Guía docente del curso: INGENIERÍA DEL SOFTWARE II

1.- Introducción

El objetivo principal de este curso, Ingeniería del Software II, consiste en capacitar al estudiante para diseñar, implementar, probar y mantener aplicaciones de una entidad y complejidad manifiestamente superiores a las desarrolladas en el curso previo Ingeniería del Software que está disponible en https://ocw.ehu.eus/course/view.php?id=220, en la plataforma OCW de la UPV/EHU.

La metodología que vertebra los distintos elementos utilizados en el curso es el Proceso Unificado de Desarrollo del Software (UP), considerado hoy en día un estándar, como el cuerpo de conocimiento esencial que debe tener un Ingeniero del Software. En este proceso se identifican diferentes actividades: captura de requisitos, diseño, implementación, pruebas y mantenimiento. Las tres primeras constituyen el núcleo principal del curso Ingeniería del Software. En Ingeniería del Software II se comienza trabajando las dos últimas (pruebas y mantenimiento), y se continúa después con aspectos más avanzados de las tres primeras (captura de requisitos, diseño e implementación). El objetivo final consiste en que los estudiantes sean capaces de desarrollar, a partir del prototipo desarrollado en el curso previo, un prototipo más avanzado, desde el punto de vista de las tecnologías utilizadas y desde el punto de vista del diseño, de las pruebas y del mantenimiento del software.

2.- Personas destinatarias y prerrequisitos

Este curso está pensado para personas con gran interés en el desarrollo de software, que sepan programar en un lenguaje de programación orientado a objetos como Java, y analizar y diseñar software utilizando un lenguaje de modelado como UML. Si no se dispone de esos conocimientos de análisis, diseño y programación, se recomienda que previamente se curse https://ocw.ehu.eus/course/view.php?id=220

3.- Competencias

Una vez superado el curso junto con el desarrollo del proyecto, el estudiante deberá ser capaz de:

- **CA1:** Gestionar de manera sistemática las pruebas de un sistema software.
- **CA2:** Conocer las técnicas de mantenimiento software existentes y saber aplicar las más significativas en cada caso.
- **CA3:** Diseñar un sistema software de calidad aplicando los patrones de diseño más extendidos
- **CA4:** Implementar un sistema software utilizando alguno de los frameworks existentes

4.- Objetivos

Para alcanzar las competencias anteriores, se han definido los siguientes objetivos globales de aprendizaje del curso, cada uno de ellos acompañado por otros objetivos más específicos:

1.**OA1:** Conocer la existencia de metodologías de pruebas para afrontar la enorme complejidad que puede alcanzar su gestión en un proyecto informático. Este objetivo tiene relación con la competencia **CA1**.

- Estudiar las diferentes metodologías existentes para la gestión de las pruebas y su integración en los procesos ágiles (TDD: Test Driven Development).
- Conocer los diferentes niveles de pruebas existentes (Pruebas unitarias, Pruebas de integración, Pruebas de sistema y Pruebas de aceptación), y las técnicas a aplicar en cada nivel.
- Aplicar las técnicas de Pruebas existentes a un proyecto software (caja negra, caja blanca, partición de equivalencia, valores límite)

2.**OA2:** Experimentar la necesidad de aplicar una metodología para el mantenimiento de un proyecto informático. En el ciclo de vida de un proyecto software el 80% constituye la fase de mantenimiento. Este objetivo tiene relación con la competencia **CA2**.

- Conocer los tipos de mantenimiento existente (correctivo, preventivo, predictivo y proactivo).
- Manejar los conceptos y principales diagramas de modelado utilizados en el mantenimiento del software.
- Dominar las técnicas de mantenimiento existentes para cada tipo.
- Manejar las herramientas existentes para facilitar el mantenimiento del software.

3. **OA3:** Entender la necesidad de utilizar patrones de diseño en el desarrollo software. Un patrón es una solución general a un problema en particular. Cuanto más complejas son las aplicaciones, el uso de patrones facilitará el desarrollo de software de calidad. Aunque este objetivo de aprendizaje está relacionado con la competencia **CA3**, también nos ha servido para introducir patrones de pruebas (relacionados con la **CA1**) y patrones de mantenimiento (relacionados con la **CA2**).
   - Conocer los tipos diferentes tipos de patrones de diseño.
   - Utilizar con soltura los patrones de diseño en los diferentes escenarios del proyecto.
   - Utilizar con soltura y precisión los Diagramas de Clases para modelar los patrones de diseño.
   - Entender la necesidad de utilizar patrones en todas las fases de desarrollo software.

OA4: Saber abordar las dificultades de una correcta interpretación de la especificación de diseño de un sistema informático y de su traducción en un desarrollo correcto. De este modo se efectuará la transición entre la competencia **CA3** y la **CA4** y se desarrollará plenamente esta última.

- Manejar con soltura el uso del entorno de programación (ECLIPSE) para el desarrollo, pruebas y mantenimiento de un sistema software.
- Utilizar una infraestructura gráfica (JSF) para dotar a las aplicaciones de una interfaz de usuario orientada a las necesidades del usuario.
- Utilizar un sistema de mapeo de bases de datos (Hibernate) para implementar la capa de persistencia de una aplicación.

