Programa del Curso
I Semestre, 2018

Ingeniería de Software

Datos Generales

Sigla**: IF 7100**
Nombre del curso: **Ingeniería de 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**: Martes: 9:00 a 11:50 am y Viernes: 9:00 a 11:50**
Suficiencia:
Tutoría:

Datos del profesor

Nombre del Profesor: Ing. Rolando Vargas González. MGT
Correo electrónico: rolando.vargasgonzalez@ucr.ac.cr
Horario de Consulta: Miércoles de 8:00 a 12:00 y de 13:00 a 17:00, oficina 33

Descripción del Curso

Este curso proporciona al 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 estudiante un espacio de aplicación a través del desarrollo de un proyecto de software.

Objetivo General

El curso le permitirá al 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.

Objetivos Específicos

Al finalizar el curso el 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 (1/2 SEM)

2. REQUERIMIENTOS DE SOFTWARE (2 SEM)
   - 2.1. Fundamentos de requerimientos
   - 2.2. Procesos de requerimientos
   - 2.3. Obtención de requerimientos
   - 2.4. Análisis de requerimientos
   - 2.5. Especificación de requerimientos
   - 2.6. Validación de requerimientos

3. DOCUMENTACIÓN Y MODELADO DE PROCESOS(1/2 SEM)
   - 3.1. Conceptos: proceso, características de los procesos
   - 3.2. Tipos de procesos: operativos, de apoyo y de gestión
   - 3.3. Formas alternativas para documentar los procesos
   - 3.4. Relación de los procesos con la ingeniería de software
   - 3.5. Técnicas para la definición y diseño de procesos: diagramas de actividad UML y diagramas de flujos de proceso (flujogramas)

3.6. Notación y simbología utilizada para modelado de procesos
3.7. Herramientas CASE para el modelado de procesos

4. DISEÑO DE SOFTWARE (2 SEM)
   4.1. Fundamentos de diseño
   4.2. Análisis y evaluación de la calidad del diseño
   4.3. Notación de diseño software
   4.4. Estrategias y métodos de diseño

5. ARQUITECTURA DEL SOFTWARE (2 SEM)
   5.1. Definiciones: arquitectura versus diseño, arquitectura de software, elementos-relaciones y propiedades de la arquitectura del software
   5.2. Razones por las cuales es necesaria la arquitectura de software
   5.3. Atributos de calidad de la arquitectura de software:performance, exactitud, seguridad, mantenibilidad, portabilidad
   5.4. Estilos arquitectónicos: estilo módulo, estilo componente y conector, estilo de alocación (despliegue)
   5.5. Diagrama de Contexto
   5.6. Vistas arquitectónicas 4+1
   5.7. Modelado arquitectónico del software
   5.8. Arquitectura Física: N-Tier Architecture: Tipos (Cliente-Servidor, 3-Tier)
   5.9. Beneficios Arquitectura Física: Escalabilidad, seguridad y tolerancia a fallas
   5.10. Arquitectura Lógica: N-Layer Architecture: Tipos (3-layers, n-layers)
   5.11. Beneficios: mantenibilidad, reutilización, distribución del trabajo, flexibilidad, robustez, entre otros
   5.12. Capas lógicas de implementación (aplicación, negocios y servicio de datos)
   5.13. Subsistemas

6. PATRONES DE DISEÑO (1 SEM)
   6.1. Descripción de los patrones
   6.2. Clasificación
   6.3. Creacionales, Estructurales, De Partición, De Comportamiento

7. CONSTRUCCIÓN DE SOFTWARE (1 SEM)
   7.1. Fundamentos de construcción
   7.2. Gestión de construcción

8. GESTIÓN DE CALIDAD DEL PRODUCTO(1/2 SEM)
   8.1. Concepto de Calidad
   8.2. Prevención versus Detección
   8.3. Clientes de la calidad: interno, externo y oculto
   8.4. Calidad del proyecto versus calidad del producto
   8.5. Verificación versus Validación
   8.6. Aseguramiento y Control de la Calidad del Software

9. PRUEBAS DEL SOFTWARE(2 SEM)
   9.1. Propósitos de la disciplina de pruebas

9.2. Actividades de la evaluación de la calidad del software: pruebas funcionales (caja blanca), pruebas estructurales (caja negra), análisis estático y análisis dinámico

9.3. 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.

