIDAD III				
7	Seleccionar las prácticas y controles de PSP y TSP aplicables a un proyecto de sofware, para valorar y mejorar el desempeño individual y del equipo de trabajo, a partir de un análisis de los elementos y principios de estos dos	1. El docente enuncia los conceptos generales de PSP Y TSP, enfatizando en los principios de estos procesos. 2. El docente enuncia los componentes principales	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas. Equipo: Computadora,	3 horas

	procesos de desarrollo, con una actitud crítica, reflexiva y de respeto.	(prácticas y controles) de PSP y TSP. 3. El docente propociona la descripción de la práctica a realizar. (Ver Manual de Prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 5. El alumno identifica las características principales de PSP y TSP en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno realiza una comparación entre PSP y TSP, destacando en la forma en que se complementan. 7. El alumno selecciona las prácticas y controles de PSP y TSP que desde su punto de vista son aplicables a un proyecto de sosftware. 8. El alumno prepara un reporte con la propuesta de las prácticas y controles seleccionados de PSP y TSP.	Conexión a Internet Herramientas software: Editor de texto.	
8	Seleccionar las prácticas y controles de CMMI aplicables a un proyecto, para optimizar la mejora de procesos y fomentar comportamientos productivos y eficientes que disminuyan los riesgos en el desarrollo de software, a partir de un análisis de los elementos y principios de este modelo de procesos, con una actitud crítica, reflexiva y con	1. El docente enuncia los conceptos generales de CMMI, enfatizando en los principios de este modelo de procesos. 2. El docente enuncia los componentes principales (áreas de proceso, prácticas y controles) de CMMI. 3. El docente propociona la	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas. Equipo: Computadora, Conexión a Internet Herramientas software:	3 horas

	respeto.	descripción de la práctica a realizar. (Ver Manual de Prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 5. El alumno identifica las características principales de CMMI en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno realiza una comparación entre las áreas de procesos más representativas de CMMI acordes al contexto de un proyecto académico, identificando la forma en que se pueden implementar. 7. El alumno selecciona las prácticas y controles de CMMI que desde su punto de vista son aplicables a un proyecto de sosftware realizado por estudiantes. 8. El alumno prepara un reporte con la propuesta de las prácticas y controles seleccionados de CMMI.	Editor de texto.	
9	Seleccionar las prácticas y controles de MoProSoft aplicables a un proyecto, para optimizar la mejora de procesos de desarrollo y el mantenimiento de productos software, a partir de un análisis de los elementos y principios de este modelo de procesos, con una actitud crítica, reflexiva y con respeto.	1. El docente enuncia los conceptos generales de MoProSoft, enfatizando en los principios de este modelo de procesos. 2. El docente enuncia los componentes principales (áreas de proceso, prácticas y controles) de MoProSoft.	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas. Equipo: Computadora, Conexión a Internet Herramientas software: Editor de texto.	2 horas

		3. El docente propociona la descripción de la práctica a realizar. (Ver Manual de Prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 5. El alumno identifica las características principales de MoProSoft en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno realiza una comparación entre las los procesos más representativos de MoProSoft acordes al contexto de un proyecto académico, identificando la forma en que se pueden implementar. 7. El alumno selecciona los procesos, las prácticas y controles de MoProSoft que desde su punto de vista son aplicables a un proyecto de sosftware realizado por estudiantes. 8. El alumno prepara un reporte con la propuesta de las prácticas y controles seleccionados de MoProSoft.		
UNIDAD IV				
10	Analizar las principales actividades realizadas por el humano dentro del desarrollo de software, identificando un conjunto de roles necesarios	1. El docente enuncia las principales actividades realizadas por el humano el proceso de desarrollo de software.	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas.	3 horas

