1.- DATOS DE LA ASIGNATURA

Nombre de la asignatura:	Arquitectura y Diseño de Software
Carrera:	Ingeniería en Sistemas Computacionales
Clave de la asignatura:	ISC - 1202
(Créditos) SATCA[1]:	2 - 2 - 4

2.- PRESENTACIÓN

Caracterización de la asignatura

Esta asignatura aporta al perfil del egresado un conjunto de actividades relacionada con la toma de decisiones, a menudo de naturaleza estructural. comparte con la programación una preocupación relacionada con abstraer la representación de la información y las secuencias del procesamiento, pero el grado de detalle es muy diferente en los extremos, el diseño construye representaciones coherentes y bien planeadas de los programas, que se concentran en las interrelaciones entre las partes al nivel más elevado y las operaciones lógicas en los niveles inferiores.

Intención didáctica.

La asignatura cubre la necesidad que tiene un ingeniero al enfrentarse a la implementación de estándares de modelado, así como el uso de herramientas CASE en el proceso.
Esta materia se organiza en seis unidades; la primera unidad, hace referencia a la introducción de las diferentes técnicas del modelado.
La unidad dos, presenta las técnicas y modelado del diseño.
En la unidad tres, introduce al desarrollo de software basado en arquitecturas.
La unidad cuatro ostenta las notaciones de la documentación y evaluación de arquitecturas de software.
En la unidad cinco, conduce a conocer las nuevas propuestas arquitectónicas.

3.- COMPETENCIAS A DESARROLLAR

Competencias específicas:	**Competencias genéricas:**
Aplicar diseño, patrones y estilos arquitectónicos para la construcción de software.	**Competencias instrumentales:** • Capacidad de análisis y síntesis. • Capacidad de organizar y planificar. • Conocimientos básicos de la carrera. • Comunicación oral y escrita. • Habilidades del manejo de la computadora. • Habilidad para buscar y analizar información proveniente de fuentes diversas. • Solución de problemas. • Toma de decisiones. **Competencias interpersonales:** • Capacidad crítica y autocrítica. • Trabajo en equipo. • Habilidades interpersonales. **Competencias sistémicas:** • Capacidad de aplicar los conocimientos en la práctica. • Habilidades de investigación. • Capacidad de aprender. • Capacidad de generar nuevas ideas (creatividad). • Búsqueda del logro.

4.- HISTORIA DEL PROGRAMA

Lugar y fecha de elaboración o revisión	Participantes	Observaciones (cambios y justificación)
Dirección General de Institutos Descentralizados, México D.F. Fecha: 12 al 14 de Septiembre de 2012.	Representantes: Tecnológico de Estudios Superiores de Coacalco. Instituto Tecnológico Superior de Atlixco, Comalcalco, Fresnillo, Santiago Papasquiaro, Tepexi de Rodríguez, Zapopan.	Análisis y adecuación por competencias del módulo de la especialidad "Ingeniería de Software" de la carrera de Ingeniería en Sistemas Computacionales.

5.- OBJETIVO(S) GENERAL(ES) DEL CURSO (competencias específicas a desarrollar en el curso)

Aplicar diseño, patrones y estilos arquitectónicos para la construcción de software.

6.- COMPETENCIAS PREVIAS

- Conocer modelado con UML orientado a objetos.
- Conocer metodologías de desarrollo de software.
- Capacidad de abstracción en el modelado de problemas.

7.- TEMARIO

Unidad	Temas	Subtemas
1	Introducción	1.1 Qué es Análisis y Diseño Orientado a Objetos 1.2 Referencias históricas. 1.2.1 OMT (Técnica de Modelado de Objetos). 1.2.2 Metodología Booch. 1.3 Metodología RUP (Rational Unified Process). 1.4 Diseño de Alto Nivel (HLD) y Bajo Nivel (LLD). 1.5 Comprensión de los requerimientos. 1.6 Casos de uso. 1.6.1 Introducción. 1.6.2 Elementos de casos de uso (diagrama, Relaciones, Especificaciones, Identificación de Casos de Uso). 1.7 Modelo del Dominio. 1.7.1 Visualización de conceptos. 1.7.2 Añadir asociaciones. 1.7.3 Añadir atributos.
2	Análisis y Diseño	2.1 Representación de los Eventos del Sistema usando Diagramas de Secuencia del Sistema. 2.2 Contratos de las operaciones. 2.3 Diseño basado en responsabilidades. 2.4 Modelo de diseño. 2.6 Modelo de comportamiento. 2.7 Diagramas de diseño. 2.7.1 Diagramas de secuencia. 2.7.2 Diagramas de colaboración. 2.7.3 Diagrama de clases de diseño. 2.8 Patrones Grasp. 2.9 Patrones GoF.
3	Diseño de Arquitecturas de Software	3.1 Desarrollo de Software Basado en Arquitecturas 3.1.1 Historia 3.2 Atributos Funcionales 3.3 Atributos de Calidad 3.4 Proceso de elaboración de Arquitecturas de Software 3.4.1 Forward Engineering 3.4.2 Reverse Engineering

		3.4.2 Estilos 3.4.3 Patrones
4	Documentación y métodos de evaluación de Arquitecturas de Software	4.1 Documentación de Arquitecturas de Software 4.2 Modelo 4+1 4.3 ATAM 4.4 SAAM 4.5 ARID 4.6 ALMA 4.7 SNA
5	Tendencias	5.1 Líneas de Productos de Software. 5.2 Arquitecturas Orientadas a Servicios. 5.3 Arquitecto de Software.

