

1. Datos Generales de la asignatura

Nombre de la asignatura:	Arquitectura y Diseño de Software.
Clave de la asignatura:	ISC-1904
SATCA¹:	2-2 4
Carrera:	Ingeniería en Sistemas Computacionales

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.

¹ Sistema de Asignación y Transferencia de Créditos Académicos

3. Participantes en el diseño y seguimiento curricular del programa

Lugar y fecha de elaboración o revisión	Participantes	Observaciones
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.
Tecnológico de Estudios Superiores de Coacalco Fecha: 10 al 14 de agosto de 2015.	Representantes: Tecnológico de Estudios Superiores de Coacalco. .	Actualización del módulo de la especialidad "Ingeniería de Software" de la carrera de Ingeniería en Sistemas Computacionales
Tecnológico de Estudios Superiores de Coacalco Fecha: 21 al 22 de junio de 2019.	Representantes: Tecnológico de Estudios Superiores de Coacalco. .	Actualización de asignaturas del módulo de la especialidad "Ingeniería de Software" de la carrera de Ingeniería en Sistemas Computacionales.

4. Competencia(s) a desarrollar

Competencia(s) específica(s) de la asignatura
Aplica diseño, patrones y estilos arquitectónicos para la construcción de software.

5. Competencias previas

- Conoce el modelado con UML orientado a objetos.
- Conoce las metodologías de desarrollo de software.
- Tiene la capacidad de abstracción en el modelado de problemas.

6. Temario

Unidad	Temas	Subtemas
1	Introducción	1.1 Definiciones prácticas de software, Arquitectura y Diseño. 1.1.1. Según ISO / IEC / IEEE 42010: 2011 1.1.2. Conceptos de diseño (alcance, agregación, partición y certificación) 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. 1.5.1. Elaboración de casos de uso 1.5.2. Elaboración de modelo de Dominio 1.6 Relación entre Arquitectura y Diseño
2	Análisis y Diseño	2.1 Metodología arquitectónica para el diseño de software 2.2 Contratos de las operaciones. 2.3 Diseño basado en responsabilidades. 2.4 Modelos de representación 2.4.1. Modelos de objetivo 2.4.2. Modelos de forma 2.4.3. Modelos de función 2.4.4. Modelos de desempeño 2.4.5. Modelos de datos 2.4.6. Modelos administrativos 2.5 Diagramas de diseño. 2.5.1 Diagramas de secuencia. 2.5.2 Diagramas de colaboración. 2.5.3 Diagrama de clases de diseño. 2.6. Patrones Grasp. 2.7. Patrones GoF.

3	Diseño de Arquitecturas de Software	3.1. Diseño de Software Basado en Arquitecturas 3.1.1. Progresión 3.1.1. Heurísticas del diseño 3.1.3. Lenguajes de descripción de Arquitecturas 3.1.3.1. ABAP 3.1.3.2. Java 3.1.3.3. Web Services basados en SOAP / REST 3.2. Atributos de calidad del software 3.2.1. Concepto de calidad de software 3.2.2. Atributos funcionales y no funcionales de Software. 3.2.3. Evaluación de atributos. 3.2.4. Principios básicos y tareas esenciales. 3.3. Proceso de elaboración de Arquitecturas de Software 3.3.1 Forward Engineering 3.3.2 Reverse Engineering 3.3.2 Estilos 3.3.3 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. PASA 4.8. SALUTA 4.9. SNA
5	Tendencias	5.1 Líneas de Productos de Software. 5.1.1. El ciclo de producción. 5.2. Arquitecturas Orientadas a Servicios (Service Oriented Architecture SOA). 5.3 Actividades del Arquitecto de Software.

