UNIVERSIDAD AUTÓNOMA DE YUCATÁN
LICENCIATURA EN INGENIERÍA DE SOFTWARE
CONSTRUCCIÓN DE SOFTWARE

PLANEACIÓN DIDÁCTICA

DATOS GENERALES DE IDENTIFICACIÓN

Nombre de la asignatura	Construcción de Software				
Tipo	Obligatoria				
Modalidad	Mixta				
Ubicación	Quinto Semestre				
Duración total en horas	112	Horas presenciales	72	Horas no presenciales	40
Créditos	7				
Requisitos académicos previos	Ninguno				

COMPETENCIA DE LA ASIGNATURA

Desarrolla aplicaciones utilizando prácticas y técnicas de construcción de software, considerando los estándares de codificación.

CONTEXTUALIZACIÓN

Construcción de Software es importante para la formación de los estudiantes de Ingeniería de Software porque le permitirá crear código simple y legible. Esta asignatura aporta al estudiante la teoría básica relacionada con las fases de construcción y mantenimiento del software para llevar un control riguroso en el desarrollo de software.

Construcción de Software se relaciona con las asignaturas Fundamentos de Ingeniería de Software, Arquitecturas de Software, Diseño de Software, Diseño de Base de Datos, Programación Orientada a Objetos, Programación Estructurada, Algoritmia, Desarrollo de Aplicaciones Web, Métricas de Software, Aseguramiento de la Calidad del Software, Requisitos de Software, Verificación y Validación de Software, Mantenimiento de Software, Administración de Proyecto I y Administración de Proyectos II, pues contribuyen al logro de las competencias de egreso:

- Desarrolla productos de software de calidad de pequeña a gran escala aplicando técnicas, herramientas, métodos y procedimientos, a través de un enfoque sistemático, disciplinado y cuantificable.
- Mantiene productos de software heredados en diferentes dominios de la aplicación, optimizando los recursos humanos, materiales, económicos y de tiempo, y atendiendo las necesidades de la organización.

COMPETENCIAS DISCIPLINARES QUE SE MOVILIZAN EN LA ASIGNATURA

COMPETENCIAS DISCIPLINARES

- Diseña algoritmos computacionales eficientes aplicando conceptos básicos de matemáticas discretas, lógica, algoritmia y estructuras de datos.
- Resuelve problemas computacionales aplicando el conocimiento de la estructura, organización, funcionamiento, programación e interconexión de sistemas de cómputo.

UNIDADES Y COMPETENCIAS

Unidades	Competencias	Duración	
		HP	HNP
I. Fundamentos y prácticas de construcción de software	Aplica de manera eficaz, las técnicas de construcción de código, para el aseguramiento de la calidad en el mismo.	30	15
II. Refactorización de software	Genera código simple y legible, mediante la refactorización del mismo, sin alteración en el resultado de su ejecución.	30	15
III. Herramientas para la construcción	Utiliza herramientas de construcción de software durante la escritura del código, de manera ágil.	12	10
	Total	**72**	**40**

DESARROLLO DE LAS COMPETENCIAS GENÉRICAS DE LA ASIGNATURA

COMPETENCIAS GENÉRICAS	UNIDAD I	UNIDAD II	UNIDAD III
Se comunica en español en forma oral y escrita en sus intervenciones profesionales y en su vida personal, utilizando correctamente el idioma.	X	X	X
Gestiona el conocimiento en sus intervenciones profesionales y en su vida personal, de manera pertinente.	X	X	

Desarrolla su pensamiento en intervenciones profesionales y personales, de manera crítica, reflexiva y creativa.	X	X	
Toma decisiones en su práctica profesional y personal, de manera responsable.		X	X
Pone de manifiesto su compromiso con la calidad y la mejora continua en su práctica profesional y en su vida personal de manera responsable.	X	X	
Establece relaciones interpersonales, en los ámbitos en los que se desenvuelve, de manera positiva y respetuosa.			X

SECUENCIA DIDÁCTICA UNIDAD I

Unidad I	Fundamentos y técnicas de Construcción de Software
Competencia	Aplica de manera eficaz, las técnicas de construcción de código, para el aseguramiento de la calidad en el mismo.

