UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

COORDINACIÓN GENERAL DE FORMACIÓN PROFESIONAL

PROGRAMA DE UNIDAD DE APRENDIZAJE

I. DATOS DE IDENTIFICACIÓN

1. Unidad Académica: Facultad de Ingeniería, Arquitectura y Diseño, Ensenada; Facultad Ciencias Químicas e Ingeniería, Tijuana; y Facultad de Ciencias de la Ingeniería y Tecnología, Valle de las Palmas

2. Programa Educativo: Ingeniero en Software y Tecnologías Emergentes

3. Plan de Estudios: 2022-1

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

5. Clave:40010

6. HC: 02 **HT:** 01 **HL:** 02 **HPC:** 00 **HCL:** 00 **HE:** 02 **CR:** 07

7. Etapa de Formación a la que Pertenece: Disciplinaria

8. Carácter de la Unidad de Aprendizaje: Obligatoria

9. Requisitos para Cursar la Unidad de Aprendizaje: Ninguno

Equipo de diseño de PUA
Guillermo Licea Sandoval
Irma Alejandra Amaya Patrón

Vo.Bo. de subdirector(es) de Unidad(es) Académica(s)
Humberto Cervantes De Ávila
Daniela Mercedes Martínez Platas
Noemí Hernández Hernández

Fecha: 23 de febrero de 2021

II. PROPÓSITO DE LA UNIDAD DE APRENDIZAJE

La programación orientada a objetos permite al profesionista resolver problemas del mundo real, mediante la abstracción utilizando un lenguaje de programación moderno. El Paradigma orientado a objetos es uno de los más utilizados debido a su potencial para crear arquitecturas robustas, fáciles de mantener y con alto nivel de reusabilidad.

Los patrones de software apoyan la reusabilidad de código y de diseño para potenciar el paradigma orientado a objetos. A través del uso de un catálogo de patrones, el desarrollador puede reducir el tiempo de desarrollo y mejorar el diseño de la arquitectura o de los subsistemas que integran el sistema de software.

La refactorización permite mejorar la estructura interna del sistema de software sin modificar el comportamiento externo. Mediante la refactorización se busca mejorar el código escrito mejorando el diseño y reduciendo errores o fallas del sistema.

Se ubica en la etapa disciplinaria con carácter obligatorio y pertenece al área de conocimiento Métodos y Tecnologías de Software.

Se recomienda haber cursado Lenguajes de Programación Orientada a Objetos.

III. COMPETENCIA GENERAL DE LA UNIDAD DE APRENDIZAJE

Diseñar e implementar programas de computadora eficientes, a través de la selección de los patrones de software adecuados, para solucionar problemas de procesamiento de información, de manera creativa, organizada y honesta.

IV. EVIDENCIA(S) DE APRENDIZAJE

Reporte de un proyecto de software cuya estructura debe identificar los patrones seleccionados para el diseño e implementación.

V. DESARROLLO POR UNIDADES
UNIDAD I. Antecedentes de los patrones de software

Competencia:
Identificar patrones de software de acuerdo a su nivel, utilizando la definición formal de patrón de software, para integrarlos en el desarrollo de un sistema de cómputo, de manera creativa y organizada.

Contenido: **Duración:** 2 horas

1.1. Historia de los patrones de software
1.2. Definición y estructura de los patrones de software
1.3. Niveles en los patrones de software

UNIDAD II. Patrones para la arquitectura del software

Competencia:
Identificar los patrones de arquitectura más utilizados para estructurar sistemas de cómputo, mediante el análisis del sistema propuesto, para definir la arquitectura del software a desarrollar, de manera creativa y organizada.

Contenido: **Duración:** 6 horas

2.1. Definición de patrón arquitectónico
2.2. Patrones estructurales
2.3 Patrones para sistemas distribuidos
2.4. Patrones para sistemas interactivos
2.5. Patrones para sistemas adaptables

UNIDAD III. Patrones de diseño

Competencia:
Integrar patrones de diseño en el desarrollo de sistemas, utilizando los catálogos de patrones más conocidos, para dar soluciones a problemas de procesamiento de información, de manera creativa y organizada.

Contenido: **Duración:** 16 horas

3.1. Definición de patrón de diseño
3.2 Organización de los patrones de diseño
3.3 Patrón observador
3.4 Patrón decorador
3.5 Patrones fábrica
3.6 Patrón singleton
3.7 Patrón comando
3.8 Patrón adaptador y fachada
3.9 Patrón método plantilla
3.10 Patrón iterador y composición
3.11 Patrón estado
3.12 Patrón proxy
3.13 Patrones de patrones

UNIDAD IV. Refactorización

Competencia:
Aplicar los conceptos de refactorización, para mejorar la estructura interna del software, utilizando las técnicas de identificación de olores dentro y entre clases, de manera creativa y organizada.