8.- SUGERENCIAS DIDÁCTICAS

- Propiciar la búsqueda y selección de información de sobre análisis y diseño orientados a objetos, mediante equipos analizar y discutir en clase.
- Realizar las prácticas propuestas para poder alcanzar el objetivo de la materia.
- Ejercicios de documentación de arquitecturas bajo una técnica.
- Analizar prácticas modelo para comprender su funcionamiento.
- Realizar búsquedas de información sobre temas afines.
- Elaborar reportes o informes de las prácticas.
- Uso de alguna herramienta UML.
- Buscar oportunidades para la planificación y modelado de un sistema computacional en las diferentes organizaciones de la localidad.
- Aplicar arquitectura orientada a servicios como decisión de arquitectura de software.

9.- SUGERENCIAS DE EVALUACIÓN

- Revisión del diseño de la arquitectura.
- Mapas conceptuales, cuestionarios sobre conceptos relacionados con la arquitectura.
- Revisión de la documentación del modelado.
- Exámenes escritos para comprobar el manejo de aspectos teóricos.

10.- UNIDADES DE APRENDIZAJE

UNIDAD 1. Introducción

Competencia específica a desarrollar	Actividades de Aprendizaje
Conoce metodologías y técnicas aplicables al diseño.	• Analizar y discutir las fases que implica el modelado del negocio. • Aplicar el Lenguaje Unificado • Modelado (UML) realizando prácticas sobre el modelado del negocio para un caso concreto. • Documentar los requerimientos denotados por UML para un sistema concreto. • Realizará todos los componentes del proceso que involucra un caso de uso para el sistema considerado en el punto anterior

UNIDAD 2. Análisis y Diseño

Competencia específica a desarrollar	Actividades de Aprendizaje
Aplica los patrones de diseño GRASP y GoF involucrados en el análisis y diseño de un sistema concreto.	• Realizar la representación de los eventos del sistema, así como los contratos de las operaciones. • Utilizando los patrones de diseño GRASP y GoF, realizar los modelos correspondientes al sistema. • Elaborar los diagramas denotados por la metodología que correspondan a los modelos anteriores.

UNIDAD 3. Diseño de Arquitecturas de Software

Competencia específica a desarrollar	Actividades de Aprendizaje
Identifica los conceptos fundamentales de arquitectura de software y su relevancia.	• Realizar un ejercicio que muestre la aplicación del concepto de desarrollo de software basado en arquitecturas. • Buscar, seleccionar y evaluar información de atributos funcionales del negocio, de usuario, de sistema, entre otros. • Buscar, seleccionar y evaluar información de atributos de calidad como: desempeño, confiabilidad, seguridad, facilidad de modificación, facilidad de uso, robustez, portabilidad, escalabilidad, reutilización, disponibilidad, etcétera. • Realizar el diseño de la arquitectura del sistema que se está realizando, realizar ejercicios que permitan modelar la arquitectura de sistemas existentes.

UNIDAD 4. Documentación y métodos de evaluación de Arquitecturas de Software.

Objetivo Educacional	Actividades de Aprendizaje
Documenta y evalúa una arquitectura de software.	• Buscar, seleccionar y evaluar información de documentación de arquitecturas de Software. • Realizar ejercicios con el lenguaje UML. • Analizar en equipos, el Modelo 4+1 e identificar sus vistas y aplicaciones. • Aplicar el método de evaluación ATAM a un ejemplo de diseño de arquitectura • Aplicar el método de evaluación SAAM a un ejemplo de diseño de arquitectura • Aplicar el método de evaluación ARID a un ejemplo de diseño de arquitectura • Aplicar el método de evaluación ALMA a un ejemplo de diseño de arquitectura • Aplicar el método de evaluación SNA a un ejemplo de diseño de arquitectura • Realizar una comparación de los diferentes métodos de evaluación y aplicar uno de los métodos al diseño de arquitectura del proyecto dosificado.

UNIDAD 5. Tendencias.

Objetivo Educacional	Actividades de Aprendizaje
Identifica nuevos paradigmas y perspectivas de desarrollo de software.	• Analizar la aplicación del paradigma de Líneas de Productos de Software para conocer sus ventajas y beneficios. • Implementar Servicios Web que conformen una Arquitectura Orientada a Servicios. • Realizar un análisis para identificar las capacidades del ingeniero de software.

