

**PROGRAMA CURSO: IF7100 INGENIERÍA DEL SOFTWARE**  
I Semestre, 2017

**Datos Generales**

---

**Sigla:** IF7100  
**Nombre del curso:** Ingeniería del Software  
**Tipo de curso:** Teórico - Práctico  
**Número de créditos:** 4  
**Número de horas semanales presenciales:** 6  
**Requisitos:** IF-6100 Análisis y Diseño de Sistemas  
**Co-requisitos:** Ninguno  
**Ubicación en el plan de estudio:** VII Ciclo  
**Tutoría:** (NO)  
**Suficiencia:** (NO)  
**Modalidad:** (Presencial, bajo virtual)

**Horario del curso:**

L 17:00 a 19:50
V: 15:00 a 17:50

**Datos del Profesor**

---

**Nombre:** Oscar Alfaro Solís  
**Correo Electrónico:** [osky.alfaro@gmail.com](mailto:osky.alfaro@gmail.com)

**Horas consulta presencial:** Días de Clase, reuniones de seguimiento de proyectos.

**Horas consulta remota:** Martes, Miércoles y Viernes 4:00 pm a 06:00 pm.

**Medio oficial para consulta remota:** Skype: oscar.alfaro.samtec

**1. Descripción del curso**

Este curso proporciona al y la estudiante conocimientos sobre conceptos, métodos, metodologías y herramientas de la ingeniería de software empleadas en el diseño, desarrollo, pruebas, operación y mantenimiento de productos de software.

El curso pone especial énfasis en aspectos de modelado arquitectónico del software, herramientas CASE de soporte al proceso y en la gestión de la calidad del producto a través de pruebas exhaustivas de distinta índole. Adicionalmente, el curso le proporciona al y la estudiante un espacio de aplicación a través del desarrollo de un proyecto de software.



---

## Objetivos

### 2. Objetivo General

El curso le permitirá al y la estudiante:

Adquirir una visión integral de los procesos de la ingeniería de software así como desarrollar sus conocimientos y habilidades a nivel de buenas prácticas de la ingeniería de software para producir aplicaciones de software de alta calidad.

---

### 3. Objetivos Específicos

Al finalizar el curso el o la estudiante estará en capacidad de:

- Reconocer el carácter transdisciplinario de la ingeniería de software.
- Proponer el modelo de diseño de los productos de software que construya.
- Proponer una arquitectura de software que atienda los atributos calidad de la misma en función del contexto del proyecto de software.
- Realizar la implementación del producto de software en función del diseño y la arquitectura que haya formulado.
- Vivenciar el proceso integración de productos intermedios y de módulos en un proyecto de desarrollo de software
- Diseñar estrategias de pruebas pertinentes al producto de software que se esté desarrollando.
- Administrar los procesos de implantación de las aplicaciones que desarrolle.
- Gestionar el proceso de administración de configuración del software.
- Emplear eficazmente herramientas CASE modernas para soportar los procesos de desarrollo de software de manera más estructurada y sistematizada
- Adoptar una actitud crítica ante la diversidad de temáticas asociadas con la ingeniería de software

---

### 4. Contenidos

#### 1. INTRODUCCIÓN INGENIERÍA DE SOFTWARE

#### 2. GESTIÓN DE INGENIERIA DE SOFTWARE

1. Definición inicial y alcances
2. Planificación proyectos



BACHILLERATO EN INFORMÁTICA EMPRESARIAL

3. Revisión y evaluación
4. Cierre proyectos
  
3. REQUERIMIENTOS DE SOFTWARE
  1. Fundamentos de requerimientos
  2. Procesos de requerimientos
  3. Obtención de requerimientos
  4. Análisis de requerimientos
  5. Especificación de requerimientos
  6. Validación de requerimientos
  
4. DOCUMENTACIÓN Y MODELADO DE PROCESOS
  1. Conceptos: proceso, características de los procesos
  2. Tipos de procesos: operativos, de apoyo y de gestión
  3. Formas alternativas para documentar los procesos
  4. Relación de los procesos con la ingeniería de software
  5. Técnicas para la definición y diseño de procesos: diagramas de actividad UML y diagramas de flujos de proceso (flujogramas)
  6. Notación y simbología utilizada para modelado de procesos
  7. Herramientas CASE para el modelado de procesos
  
5. DISEÑO DE SOFTWARE
  1. Fundamentos de diseño
  2. Análisis y evaluación de la calidad del diseño
  3. Notación de diseño software
  4. Estrategias y métodos de diseño
  