5.- Descripción del curso

El equipo docente del curso está formado por los profesores Jon Iturrioz y Alfredo Goñi, que han impartido en el Grado en Ingeniería Informática de la UPV/EHU las dos asignaturas básicas de Ingeniería del Software durante los últimos años.
Para completar este curso, el estudiante deberá dedicar unas 6-7 horas por semana durante un total de 12 semanas.

6.- Programa del curso

El programa está formado por los siguientes 4 temas, donde cada tema está dedicado a trabajar cada una de las 4 competencias anteriores: **CA1**, **CA2**, **CA3** y **CA4.**

TEMA 1: Verificación del software
- 1.1- Diseño de las pruebas
- 1.2.- Implementación de las pruebas

TEMA 2: Mantenimiento del software
- 2.1.- Aspectos generales
- 2.2.- Refactorización

TEMA 3: Diseño del software
- 3.1.- Principios SOLID
- 3.2.- Patrones de diseño

TEMA 4: Implementación: Frameworks
- 4.1.- Introducción
- 4.2.- Java Server Faces (JSF)
- 4.3.- Hibernate

7.- Metodología para el estudio

El método diseñado para este curso va a tratar, por un lado, de que los estudiantes adquieran unos determinados conocimientos teóricos y prácticos a lo largo de las 12 semanas de las que consta el curso. Y, por otro lado, a lo largo de todo el curso, los estudiantes irán desarrollando un proyecto final donde consoliden la adquisición de las competencias anteriormente descritas. Dicho desarrollo se realizará sobre un prototipo desarrollado en el curso previo (https://ocw.ehu.eus/course/view.php?id=220), el cual se proporcionará al inicio del curso.

El material de estudio que se proporciona para cada uno de los temas del programa del curso consiste en:

- Un conjunto de diapositivas que explican los conceptos teóricos de la asignatura.
- Un test de autoevaluación que pretende que el estudiante compruebe si ha entendido dichos conceptos teóricos.
- Uno o más laboratorios guiados (tutoriales o actividades prácticas) que enseñan paso a paso cómo aplicar la teoría explicada utilizando tecnología apropiada.
- Una serie de ejercicios prácticos que los estudiantes deben realizar de manera autónoma. Para algunos de ellos se proporcionan las soluciones, para que puedan comprobar si han adquirido las competencias.

Además, se propone en paralelo un proyecto final de desarrollo de software de más envergadura, que consiste en proporcionar una nueva implementación del prototipo desarrollado en el curso previo (Ingeniería del Software), utilizando los conceptos de pruebas y diseño del software, así como la tecnología explicada en el tema de implementación, donde se tratan los Frameworks JSF e Hibernate.

8.- Cronograma

Las actividades propuestas tendrán las siguientes duraciones estimadas. Nótese que las actividades planificadas para el desarrollo del proyecto se realizan en paralelo a lo largo de las 12 semanas de curso.

TEMA	ACTIVIDADES	Semana	Horas
TEMA 1: VERIFICACIÓN DEL SOFTWARE	Lectura diapositivas	1	3
	Test autoevaluación	1	0,5
	Realización laboratorio guiado JUnit	1	1,5
	Resolución ejercicios pruebas	2	1,5
TEMA 2: MANTENIMIENTO DEL SOFTWARE	Lectura diapositivas	2	3
	Test autoevaluación	2	0,5
	Realización laboratorio Refactorización	3	1,5
	Realización laboratorio guiado JDeodorant	3	1,5
	Resolución ejercicios mantenimiento	4	3,5
TEMA 3: DISEÑO DEL SOFTWARE	Lectura diapositivas Principios SOLID	4	3
	Test autoevaluación	4	0,5
	Resolución ejercicios Principios SOLID	5	3
	Lectura diapositivas Patrones de Diseño	5	3
	Test autoevaluación	5	0,5
	Realización laboratorio Patrones de Diseño	6	1,5
	Resolución ejercicios Patrones de Diseño	7	3
TEMA 4: IMPLEMENTACIÓN FRAMEWORKS	Lectura diapositivas Framework JSF	8	2
	Test autoevaluación	8	0,5
	Realización laboratorio JSF	8	4
	Resolución ejercicios JSF	9	3
	Lectura diapositivas Framework Hibernate	10	2
	Test autoevaluación	10	0,5
	Realización laboratorio guiado Hibernate	11	4
	Resolución ejercicios Hibernate	11	3
DESARROLLO DEL PROYECTO	Realización de las pruebas del proyecto	1-2-3	7
	Rediseño del proyecto	6-7	7
	Realización proyecto aplicando JSF	9-10	7
	Realización proyecto aplicando Hibernate	12	6
	Test final autoevaluación	12	0,5

9.- Información adicional

Para realizar los ejercicios y actividades de este curso se necesitará que los estudiantes dispongan de un equipo con una instalación apropiada del entorno de desarrollo Eclipse con algunos plugins específicos. En el documento "Instalación del software necesario" se explica cómo realizarlo.