	enfatizando en los principales, tales como ingeniero de requerimientos, analista, arquitecto, programador y líder de equipo, reconociendo los conocimientos y habilidades que debe tener quien ocupa un determinado rol, con actitud crítica, reflexiva.	2. El docente enuncia los principales roles identificados en el proceso de desarrollo de software. 3. El docente propociona la descripción de la práctica a realizar. (Ver Manual de Prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica). 5. El alumno identifica las principales funciones realizadas por el humano en el proceso de desarrollo de software, en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno realiza una comparación entre las diferentes funciones de cada rol. 7. El alumno prepara un reporte con las diferentes funciones de cada rol en el proceso de desarrollo de software.	Equipo: Computadora, Conexión a Internet Herramientas software: Editor de texto.	
11	Formular un plan de roles, para integrar un equipo de trabajo auto-organizado y eficiente acorde a una metodología/modelo, mediante el análisis de las propuestas de roles de cada metodología/modelo y las características de los posibles integrantes del equipo, con actitud crítica, reflexiva y de respeto.	1. El docente enuncia la necesidad de integrar un plan de roles con funciones específicas 2. El docente enfatiza la necesidad de la correspondencia de las capacidades de los integrantes del equipo con la naturaleza de cada rol. 3. El docente propociona la descripción de la práctica a realizar. (Ver Manual de Prácticas).	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas. Equipo: Computadora, Conexión a Internet Herramientas software: Editor de texto.	3 horas

		4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica). 5. El alumno identifica en forma detallada las funciones de cada rol dentro de una metodología/modelo seleccionada, en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno realiza un recuento de los conocimientos y habilidades de cada integrante del equipo de trabajo para hacer una asignación de rol. 7. El alumno prepara un reporte con el plan de roles del equipo de trabajo para el desarrollo de un proyecto de software.		
12	Formular un plan de comunicación, para lograr equipos de trabajo altamente cohesivos, considerando técnicas y herramientas de comunicación eficaces y de actualidad que permitan la interacción efectiva en forma presencial y a distancia, incluyendo herramientas de social media, con una actitud reflexiva, propositiva y emprendedora.	1. El docente explica la necesidad e importancia de una comunicación adecuada en el equipo de trabajo y para un proyecto. 2. El docente enfatiza los elementos necesarios para una comunicación efectiva. 3. El docente propociona la descripción de la práctica a realizar. (Ver Manual de Prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica). 5. El alumno identifica los	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas. Equipo: Computadora, Conexión a Internet Herramientas software: Editor de texto, herramientas de comunicación y social media.	3 horas

		elementos y mecanismos para una comunicación efectiva, en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno identifica un conjunto de herramientas para lograr la comunicación efectiva tanto en forma presencial como a distancia. 7. El alumno formula un plan de comunicación con todos los "stakeholders" del proyecto y para el equipo de desarrollo. 8. El alumno prepara un reporte con el plan de comunicación para un proyecto de software.		
UNIDAD V				
13	Estimar la factibilidad técnica de un proyecto de software, para asegurar la realización del mismo en tiempo y con calidad para satisfacer las necesidades del cliente, mediante la aplicación de técnicas de análisis acordes al tipo y complejidad del proyecto, con una actitud crítica, reflexiva y responsable.	1. El docente enuncia los conceptos generales del análisis de factibilidad técnica. 2. El docente enuncia los aspectos y componentes a considerar en un análisis de factibilidad técnica. 3. El docente propociona la descripción de la práctica a realizar. (Ver Manual de Prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 5. El alumno identifica los aspectos y componentes a considerar en un análisis de factibilidad técnica en los apuntes	Material didáctico: Apuntes del curso, literatura a consultar, recursos Web. Manual de prácticas. Equipo: Computadora, Conexión a Internet Herramientas software: Editor de texto, Hoja de Cálculo.	4 horas

