Datos Generales de la asignatura

Nombre de la asignatura:	**Arquitectura y Diseño de Software**
Clave de la asignatura:	**ISC-2604**
SATCA[1]:	**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 de Arquitectura y Diseño de Software responde a la necesidad de que el ingeniero en sistemas comprenda, diseñe y evalúe estructuras de software sostenibles, escalables y alineadas con los objetivos del negocio y de la tecnología. Permite al estudiante integrar principios, estilos y patrones arquitectónicos con el uso de herramientas modernas de documentación, modelado y evaluación, fortaleciendo su capacidad de análisis, abstracción y toma de decisiones técnicas fundamentadas. La materia se organiza en **siete unidades**. La **primera unidad** introduce los fundamentos de la arquitectura de software, brindando los conceptos básicos, principios de calidad y la relación entre la arquitectura y el ciclo de vida del software. La **segunda unidad** aborda los estilos y patrones arquitectónicos, analizando las principales estructuras empleadas en la industria, sus ventajas y aplicaciones en contextos reales.

[1] Sistema de Asignación y Transferencia de Créditos Académicos

La **tercera unidad** profundiza en el diseño de software orientado a la sostenibilidad, promoviendo el desarrollo de soluciones mantenibles, reutilizables y eficientes en el uso de recursos.

La **cuarta unidad** se centra en la documentación y modelado de arquitecturas, empleando distintos lenguajes y modelos para representar y comunicar de manera efectiva las decisiones arquitectónicas.

La **quinta unidad** desarrolla la evaluación y validación de arquitecturas, mediante métodos sistemáticos de análisis de riesgos, trazabilidad y pruebas de concepto.

La **sexta unidad** explora las tendencias modernas en arquitectura de software, incluyendo microservicios, arquitecturas reactivas, serverless y sistemas distribuidos, que reflejan el entorno tecnológico actual.

Finalmente, la **séptima unidad** integra herramientas y prácticas profesionales de diseño, fomentando el uso de principios SOLID, refactorización, pruebas y herramientas automatizadas para elevar la calidad del producto software.

Con ello, la asignatura forma profesionistas capaces de diseñar y evaluar soluciones arquitectónicas robustas, sostenibles y adaptadas a las exigencias actuales de la ingeniería de software.

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	Representantes:	Actualización de asignaturas del módulo de la especialidad "Ingeniería de Software" de la carrera

Fecha: 21 al 22 de junio de 2019. Tecnológico de Estudios Superiores de Coacalco Octubre del 2025.	Tecnológico de Estudios Superiores de Coacalco. Representantes: Docentes de la Academia de Ingeniería en Sistemas Computacionales del Tecnológico de Estudios Superiores de Coacalco.	de Ingeniería en Sistemas Computacionales. Análisis, formulación de competencias, actualización de temario, formulación de actividades y prácticas de la especialidad de "Ingeniería de Software y Arquitectura de Sistemas" 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

No.	Temas	Subtemas
1	**Fundamentos de la Arquitectura de Software**	1.1 Conceptos básicos de arquitectura de software 1.2 Principios y atributos de calidad (modificabilidad, escalabilidad, rendimiento, etc.) 1.3 Roles del arquitecto de software 1.4 Relación entre arquitectura y ciclo de vida del software
2	**Estilos y Patrones Arquitectónicos**	2.1 Estilos arquitectónicos: monolítico, cliente-servidor, microservicios, orientado a eventos 2.2 Patrones arquitectónicos comunes: MVC, MVVM, capas, pipes & filters 2.3 Selección de estilos y patrones adecuados según el contexto 2.4 Casos de uso reales de aplicaciones con diferentes estilos

3	**Diseño de Software Orientado a la Sostenibilidad**	3.1 Principios de diseño sostenible y de bajo acoplamiento 3.2 Reutilización de componentes y diseño para la mantenibilidad 3.3 Evaluación del impacto del diseño en los recursos computacionales 3.4 Prácticas de diseño centrado en el usuario y accesibilidad
4	**Documentación y Modelado de Arquitecturas**	4.1 Documentación con vistas arquitectónicas (4+1 de Kruchten, C4 Model) 4.2 Lenguajes de modelado: UML, ArchiMate 4.3 Herramientas para documentación y modelado 4.4 Uso de modelos en la toma de decisiones técnicas
5	**Evaluación y Validación de Arquitecturas**	5.1 Métodos de evaluación de arquitectura (ATAM, SAAM) 5.2 Análisis de riesgos arquitectónicos 5.3 Pruebas de concepto (PoC) y prototipos arquitectónicos 5.4 Trazabilidad entre requisitos y decisiones arquitectónicas
6	**Tendencias en Arquitectura de Software Moderna**	6.1 Arquitectura basada en servicios: SOA vs. microservicios 6.2 Serverless, contenedores y orquestadores (Docker, Kubernetes) 6.3 Arquitecturas reactivas y event-driven 6.4 Arquitectura para sistemas distribuidos y escalables
7	**Herramientas y Prácticas de Diseño de Software**	7.1 Principios SOLID y buenas prácticas de diseño 7.2 Refactorización y patrones de diseño 7.3 Integración de pruebas en el diseño 7.4 Herramientas de apoyo: editores de diagramas, linters, análisis estático