7. Actividades de aprendizaje de los temas

UNIDAD 1. Introducción	
Competencias	Actividades de aprendizaje
Específica(s): Comprende los conceptos de diseño y arquitectura de software, así como la relación que existe entre ellos y las metodologías de desarrollo de software. Genéricas: • Capacidad de análisis y síntesis • Capacidad de organizar y planificar • Comunicación oral y escrita • Habilidad para buscar y analizar información proveniente de fuentes diversas • Solución de problemas	• Explica en plenaria los conceptos de Arquitectura y diseño de software. • Describe las fases de cada una de las metodologías siguientes: OMT, Booch, RUP, HLD y LLD). • Aplica el Lenguaje Unificado de Modelado (UML) realizando prácticas sobre un caso concreto. • Documenta los requerimientos denotados por UML para el caso anterior. • Realiza la asociación entre arquitectura y diseño, así como con las metodologías de desarrollo de software.
UNIDAD 2. Análisis y Diseño	
Competencias	Actividades de aprendizaje
Específica(s): Identifica los componentes de la metodología arquitectónica para el diseño de software y elabora el contrato de operaciones. Compara y determina el modelo de representación y los diagramas de diseño que debe utilizar para un estudio de caso concreto. Aplica los patrones de diseño GRASP y GoF involucrados en el análisis y diseño del caso concreto. Genéricas: • Capacidad de análisis y síntesis. • Capacidad de organizar y planificar • Habilidad para buscar y analizar información proveniente de fuentes diversas • Solución de problemas • Capacidad crítica y autocrítica	• Investiga y resume los componentes de la metodología arquitectónica. • Enlista los elementos que se establecen en el contrato de operaciones. • Realiza la representación de los componentes de la metodología arquitectónica para el caso concreto, así como los contratos de las operaciones. • Investiga y resume los diferentes modelos de diseño. • Elabora los diagramas denotados por la metodología que correspondan a los modelos anteriores. • Compara los diferentes patrones de diseño. • Utilizando los patrones de diseño GRASP y GoF, realizar los

- Trabajo en equipo - Habilidades interpersonales	modelos correspondientes al caso concreto.
UNIDAD 3. Diseño de Arquitecturas de Software	
Competencias	Actividades de aprendizaje
Específica(s): Identifica los conceptos de diseño de software basado en arquitectura. Clasifica los atributos de calidad y define la forma de evaluarlos. Elabora el diseño arquitectónico para un caso concreto. Genéricas: - Solución de problemas - Capacidad crítica y autocrítica - Trabajo en equipo	- Investiga y explica en plenaria los conceptos de progresión, heurística del diseño y los lenguajes de descripción de arquitecturas. - Realiza un cuadro comparativo entre los lenguajes de descripción de arquitecturas. - Busca, selecciona y evalúa información de atributos funcionales y no funcionales del negocio, de usuario, de sistema, entre otros. - Busca, selecciona y evalúa información de atributos de calidad como: desempeño, confiabilidad, seguridad, facilidad de modificación, facilidad de uso, robustez, portabilidad, escalabilidad, reutilización, disponibilidad, etcétera. - Compara los diferentes procesos de arquitectura de software. - Selecciona y realiza el diseño de la arquitectura del caso de estudio propuesto. - Investiga y expone ejemplos que permitan modelar la arquitectura de sistemas existentes.
UNIDAD 4. Documentación y métodos de evaluación de Arquitecturas de Software.	
Competencias	Actividades de aprendizaje
Específica(s): Documenta y evalúa una arquitectura de software Genéricas: - Capacidad de aplicar los conocimientos en la práctica - Habilidades de investigación	- Busca, selecciona y evalúa información de documentación de arquitecturas de Software. - Realiza ejercicios con el lenguaje UML. - Analiza en equipos, el Modelo 4+1 e identificar sus vistas y aplicaciones.

• Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad).	• Aplica el método de evaluación ATAM a un ejemplo de diseño de arquitectura • Aplica el método de evaluación SAAM a un ejemplo de diseño de arquitectura • Aplica el método de evaluación ARID a un ejemplo de diseño de arquitectura • Aplica el método de evaluación ALMA a un ejemplo de diseño de arquitectura • Aplica el método de evaluación PASA a un ejemplo de diseño de arquitectura • Aplica el método de evaluación SALUTA a un ejemplo de diseño de arquitectura • Aplicar el método de evaluación SNA a un ejemplo de diseño de arquitectura • Realiza una comparación de los diferentes métodos de evaluación y aplicar uno de los métodos al diseño de arquitectura del proyecto dosificado.
UNIDAD 5. Tendencias	
Competencias	Actividades de aprendizaje
Específica(s): Identifica nuevos paradigmas y perspectivas de arquitecturas para el desarrollo de software. Comprende las actividades del arquitecto de software. Genéricas: • Capacidad de aplicar los conocimientos en la práctica • Habilidades de investigación • Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad).	• Investiga y representa el ciclo de producción. • Analiza la aplicación del paradigma de Líneas de Productos de Software para conocer sus ventajas y beneficios. • Implementa Servicios Web que conformen una Arquitectura Orientada a Servicios. • Realiza un análisis para identificar las capacidades del ingeniero de software. • Realiza un FODA sobre su perfil como arquitecto de software.

8. Práctica(s)

- 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.
- Realizar el análisis y diseño para el sistema determinado siguiendo los modelos, patrones y estilos; apoyándose en 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 UML.

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-profesión, 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

Se recomienda evaluar los siguientes puntos:

- Evaluación Teórica.
- Prácticas de laboratorio con el uso de herramientas case.
- Desarrollo y evaluación de un proyecto de software.

La ponderación de la evaluación debe ser propuesta por la academia correspondiente de cada institución.

11. Fuentes de información

- Larman, Craig (2003), UML y patrones, Ed. Pearson.
- Grandy Booch - James Rumbaugh - Ivar Jacobson, El lenguaje unificado de modelado, Addison Wesley 2000
- Ivar Jacobson - Grady Booch - James Rumbaugh, El proceso unificado de desarrollo de software, Addison Wesley 2000
- Martin Fowler-Kendall Scott, UML Gota a Gota, Addison Wesley
- Robert Cecil Martín, UML para Programadores en Java, Addison Wesley
- Perdita Stevens - Rob Pooley : Universidad de Edimburgo, Utilización del UML, Humphrey Watts S., A Discipline for Software Engineering, Addison Wesley
- 8. Philippe Kruchten, The Rational Unified Process. An Introduction, Second Edition, Addison Wesley. 2000.
- Shaw, Mary & Garlan, David. Software Architectures: Perspectives on an Emerging Discipline. Upper Saddle River, NJ: Prentice
- Bass, Len, Clements, Paul, and Kazman, Rick. Software Architecture in Practice. Reading, MA: Addison
- Jackson, Michael. Software Requirements & Specifications: a lexicon of practice, principles, and prejudices. Reading, MA: Addison
- 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
- Phillipe Kruchten, "The 4+1 View Model of Architecture", IEEE Software, Volume 12 , Issue 6 (November 1995), pp. 42 – 50.
- Clements, Paul & Northrop Linda, Software Product Lines, SEI Series in Software Engineering, Addison-Wesley, 2001.
- Bosch Jan, Design & Use of Software Architectures: Adopting and evolving a product-line approach, Addison-Wesley, 2000.
- Dikel M. David, Kane David, Wilson R. Wilson, Software Architecture Organizational Principles and Patterns, Software Architecture Series Prentice Hall 2001.
- Clements Paul, Kazman Rick, Klein Mark, Evaluating Software Architectures: Methods and Case Studies, SEI Series in Software Engineering, Addison-Wesley, 2002.
- Hofmeister Christine, Nord Robert, Soni Dilip, Applied Software Architecture, 1999.

- 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.
- Mead, Nancy R., Robert J. Ellison, Richard C. Linger, Thomas Longstaff and John McHugh. "Survivable Network Analysis Method." CMU/SEI, 2000.
- 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
- Dobrica, Liliana and Eila Niermela. "A Survey on Software Architecture Analysis Methods." IEEE Transactions on Software Engineering, pp 638-562, 2002.
- 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.
- 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.
- 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.
- 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.
- 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.
- 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.
- Lemus Olalde Cuauhtémoc, "Software Architecture-Based Development" White Paper, CIMAT, 2003.