9.4. 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

9.5. Herramientas CASE para efectuar pruebas del software

10. IMPLANTACIÓN (1 SEM)
    10.1. Propósitos de la disciplina
    10.2. Estrategias para la implantación del producto
    10.3. Migración de datos
    10.4. Capacitación de usuarios

11. MANTENIMIENTO DEL SOFTWARE(1 SEM)
    11.1. Propósitos de la disciplina
    11.2. Elementos claves del mantenimiento: usuario, ambiente, ambiente operativo, ambiente organizacional, proceso de mantenimiento, producto de software, personal de mantenimiento.
    11.3. Cambios del software: correctivos, adaptivos, perfectivos, preventivos
    11.4. Procesos de mantenimiento
    11.5. Técnicas para mantenimiento

12. GESTIÓN DE INGENIERÍA DE SOFTWARE (1 SEM)
    12.1. Definición inicial y alcances
    12.2. Planificación proyectos
    12.3. Revisión y evaluación
    12.4. Cierre proyectos

13. ADMINISTRACIÓN DE LA CONFIGURACIÓN DEL SOFTWARE (ACS) (1 SEM)
    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(1 SEM)

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

- Los 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 estudiantes entregarán los elementos o artefactos definidos en el enunciado del proyecto.

- Además, los 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 estudiantes serán responsables de instalar y poner en funcionamiento versiones de prueba así como mostrar ejemplos básicos de uso sobre las mismas.

- Los estudiantes resolverán casos de estudio, en clase o extra-clase, en donde se plantean escenarios asociados con procesos de la ingeniería de software.

Políticas de Evaluación

Descripción	Porcentaje
Evaluaciones parciales escritas: - I Parcial: 25% - II Parcial: 25%	50%
Tareas - Tarea 1: 5% - Tarea 2: 5% - Tarea 3: 5%	15%
Proyecto	15%
Investigaciones - Investigación 1: 5% - Investigación 2: 5%	10%
Evaluaciones cortas y/o resúmenes semanales	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**
- No se reciben trabajos que carezcan de portada y no estén engrapados.

Cronograma del Curso

Semana 1	**Actividades**
12 al 18 marzo	Entrega y lectura carta del estudiante. Introducción ingeniería de software. Asignación de proyectos. Asignación de investigaciones.
Semana 2	**Actividades**
19 al 25 marzo	Documentación y modelado de procesos.
	Actividades
26 marzo al 1° abril	Semana Santa
Semana 3	**Actividades**
2 al 8 abril	Requerimientos de software.
Semana 4	**Actividades**
9 al 15 abril	**Requerimientos de software**. Presentación Investigación de Notaciones: LUM (UML), BPMN, DFD, CASE
Semana 5	**Actividades**
16 al 22 abril	Diseño de software. Tarea: Prueba de Unidad
Semana 6	**Actividades**
23 al 29 abril	Diseño de software. Semana Universitaria
Semana 7	**Actividades**
30 abril al 6 mayo	Arquitectura del software
Semana 8	**Actividades**
7 al 13 mayo	Arquitectura del software
Semana 9	**Actividades**
14 al 20 mayo	**Patrones de diseño.** Presentación Investigación de Modelos de proceso de Software: Codificar y Corregir, Lineal Secuencial, Construcción de prototipos, DRA, Procesos Evolutivos de SW (incremental y Espiral), Formales y Técnicas de 4G
Semana 10	**Actividades**
21 al 27 mayo	**Construcción de software**. Tarea: Modelos y Ciclos de vida del desarrollo de SW: Cascada o Clásico, Prototipos, Espiral, Etapas, Incremental o iterativo, Estructurado, Orientados a Objetos, RAD, Desarrollo Concurrente, Proceso Unificados de Desarrollo (RUP). **Primer examen parcial.**
Semana 11	**Actividades**
28 mayo al 3 junio	Gestión de calidad del producto.
Semana 12	**Actividades**
4 al 10 de junio	**Pruebas del software**. Tarea: Pruebas: Caja Blanca, Camino Básico, Complejidad Ciclomática, Estructura de Control, Caja Negra, Entornos Especializados Arquitecturas y Aplicaciones. **Gira a empresas: INTEL y GBM**

