782/18

Aprobado por resolución N°113 del CFI de fecha 04.07.2017

FACULTAD DE INGENIERIA

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Programa de Fundamentos de Ingeniería de Software (FIS)

1. NOMBRE DE LA UNIDAD CURRICULAR

Fundamentos de Ingeniería de Software

2. CRÉDITOS

8 créditos

3. OBJETIVOS DE LA UNIDAD CURRICULAR

Brindar un panorama de los aspectos técnicos y administrativos más relevantes de la Ingeniería de Software para estudiantes de ingeniería que no son del área informática.
Brindar conocimientos de algunas técnicas y herramientas utilizadas en la producción industrial de software.
Brindar conocimientos del proceso de desarrollo de software y de los subprocesos de las disciplinas centrales del desarrollo de software.
Este curso está pensado para carreras que no son de informática pero que requieren que el estudiante tenga conocimiento sobre la producción industrial de software. Por ende, se trabajará a nivel de conocimiento en las áreas más técnicas del software, y a nivel de conocimiento y adquisición de habilidades en las áreas que tratan aspectos de gestión y procesos de la producción de software (como gestión de proyectos de software, gestión de la calidad de software, gestión de la configuración, proceso de desarrollo de software, proceso de ingeniería de requisitos, entre otras).
Entonces, el objetivo central de esta unidad curricular es que el estudiante reconozca las diferencias entre la producción de software y otros tipos de producción en ingeniería y que sea capaz de entender las tareas de gestión y procesos de la producción de software. En conjunto con otras unidades curriculares que el estudiante pueda tomar y con la adquisición de experiencia profesional, podrá participar o co-participar en actividades de gestión de software en proyectos de desarrollo de software.

4. METODOLOGÍA DE ENSEÑANZA

2 clases por semana de 2 horas cada una, durante 15 semanas.
Cada clase tendrá contenido tanto teórico como práctico.
Los estudiantes deberán ir a las clases con los temas ya leídos del libro para poder participar en la discusión de los distintos tópicos.
Los trabajos prácticos serán distribuidos, algunos se realizarán durante el horario de clase mientras que para otros se solicitará se realicen de forma individual para su corrección en clase. Esta tarea es de carácter obligatorio.
Durante el laboratorio se espera que el estudiante realice los trabajos solicitados, y como resultado haga entrega de un informe y realice una presentación sobre el mismo. Esta tarea es de carácter obligatorio.
La dedicación del estudiante se distribuye de la siguiente manera:

- 60 horas de clases (30 clases de 2 horas distribuidas en 15 semanas)
- 60 horas de estudio individual para el seguimiento de las clases, realización de los prácticos, laboratorio y preparación de parciales.

5. TEMARIO

1. Introducción: qué es la ingeniería de software, motivación, surgimiento, importancia, principales características de la industria de software y su evolución.
   1.1. Definición de la ingeniería de software como disciplina.
   1.2. Ética en ingeniería de software
   1.3. La ingeniería de software como una profesión.
   1.4. Habilidades del ingeniero de software

2. Procesos de desarrollo de software: qué es un proceso de desarrollo, para qué sirven; se analizan diferentes modelos de procesos de desarrollo.
   2.1. Qué son y para qué se utilizan los procesos de desarrollo
   2.2. Procesos de desarrollo tradicionales
   2.3. Procesos de desarrollo ágiles

3. Ingeniería de Requisitos: qué son los requisitos de software, cuál es la importancia de su especificación. Técnicas para el relevamientos, especificación y análisis de requisitos.
   3.1. Requisitos funcionales y no funcionales
   3.2. El documento de requisitos de software
   3.3. Especificación de requisitos
   3.4. Proceso de ingeniería de requisitos
   3.5. Modelado con Casos de Uso

4. Arquitectura y diseño de software: qué es la arquitectura y diseño de software, por qué es importante su definición. Principales arquitecturas. Modelos de UML utilizados durante el proceso de desarrollo.

   4.1. Qué es la arquitectura de software y su importancia

   4.2. Vistas de la arquitectura de software: de casos de uso, lógica, de procesos, de desarrollo y física (se explicará y mostrará un ejemplo de cada vista)

   4.3. Arquitectura en capas y arquitectura cliente-servidor (se explicarán estos dos tipos de arquitectura en particular)

   4.4. Diagrama de clases y de interacción para diseño de bajo nivel

   Nota: Este punto se tratará a nivel de conocimientos ya que las habilidades necesarias para desarrollar arquitecturas de software o diseños de bajo nivel no son necesarios para este curso. El estudiante debe comprender la importancia de la arquitectura y el diseño de software así como los modelos más conocidos utilizados en el desarrollo. Para este tema no se utilizará el libro del curso.

