1. Datos Generales de la asignatura

Nombre de la asignatura:	Ingeniería de Software
Clave de la asignatura:	CDC-2410
SATCA[1]:	2-2-4
Carrera:	Ingeniería en Ciencia de Datos

2. Presentación

Caracterización de la asignatura

La asignatura de Ingeniería de Software contribuye al perfil de egreso proporcionando a los estudiantes las habilidades y conocimientos necesarios para desarrollar software de calidad, cumpliendo con estándares profesionales y adaptándose a las demandas del mercado. Esto incluye competencias en análisis, diseño, implementación, pruebas y mantenimiento de sistemas de software, así como habilidades para trabajar en equipo, comunicarse efectivamente y adaptarse a los cambios tecnológicos.

La Ingeniería de Software es crucial en la actualidad debido a la creciente dependencia de la tecnología en todos los aspectos de la vida y los negocios. Un software bien diseñado y desarrollado puede mejorar la eficiencia, la productividad y la calidad de vida, mientras que el software deficiente puede causar problemas graves, desde pérdidas económicas hasta riesgos para la seguridad y la salud. Por lo tanto, entender los principios y prácticas de la ingeniería de software es fundamental para garantizar el éxito en el desarrollo de sistemas de software.

La asignatura de Ingeniería de Software se centra en los principios, métodos y técnicas para desarrollar software de manera sistemática y eficiente. Esto incluye temas como la gestión de requisitos, el modelado de sistemas, la arquitectura de software, el diseño orientado a objetos, las metodologías de desarrollo de software (como Scrum o DevOps), la calidad del software y la gestión de proyectos de software. Los estudiantes también pueden aprender sobre herramientas y tecnologías específicas utilizadas en el desarrollo de software.

La Ingeniería de Software se relaciona con varias asignaturas, como Análisis y Diseño de Sistemas, Bases de Datos, Programación, Gestión de Proyectos, Calidad de Software y Tecnologías de la Información. Estas asignaturas proporcionan los fundamentos necesarios en áreas como el análisis de problemas, el diseño de soluciones, la implementación de sistemas, la gestión de recursos y la evaluación de la calidad, que son aplicables en el contexto de la ingeniería de software. La identificación y generación de proyectos integradores pueden surgir al combinar conocimientos y habilidades de estas asignaturas para abordar problemas complejos de desarrollo de software en un contexto multidisciplinario.

[1] Sistema de Asignación y Transferencia de Créditos Académicos

Intención didáctica

La intención didáctica de la asignatura de Ingeniería de Software se enfoca en proporcionar a los estudiantes una formación integral que les permita comprender y aplicar los principios, métodos y técnicas fundamentales para el desarrollo efectivo de software.

Los contenidos de la asignatura se abordan de manera estructurada y progresiva, comenzando por los conceptos básicos y avanzando hacia temas más complejos. Se utilizan ejemplos y casos prácticos para ilustrar los conceptos teóricos y facilitar su comprensión.

Los contenidos se tratan desde un enfoque práctico y orientado a la resolución de problemas reales. Se enfatiza la importancia de entender las necesidades del usuario, analizar los requisitos del sistema y diseñar soluciones eficientes y efectivas.

Los contenidos se tratan de manera suficientemente amplia y profunda para proporcionar a los estudiantes una comprensión sólida de los principios y técnicas de la Ingeniería de Software. Se cubren aspectos clave como la gestión de requisitos, el modelado de sistemas, el diseño de software, la implementación y pruebas, la gestión de proyectos y la calidad del software.

Se resaltan actividades como la resolución de problemas, el trabajo en equipo, la comunicación efectiva y el pensamiento crítico. Los estudiantes participan en proyectos prácticos que les permiten aplicar los conocimientos adquiridos en situaciones reales y desarrollar habilidades transferibles.

El tratamiento de los contenidos de la asignatura permite el desarrollo de competencias genéricas como el trabajo en equipo, la comunicación efectiva, el pensamiento crítico, la resolución de problemas y la adaptabilidad. Estas competencias son esenciales para el éxito profesional en el campo de la Ingeniería de Software y en otros ámbitos laborales.

El docente desempeña un papel activo y facilitador en el proceso de enseñanza y aprendizaje. Proporciona orientación y apoyo a los estudiantes, estimula la participación activa en clase, fomenta el debate y la reflexión crítica, y proporciona retroalimentación constructiva sobre el desempeño de los estudiantes. Además, el docente actúa como un modelo a seguir, demostrando habilidades y actitudes profesionales relevantes para la práctica de la Ingeniería de Software.