6. ARQUITECTURA DEL SOFTWARE
  - 6.1 Definiciones: arquitectura versus diseño, arquitectura de software, elementos-relaciones y propiedades de la arquitectura del software
  2. Razones por las cuales es necesaria la arquitectura de software
  3. Atributos de calidad de la arquitectura de software: performance, exactitud, seguridad, mantenibilidad, portabilidad
  4. Estilos arquitectónicos: estilo módulo, estilo componente y conector, estilo de alocaación (despliegue)
  5. Diagrama de Contexto
  6. Vistas arquitectónicas 4+1



BACHILLERATO EN INFORMÁTICA EMPRESARIAL

7. Modelado arquitectónico del software
  8. Arquitectura Física: N-Tier Architecture: Tipos ( Cliente-Servidor, 3-Tier )
  9. Beneficios Arquitectura Física: Escalabilidad, seguridad y tolerancia a fallas
  10. Arquitectura Lógica: N-Layer Architecture: Tipos ( 3-layers, n-layers )
  11. Beneficios: mantenibilidad, reutilización, distribución del trabajo, flexibilidad, robustez, entre otros
  12. Capas lógicas de implementación (aplicación, negocios y servicio de datos)
  13. Subsistemas
- 
7. PATRONES DE DISEÑO
    1. Descripción de los patrones
    2. Clasificación
    3. Creacionales, Estructurales, De Partición, De Comportamiento
- 
8. CONSTRUCCIÓN DE SOFTWARE
    1. Fundamentos de construcción
    - 8.2 Gestión de construcción
- 
9. GESTIÓN DE CALIDAD DEL PRODUCTO
    1. Concepto de Calidad
    2. Prevención versus Detección
    3. Clientes de la calidad: interno, externo y oculto
    4. Calidad del proyecto versus calidad del producto
    5. Verificación versus Validación
    6. Aseguramiento y Control de la Calidad del Software
- 
10. PRUEBAS DEL SOFTWARE
    1. Propósitos de la disciplina de pruebas
    2. Actividades de la evaluación de la calidad del software: pruebas funcionales (caja blanca), pruebas estructurales (caja negra), análisis estático y análisis dinámico
    3. Flujo de trabajo general de la disciplina: Identificación del objetivo de las pruebas, selección de las entradas, definición de salidas esperadas, configuración del ambiente de ejecución, ejecución del programa, análisis de los resultados.
    4. Implementación de Pruebas
      - Pruebas de Unidad
      - Pruebas de Integración
      - Pruebas de Validación



BACHILLERATO EN INFORMÁTICA EMPRESARIAL

- Pruebas del Sistema: funcionalidad, seguridad, robustez, cargado, estabilidad, de estrés, performance y fiabilidad
  - Pruebas de Aceptación
  - Pruebas Funcionales: Casos de prueba, matriz de cobertura de pruebas, matriz de casos de prueba, procedimientos de prueba, escenarios de prueba, scripts de prueba
5. Herramientas CASE para efectuar pruebas del software

11. IMPLANTACIÓN

1. Propósitos de la disciplina
2. Estrategias para la implantación del producto
3. Migración de datos
4. Capacitación de usuarios

12. MANTENIMIENTO DEL SOFTWARE

1. Propósitos de la disciplina
2. Elementos claves del mantenimiento: usuario, ambiente, ambiente operativo, ambiente organizacional, proceso de mantenimiento, producto de software, personal de mantenimiento.
3. Cambios del software: correctivos, adaptivos, perfectivos, preventivos
4. Procesos de mantenimiento
5. Técnicas para mantenimiento

13. ADMINISTRACIÓN DE LA CONFIGURACIÓN DEL SOFTWARE (ACS)

1. Definición de la Administración de la Configuración del Software
2. Actividades de la disciplina: identificación, almacenamiento, control del cambio y reporte del estado
3. Mejores prácticas de la disciplina.
4. Conceptos: metadata, línea base, roles y responsabilidades, ítem de configuración, árbol de versiones (versión y revisión), repositorio
5. El proceso de la Administración de Configuración de Software
6. Control de versiones, Control del Cambio, Reporte Estado, Auditoría de Configuración
7. Herramientas CASE para la ACS

14. METODOLOGÍAS DE DESARROLLO DEL SOFTWARE



1. Concepto de metodología
2. Modelo de procesos de desarrollo empleados en las metodologías (cascada, iterativo-incremental)
3. Aspectos fundamentales de las metodologías: fases, disciplinas e hitos (milestones), flujos de trabajo, artefactos y productos de entrada y salida de las fases, roles y responsabilidades
4. Metodología estructurada: Proceso Unificado de Rational
5. Metodologías ágiles: Extreme Programming y SCRUM

---

## 5. Metodología

El curso presenta un eje de desarrollo teórico-práctico.