5. Pruebas (testing) de software: qué es el testing de software, por qué es importante. Proceso de pruebas de software en el contexto del proceso de desarrollo. Técnicas de verificación, centrado en la especificación de casos de prueba.

   5.1. Definiciones básicas de pruebas de software

   5.2. Pruebas durante el desarrollo de software: unitarias, de integración, de sistema, aceptación y de liberación.

   5.3. Técnicas de verificación.

6. Evolución del software: proceso y principales características del ciclo de mantenimiento y evolución del software.

   6.1. Proceso de evolución

   6.2. Mantenimiento de software

7. Gestión de proyectos de software: conceptos de gestión de proyectos aplicados a proyectos de software; particularidades de esta industria.

   7.1. Nociones básicas y diferencias con la gestión proyectos de otras ramas de la ingeniería

   7.2. Gestión del riesgo

   7.3. Gestión de recursos humanos

   7.4. Trabajo en equipos

8. Planificación de proyectos de software: conceptos de planificación de proyectos aplicados a proyectos de software; particularidades de esta industria.

   8.1. Precio del software

   8.2. Desarrollo guiado por el plan de proyecto

   8.3. Calendario del proyecto

   8.4. Técnicas de estimación de software

9. Gestión de la calidad de software: principales conceptos de calidad aplicados a software; particularidades de esta industria.

   9.1. Calidad de software

   9.2. Estándares de software

   9.3. Revisiones e inspección

   9.4. Medición y medidas de software

10. Gestión de la configuración de software: principales conceptos de la gestión de la configuración, gestión de cambio, gestión de versiones y gestión de liberación de software.

    10.1. Gestión del cambio

    10.2. Control y gestión del versionado 10.3. Construcción del sistema

    10.4. Gestión de la liberación (release)

6. BIBLIOGRAFÍA

Tema	Básica	Complementaria
1. Introducción	(1) capítulo 1	(2) (3) (4) (5)
2. Procesos de Desarrollo de Software	(1) capítulo 2	
3. Ingeniería de Requisitos	(1) capítulo 4	
4. Arquitectura y Diseño de Software	(1) capítulo 6	
5. Pruebas (testing) de Software	(1) capítulo 8	
6. Evolución del Software	(1) capítulo 9	
7. Gestión de Proyectos de Software	(1) capítulo 22	
8. Planificación de Proyectos de Software	(1) capítulo 23 sin sección 23.4	
9. Gestión de la Calidad de Software	(1) capítulo 24	
10. Gestión de la Configuración de Software	(1) capítulo 25	

6.1 Básica

1. [Sommerville 10] Ian Sommerville; Software Engineering, 9th Edición, Addison-Wesley, 2010, ISBN 978-0137035151

6.2 Complementaria

2. [McConell 13] Steven McConnel and Leonard L. Tripp; Software Engineering Professional Practices, en Software Engineering Essentials, Volume II: The Supporting Processes, Richard Hall Thayer and Merlin Dorfman, Eds., 2013, cap. 11
3. [Parnas 11] David Parnas; Software Engineering - Missing in Action: A Personal Perspective, IEEE Computer, vol. 44, no. 10, pp. 54-58, 2011.
4. [Rivera-Ibarra 10] Rivera-Ibarra, J.G.; Rodriguez-Jacobo, J.; Serrano-Vargas, M.A., Competency Framework for Software Engineers, Software Engineering Education and Training (CSEE&T), 2010 23rd IEEE Conference on , vol., no., pp.33,40, 9-12 March 2010
5. [SECEPP] Joint Task Force on Software Engineering Ethics and Professional Practices, IEEE Computer Society and Association for Computing Machinery; Software Engineering Code of Ethics and Professional Practice;1999