3. Participantes en el diseño y seguimiento curricular del programa

Lugar y fecha de elaboración o revisión	Participantes	Observaciones
Instituto Tecnológico Superior de Alvarado del 21 al 23 agosto de 2023.	Representante del Instituto Tecnológico Superior de Alvarado.	Propuesta inicial.
Tecnológico Nacional de México 30 octubre 2023	Representante del Instituto Tecnológico de: Querétaro y del Instituto Tecnológico Superior de Alvarado.	Presentación de la propuesta de la carrera de Ingeniería en Ciencia de Datos.
Instituto Tecnológico de Querétaro Campus Norte del 19 al 22 de marzo 2024.	Representantes de los Institutos Tecnológicos de: Morelia, Puebla, Querétaro, Tehuacán. Instituto Tecnológico Superior de Alvarado. CENIDET. Representante de Ciencias Básica de los Institutos de: Celaya, Morelia y CIIDET.	Diseño y/o desarrollo curricular de la carrera de Ingeniería en Ciencia de Datos.
Tecnológico Nacional de México del 22 al 24 de abril del 2024	Representante del Instituto Tecnológico de Querétaro e Instituto Tecnológico Superior de Alvarado.	Contraste y ajuste de las asignaturas de Ingeniería en Ciencia de Datos con respecto a las de Ing. en Inteligencia Artificial, Ing. en Desarrollo WEB e Ing. en Ciberseguridad
Tecnológico Nacional de México del 27 al 31 de mayo del 2024.	Representantes de los Institutos Tecnológicos de: Morelia, Querétaro. Instituto Tecnológico Superior de Alvarado. CENIDET.	Consolidación curricular de la carrera de Ingeniería Ciencia de Datos

4. Competencia(s) a desarrollar

Competencia(s) específica(s) de la asignatura
Aplica modelos y/o técnicas de desarrollo de software con la finalidad de implementar sistemas eficientes en base a requerimientos específicos bajo lineamientos y estándares para el aseguramiento de calidad.

5. Competencias previas

Aplica herramientas metodológicas de investigación en la elaboración de escritos académicos, producto del desarrollo de la investigación documental en temáticas de su área, que lo habiliten para ser autónomo en la adquisición y construcción de conocimientos que fortalezcan su desarrollo profesional.

Aplica los principios generales de la administración y su proceso en las estructuras y funciones fundamentales de las organizaciones acorde a las necesidades de la misma, para contribuir sustantivamente con los procesos de planeación y toma de decisiones, con una visión crítica del contexto empresarial.

Describe los conceptos principales del paradigma de programación orientada a objetos para modelar situaciones reales.
Construye un plan de negocios para crear una empresa considerando el análisis de mercado, estudio técnico, organización, análisis financiero y estados financieros del proyecto

Requiere de competencias previas como:
Manejo de un lenguaje de modelado, dominio en el uso de herramientas CASE, uso de algún Sistema Manejador de Bases de Datos, dominio de algún lenguaje de programación orientado a objetos, identificación de las etapas del ciclo de desarrollo de sistemas y de las diferentes plataformas operativas.

6. Temario

No.	Temas	Subtemas
1	Introducción a la ingeniería del software.	1.1. Definición y objetivos de la Ingeniería de Software. 1.2. Principales desafíos y tendencias actuales en el desarrollo de software. 1.3. Los sistemas de información: concepto, características, estructuras, procesos, clasificación, ERP´s, CRM, SCM. 1.4. Metodologías ágiles (Scrum, Kanban, XP) y su aplicación en el desarrollo de software. 1.5. Introducción a la metodologías de gestión de proyectos. 1.6. Aplicación de los conceptos en su proyecto.

2	Planificación del proyecto.	2.1. Ámbito del software: recursos humanos, recursos de software reutilizables, recursos del entorno. 2.2. Métricas orientadas al tamaño, al esfuerzo y a los costos. 2.3. Aplicación de herramientas para estimación de tiempos y costos del desarrollo de software: GANTT, PERT/CPM, uso de software para la estimación de tiempos y costos. 2.4. Análisis y gestión del riesgo: estrategias, identificación, proyección, refinamiento, reducción, supervisión y gestión del riesgo. 2.5. Incorporación de elementos de planificación en proyecto específico.
3	Análisis y modelado del proyecto de software.	3.1. Técnicas de recopilación de requisitos y herramientas CASE para su uso. 3.2. Estudio de viabilidad. 3.3. Análisis de requerimientos funcionales y no funcionales. 3.4. Arquitectura del sistema basada en UML. 3.5. Incorporación del análisis y modelado en proyecto específico.
4	Calidad de Software.	4.1. Conceptos de la calidad del software y su impacto. 4.2. Modelos de calidad y ejecución: ISO, SPICE, CMMI, Moprosoft. 4.3. Métricas orientadas a la calidad. 4.4. Aseguramiento de la calidad (SQA). 4.5. Incorporación de elementos de la calidad del software en un proyecto específico.

