1. Datos Generales de la asignatura

Nombre de la asignatura:	**Desarrollo de Proyectos de Software I**
Clave de la asignatura:	INC-1701
SATCA:	2-2-4
Carrera:	**Ingeniería en Sistemas Computacionales**

2. Presentación

Caracterización de la asignatura

El desarrollo de proyectos de software ha permitido apoyar las actividades diarias en empresas, en procesos de manejo de información o en sistemas de producción donde se ha sustituido la mano del hombre por máquinas que controlan la producción a través de dispositivos programados y el uso de las nuevas tecnologías.

Esta asignatura aporta al perfil del Ingeniero en Sistemas Computacionales las competencias profesionales para aplicar métodos y técnicas de diseño que permitan desarrollar soluciones de software, considerando conceptos de arquitectura de software 4+1 vistas, modelado orientado a objetos con UML, reglas de interfaz de usuario, diseño de base de datos y el uso de patrones arquitectónico MVC y de diseño de software.

La importancia de esta asignatura es que permite al alumno continuar con las fases del ciclo de vida del desarrollo de cualquier tipo de software, en cuanto a diseño. Esta asignatura, es la aplicación práctica del conocimiento científico, a través de los métodos y técnicas adecuados, para el diseño en el desarrollo de software.

La disciplina de Desarrollo de Proyectos de Software I se relaciona con materias precedentes como: Programación Orientada A Objetos, Estructura De Datos, Tópicos Avanzados De Programación, Fundamentos de Base de Datos, Sistemas Operativos, Arquitectura De Computadoras, Telecomunicaciones, Ingeniería De Software y Fundamentos De Ingeniería de Software.

Posteriores: Desarrollo de Proyectos de Software II, Gestión de Proyectos de Software, Programación Web, Metodologías Ágiles.

Requiere de competencias previas como: Manejo de un lenguaje de modelado, dominio en

el uso de herramientas CASE, uso de algún Sistema Manejador de Bases de Datos, dominio de algún lenguaje de programación orientado a objetos, identificación de las etapas del ciclo de desarrollo de sistemas y de las diferentes plataformas operativas.
Intención didáctica
La asignatura debe ser teórico – práctico y capaz de desarrollar en el estudiante la habilidad para la aplicación de las diferentes técnicas de diseño para el desarrollo de software, considerando siempre los principios de la ingeniería de software, para lo cual se organiza el temario en siete bloques o unidades temáticas. En el bloque uno, conceptos fundamentales de diseño tales como la abstracción, componentes de interfaces y descomposición y modularización. Así mismo, revisar y aplicar métodos de diseño estructurados, orientados a datos y diseño orientado a objetos e involucra aspectos como espíritu emprendedor creatividad y trabajo en equipo. El bloque dos permitirá investigar en cuanto a arquitecturas de software y aplicar la arquitectura 4+1 vistas, para identificar subsistemas y establecer un marco de trabajo para su control y comunicación, contemplando las actividades relativas a la especificación del software, el desarrollo, la validación y la evolución. Utiliza el modelado orientado a objetos con UML elaborando la vista Lógica utilizando el diagrama de vista Lógica a través del diagrama de clases y la Vista de procesos a través de algunos de los diagramas de estado, actividades, secuencia o colaboración según sea el tipo de sistema. En el bloque tres se revisan las reglas de diseño de interfaz de usuario y la integración de la interfaz al caso de uso, elaborando la vista de usuarios del proyecto considerando la implementación del patrón arquitectónico MVC) En el bloque cinco se estudian los *frameworks*, *plug-ins* y componentes de software. En el bloque seis y siete se revisan y estudian los patrones arquitectónico MVC y de diseño de software para su uso según lo requieran las necesidades del software.

3. Participantes en el diseño y seguimiento curricular del programa

Lugar y fecha de elaboración o revisión	Participantes	Evento
Instituto Tecnológico de Hermosillo Agosto – Diciembre 2016	MSI Bettina Elisa Santa Cruz Welsh M.C. Martha Patricia Sevilla Zazueta	Jornadas Curriculares para definir las especialidades de las carreras de ISC e II.

