PROGRAMA CURSO: IF7100

I Semestre, 2016

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
Número de horas semanales de trabajo independiente del estudiante: 5
Requisitos: IF-6100 Análisis y Diseño de Sistemas
Correquisitos: Ninguno
Ubicación en el plan de estudio: VII Ciclo
Horario del curso:

Grupo 01:

Grupo 02:

V: 15:00 a 17:50
S: 9:00 a 12:00

L : 18:00 a 20:50
K : 18:00 a 19:50

Datos del Profesor

Grupo 1
Nombre: Jose Eliecer Montero
Correo Electrónico: jose.montero@ucrso.info, jelimv@hotmail.com
Skype: **eliecermv**

Horario de Consulta: Dias de Clase, reuniones de seguimiento de proyectos.

Grupo 2
Nombre: Juan Carlos Miranda
Correo Electrónico: jmiranda@scgint.com
juancarlos.miranda@ucr.ac.cr
Skype: juankamiranda

Horario de Consulta: Dias de Clase, reuniones de seguimiento de proyectos.

PROGRAMA CURSO: IF7100

I Semestre, 2016

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.

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.

Objetivo 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

Contenido del Curso

1. INTRODUCCIÓN INGENIERÍA DE SOFTWARE

2. GESTIÓN DE INGENIERIA DE SOFTWARE

2.1 Definición inicial y alcances
2.2 Planificación proyectos
2.3 Revisión y evaluación
2.4 Cierre proyectos

3. REQUERIMIENTOS DE SOFTWARE
3.1 Fundamentos de requerimientos
3.2 Procesos de requerimientos
3.3 Obtención de requerimientos
3.4 Análisis de requerimientos
3.5 Especificación de requerimientos
3.6 Validación de requerimientos

4. DOCUMENTACIÓN Y MODELADO DE PROCESOS
4.1 Conceptos: proceso, características de los procesos
4.2 Tipos de procesos: operativos, de apoyo y de gestión
4.3 Formas alternativas para documentar los procesos
4.4 Relación de los procesos con la ingeniería de software

4.5 Técnicas para la definición y diseño de procesos: diagramas de actividad UML y diagramas de flujos de proceso (flujogramas)
4.6 Notación y simbología utilizada para modelado de procesos
4.7 Herramientas CASE para el modelado de procesos

5. DISEÑO DE SOFTWARE
5.1 Fundamentos de diseño
5.2 Análisis y evaluación de la calidad del diseño
5.3 Notación de diseño software
5.4 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
6.2 Razones por las cuales es necesaria la arquitectura de software
6.3 Atributos de calidad de la arquitectura de software: performance, exactitud, seguridad, mantenibilidad, portabilidad
6.4 Estilos arquitectónicos: estilo módulo, estilo componente y conector, estilo de alocación (despliegue)
6.5 Diagrama de Contexto
6.6 Vistas arquitectónicas 4+1
6.7 Modelado arquitectónico del software
6.8 Arquitectura Física: N-Tier Architecture: Tipos (Cliente-Servidor, 3-Tier)
6.9 Beneficios Arquitectura Física: Escalabilidad, seguridad y tolerancia a fallas
6.10 Arquitectura Lógica: N-Layer Architecture: Tipos (3-layers, n-layers)
6.11 Beneficios: mantenibilidad, reutilización, distribución del trabajo, flexibilidad, robustez, entre otros
6.12 Capas lógicas de implementación (aplicación, negocios y servicio de datos)
6.13 Subsistemas

7. PATRONES DE DISEÑO
7.1 Descripción de los patrones
7.2 Clasificación
7.3 Creacionales, Estructurales, De Partición, De Comportamiento

8. CONSTRUCCIÓN DE SOFTWARE
8.1 Fundamentos de construcción

8.2 Gestión de construcción

9. GESTIÓN DE CALIDAD DEL PRODUCTO
9.1 Concepto de Calidad
9.2 Prevención versus Detección
9.3 Clientes de la calidad: interno, externo y oculto
9.4 Calidad del proyecto versus calidad del producto
9.5 Verificación versus Validación
9.6 Aseguramiento y Control de la Calidad del Software

10. PRUEBAS DEL SOFTWARE
10.1 Propósitos de la disciplina de pruebas
10.2 Actividades de la evaluación de la calidad del software: pruebas funcionales (caja blanca), pruebas estructurales (caja negra), análisis estático y análisis dinámico
10.3 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.
10.4 Implementación de Pruebas
- Pruebas de Unidad
- Pruebas de Integración
- Pruebas de Validación
- 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