7. Actividades de aprendizaje de los temas

1. Introducción a la ingeniería del software.	
Competencias	Actividades de aprendizaje
Específica(s): Identifica los fundamentos teóricos que integran la ingeniería de software y los sistemas de información con la finalidad de comprender la relación entre sus elementos. *Genérica(s):* • Habilidad para buscar y analizar información proveniente de fuentes diversas. • Capacidad de análisis y síntesis. • Habilidades básicas de manejo de la computadora.	• Realizar una investigación y síntesis sobre las características y elementos de la ingeniería de software. • Investigar en diferentes fuentes de información la importancia e historia de la ingeniería del software y plasmarlo en una línea de tiempo. • Gestionar información sobre el concepto y componentes del modelado de negocios para su discusión en grupo. • Gestionar información por equipo de los diferentes estándares y notaciones del modelado de negocios para su exposición al grupo. • Realizar una investigación y síntesis sobre el concepto, las características, y elementos de los sistemas de información. • Organizar juegos de rol o simulaciones en clase donde los estudiantes puedan representar diferentes roles en un equipo de desarrollo de software, como desarrolladores, analistas de negocios, gerentes de proyecto, etc. Estas actividades ayudarán a los estudiantes a comprender mejor las dinámicas de trabajo en equipo y los procesos de desarrollo de software. • Organizar visitas a empresas de tecnología o centros de desarrollo de software donde los estudiantes puedan conocer de cerca el entorno de trabajo y las prácticas de ingeniería de software en la industria. Durante la visita, los estudiantes pueden interactuar con profesionales de la industria, hacer preguntas y observar el proceso de desarrollo de software en acción.
2. Planificación del proyecto	
Competencias	Actividades de aprendizaje
Específica(s): Identifica los diferentes modelos de desarrollo de software para identificar el más adecuado y diseñar sistemas de información eficientes.	• Proporcionar a los estudiantes una serie de casos de estudio de proyectos de software reales, que abordan diferentes aspectos de la planificación del proyecto, como la gestión de requisitos, la estimación de esfuerzo, la asignación de recursos, el seguimiento del progreso y la resolución de problemas. Después de la lectura, organizar sesiones de discusión para analizar los casos y extraer lecciones aprendidas.

Genérica(s): • Capacidad de análisis y síntesis • Habilidades básicas de manejo de la computadora. • Solución de problemas. • Capacidad de aplicar conocimientos en la práctica.	• Organizar una actividad donde los estudiantes simulen la planificación de un proyecto de software en grupos pequeños. Proporciona a cada grupo un escenario ficticio y pídeles que utilicen herramientas como diagramas de Gantt, listas de tareas, y matrices de riesgos para crear un plan detallado del proyecto. Después de la simulación, discute los planes de proyecto de cada grupo y resalta los puntos clave relacionados con la planificación del proyecto. • Dividir a los estudiantes en equipos y asignar a cada equipo un proyecto de software real o ficticio para planificar. Pide a los equipos que trabajen juntos para desarrollar un plan detallado del proyecto, que incluya la definición de objetivos, la identificación de requisitos, la estimación de recursos, la programación de actividades, la asignación de responsabilidades y la elaboración de un presupuesto. Después de completar el plan, los equipos pueden presentar sus resultados en clase y recibir retroalimentación. • Solicitar a los estudiantes que investiguen y analicen diferentes herramientas de planificación de proyectos de software disponibles en el mercado, como Microsoft Project, Jira, Trello, etc. Pide a los estudiantes que comparen las características, ventajas y desventajas de cada herramienta, y que elijan la más adecuada para un proyecto específico basándose en sus necesidades y requisitos.