	MSI Francisca Lorena Zepeda Miramontes MCC Ana Luisa Millán Castro MCC José Miguel Rodríguez Pérez	

4. Competencia(s) a desarrollar

Competencia(s) específica(s) de la asignatura
Desarrollar soluciones de software, considerando los aspectos de diseño, componentes y abstracción, mediante la implementación de la arquitectura 4+1 vistas y el uso de UML para el modelado de la solución, así también, considerando las reglas de diseño de interfaz de usuario y de diseño de bases de datos y el uso de diseño de patrones Arquitectónico MVC y de diseño software. (Diseño OO, Arquitectura de SW, Patrones Arquitectónicos MVC).

5. Competencias previas

- Utiliza algún lenguaje de programación orientado a objetos.
- Aplica razonamiento lógico.
- Identifica conceptos básicos de Bases de Datos.
- Aplicar modelos, técnicas y herramientas para cada una de las etapas del ciclo de vida de desarrollo de software.

6. Temario

No.	**Temas**	**Subtemas**
1	Arquitectura de software.	1.1 Introducción. 1.1.1 Conceptos fundamentales. 1.1.2 Campos de la Arquitectura de Software. 1.1.3 Modalidades y tendencias. 1.1.4 Diferencias entre Arquitectura y Diseño. 1.1.5 Relevancia de la Arquitectura de Software. 1.1.6 Rol y responsabilidades del arquitecto de software. 1.2 Estilos y patrones. 1.2.1 Introducción.

		1.3 Tipos de arquitectura y sus capas. 1.3.1 Cliente/Servidor. 1.3.2 Capas. 1.3.3 SOA. 1.4 Modelos y escenarios: 4+1 Vistas. 1.4.1 El Modelo de 4+1 Vistas. 1.4.2 La Arquitectura Lógica. 1.4.3 La Vista de Procesos. 1.4.4 Vista de Desarrollo. 1.4.5 Arquitectura Física. 1.4.6 Escenarios. 1.4.7 Correspondencia entre las Vistas. 1.4.8 Confeccionando el Modelo. 1.4.9 Documentación de la Arquitectura. 1.5 Lenguajes de descripción arquitectónica ADL. 1.5.1 Criterios de definición de un ADL. 1.5.2 Lenguajes: Acme / Armani ADLs: Acme /Armani – ADML – Aesop – ArTek – C2 SADL – CHAM – Darwin – Jacal – LILEANNA – MetaH / AADL – Rapide – UML – UniCon – Wright –xArch / xADL. 1.6 UML. 1.6.1 Introducción. 1.6.2 Modelos computacionales y paradigmas de modelado. 1.6.3 Componentes y restricciones. 1.6.4 Diagramas de UML
2	Frameworks, plug-ins y componentes de Software.	2.1 Introducción. 2.2 Relación de los framework con los estilos arquitectónicos. 2.3 Familias de programas y frameworks 2.4 Plug-ins 2.5 Componentes de software 2.6 Desarrollo de Software basado en Componentes
3	Patrón Arquitectónico MVC.	3.1. De estructura 3.1.1. Capas

		3.1.2. Pipas y filtros 3.1.3. Blackboard 3.2. De distribución 3.2.1. Broker 3.2.2. Cliente servidor 3.2.3. Peer to Peer 3.3. De sistemas interactivos 3.1. Modelo-Vista- Controlador 3.2. Controlador- Supervisor 3.3. Presentación- abstracción-control 3.4. De sistemas adaptables 3.4.1. Microckernel 3.4.2. Reflection
4	Los patrones de diseño software.	4.1 Documentación de los patrones de diseño 4.2. Tipos de patrones 4.2.1. Básicos 4.2.2. Creación de objetos 4.2.3. De colecciones 4.2.4. De estructura 4.2.5. De comportamiento 4.2.6. De concurrencia 4.3. Patrones comunes 4.3.1. Interfaces y clases abstractas 4.3.2. Métodos de acceso 4.3.3. Observador (publisher - subscriber) 4.3.4. Decorador 4.3.5. Método Factory 4.3.6. Singleton 4.3.7. Command 4.3.8. Adaptador 4.3.9. Método Template 4.3.10. Iterator 4.3.11. Composite 4.3.12. Estado (State) 4.3.13. Proxy