Semana 13	Actividades
11 al 17 de junio	Implantación. Tarea: Prueba de Integración descendente, integración ascendente, regresión, humo, validación, sistema, depuración.
Semana 14	**Actividades**
18 al 24 de junio	Mantenimiento del software
Semana 15	**Actividades**
25 junio al 1° julio	Gestión de ingeniería de software
Semana 16	**Actividades**
2 al 8 de julio	Administración de la configuración del software. Metodologías de desarrollo del software. **Segundo Examen parcial**
Semana 17	**Actividades**
9 al 15 de julio	Presentación de proyectos
Semana 18	**Actividades**
16 al 22 de julio	**Examen de Ampliación**

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

Evaluación de exposiciones de trabajos de investigación y presentación de proyectos

Trabajo Oral			Propuesta tiempo estimado 20', cantidad de oradores decisión grupal, recomendación máxima de dos
	Valor		Observaciones
Aspectos administrativos	20%		
Presentación de integrantes	2		
Agenda	2		
Hilo conductor	4		No se perciben los cambios entre secciones o etapas. Orden y apego a la agenda. Respeto de la sección
Estándar en la presentación	2		
Balance en la carga de cada filmina	2		
Utiliza gráficos o dibujos	2		Un gráfico dice mas que mil palabras, sin embargo no debe abusarse de la técnica
Respeto de la hora de inicio	2		
Respeto de la hora final	2		Al cumplirse el tiempo el orador posee dos minutos para terminar la presentación de todos los elementos
Referencias bibliográficas, autores o estándares	2		
Aspectos de comunicación	20%		
Dominio del tema	1		
Tono de voz	1		
Seguridad	1		
Velocidad	1		
Dirección a la audiencia	1		
Comunicación en función de la audiencia	1		
Ademánes	1		
Capta la atención de la audiencia	3		
Gana adeptos con sus comentarios	3		Influye positivamente a la audiencia
Asegura el entendimiento	3		
No hay redundancias	1		Explica una y otra vez la misma situación
La ayuda visual es un respaldo	1		No hay lectura total de la ayuda visual, o lectura para recordar
Comentarios alineados a la ayuda visual	1		
Mantiene el interés durante toda la presentación	1		Entretener-Informar-Persuadir
Aporte al temario del curso	50%		
Preguntas y respuestas	10%		Aplican para todos los miembros del grupo
Responde concretamente	5		No divaga
Velocidad de la respuesta	2		
Elimina dudas	3		No genera mas dudas con sus respuestas
" Si usted no puede decir lo que quiere en veinte minutos, debería irse y escribir un libro sobre eso" , Victor J. Zuñiga, MBA.			
Ing. Rolando Vargas González, MGT.			

Trabajo Escrito	Requerido	Opcional	Observaciones
Portada	X		
Introducción	X		
Resumen Ejecutivo	X		
Marco de Referencia		X	Si el trabajo se llevó a cabo en una Empresa incluir su contexto
Marco Conceptual	X		Conceptos técnicos utilizados
Desarrollo	X		
Conclusiones	X		
Bibliografía	X		
Distribución			
Contenido	70%		Aporte al temario del curso, calidad bibliográfica o referencial
Línea base	30%		Portada, Introducción, Resumen, Conceptos, Conclusiones, Bibliografía
Ing. Rolando Vargas González, MGT.			