10.5 Herramientas CASE para efectuar pruebas del software

11. IMPLANTACIÓN

11.1 Propósitos de la disciplina

11.2 Estrategias para la implantación del producto
11.3 Migración de datos
11.4 Capacitación de usuarios

12. MANTENIMIENTO DEL SOFTWARE
12.1 Propósitos de la disciplina
12.2 Elementos claves del mantenimiento: usuario, ambiente, ambiente operativo, ambiente organizacional, proceso de mantenimiento, producto de software, personal de mantenimiento.
12.3 Cambios del software: correctivos, adaptivos, perfectivos, preventivos
12.4 Procesos de mantenimiento
12.5 Técnicas para mantenimiento

13. ADMINISTRACIÓN DE LA CONFIGURACIÓN DEL SOFTWARE (ACS)
13.1. Definición de la Administración de la Configuración del Software
13.2. Actividades de la disciplina: identificación, almacenamiento, control del cambio y reporte del estado
13.3. Mejores prácticas de la disciplina.
13.4. Conceptos: metadata, línea base, roles y responsabilidades, ítem de configuración, árbol de versiones (versión y revisión), repositorio
13.5. El proceso de la Administración de Configuración de Software
13.6. Control de versiones, Control del Cambio, Reporte Estado, Auditoría de Configuración
13.7. Herramientas CASE para la ACS

14. METODOLOGÍAS DE DESARROLLO DEL SOFTWARE

14.1. Concepto de metodología
14.2. Modelo de procesos de desarrollo empleados en las metodologías (cascada, iterativo-incremental)
14.3. 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
14.4. Metodología estructurada: Proceso Unificado de Rational

14.5. Metodologías ágiles: Extreme Programming y SCRUM

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 (herramientas CASE). Los y las estudiantes serán responsables de instalar y poner en funcionamiento versiones de prueba así como mostrar ejemplos básicos de uso sobre las mismas.

Políticas de Evaluación

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

Notas y Aclaraciones

- 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
- 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.

PROGRAMA CURSO: IF7100

I Semestre, 2016

Cronograma del Curso

Semana 1 - 07 al 12 Marzo	**Actividades**
Inicio de clases	Entrega y Lectura carta del estudiante Introducción ingeniería de software
Semana 2 - 14 al 19 Marzo	**Actividades**
	GESTIÓN DE INGENIERIA DE SOFTWARE
Semana 3 – 21 al 26 Marzo	**Actividades**
	SEMANA SANTA
Semana 4 - 28 Marzo al 02 Abril	**Actividades**
	Exposición 1: Modelado de Procesos REQUERIMIENTOS DE SOFTWARE
Semana 5 – 04 al 09 Abril	**Actividades**
	REQUERIMIENTOS DE SOFTWARE **Exposición 2: Metodologías de desarrollo del Software**
Semana 6 – 11 al 16 de Abril	**Actividades**
	Primer Examen Parcial
Semana 7 – 18 al 23 de Abril	**Actividades**
	Exposición 3: Patrones de Diseño DISEÑO DE SOFTWARE
Semana 8 – 25 al 30 Abril	**Actividades**
	SEMANA UNIVERSITARIA ARQUITECTURA DEL SOFTWARE
Semana 9 – 2 al 07 Mayo	**Actividades**
	CONSTRUCCIÓN DE SOFTWARE
Semana 10 – 09 al 14 Mayo	**Actividades**
	GESTIÓN DE CALIDAD DEL PRODUCTO

PROGRAMA CURSO: IF7100

I Semestre, 2016

Semana 11 – 16 al 21 Mayo	Actividades
	PRUEBAS DEL SOFTWARE **Exposición 4: Herramientas de automatización de pruebas**
Semana 12 – 23 al 28 Mayo	**Actividades**
	IMPLANTACIÓN **Exposición 5: Product Management**
Semana 13 – 30 al 04 Junio	**Actividades**
	MANTENIMIENTO DEL SOFTWARE **Exposición 6: Internet of things**
Semana 14 – 06 al 11 Junio	**Actividades**
	ADMINISTRACIÓN DE LA CONFIGURACIÓN DEL SOFTWARE (ACS) **Exposición 7: In Memory Computing**
Semana 15 – 13 al 18 Junio	**Actividades**
	SEGUNDO EXAMEN PARCIAL
Semana 16 – 20 al 25 Junio	**Actividades**
	REVISION DE AVANCE DE PROYECTOS
Semana 17 – 27 al 02 Julio	**Actividades**
	PRESENTACION FINAL DE PROYECTOS
Semana 18 – 04 al 09 Julio	**Actividades**
	ENTREGA PROMEDIOS
Semana 19 – 11 al 16 Julio	**Actividades**
	AMPLIACION

Bibliografía del Curso

- 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. **Ingenieria 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 Contex Diagrams

- Ferm, Fredrik. T**he what, why, and how of a subsystem**. IBM DeveloperWorks, 25 Noviembre 2003
  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