11. FUENTES DE INFORMACIÓN

1. Larman, Craig (2003), UML y patrones, Ed. Pearson.
2. Grandy Booch - James Rumbaugh - Ivar Jacobson, El lenguaje unificado de modelado, Addison Wesley 2000
3. Ivar Jacobson - Grady Booch - James Rumbaugh, El proceso unificado de desarrollo de software, Addison Wesley 2000
4. Martin Fowler-Kendall Scott, UML Gota a Gota, Addison Wesley
5. Robert Cecil Martín, UML para Programadores en Java, Addison Wesley
6. Perdita Stevens - Rob Pooley : Universidad de Edimburgo, Utilización del UML, Humphey Watts S., A Discipline for Software Engineering, Addison Wesley
8. Philippe Kruchten, The Rational Unified Process. An Introduction, Second Edition, Addison Wesley. 2000.
9. Shaw, Mary & Garlan, David. Software Architectures: Perspectives on an Emerging Discipline. Upper Saddle River, NJ: Prentice
10. Bass, Len, Clements, Paul, and Kazman, Rick. Software Architecture in Practice. Reading, MA: Addison
11. Jackson, Michael. Software Requirements & Specifications: a lexicon of practice, principles, and prejudices. Reading, MA: Addison
12. Buschmann, Frank; Meunier, Regine; Rohnert, Hans; Sommerlad, Peter & Stah, Michael. Pattern-Oriented Software Architecture, A System of Patterns, John Wiley and Sons, Ltd, West Sussex PO19 1UD, England, 1996
13. Phillipe Kruchten, “The 4+1 View Model of Architecture”, *IEEE Software*, Volume 12 , Issue 6 (November 1995), pp. 42 – 50.
14. Clements, Paul & Northrop Linda, Software Product Lines, SEI Series in Software Engineering, Addison-Wesley, 2001.
15. Bosch Jan, Design & Use of Software Architectures: Adopting and evolving a product-line approach, Addison-Wesley, 2000.
16. Dikel M. David, Kane David, Wilson R. Wilson, Software Architecture Organizational Principles and Patterns, Software Architecture Series Prentice Hall 2001.
17. Clements Paul, Kazman Rick, Klein Mark, Evaluating Software Architectures: Methods and Case Studies, SEI Series in Software Engineering, Addison-Wesley, 2002.
18. Hofmeister Christine, Nord Robert, Soni Dilip, Applied Software Architecture, 1999.
19. Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. Design Patterns Elements of Reusable Object-Oriented Software. Edited by Addison-Wesley professional computing series: Addison Wesley, 1995.
20. Mead, Nancy R., Robert J. Ellison, Richard C. Linger, Thomas Longstaff and John McHugh. "Survivable Network Analysis Method*." CMU/SEI, 2000*.
21. Ali Babar, Muhammad and Ian Gorton. "Comparison of Scenario-Based Software Architecture Evaluation Methods." Paper presented at the *11th Asia-Pacific Software Engineering Conference* 2004
22. Dobrica, Liliana and Eila Niermela. "A Survey on Software Architecture Analysis Methods." *IEEE Transactions on Software Engineering*, pp 638-562, 2002.
23. Bengtsson, PerOlof, Nico Lassing and Jan Bosch, Vliet, Hans van. "Architecture-Level Modifiability Analysis (ALMA)." *The Journal of Systems and Software* , vol. 69, pp. 129-147, 2004.
24. Kazman, Rick, Len Bass, Gregory Abowd and Mike Webb. "Saam: A Method for Analyzing the Properties of Software Architectures." Paper presented at the Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy 1994.

25. Kazman, Rick, Mark Klein, Mario Barbacci, Thomas Longstaff, Lipson Howard F. and S. Jeromy Carriere. "The Architecture Tradeoff Analysis Method." Paper presented at the Proceedings of ICECCS, Monterey, CA 1998.
26. Gómez Gómez Salvador, Lemus Olalde Cuauhtémoc, “Proceso de evaluación para arquitecturas de software usadas en el sector empresarial”, Reporte Técnico Maestría en Ingeniería de Software, CIMAT, Julio 2004.
27. Nuñez Mora Araceli, Lemus Olalde Cuauhtémoc, “Proceso para el desarrollo de Arquitecturas de Software basado en DFSS”, Reporte Técnico Maestría en Ingeniería de Software, CIMAT, Julio 2005.
28. Clements Paul, Bachman Felix, Bass Len, Garlan David, Ivers James, Little Reed, Nord Robert, Stafford Judith, Documenting Software Architectures: Views and Beyond, SEI Series in Software Engineering, Addison-Wesley, 2002.
29. Lemus Olalde Cuauhtémoc, “Software Architecture-Based Development” White Paper, CIMAT, 2003.

12. PRACTICAS

- Dado un sistema concreto real o ficticio realizar representación del modelado del negocio siguiendo las especificaciones denotadas por UML para este proceso. Realizar el análisis de requerimientos del sistema dado, contemplando: recolección, representación y validación de los datos.
- Representar los elementos que conforman un caso de uso para el sistema especificado.
- Realizar el análisis y diseño para el sistema determinado siguiendo los patrones especificados por la notación UML.
- Desarrollo de un proyecto dosificado durante el semestre, involucrando la elaboración, evaluación y documentación de una arquitectura de software, donde se aplique al menos uno de los métodos de evaluación y al menos dos de las vistas del modelo 4+1 utilizando el Lenguaje Unificado de Modelado.