Secuencia de contenidos	Resultados de aprendizaje	Desagregado de contenidos	Estrategias de enseñanza y aprendizaje	Actividades de aprendizaje	Duración	
				Descripción	HP	HNP
1. Introducción a la construcción de software	Identifica los principales conceptos de construcción de software, considerando su importancia y áreas de aplicación.	1. Tópicos de la construcción de software. 2. Importancia de la construcción de software.	Enseñanza por explicación y exposición Organizadores gráficos Aprendizaje autónomo y reflexivo Aprendizaje colaborativo Prácticas de programación	De manera individual realizar un reporte en el que se incluya un mapa conceptual de los elementos fundamentales de la construcción de software. En el salón de clases, participar en una lluvia de ideas. **Recursos y materiales:** • Notas de curso • Referencias 1, 5, 6. En grupos, utilizar bloques de código preexistente y medir su calidad aplicando métricas de construcción de código. Posteriormente, corregir el código aplicando técnicas y estándares de codificación, y realizar nuevamente la medición de calidad. Finalmente, elaborar un reporte en el que se incluya un cuadro comparativo entre los resultados.	3	2
2. Características de un código de calidad	Aplica de manera apropiada, técnicas de construcción de software en código preexistente, reconociendo su importancia en la construcción de software.	1. Importancia de la construcción de código de calidad. 2. Métricas básicas en la construcción de Software. 3. Estándares de codificación.			3	3

3. Técnicas de construcción de código	Utiliza técnicas y estándares de construcción de código para la elaboración de programas de software, de manera estructurada y funcional.	1. Técnicas de construcción de variables y tipos de datos fundamentales. 2. Técnicas de organización de sentencias. 3. Técnicas de construcción de estructuras de control de flujo. 4. Técnicas de construcción de procedimientos. 5. Técnicas de construcción de clases 6. Técnicas de documentación de código 7. Estándares de codificación		**Recursos y materiales:** • Notas de curso • Referencia 4 • IDE del lenguaje de programación **Prácticas de Programación** En grupos, desarrollar programas de software aplicando técnicas y estándares de construcción de código. **Recursos y materiales:** • Notas de curso • Referencia 4 y 5 • IDE del lenguaje de programación	24	10

SECUENCIA DIDÁCTICA UNIDAD II	
Unidad II	Refactorización de software
Competencia	Genera código simple y legible, mediante la refactorización del mismo, sin alteración en el resultado de su ejecución.

Secuencia de contenidos	Resultados de aprendizaje	Desagregado de contenidos	Estrategias de enseñanza y aprendizaje	Actividades de aprendizaje	Duración	
				Descripción	HP	HNP
1. Introducción a la refactorización de software	Identifica el concepto de refactorización de software, considerando su importancia y problemas asociados.	1. Definición de refactorización de software. 2. Importancia de la refactorización. 3. Problemas asociados con la refactorización.	Enseñanza por explicación y exposición Organizadores gráficos	De manera individual, utilizar bloques de código preexistente y medir su calidad aplicando métricas de construcción de código. Posteriormente, reestructurar el código aplicando prácticas de refactorización, y realizar nuevamente la medición de calidad. Finalmente, realizar un reporte en el que	3	2

2. Prácticas de refactorización	Genera código simple y legible, mediante las prácticas de refactorización de software a nivel de clases y objetos.	1. Métodos de composición. 2. Movimiento de funciones entre objetos. 3. Organización de datos. 4. Simplificación de expresiones condicionales. 5. Simplificación de las llamadas a métodos. 6. Manejo de la generalización.	Aprendizaje autónomo y reflexivo Aprendizaje colaborativo Prácticas de programación	se incluya un cuadro comparativo entre los resultados. **Recursos y materiales:** • Notas de curso • Referencia 2 • Métricas de software • IDE del lenguaje de programación	20	10
3. Grandes refactorizaciones	Aplica prácticas de refactorización de software a nivel sistema, obteniendo código simple y legible.	1. Herencias confusas. 2. Conversión del diseño procedural a objetos. 3. Separación del dominio de la presentación. 4. Extracción de jerarquías.		**Prácticas de Programación** En grupos, reestructurar programas de software aplicando prácticas de refactorización. **Recursos y materiales:** • Notas de curso • Referencia 2 • IDE del lenguaje de programación	7	3

SECUENCIA DIDÁCTICA UNIDAD III	
Unidad III	Herramientas para la construcción de software
Competencia	Utiliza herramientas de construcción de software durante la escritura del código, de manera ágil.

Secuencia de contenidos	Resultados de aprendizaje	Desagregado de contenidos	Estrategias de enseñanza y aprendizaje	Actividades de aprendizaje	Duración	
				Descripción	HP	HNP
1. Pruebas Unitarias.	Ejecuta pruebas unitarias al código de manera pertinente, para el aseguramiento de la calidad del software.	1. Introducción a las pruebas. 2. Pruebas unitarias. 3. Herramientas para ejecución de pruebas unitarias.	Enseñanza por explicación y exposición Aprendizaje autónomo y reflexivo Prácticas de programación	**Práctica** De manera individual, utilizar bloques de código preexistente y realizar las pruebas unitarias, implementar una herramienta para controlar su versión y generar la documentación del mismo. **Recursos y materiales:** • Notas de curso • JUnit • Javadoc • Git • IDE del lenguaje de programación	4	4
2. Control de versiones del código	Aplica un adecuado control de versión del código que resguarde de manera eficaz las distintas versiones del mismo, para su implementación y rescate.	2.1 Elementos básicos del control de versiones. 1. Estrategias para un buen control de versiones 2. Uso de una herramienta para el control de versiones.			5	4
3. Documentación del código	Aplica estándares de documentación de código que permita su legibilidad, entendimiento y posible modificación.	1. Elementos de la documentación del código. 2. Herramientas para generar la documentación visual del código.			3	2