7. Actividades de aprendizaje de los temas

1. Arquitectura de software	
Competencias	Actividades de Aprendizaje
Específica(s): Crear la arquitectura 4+1 Vistas y usa UML para el modelado de la solución, elaborando la vista lógica a través del diagrama de clases de UML y la vista de procesos a través de los diagramas de estado, de actividades, secuencia o colaboración según sea el tipo de sistema de su caso de estudio. Elaborar, así también, la vista de usuarios del proyecto según su caso de estudio. **Genéricas:** **a) instrumentales** • Capacidad de análisis y síntesis • Capacidad de organizar y planificar • Comunicación oral y escrita • Habilidad analizar información • Solución de problemas • Toma de decisiones. **b) Interpersonales** • Capacidad crítica y autocrítica • Trabajo en equipo • Habilidades interpersonales **c) Sistémicas** • Capacidad de aplicar los conocimientos en la práctica • Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad) • Habilidad para trabajar en forma autónoma • Búsqueda del logro	• Fomentar el uso de las tecnologías de información y comunicación. • Realizar visitas a empresas que para observar los roles del Arquitecto de software, los modelos utilizados y alcances de las actividades de arquitectura de software. • Solicitar al estudiante que realice investigaciones en diversas fuentes de información sobre arquitectura de software. • Relacionar los contenidos de este tema con los obtenidos en las demás del plan de estudios y de la especialidad, reforzando la importancia de tener una visión y práctica interdisciplinaria para alcanzar las metas académicas, profesionales y empresariales. • Analizar un conjunto de casos prácticos. • Cuando los temas lo requieran, utilizar medios audiovisuales para una mejor comprensión del estudiante. • Integrar equipos de trabajo en donde se compartan conocimientos y experiencias académicas y laborales. • Discutir en grupo la información generada por los equipos de trabajo. • Analizar cada una de las vistas del modelo, los escenarios, las correspondencias entre las vistas, modelado y documentación y plasmar. • Investigar las reglas de diseño de interfaz de usuario e integrar la interfaz al caso de uso, elaborando la vista de usuarios del proyecto según su caso de estudio. • Diseñar la base de datos de su proyecto según su caso de estudio y considera buenas prácticas de diseño, por ejemplo: normaliza

	la base de datos). Elaborar el diagrama E-R en una herramienta de software apropiada.
2. Frameworks, plug-ins y componentes de Software	
Competencias	Actividades de Aprendizaje
Específica(s): Identifica los conceptos de *Frameworks*, *plug-ins* y componentes de Software y toma decisión de su uso según el proyecto de su caso de estudio **Genéricas:** **a) instrumentales** • 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 • Toma de decisiones. **b) Interpersonales** • Capacidad crítica y autocrítica • Trabajo en equipo • Habilidades interpersonales **c) Sistémicas** • Capacidad de aplicar los conocimientos en la práctica • Habilidades de investigación • Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad) • Habilidad para trabajar en forma autónoma • Búsqueda del logro	• Fomentar el uso de las tecnologías de información y comunicación. • Solicitar al estudiante que realice investigaciones en diversas fuentes de información sobre frameworks, plug-ins y componentes de software. • Relacionar los contenidos de este tema con los obtenidos en las demás del plan de estudios y de la especialidad, reforzando la importancia de tener una visión y práctica interdisciplinaria para alcanzar las metas académicas, profesionales y empresariales. • Elaborar un conjunto de casos prácticos. • Cuando los temas lo requieran, utilizar medios audiovisuales para una mejor comprensión del estudiante. • Integrar equipos de trabajo en donde se compartan conocimientos y experiencias académicas y laborales. • Identificar en un proyecto de Software el framework, plug-ins y componentes utilizados. • Discutir en grupo la información generada por los equipos de trabajo.
3. Patrón Arquitectónico MVC	
Competencias	Actividades de Aprendizaje
Específica(s): Considera la implementación del patrón	• Fomentar el uso de las tecnologías de información y comunicación.