		del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno, tomando en consideración un proyecto de software a desarrollar, identifica los aspectos y componentes para factibilidad técnica. 7. El alumno, tomando en consideración un proyecto de software a desarrollar, realiza el análisis de factibilidad técnica. 8. El alumno prepara un reporte con el análisis de factibilidad técnica.		
14	Estimar un proyecto de software en términos de tamaño, esfuerzo y complejidad, con el fin de maximizar el aprovechamiento de recursos, las utilidades y minimizar los riesgos, empleando técnicas de estimación acordes a la metodología seleccionada para el desarrollo del software, con actitud crítica, reflexiva y con responsabilidad.	1. El docente enuncia los conceptos generales de la estimación de proyectos. 2. El docente enuncia los aspectos para la estimación, enfatizando en: tamaño, complejida, esfuerzo, costo. 3. El docente propociona la descripción de la práctica a realizar. (Ver Manual de Prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 5. El alumno identifica los aspectos y elementos a considerar en la estimación en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno, tomando en consideración un proyecto de	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet Herramientas software: Hoja de cálculo, Editor de texto.	4 horas

		software a desarrollar, los aspectos y componentes para factibilidad técnica. 7. El alumno, tomando en consideración un proyecto de software a desarrollar, realiza la estimación del proyecto. 8. El alumno prepara un reporte con la estimación realizada.		
15	Formular el plan de realización de un proyecto de software, para guiar la ejecución en términos de calendario de trabajo, recursos humanos requeridos, entradas y entregables, empleando técnicas de planeación de proyectos acordes a la metodología seleccionada para el desarrollo del software, con actitud crítica, reflexiva y responsabilidad.	1. El docente enuncia los conceptos generales de la planeación de un proyecto. 2. El docente enuncia los aspectos y elementos a considerar para formular un plan de realización de proyecto. 3. El docente propociona la descripción de la práctica a realizar. (Ver manual de prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 5. El alumno identifica los aspectos y componentes a considerar para formular el plan de realización de proyecto, en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno, tomando en consideración un proyecto de software a desarrollar, identifica los aspectos y componentes para la formulación del plan de realización de proyecto. 7. El alumno, tomando en	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet Herramientas software: Hoja de cálculo, Software para ruta crítica, Herramienta de administración de proyectos (Dot Project, Microsoft Project), Editor de texto.	4 horas

		consideración un proyecto de software a desarrollar, realiza la formulación del plan de realización de proyecto. 8. El alumno prepara un reporte con el plan de realización de proyecto.		
16	Formular un plan de manejo de riesgos para un proyecto de software, con el fin de precisar estrategias que permitan disminuir interferencias y lograr el proyecto en tiempo, costo y con la calidad requerida por el cliente, empleando técnicas de identificación de amenazas y creación de contingencias, con actitud crítica, reflexiva y con responsabilidad.	1. El docente enuncia los conceptos generales del manejo de riesgos. 2. El docente enuncia los aspectos a considerar en un plan de manejo de riesgos. 3. El docente propociona la descripción de la práctica a realizar. (Ver Manual de Prácticas). 4. El alumno lee la descripción de la práctica en el manual de prácticas. (Ver la descripción específica de la práctica) 5. El alumno identifica los aspectos a considerar en un plan de manejo de riesgos, en los apuntes del curso y otras fuentes bibliográficas, según lo requerido en la práctica. 6. El alumno, tomando en consideración un proyecto de software a desarrollar, identifica los aspectos y elementos para formular el plan de manejo de riesgos. 7. El alumno, tomando en consideración un proyecto de software a desarrollar, formula el plan de manejo de riesgos. 8. El alumno prepara un reporte	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet Herramientas software: Hoja de cálculo, Editor de texto.	4 horas

		con el Plan de Manejo de Riesgos.		

VII. MÉTODO DE TRABAJO

Encuadre: El primer día de clase el docente debe establecer la forma de trabajo, criterios de evaluación, calidad de los trabajos académicos, derechos y obligaciones docente-alumno.

Estrategia de enseñanza (docente)

- Exposición de temas y conceptos por medios electrónicos.
- Demostraciones de técnicas de integración de equipos.
- Demostraciones de planeación y desarrollo de proyecto.