El profesor desarrolla las clases soportado en diapositivas, modelos UML, plantillas de artefactos y código fuente. Algunas clases se realizarán en el laboratorio y estarán acompañadas de ejercicios prácticos sobre las temáticas siendo estudiadas.

Los materiales didácticos estarán disponibles en un grupo virtual creado para el curso. Visitar y registrarse en: dirección URL

Los y las estudiantes realizan lecturas semanales y presentan resúmenes. De esta forma se pretende que durante la clase los y las estudiantes tengan conocimiento del tema para poder participar. Los resúmenes serán de carácter individual, únicamente.

Los y las estudiantes desarrollan un proyecto de software. El mismo lo gestionarán mediante el marco de trabajo denominado Proceso Unificado. Los requerimientos técnicos y las políticas de evaluación del proyecto estarán consignadas en un documento de proyecto que se entregará de forma oportuna. El proyecto tendrá puntos de control en los cuales los y las estudiantes entregarán los elementos o artefactos definidos en el enunciado del proyecto.

Además, los y las estudiantes realizarán exposiciones acerca de temáticas de interés que complementen el desarrollo del curso. Los temas de exposición serán entregados en la segunda semana de clase, así como las fechas y los aspectos a ser evaluados. Algunas exposiciones serán de carácter práctico.

---

## 6. Evaluación



Descripción	Porcentaje
I Parcial	20%
II Parcial	20%
Evaluaciones cortas y Tareas	10%
Proyecto	40%
Exposiciones	10%

### Consideraciones sobre la evaluación

- ❑ Evaluaciones cortas semanales. No se harán reposiciones.
- ❑ La entrega de proyecto y de los puntos de control debe ir acompañada de la respectiva documentación. No se recibe documentación posterior a la defensa del proyecto.
- ❑ La comprobación de que alguna tarea individual, proyecto o examen es una copia aplicará las sanciones que contemple el reglamento. Consultar en:  
[http://cu.ucr.ac.cr/normativ/regimen\\_academico\\_estudiantil.pdf](http://cu.ucr.ac.cr/normativ/regimen_academico_estudiantil.pdf)
- ❑ Sólo se recibirán discos etiquetados y cuyo contenido sea exclusivamente la asignación respectiva.
- ❑ No se reciben trabajos que carezcan de portada y no estén engrapados.

### 7. Cronograma

Semana 1 - 13 al 18 Marzo	Actividades
Inicio de clases	Entrega y Lectura carta del estudiante Introducción ingeniería de software
Semana 2 - 20 al 25 Marzo	Actividades
	GESTIÓN DE INGENIERIA DE SOFTWARE
Semana 3 - 27 de Marzo al 01 de Abril	Actividades
	REQUERIMIENTOS DE SOFTWARE



BACHILLERATO EN INFORMÁTICA EMPRESARIAL

<b>Semana 4 - 3 al 8 Abril</b>	<b>Actividades</b>
	REQUERIMIENTOS DE SOFTWARE
<b>Semana 5 - 10 al 15 Abril</b>	<b>Actividades</b>
	Semana Santa
<b>Semana 6 - 17 al 22 de Abril</b>	<b>Actividades</b>
	Diseño del Software
<b>Semana 7 - 24 al 29 de Abril</b>	<b>Actividades</b>
	<b>SEMANA UNIVERSITARIA</b> ARQUITECTURA DEL SOFTWARE
<b>Semana 8 - 01 al 06 Mayo</b>	<b>Actividades</b>
	ARQUITECTURA DEL SOFTWARE
<b>Semana 9 - 2 al 07 Mayo</b>	<b>Actividades</b>
	<b>Primer Examen Parcial</b> CONSTRUCCIÓN DE SOFTWARE
<b>Semana 10 - 15 al 20 Mayo</b>	<b>Actividades</b>
	GESTIÓN DE CALIDAD DEL PRODUCTO
<b>Semana 11 - 22 al 27 Mayo</b>	<b>Actividades</b>
	GESTIÓN DE CALIDAD DEL PRODUCTO
<b>Semana 12 - 29 de Mayo al 03 de Junio</b>	<b>Actividades</b>
	PRUEBAS DEL SOFTWARE
<b>Semana 13 - 05 al 10 Junio</b>	<b>Actividades</b>
	IMPLANTACION DEL SOFTWARE
<b>Semana 14 - 12 al 17 Junio</b>	<b>Actividades</b>
	MANTENIMIENTO DEL SOFTWARE
<b>Semana 15 - 19 al 24 Junio</b>	<b>Actividades</b>
	ADMINISTRACION DE LA CONFIGURACION DEL SOFTWARE
<b>Semana 16 - 26 de Junio al 01 de Julio</b>	<b>Actividades</b>