arquitectónico MVC a través de algún framework o algún IDE dando inicio con la fase de desarrollo del proyecto. **Genéricas:** **a) instrumentales** • 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 • Toma de decisiones. **b) Interpersonales** • Capacidad crítica y autocrítica • Trabajo en equipo • Habilidades interpersonales **c) Sistémicas** • Capacidad de aplicar los conocimientos en la práctica • Habilidades de investigación • Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad) • Habilidad para trabajar en forma autónoma • Búsqueda del logro	• Solicitar al estudiante que realice investigaciones en diversas fuentes de información sobre patrones arquitectónicos según estructura, distribución, de sistemas interactivos y de sistemas adaptables. • Solicitar al estudiante que realice investigaciones en diversas fuentes de información sobre el patrón arquitectónico MVC específicamente y analizar sus características y detalles importantes a considerar. • Relacionar los contenidos de este tema con los obtenidos en las demás del plan de estudios y de la especialidad, reforzando la importancia de tener una visión y práctica interdisciplinaria para alcanzar las metas académicas, profesionales y empresariales. • Elaborar un conjunto de casos prácticos. •Integrar equipos de trabajo en donde se compartan conocimientos y experiencias académicas y laborales. • Discutir en grupo la información generada por los equipos de trabajo. • Implementar el patrón arquitectónico MVC a través de algún framework o algún IDE dando inicio con la fase de desarrollo del proyecto. • Empezar a codificar los catálogos del sistema, aplicando el patrón arquitectónico, por ejemplo, hacer la vista de usuario, el controlador de usuario y modelo de datos de usuario, y así con el resto de los catálogos.
4. Los patrones de diseño software	
Competencias	Actividades de Aprendizaje
Específica(s): Aprende mediante ejemplos la aplicación de los patrones de diseño más e implementa al menos 4 patrones de diseño en el desarrollo de su codificación.	• Fomentar el uso de las tecnologías de información y comunicación. • Solicitar al estudiante que realice investigaciones en diversas fuentes de información sobre patrones de diseño.

Genéricas: **a) instrumentales** • 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 • Toma de decisiones. **b) Interpersonales** • Capacidad crítica y autocrítica • Trabajo en equipo • Habilidades interpersonales **c) Sistémicas** • Capacidad de aplicar los conocimientos en la práctica • Habilidades de investigación • Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad) • Habilidad para trabajar en forma autónoma. • Búsqueda del logro	• Solicitar al estudiante que realice investigaciones en diversas fuentes de información sobre la documentación de los patrones de diseño, tipos de patrones y los patrones comunes • Relacionar los contenidos de este tema con los obtenidos en las demás del plan de estudios y de la especialidad, reforzando la importancia de tener una visión y práctica interdisciplinaria para alcanzar las metas académicas, profesionales y empresariales. • Elaborar un conjunto de casos prácticos. •Integrar equipos de trabajo en donde se compartan conocimientos y experiencias académicas y laborales. • Discutir en grupo la información generada por los equipos de trabajo. • Implementar al menos 4 patrones de diseño en el desarrollo de su codificación.

8. Prácticas

• Integrar equipos de trabajo • Investigar los Antecedentes históricos que han dado lugar a las prácticas de la Arquitectura de Software hoy en día. • Realizar un mapa mental de los conceptos fundamentales de la Arquitectura de Software • Realizar un cuadro sinóptico de las diferencias entre Arquitectura y Diseño de Software • Discutir en el grupo a manera de mesa redonda a cerca de los campos de la arquitectura de software, modalidades y tendencias. • Realizar un ensayo a cerca de la relevancia de la Arquitectura del Software en el desarrollo • Regional y en el desarrollo global del software • Investigar las Características y competencias del Arquitecto de Software • Identificar en un Diagrama, las fases en que participa el Arquitecto de Software