3. Análisis y Modelado del proyecto de software	
Competencias	Actividades de aprendizaje
Específica(s): • Planifica actividades con la finalidad de estimar tiempos y costos del proyecto de software. • Aplica métricas orientadas al tamaño para medir y controlar los recursos necesarios para el desarrollo del proyecto de software. *Genérica(s):* • Capacidad de análisis y síntesis. • Capacidad de organizar y planificar. • Solución de problemas. • Toma de decisiones. • Trabajo en equipo. • Capacidad de aplicar los conocimientos en la práctica. • Capacidad de generar nuevas ideas (creatividad).	• Proporcionar a los estudiantes una serie de casos de estudio que involucren el análisis y el modelado de proyectos de software reales. Los casos podrían incluir proyectos exitosos o fracasados, y abordar temas como la identificación de requisitos, el diseño de la arquitectura, la elaboración de diagramas UML, etc. Después de la lectura, organiza sesiones de discusión para analizar los casos y extraer lecciones aprendidas. • Organizar talleres prácticos donde los estudiantes aprendan a elaborar diferentes tipos de diagramas UML (Diagrama de Casos de Uso, Diagrama de Clases, Diagrama de Secuencia, Diagrama de Actividad, etc.). Proporcionar escenarios específicos y solicitar a los estudiantes que trabajen en grupos para crear diagramas UML que representen los diferentes aspectos del proyecto de software. • Organizar sesiones de prototipado donde los estudiantes puedan crear prototipos de baja fidelidad (bocetos, wireframes) o prototipos de alta fidelidad (maquetas interactivas) para representar la interfaz de usuario y el flujo de trabajo del proyecto de software. Los estudiantes pueden utilizar herramientas de diseño como Adobe XD, Sketch, Figma, Balsamiq, etc., para crear los prototipos y recibir retroalimentación de sus compañeros y del instructor.
4. Calidad del Software	
Competencias	Actividades de aprendizaje
Específica(s): Analiza y comprende el enfoque de calidad aplicada al software para determinar los procesos de implementación y aseguramiento de la misma en un entorno globalizado.	• Proporciona a los estudiantes una serie de casos de estudio que abordan diferentes aspectos de la calidad del software, como problemas de calidad en proyectos reales, impacto de la falta de calidad en el negocio, implementación exitosa de prácticas de calidad, etc. Después de la lectura, organiza sesiones de discusión para analizar los casos y extraer lecciones aprendidas.

Genérica(s): • Capacidad de análisis y síntesis. • Comunicación oral y escrita. • Habilidad para buscar y analizar información proveniente de fuentes diversas. • Solución de problemas. • Capacidad de aplicar los conocimientos en la práctica.	• Organizar talleres prácticos donde los estudiantes puedan revisar y analizar el código de proyectos de software reales o ficticios. Proporciona herramientas de análisis estático de código (como SonarQube, ESLint, PMD) y pide a los estudiantes que identifiquen posibles problemas de calidad, como código duplicado, falta de comentarios, violaciones de estándares de codificación, etc. • Divide a los estudiantes en equipos y asigna a cada equipo un proyecto de software para el cual deben diseñar e implementar un plan de pruebas. Los estudiantes pueden realizar pruebas unitarias, de integración, de sistema y de aceptación, utilizando diferentes técnicas de prueba (caja blanca, caja negra, pruebas exploratorias, etc.). Después de completar las pruebas, los equipos pueden presentar sus resultados en clase y discutir los hallazgos. • Proporcionar a los estudiantes un conjunto de métricas de calidad del software (como la tasa de defectos, la cobertura de código, el tiempo medio entre fallos, etc.) y pide a los estudiantes que analicen datos de proyectos de software reales para evaluar su calidad. Los estudiantes pueden utilizar herramientas de gestión de proyectos y seguimiento de defectos para recopilar y analizar los datos. • Divide a los estudiantes en equipos y pide a cada equipo que desarrolle un plan de aseguramiento de la calidad para un proyecto de software específico. El plan debe incluir actividades de aseguramiento de la calidad en todas las etapas del ciclo de vida del proyecto, desde la planificación hasta la entrega. Después de completar el plan, los equipos pueden presentar sus resultados en clase y recibir retroalimentación.