7. Actividades de aprendizaje de los temas

UNIDAD 1. Fundamentos de la Arquitectura de Software	
Competencias	Actividades de aprendizaje
Específica(s): Comprende los conceptos, principios y atributos de calidad de la arquitectura de	• Investigar el concepto de arquitectura de software y exponer ejemplos de su aplicación en proyectos reales. • Analizar los atributos de calidad en diferentes tipos de sistemas.

software, el papel del arquitecto y su relación con el ciclo de vida del 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	• Desarrollar un mapa conceptual sobre el rol del arquitecto en el ciclo de vida del software. • Elaborar un resumen comparativo de arquitecturas documentadas por empresas tecnológicas.
UNIDAD 2. Estilos y Patrones Arquitectónicos	
Competencias	Actividades de aprendizaje
Específica(s): Identifica y diferencia los principales estilos y patrones arquitectónicos. Seleccionando patrones adecuados según el contexto del problema. 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 • Trabajo en equipo • Habilidades interpersonales	• Elaborar una tabla comparativa de estilos arquitectónicos (monolítico, cliente-servidor, microservicios, etc.). • Analizar casos de uso reales donde se apliquen diferentes patrones. • Diseñar un esquema arquitectónico aplicando un patrón específico (por ejemplo, MVC o capas). • Presentar un informe de ventajas y desventajas de cada estilo frente a un problema dado.
UNIDAD 3. Diseño de Software Orientado a la Sostenibilidad	
Competencias	Actividades de aprendizaje
Específica(s): Diseña soluciones sostenibles y mantenibles mediante principios de bajo acoplamiento y reutilización. Evalúa el impacto del diseño en la eficiencia y los recursos computacionales. Genéricas:	• Realizar un estudio de caso sobre sostenibilidad en diseño de software. • Diseñar componentes reutilizables con bajo acoplamiento. • Evaluar el consumo de recursos en diferentes alternativas de diseño. • Debatir sobre accesibilidad y usabilidad en el diseño de software moderno.

• Solución de problemas • Capacidad crítica y autocrítica • Trabajo en equipo	
UNIDAD 4. Documentación y Modelado de Arquitecturas.	
Competencias	Actividades de aprendizaje
Específica(s): Documenta arquitecturas usando vistas y modelos reconocidos (C4, 4+1, UML) utilizando herramientas de modelado para comunicar decisiones arquitectónicas. Genéricas: • Capacidad de aplicar los conocimientos en la práctica • Habilidades de investigación • Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad).	• Elaborar diagramas UML para representar una arquitectura de software existente. • Aplicar el modelo C4 para documentar un sistema real o académico. • Comparar herramientas de modelado (ArchiMate, PlantUML, Draw.io, etc.). • Documentar decisiones arquitectónicas con base en requisitos funcionales y no funcionales.
UNIDAD 5. Evaluación y Validación de Arquitecturas	
Competencias	Actividades de aprendizaje
Específica(s): Aplica métodos de evaluación de arquitecturas (ATAM, SAAM). Identifica riesgos y establece trazabilidad entre decisiones y requisitos. Genéricas: • Capacidad de aplicar los conocimientos en la práctica • Habilidades de investigación • Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad).	• Aplicar ATAM a un caso de estudio para identificar riesgos arquitectónicos. • Diseñar una matriz de trazabilidad entre decisiones y requisitos. • Desarrollar una prueba de concepto (PoC) para validar una hipótesis de diseño. • Redactar un informe de evaluación comparando dos alternativas arquitectónicas.
UNIDAD 6. Tendencias en Arquitectura de Software Moderna.	
Específica(s): Analiza arquitecturas modernas basadas en servicios, contenedores y sistemas distribuidos.	• Realizar un seminario sobre microservicios, contenedores y orquestadores. • Comparar arquitecturas SOA vs. microservicios.