• Identificar en un proyecto de Software desarrollado en alguna asignatura anteriormente, las fases en que participa el Arquitecto del Software y los tipos de arquitectos que pueden trabajar en ese proyecto de manera justificada.
• Investigar las características principales de los ADL y los criterios de Definición de los mismos.
• Crear un mapa conceptual de los lenguajes más relevantes de descripción arquitectónica con sus principales características.
• Realizar una matriz comparativa, evaluarla y discutir en grupo.
• Crear la arquitectura 4+1 Vistas para el modelado de la solución, elaborando la vista lógica a través del diagrama de clases de UML y la vista de procesos a través de los diagramas de estado, de actividades, secuencia o colaboración según sea el tipo de sistema de su caso de estudio.
• Elaborar la vista de usuarios del proyecto según su caso de estudio.
• Documentar la arquitectura
• Elaborar la vista de usuarios del proyecto según su caso de estudio.
• Diseñar la base de datos de su proyecto según su caso de estudio considerando buenas prácticas de diseño, por ejemplo: normaliza la base de datos).
• Elaborar el diagrama E-R en una herramienta de software apropiada.
• Analizar las características de los diferentes frameworks y plug-ins más utilizados en el desarrollo del software.
• Investigar los Beneficios del Desarrollo de Software basado en Componentes.
• Identificar en un proyecto de Software desarrollado el framework, plug-ins y componentes necesarios a utilizar en su caso de estudio.
• Identificar y explicar las características de un modelo de referencia, estilo o patrón de arquitectura, arquitectura de referencia y arquitectura de software.
• Investigar sobre ejemplos de sistemas de software construidos con base en cada uno de los distintos estilos de arquitectura.
• Elaborar un cuadro sinóptico que identifique las principales diferencias y similitudes entre los distintos estilos de arquitecturas de software, así como el tipo de problema que busca resolver cada estilo.
• Elaborar un diagrama que muestre la evolución histórica de los estilos de arquitecturas de software.
• Elaborar una plantilla que especifique los principales elementos que debe contener la documentación de la arquitectura de un sistema de software.
Investigar qué son los patrones de arquitectura, para qué sirven, y cómo se documentan.
• Elaborar un mapa conceptual que describa y clasifique los diversos patrones arquitectónicos.
• Analizar y discutir en grupo el tipo de problema que busca resolver cada patrón arquitectónico.

• Proponer al menos un ejemplo de sistema en el que pudiera ser aplicado cada tipo de patrón arquitectónico.
• Reconocer algunos de los patrones de arquitectura más representativos, aplicables en el diseño de una arquitectura de software.
• Empezar a codificar los catálogos del sistema, aplicando el patrón arquitectónico, por ejemplo, hacer la vista de usuario, el controlador de usuario y modelo de datos de usuario, y así con el resto de los catálogos.
• Diseñar un mapa mental a cerca del proceso de documentación de los Patrones de diseño que incluya lo siguiente:
• Tipos de patrones básicos y comunes.
• Elaborar un cuadro sinóptico a cerca de diferencias y semejanza entre los diferentes tipos de patrones.
• Analizar casos reales y generar un ensayo donde se reflejen las conclusiones
• Identificar los posibles patrones según su caso de estudio
• Elección del (los) patrón(es) a utilizar
• Implementa al menos 4 patrones de diseño en el desarrollo de su codificación en el modelo de diseño.
• Documentación de la arquitectura y el diseño.

9. Proyecto de Asignatura

Desarrollar un proyecto de software real, considerando continuar con el proyecto que se viene desarrollando desde los cursos de fundamentos de Ingeniería de Software e Ingeniería de Software para darle continuidad con las fases de desarrollo según los temas desarrollados en clase.

Así, 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

Para evaluar las actividades de aprendizaje se recomienda solicitar:

- Solución de casos prácticos solicitados durante las actividades, así como sus conclusiones de forma escrita.
- Reportes de investigación.
- Reportes de prácticas.
- Ejercicios realizados.
- Tareas.
- Exposición.
- Participación en clase.
- Proyecto.
- Ensayos.
- Cuadros sinópticos
- Portafolio de evidencias
- Exámenes teóricos y/o prácticos.

Para verificar el nivel de logro de las competencias del estudiante se recomienda utilizar:

- Listas de cotejo.
- Guías de observación.
- Coevaluación y autoevaluación
- Realizar rúbricas para cada caso