	<b>SEGUNDO EXAMEN PARCIAL</b> AVANCES DEL PROYECTO
Semana 17 - 03 al 08 Julio	Actividades
	Avances del proyecto
Semana 18 - 10 al 15 Julio	Actividades
	PRESENTACION DE PROYECTOS ENTREGA DE NOTAS
Semana 19 - 11 al 16 Julio	Actividades
	AMPLIACION

---

## 8. Normativa del Curso

## 9. Bibliografía

- <http://www.swebok.org/> (IEEE - Guide to the Software Engineering Body of Knowledge)
- <http://www.computer.org/tse/> (IEEE Transactions in Software Engineering)
- <http://www.sei.cmu.edu/> (SEI - Software Engineering Institute)
- [Bass, L. et al. 2003] Bass, Len; Clements, Paul; Kazman, Rick. **Software Architecture in Practice**. Segunda edición. Addison Wesley, 2003
- [Braude 2005] Braude, Eric J. **Ingeniería de Software: Una Perspectiva Orientada A Objetos**. Alfaomega, 2005 - *Cap. 1 Introducción*
- [Clements, P y otros, 2010] Clements, Paul; Bachmann, Felix; Bass, Len; Garlan, David; Ivers, James; Little, Reed; Merson, Paulo; Nord, Robert; Stafford, Judith. **Documenting Software Architectures: Views and Beyond**. Second Edition. USA, Pearson Education Cap. 6 – Sección 6.3 Context Diagrams
- Ferm, Fredrik. **The what, why, and how of a subsystem**. IBM DeveloperWorks, 25 Noviembre 2003



BACHILLERATO EN INFORMÁTICA EMPRESARIAL

Disponible en: <http://www.ibm.com/developerworks/rational/library/1761.html>

Fecha de consulta: 30/3/2010

- [Hunt, J. 2003] Hunt, John. **Guide to the Unified Process featuring UML, Java and Design Patterns**. Springer, UK, 2003 - Cap. 4.
- Kontio, Mikko. **Architectural manifesto: Designing software architectures, Part 5**. IBM DeveloperWorks, 02 Feb 2005  
Disponible en: <http://www.ibm.com/developerworks/wireless/library/wi-arch11/>  
Fecha de consulta: 21/3/2010
- [Kurbel, K. 2008] Kurbel K. E. **The Making of Information Systems: Software Engineering and Management in a Globalized World**. Springer-Verlag Berlin Heidelberg, Germany, 2008 – Cap. 3
- [Lhotka, R. 2009] Lhotka, Rockford. **Expert C# 2008 Business Objects**. Apress, USA, 2009 – Cap. 1 Distributed Architecture
- [Manassis, E. 2003] Manassis, Enricos. **Practical Software Engineering: Analysis and Design for the .NET Platform** Addison Wesley, 2003 – *Cap. 1 Introducción y Cap. 9 Testing*
- Mitra, Tilak. **Documenting software architecture, Part 2: Develop the system context**. IBM DeveloperWorks, 13 May 2008  
Disponible en: <http://www.ibm.com/developerworks/library/ar-archdoc2/>  
Fecha de consulta: 21/3/2010
- [Pressman, Roger 2005 ] **Ingeniería del Software, Un Enfoque Práctico**, Sexta Edición. USA, Mc Graw Hill
- [Schach, S. 2007] Schach, Stephen. **Ingeniería de software clásica y orientada a objetos**. Sétima edición. USA. McGraw-Hill. 2007.
- [Weitzenfeld, 2005] Weitzenfeld, Alfredo. **Ingeniería de software orientada a objetos con UML**. México. Thomson
- 
- [Sommerville, Ian 2005] **Ingeniería de Software**. Séptima edición, Prentice Hall, 2005
- [Zielinski, 2005] Zielinski, Krzysztof. **Software Engineering: Evolution and Emerging Technologies**. Amsterdam, Holanda. IOS Press