7. CONOCIMIENTOS PREVIOS EXIGIDOS Y RECOMENDADOS

7.1 Conocimientos Previos Exigidos: N/A

7.2 Conocimientos Previos Recomendados:

1. Conocimientos básicos de programación
2. Conocimientos básicos de producción

Aprobado por resolución N°113 del CFI de fecha 04.07.2017

ANEXO A
Para todas las Carreras

A1) INSTITUTO

Instituto de Computación

A2) CRONOGRAMA TENTATIVO

Semana 1	Tema 1. Introducción (4 hs de clase)
Semana 2	Tema 2. Procesos de Desarrollo de Software (4 hs de clase)
Semana 3	Tema 3. Ingeniería de Requisitos (4 hs de clase)
Semana 4	Primer parcial (1 hs de clase) Tema 3. Ingeniería de Requisitos (3 hs de clase)
Semana 5	Tema 3. Ingeniería de Requisitos (1 hs de clase) Tema 4. Arquitectura y diseño de software (3 hs de clase)
Semana 6	Tema 4. Arquitectura y diseño de software (2 hs de clase) Tema 5. Pruebas de software (2 hs de clase)
Semana 7	Laboratorio – Presentaciones Obligatorio 1 (3 hs de clase) Tema 5. Pruebas de software (1 hs de clase)
Semana 8	Tema 5. Pruebas de software (3 hs de clase) Segundo parcial (1 hs de clase)
Semana 9	Tema 5. Pruebas de software (2 hs de clase) Tema 6. Evolución de software (2hs de clase)
Semana 10	Tema 7. Gestión de proyectos de software (4hs de clase)
Semana 11	Tema 7. Gestión de proyectos de software (4hs de clase)
Semana 12	Tema 8. Planificación de proyectos de software (3hs de clase) Tercer parcial (1 hs de clase)
Semana 13	Tema 8. Planificación de proyectos de software (2hs de clase) Tema 9. Gestión de calidad de software (2hs de clase)
Semana 14	Tema 10. Gestión de configuración de software (2hs de clase) Retrospectiva del curso (2 hs de clase)
Semana 15	Cuarto parcial (1 hs de clase) Laboratorio – Presentaciones Obligatorio 1 (3 hs de clase)

A3) MODALIDAD DEL CURSO Y PROCEDIMIENTO DE EVALUACIÓN

Modalidad de curso:
Clases teórico-prácticas obligatorias. Lecturas de libro y artículos recomendados. Elaboración de ejercicios prácticos y laboratorios requeridos por el docente.

Procedimiento de evaluación:

- Asistencia a clase obligatoria. Los estudiantes deben asistir al 80% de las clases para poder exonerar el curso. Es importante la asistencia a clase para fomentar la discusión de los temas planteados.
- Se tomarán pruebas parciales y continuas durante el curso. Para exonerar el estudiante debe obtener al menos un 40% en cada prueba individual y un 60% del puntaje obtenido en el total de las pruebas.
- Se realizarán dos trabajos obligatorios grupales que deberán ser presentados y defendidos en clase. El resultado de cada obligatorio es aprobado o reprobado. Para exonerar el curso ambos obligatorios deben ser aprobados.
- Se solicitará la elaboración de ejercicios prácticos que serán corregidos en clase.
- El resultado que se puede obtener es exoneración del curso o reprobación del mismo. En este último caso el estudiante tendrá la posibilidad de volver a cursarlo en una próxima edición del mismo.

A4) CALIDAD DE LIBRE

Esta unidad curricular no adhiere a la resolución de calidad de libre.

A5) CUPOS DE LA UNIDAD CURRICULAR

Sin cupo.

Aprobado por resolución Nº113 del CFI de fecha 04.07.2017

ANEXO B para la(s) carrera(s) Ingeniería en Producción

B1) ÁREA DE FORMACIÓN

Ingeniería en Computación.

B2) UNIDADES CURRICULARES PREVIAS

Para el Curso: El estudiante debe haber aprobado al menos 270 créditos en la carrera.

Para el Examen: No aplica.

Ref exped 060120-003053-17

ANEXO B para la carrera Ingeniería Eléctrica

B1) ÁREA DE FORMACIÓN

Informática

B2) UNIDADES CURRICULARES PREVIAS

El curso tiene como previas el exámenes de Programación 1 y 270 créditos aprobados.

Examen: No tiene

(Las unidades curriculares previas serán definidas por las carreras que tomen la unidad curricular en cuestión, teniendo en cuenta los conocimientos exigidos que figuran en el programa.)

APROB. RES. CONSEJO DE FAC. ING.
techu 3/7/18 Exp. 06 0120-003053-17