Contenido: **Duración:** 8 horas

4.1. Definición de refactorización
4.2. Definición de olor y su clasificación
4.3. El ciclo de vida de la refactorización
4.4. Olores dentro de las clases
    4.4.1. Medibles
    4.4.2. Nombres
    4.4.3. Complejidad innecesaria
    4.4.4. Duplicado
    4.4.5. Lógica condicional
4.5. Olores entre las clases
    4.5.1. Datos
    4.5.2. Herencia
    4.5.3. Responsabilidad
    4.5.4. Ajuste al cambio
    4.5.5. Biblioteca de clases
4.6. Ejemplos de refactorización de programas

VI. ESTRUCTURA DE LAS PRÁCTICAS DE TALLER				
No.	**Nombre de la Práctica**	**Procedimiento**	**Recursos de Apoyo**	**Duración**
UNIDAD I				
1	**Identificar el nivel de los patrones**	1. Identifica el nivel de los patrones descritos por el docente. 2. Clasifica los patrones de acuerdo al nivel identificado. 3. Elabora y entrega al docente reporte que incluye los patrones clasificados.	• Pizarrón. • Presentación de los patrones a clasificar. • Papel y lápiz.	1 hora
UNIDAD II				
2	**Clasificación de sistemas de acuerdo a su arquitectura**	1. Identifica el tipo de arquitectura de un sistema. 2. Establece correspondencia con algún patrón arquitectónico. 3. Elabora y entrega al docente reporte que describe la arquitectura del sistema y su patrón correspondiente.	• Pizarrón. • Presentación de los sistemas a clasificar. • Papel y lápiz.	3 horas
UNIDAD III				
3 a 10	**Estudio de casos prácticos de utilización de patrones de diseño**	1. Identifica el propósito de cada patrón. 2. Analiza la aplicación de cada patrón en un ejemplo práctico. 3. Elabora y entrega al docente reporte de cada uno de los casos prácticos de la utilización de los patrones.	• Pizarrón. • Presentación de los casos prácticos. • Papel y lápiz.	8 horas
UNIDAD IV				
11 a 14	**Estudio de casos prácticos de utilización de refactorizaciones**	1. Identifica el propósito de cada refactorización.	• Pizarrón.	4 horas

		2. Analiza la aplicación de cada refactorización en un ejemplo práctico. 3. Elabora y entrega al docente reporte de cada uno de los casos prácticos de la utilización de las refactorizaciones.	• Presentación de los casos prácticos. • Papel y lápiz.	

VI. ESTRUCTURA DE LAS PRÁCTICAS DE LABORATORIO				
No.	**Nombre de la Práctica**	**Procedimiento**	**Recursos de Apoyo**	**Duración**
UNIDAD III				
1	**Patrón observador**	1. Programa las clases correspondientes al patrón. 2. Utiliza las clases del patrón en un caso práctico. 3. Realiza pruebas unitarias del software desarrollado. 4. Entrega código y reporte de pruebas al docente.	• Pizarrón. • Computadora. • Proyector. • Internet. • Ambiente de programación orientado a objetos.	2 horas
2	**Patrón decorador**	1. Programa las clases correspondientes al patrón. 2. Utiliza las clases del patrón en un caso práctico. 3. Realiza pruebas unitarias del software desarrollado. 4. Entrega código y reporte de pruebas al docente.	• Pizarrón. • Computadora. • Proyector. • Internet. • Ambiente de programación orientado a objetos.	2 horas
3	**Patrón fábrica**	1. Programa las clases correspondientes al patrón. 2. Utiliza las clases del patrón en un caso práctico.	• Pizarrón. • Computadora. • Proyector. • Internet.	2 horas

		3. Realiza pruebas unitarias del software desarrollado. 4. Entrega código y reporte de pruebas al docente.	• Ambiente de programación orientado a objetos.	
4	**Patrón singleton**	1. Programa las clases correspondientes al patrón. 2. Utiliza las clases del patrón en un caso práctico. 3. Realiza pruebas unitarias del software desarrollado. 4. Entrega código y reporte de pruebas al docente.	• Pizarrón. • Computadora. • Proyector. • Internet. • Ambiente de programación orientado a objetos.	2 horas
5	**Patrón comando**	1. Programa las clases correspondientes al patrón. 2. Utiliza las clases del patrón en un caso práctico. 3. Realiza pruebas unitarias del software desarrollado. 4. Entrega código y reporte de pruebas al docente.	• Pizarrón. • Computadora. • Proyector. • Internet. • Ambiente de programación orientado a objetos.	2 horas
6	**Patrón adaptador y fachada**	1. Programa las clases correspondientes al patrón. 2. Utiliza las clases del patrón en un caso práctico. 3. Realiza pruebas unitarias del software desarrollado. 4. Entrega código y reporte de pruebas al docente.	• Pizarrón. • Computadora. • Proyector. • Internet. • Ambiente de programación orientado a objetos.	2 horas
7	**Patrón método plantilla**	1. Programa las clases correspondientes al patrón. 2. Utiliza las clases del patrón en un caso práctico. 3. Realiza pruebas unitarias del software desarrollado. 4. Entrega código y reporte de pruebas al docente.	• Pizarrón. • Computadora. • Proyector. • Internet. • Ambiente de programación orientado a objetos.	2 horas
8	**Patrón iterador y composición**	1. Programa las clases correspondientes al patrón.	• Pizarrón. • Computadora.	2 horas