Estrategia de aprendizaje (alumno)

- Participación para mostrar aprendizaje de los temas contenidos en cada unidad.
- Realización de prácticas en el laboratorio de cómputo mediante las cuales se pueda fortalecer y afianzar el conocimiento, trabajando en equipo y usando computadoras personales y herramientas que permitan el modelado, creación, compilación y ejecución de sistemas de información.
- Presentación de entregables relacionados con prácticas realizadas y el proyecto, los cuales permitan visualizar claramente las soluciones dadas a los problemas planteados y el proyecto.

VIII. CRITERIOS DE EVALUACIÓN

La evaluación será llevada a cabo de forma permanente durante el desarrollo de la unidad de aprendizaje de la siguiente manera:

Criterios de acreditación

- Para tener derecho a examen ordinario y extraordinario, el estudiante debe cumplir los porcentajes de asistencia que establece el Estatuto Escolar vigente.
- Calificación en escala del 0 al 100, con un mínimo aprobatorio de 60.

Criterios de evaluación

- Participación en clase……10%
- Evaluaciones parciales……40%
- Realización de prácticas y entregables ……20%
- Evidencia de desempeño……30%
  (Entrega de un proyecto de un sistema de información)

Total…….. 100%

IX. REFERENCIAS

Básicas

Gruhn, V. y Striemer, R. (Eds.). (2018) .*The Essence of Software Engineering*. Kindle Edition: Springer.

Jacobson, I. Lawson, H., NG, Pan-We. (2019). *The Essentials of Modern Software Engineering: Free the Practices from the Method Prisons*! (ACM Books). i. ACM Books.

Pressman, R.S. y Maxim, B. (2014). *Software Engineering: A Practitioner's Approach* (8ª ed.). Nueva York: McGraw-Hill Education. [clásica]

Sadowski, C. Zimmermann, T. (Eds.). (2019). *Rethinking Productivity in Software Engineering* Edición; Kindle edition.

Sommerville, I. (2016). *Software Engineering* (10ª ed). Edimburgo, Escocia: Pearson Education Limited.

Stephens, R. (2015). *Beginning Software Engineering*. Estados Unidos: John Wiley & Sons, Inc.

Thayer R. H. y Dorfman, M. (2013*). Software Engineering Essentials, Volume II: The Supporting Processes: A Detailed Guide to the IEEE SWEBOK and the IEEE CSDP/CSDA Exam* (4ª ed.). California, Estados Unidos: Software Management Training Press. [clásica]

Tsui, F., Karam, O., y Bernal, B. (2016). *Essentials of Software Engineering* (4ª ed.). Massachusetts, Estados Unidos: Jones & Bartlett Learning.

Complementarias

Cohn, M. (2019). *Succeeding with Agile: Software Development Using Scrum*. Masachusetts, Estados Unidos: Addison-Wesley Professional.

Xiang, D., Neylon, B., XIANG, D. (2018). *Software Developer Life: Career, Learning, Coding, Daily Life, Stories* -Audiolibro. DAVID XIANG (Ed.)

Martin, R.C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.

Oktaba, H., Alquicira, C., Ramos, A.S., Martínez, A. Quintanilla, G, Ruvalcaba, M., López, F., Rivera, M., Flores, M.A. (Eds.). (2015). *Modelo de Procesos para la Industria de Software, MoProSoft*, Versión *1.3*. México: Secretaría de Economía.

Pomeroy-Huff, M., Cannon, R., Chick, T.A., Mullaney, J. L., and Nichols, W. (2009). *The Personal Software Process (PSP) Body of Knowledge, Version 2.0.* Pensilvania, Estados Unidos: Software Engineering Institute. [clásica]

X. PERFIL DEL DOCENTE

El docente que imparta Ingeniería de Software debe contar con título de nivel licenciatura en ingeniería de software o área afín; preferentemente con posgrado (maestría y doctorado), con experiencia docente en el área, experiencia laboral y práctica en el campo disciplinar, liderazgo en el campo disciplinar; y debe poseer cualidades de liderazgo, capacidad de dirección de proyectos, comunicación, capacidad de motivación, emprendimiento, e innovador.