8. Práctica(s)

- Diseña proyectos prácticos que permitan a los estudiantes aplicar los conceptos teóricos en situaciones del mundo real. Por ejemplo, pueden trabajar en equipos para desarrollar una aplicación web o móvil desde cero, utilizando metodologías ágiles como Scrum.
- Proporciona estudios de caso basados en proyectos de la vida real en la industria del software. Los estudiantes pueden analizar problemas reales, identificar soluciones y discutir las implicaciones éticas y técnicas de cada decisión.
- Fomenta el trabajo colaborativo entre los estudiantes, donde cada uno aporta sus habilidades y conocimientos únicos al proyecto. Esto refleja el entorno de trabajo en la industria del software, donde el trabajo en equipo es esencial.
- Implementa sesiones regulares de revisión de código, donde los estudiantes puedan compartir su código con sus compañeros y recibir retroalimentación constructiva. Esto les ayuda a mejorar sus habilidades de programación y a aprender mejores prácticas de codificación.
- Organiza eventos de hackatones donde los estudiantes puedan trabajar en proyectos durante un período intensivo de tiempo. Esto fomenta la creatividad, la resolución de problemas y la colaboración bajo presión, habilidades importantes en la industria del software.
- Invita a profesionales de la industria del software para que den conferencias o charlas sobre temas relevantes y tendencias actuales en ingeniería de software. Esto ayuda a los estudiantes a mantenerse actualizados y a obtener perspectivas del mundo real.
- Utiliza herramientas de laboratorio virtual que permitan a los estudiantes experimentar con diferentes tecnologías y entornos de desarrollo de software de forma segura y práctica.
- Crea simulaciones o juegos de roles que imitan situaciones comunes en el desarrollo de software, como la resolución de conflictos en equipos, la gestión de proyectos y la toma de decisiones bajo presión.
- Presenta a los estudiantes problemas complejos relacionados con la ingeniería de software y guíalos en el proceso de investigación y resolución de los mismos. Esto fomenta el pensamiento crítico y la autonomía en el aprendizaje.
- Utiliza plataformas de aprendizaje en línea donde los estudiantes puedan acceder a recursos adicionales, realizar actividades interactivas y participar en discusiones con otros compañeros y profesores.

9. Proyecto de asignatura

El objetivo del proyecto que planteé el docente que imparta esta asignatura, es demostrar el desarrollo y alcance de la(s) competencia(s) de la asignatura, considerando las siguientes fases:

Fundamentación: marco referencial (teórico, conceptual, contextual, legal) en el cual se fundamenta el proyecto de acuerdo con un diagnóstico realizado, mismo que permite a los estudiantes lograr la comprensión de la realidad o situación objeto de estudio para definir un proceso de intervención o hacer el diseño de un modelo.

Planeación: con base en el diagnóstico en esta fase se realiza el diseño del proyecto por parte de los estudiantes con asesoría del docente; implica planificar un proceso: de intervención empresarial, social o comunitario, el diseño de un modelo, entre otros, según el tipo de proyecto, las actividades a realizar los recursos requeridos y el cronograma de trabajo.

- **Ejecución:** consiste en el desarrollo de la planeación del proyecto realizada por parte de los estudiantes con asesoría del docente, es decir en la intervención (social, empresarial), o construcción del modelo propuesto según el tipo de proyecto, es la fase de mayor duración que implica el desempeño de las competencias genéricas y específicas a desarrollar.

- **Evaluación:** es la fase final que aplica un juicio de valor en el contexto laboral-profesional, social e investigativo, ésta se debe realizar a través del reconocimiento de logros y aspectos a mejorar se estará promoviendo el concepto de "evaluación para la mejora continua", la metacognición, el desarrollo del pensamiento crítico y reflexivo en los estudiantes.

10. Evaluación por competencias

La evaluación debe hacerse diagnóstica, formativa y sumativa. De igual manera, para fortalecer la parte actitudinal, se recomienda guiar al estudiante hacia la introspección para utilizar la autoevaluación y la coevaluación.

En el caso de las actividades de aprendizaje se sugiere el uso de estrategias metacognitivas como: mapas conceptuales, reportes de prácticas, exposiciones en clase, ensayos, resúmenes, reportes de visitas industriales, trípticos, guías de entrevista, observación y cuestionarios. Mientras que para verificar el nivel del logro de las competencias del estudiante se recomienda utilizar: el portafolio de evidencias, listas de cotejo, rúbricas, matrices de valoración, guías de observación, además de estrategias en las que se logren las competencias blandas.

11. Fuentes de información

1. Pressman, R. S. (2014). Software engineering: a practitioner's approach (8th ed.). McGraw-Hill Education.
2. Sommerville, I. (2015). Software engineering (10th ed.). Pearson Education Limited.
3. Pfleeger, S. L., & Atlee, J. M. (2015). Software engineering: theory and practice (4th ed.). Pearson.
4. Sommerville, I. (2016). Software engineering (10th ed.). Pearson Education.
5. Roger S. Pressman, B. A. (2015). Ingeniería del software: un enfoque práctico (7th ed.). McGraw-Hill.
6. Ghezzi, C., Jazayeri, M., & Mandrioli, D. (2010). Fundamentals of Software Engineering (2nd ed.). Pearson.
7. Bass, L., Clements, P., & Kazman, R. (2012). Software architecture in practice (3rd ed.). Addison-Wesley.
8. Fowler, M. (2002). Patterns of enterprise application architecture. Addison-Wesley Professional.
9. Martin, R. C. (2008). Clean code: a handbook of agile software craftsmanship. Prentice Hall.
10. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: elements of reusable object-oriented software. Addison-Wesley.