11.- Fuentes de Información

1. Sommerville, Ian. Ingeniería de Software. Prentice Hall. 2001.
2. Pressman Roger S. Ingeniería del Software, 5/E. Mc.Gaw-Hill. 2001
3. Booch, Grady, El Lenguaje Unificado de Modelado. Addison Wesley Iberoamericana, 1999.
4. Booch, Grady. Object-Oriented Analysis and Design. Second Edition. Benjamin/Cummings, Redwood: 1994.
5. Jacobson, Ivar. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, 1992.
6. Jacobson, Ivar, Grady Booch, and James Rumbaugh. El Proceso Unificado de Desarrollo de Software. México: Addison-Wesley, 1999.
7. J. Rumbaugh. Object-Oriented Modeling and Design. et al.Prentice- Hall,1991.
8. Larman, Craig UML y Patrones. Prentice Hall, 1999.
9. Larman, Craig. UML y Patrones, Introducción al análisis y diseño orientado a objetos. México: Prentice Hall, 1999.
10. Oestereich, Bernd. Developing software with UML Object-oriented analysis and design in practice. Addison-Wesley, 2001.
11. Bruegge, Bernd. Ingeniería de software Orientada a objetos. Prentice Hall. 2000
12. Kruchten, Philippe. "Architectural Blueprints--The 4+1 View Model of Software Architecture". IEEE Software, Institute of Electrical and Electronics Engineers. November 1995, pp. 42-50.

13. Martin, Robert C. "Design Principles and Design Patterns". Objectmentor

14. Muller, Pierre-Alain. Modèlisation Object avec UML. Paris: Eyrolles, 1997.

15. Wilson, Scott F. Analyzing Requirements and Defining Solution Architectures. Redmond: Microsoft Press, 1999.

16. Fernández Aramayo, David Ricardo. Arquitectura de Software. Universidad Tecmilenio, ITESM

17. Zapata Sanchez, Andres felipe. Arquitectura de Software www.fi.uba.ar

18. Meylin Siguas Villavicencio www.unpmsn.org

19. Rozanski, N., Woods,E., "Software Systems Architecture", Addison Wesley. 2005

20. Bass, L. Clements, P., Kazman, R., "Software Architecture in Practice", Addison-Wesley, Second Edition, 2006.

21. Paul Clements et al, "Documenting Software Architectures: Views and Beyond", Addison Wesley, 2002.

22. Paul Clements et al, "Evaluating Software Architectures", Addison Wesley, 2002.

23. Erl, T., "SOA Principles of Service Design", Prentice Hall, 2008

24. Richard Taylor, Nenad Medvidovic, Eric Dashofy. "Software Architecture Foundations, Theory and Practice, 2009.

25. UML Resource Center. Rational Software. http://www.rational.com/uml/

26. Rose-Hulman Institute of Technology, Terre Haute, IN, USA. CSSE 477, Software Architecture. http://www.rose-hulman.edu/class/csse/csse477/
Versión: circa 2011-12.

27. Eddie Burris, Programming in the Large with Design Patterns, Pretty Print Press, 2012.

28. Partha Kuchana, Software Architecture Design Patterns in Java, AUERBACH, Boca Raton, USA, 2004.

29. Elisabeth Freeman, Eric Freeman, Bert Bates and Kathy Sierra, Head First Design Patterns - O'Reilly, 2004.

30. Jason McC. Smith, Elemental Design Patterns, Addison-Wesley, 2012.

31. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns. Elements of Reusable Object-Oriented Software - Addison Wesley, 1994.

32. Buschmann, Meunier, Rohnert, Sommerlad, Stal, Pattern Oriented Software Architecture Volume 1: A System of Patterns - Wiley, 1996.

33. Schmidt, Stal, Rohnert, Buschmann, Pattern Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects - Wiley, 2000

34. Kircher, Jain, Pattern Oriented Software Architecture Volume 3: Patterns for Resource Management - Wiley, 2004.

35. Buschmann, Henney, Schmidt, Pattern Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing - Wiley, 2007.

36. Buschmann, Henney, Schmidt, Pattern Oriented Software Architecture Volume 5: On Patterns and Patterns Language - Wiley, 2007.

37. AntiPatterns. Refactoring Software, Architectures and Projects in Crisis - Brown, Malveau, McCormick Mowbray – Wiley

38. Patterns in Java - Mark Grand – Wiley

39. Frameworks y Componentes (... ¡¡¡reutilizar, reutilizar, reutilizar!!! ...) Universidad de los Andes Demián Gutierrez Abril 2010

http://www.codecompiling.net/files/slides/IS_clase_10_frameworks_componentes.pdf

40. Desarrollo de Software basado en Componentes
https://msdn.microsoft.com/es-es/library/bb972268.aspx