EVALUACIÓN DEL DESEMPEÑO

EVALUACIÓN DE PROCESO		
Estrategia de evaluación	**Criterios de evaluación**	**Ponderación**
2 Pruebas de desempeño: Primera: Unidad 1 Segunda: Unidad 2	• Exactitud en la respuesta a preguntas planteadas. • Entrega del programa en el tiempo establecido. • Implementación de las funcionalidades del programa solicitadas. • Aplicación de estándares y prácticas de construcción y factorización de software.	20
Reportes de análisis y resultados	• Análisis de lo solicitado y congruencia con los resultados. • Redacción y ortografía. • Entrega del reporte en el tiempo establecido.	20
Resolución de ejercicios y prácticas de programación	• Descripción del análisis de los ejercicios. • Uso correcto de herramientas (IDE) para la codificación los ejercicios. • Identificación de errores (léxicos y sintácticos). • Legibilidad y documentación del código. • Uso del paradigma de programación orientada a objetos.	20

EVALUACIÓN DE PRODUCTO		
Estrategia de evaluación	**Criterios de evaluación**	**Ponderación**
Proyecto Integrador: Programa de Cómputo (Equipos de 3-5 integrantes)	• Entrega del programa en el tiempo establecido. • Aplicación de estándares y prácticas de construcción y factorización de software. • Modelado del diseño del proyecto. • Prueba de Aceptación (Requerimientos vs Programa).	40

EVALUACIÓN DEL DESEMPEÑO	
Evaluación de proceso	60%

Evaluación de producto	40%
Total	100%

DESCRIPCIÓN DE LOS NIVELES DE DOMINIO		
Puntaje	**Categoría**	**Descripción**
90 – 100	Sobresaliente (SS)	Desarrolla aplicaciones utilizando prácticas, técnicas y estándares de la construcción de software, evidenciando una excelente calidad, funcionalidad y documentación derivado del control de estos procesos. .
80 – 89	Satisfactorio (SA)	Desarrolla aplicaciones utilizando prácticas, técnicas y estándares de la construcción de software, evidenciando una calidad aceptable, funcionalidad adecuada y documentación derivado del control de estos procesos.
70 – 79	Suficiente (S)	Desarrolla aplicaciones utilizando prácticas, técnicas y estándares de la construcción de software, evidenciando cierta calidad, funcionalidad y documentación derivado del control de estos procesos
0 - 69	No acreditado (NA)	No cumple con los atributos mínimos descritos para obtener un desempeño Suficiente (S).

ACTIVIDADES QUE FOMENTAN LA FORMACIÓN INTEGRAL	
DIMENSIONES DE LA FI	**ACTIVIDADES**
Cognitiva	• Búsqueda y evaluación de información proveniente de diferentes fuentes para promover el uso del pensamiento lógico. • Refactoring de código mal estructurado que fomentan el aprendizaje autónomo y reflexivo.
Social	• Aplicar técnicas y estándares de construcción de código en forma colaborativa. • Diseño del proyecto integrador en equipos de trabajo colaborativo.
Emocional	• Participación en una sesión de reflexión para el reconocimiento y manejo de emociones negativas, derivadas del trabajo bajo presión durante la realización de proyectos.
Valoral-actitudinal	• Interacción respetuosa con los compañeros. • Desarrollo de código original para fomentar prácticas éticas. • Desarrollo de actividades en línea para evitar el uso de papel y así fomentar el cuidado del medio ambiente.
Física	• Adopción de posturas de trabajo correctas durante el uso de computadoras. • Promoción del uso de dispositivos de cómputo ergonómicos.

REFERENCIAS

1. Parnas, D. L. (2018). Software Structures: A Careful Look. *IEEE Software, 35*(6), 68-71. doi:10.1109/ms.2018.4321239
2. Meyer, Bertrand. Object Oriented Software Construction. Prentice Hall. 1997
3. Martin Fowler, et.al, Refactoring: Improving the Design of Existing Code. Addison - Wesley Professional. 1999
4. Martin Robert C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall. 2008
5. McConnell Steve. Code Complete 2nd Edition. Microsoft Press. 2009
6. Hunt Adrew. The pragmatic programmer: from Journeyman to Master. Addison-Wesley Professional Edition. 1999

PLANEACIÓN DIDÁCTICA ELABORADA POR:	FECHA DE ELABORACIÓN:
• M.I.T. Edwin Jesús León Bojorquez • M.C. Juan Francisco Garcilazo Ortíz • L.C.C Fayné Perera Perera Huchim	• 20 de junio de 2019