Evalúa el impacto de las tecnologías emergentes (serverless, Kubernetes, arquitecturas reactivas). Genérica(s): • 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 • Trabajo en equipo	• Diseñar un esquema conceptual de una arquitectura event-driven. • Documentar un ejemplo práctico de despliegue en la nube (AWS, Azure o GCP).
UNIDAD 7. Herramientas y Prácticas de Diseño de Software.	
Específica(s): Aplica principios SOLID y buenas prácticas de diseño orientado a objetos. Integra herramientas de refactorización, pruebas y análisis estático en el proceso de diseño. Genérica(s): • 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	• Implementar refactorización de código aplicando principios SOLID. • Analizar patrones de diseño aplicados en un sistema existente. • Integrar pruebas unitarias en el proceso de diseño (TDD). • Evaluar herramientas de análisis estático y linters en proyectos reales.

8. Práctica(s)

- Modelado arquitectónico con UML y C4: representar visualmente un sistema de mediana complejidad con diferentes vistas.
- Aplicación de patrones arquitectónicos: diseñar e implementar un sistema usando MVC o microservicios.
- Evaluación de arquitectura con ATAM o SAAM: aplicar una metodología de evaluación a un caso de estudio real.
- Refactorización y pruebas: aplicar principios SOLID y refactorización en un proyecto, integrando pruebas automatizadas.
- (Opcional de ampliación) Despliegue de una arquitectura moderna con contenedores y orquestadores en un entorno cloud simulado.

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 especificas 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

- Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice* (2nd ed.). Addison-Wesley.
- Booch, G., Rumbaugh, J., & Jacobson, I. (2000). *El lenguaje unificado de modelado (UML)*. Addison-Wesley.

- Bosch, J. (2000). *Design and use of software architectures: Adopting and evolving a product-line approach*. Addison-Wesley.
- Brown, S. (2019). *Software architecture for developers: Volume 1 - Technical leadership and the balance with agility* (2nd ed.). Leanpub.
- Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-oriented software architecture: A system of patterns*. John Wiley & Sons.
- Clements, P., Bachman, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., & Stafford, J. (2002). *Documenting software architectures: Views and beyond*. Addison-Wesley.
- Clements, P., Kazman, R., & Klein, M. (2002). *Evaluating software architectures: Methods and case studies*. Addison-Wesley.
- Dobrica, L., & Niemelä, E. (2002). A survey on software architecture analysis methods. *IEEE Transactions on Software Engineering, 28*(7), 638–653. https://doi.org/10.1109/TSE.2002.1019479
- Erich, G., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.
- Ford, N., Richards, M., Parsons, P., & Cornelison, R. (2020). *Fundamentals of software architecture: An engineering approach*. O'Reilly Media.
- Fowler, M., & Scott, K. (2000). *UML distilled: A brief guide to the standard object modeling language* (2nd ed.). Addison-Wesley.
- Gómez, S., & Lemus Olalde, C. (2004). *Proceso de evaluación para arquitecturas de software usadas en el sector empresarial* [Reporte técnico de Maestría en Ingeniería de Software, CIMAT].
- Grady Booch, J., Rumbaugh, J., & Jacobson, I. (2000). *El proceso unificado de desarrollo de software*. Addison-Wesley.
- Hofmeister, C., Nord, R., & Soni, D. (1999). *Applied software architecture*. Addison-Wesley.
- Jackson, M. (1995). *Software requirements & specifications: A lexicon of practice, principles, and prejudices*. Addison-Wesley.
- Kazman, R., Bass, L., Abowd, G., & Webb, M. (1994). SAAM: A method for analyzing the properties of software architectures. *Proceedings of the 16th International Conference on Software Engineering*, Sorrento, Italy.
- Kruchten, P. (2000). *The rational unified process: An introduction* (2nd ed.). Addison-Wesley.
- Kruchten, P. (1995). The 4+1 view model of architecture. *IEEE Software, 12*(6), 42–50. https://doi.org/10.1109/52.469759
- Larman, C. (2003). *UML y patrones*. Pearson Educación.
- Lemus Olalde, C., & Núñez Mora, A. (2005). *Proceso para el desarrollo de arquitecturas de software basado en DFSS* [Reporte técnico de Maestría en Ingeniería de Software, CIMAT].
- Mead, N. R., Ellison, R. J., Linger, R. C., Longstaff, T., & McHugh, J. (2000). *Survivable network analysis method*. Carnegie Mellon University / Software Engineering Institute.
- Shaw, M., & Garlan, D. (1996). *Software architectures: Perspectives on an emerging discipline*. Prentice Hall.
- Vernon, V. (2016). *Domain-driven design distilled*. Addison-Wesley Professional.