		2. Utiliza las clases del patrón en un caso práctico. 3. Realiza pruebas unitarias del software desarrollado. 4. Entrega código y reporte de pruebas al docente.	• Proyector. • Internet. • Ambiente de programación orientado a objetos.	
9	**Patrón estado**	1. Programa las clases correspondientes al patrón. 2. Utiliza las clases del patrón en un caso práctico. 3. Realiza pruebas unitarias del software desarrollado. 4. Entrega código y reporte de pruebas al docente.	• Pizarrón. • Computadora. • Proyector. • Internet. • Ambiente de programación orientado a objetos.	2 horas
10	**Patrón proxy**	1. Programa las clases correspondientes al patrón. 2. Utiliza las clases del patrón en un caso práctico. 3. Realiza pruebas unitarias del software desarrollado. 4. Entrega código y reporte de pruebas al docente.	• Pizarrón. • Computadora. • Proyector. • Internet. • Ambiente de programación orientado a objetos.	2 horas
11	**Patrones de patrones**	1. Integra varios patrones en un caso práctico. 2. Realiza pruebas unitarias del software desarrollado. 3. Entrega código y reporte de pruebas al docente.	• Pizarrón. • Computadora. • Proyector. • Internet. • Ambiente de programación orientado a objetos.	4 horas
UNIDAD IV				
12	**Refactorización dentro de clases**	1. Entrega del caso práctico por parte del docente. 2. Identifica olores dentro del código del caso práctico. 3. Modifica el código fuente aplicando refactorización.	• Pizarrón. • Computadora. • Proyector. • Internet. • Ambiente de programación orientado a objetos. • Código fuente del caso práctico.	4 horas

		4. Realiza pruebas unitarias y funcionales del software modificado. 5. Entrega código y reporte de pruebas al docente.		
13	**Refactorización entre clases**	1. Entrega del caso práctico por parte del docente. 2. Identifica olores entre las clases del código del caso práctico. 3. Modifica el código fuente aplicando refactorización. 4. Realiza pruebas unitarias y funcionales del software modificado. 5. Entrega código y reporte de pruebas al docente.	• Pizarrón. • Computadora. • Proyector. • Internet. • Ambiente de programación orientado a objetos. • Código fuente del caso práctico.	4 horas

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):

- Presentación de conceptos formales.
- Presentación de catálogos de patrones de diferentes niveles.
- Presentación de casos prácticos de utilización de patrones.
- Presentación de casos prácticos para la aplicación de refactorizaciones.

Estrategia de aprendizaje (alumno):

- Realización de prácticas de taller para la identificación y uso de patrones.
- Realización de prácticas de laboratorio para la aplicación de patrones y refactorización en casos prácticos.
- Elaboración de reportes para cada práctica.
- Realización de la evidencia de desempeño.

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 con los porcentajes de asistencia que establece el Estatuto Escolar vigente.
- Calificación en escala del 0 al 100, con un mínimo aprobatorio de 60.
- Es necesario entregar la evidencia de desempeño para tener derecho a calificación aprobatoria.

Criterios de evaluación

- Examen final ……..... 20%
- Prácticas de taller …….. 15%
- Prácticas de laboratorio …….. 25%
- Evidencia de desempeño …….. 40%

Total…….100%

IX. REFERENCIAS

Básicas

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. (1996). *Pattern-oriented software architecture, Vol. 1*. Reino Unido: Wiley. [Clásica].

Freeman, E., y Robson, E. (2021). *Head First Design Patterns (2a ed.)*. Estados Unidos: O´Reilly.

Freeman, E., y Freeman, E. (2004). *Head First Design Patterns*. Estados Unidos: O´Reilly. [Clásica].

Fowler, M. (2019). R*efactoring. Improving the design of existing code (2a ed.)*. Estados Unidos: Addison-Wesley.

Gamma, E., Helm, R. Johnson, R., Vlissides, J. (1977). *Design patterns. Elements of reusable object-oriented software.* Estados Unidos: Addison-Wesley. [Clásica].

Complementarias

Wake, W. C. (2003). *Refactoring workbook*. Estados Unidos: Addison-Wesley. [Clásica].

Kuchana, P. (2004). *Software architecture design patterns*. Estados Unidos: Auberbach Publications. [Clásica].

Cooper, J. W. (Junio, 1998). *Using design patterns. Communications of the ACM* 41 (6), 65-68 . Recuperado de https://dl.acm.org/doi/10.1145/276609.276621

X. PERFIL DEL DOCENTE

El docente que imparta la unidad de aprendizaje Patrones de Software debe contar con título de Ingeniero de software o área afín, con conocimientos de programación orientada a objetos y patrones de software; preferentemente con estudios de posgrado en ciencias de la computación y al menos dos años de experiencia docente. Debe ser organizado, creativo y analítico.