FICHA IDENTIFICATIVA

Datos de la Asignatura

Código	34886
Nombre	Ingeniería del Software
Ciclo	Grado
Créditos ECTS	6.0
Curso académico	2019 - 2020

Titulación(es)

Titulación	Centro	Curso	Periodo
1403 - Grado de Ingeniería Telemática	Escuela Técnica Superior de Ingeniería	3	Primer cuatrimestre

Materias

Titulación	Materia	Caracter
1403 - Grado de Ingeniería Telemática	11 - Ingeniería del Software	Obligatoria

Coordinación

Nombre	Departamento
RUEDA PASCUAL, SILVIA	240 - Informática

RESUMEN

La asignatura “Ingeniería del Software” es una asignatura obligatoriaque forma parte de la materia Ingeniería del Software y Gestión de Proyectosdel Grado en Ingeniería Telemática. Tiene asignada una dedicación de 6 ECTS que se imparten durante el 1er cuatrimestre del 3er curso.

En esta asignatura, se trata de aprender a desarrollar proyectos software siguiendo un proceso sistemático y apoyándose en herramientas que permiten mejorar la calidad del software en entornos de producción.

Se introducirá al alumnado en el conocimiento y manejo de diferentes metodologías de desarrollo de sistemas de información.

Se tratará de conseguir un conocimiento suficiente del proceso de software, de forma que el alumnado sea capaz de, usando como método el Proceso Unificado de Desarrollo, capturar los requisitos, analizar, diseñar, implementar, probar e implantar proyectos software de manera concreta y con precisión.

En lo que se refiere a la parte práctica, en esta asignatura trataremos de que el alumnado utilizando el lenguaje de modelado UML y el lenguaje de programación Java, sea capaz de poner en práctica los conocimientos vistos en la parte teórica.

El objetivo principal de la asignatura es introducir al alumnado en el desarrollo de proyectos software desde el análisis de requisitos hasta implantación y verificación del producto por parte del cliente, de forma que sea capaz de:

- Conocer el origen y significado del término “Ingeniería del Software”, su evolución histórica y los desafíos actuales (con atención al contexto sociocultural de su desarrollo), y ser consciente de la responsabilidad ética y profesional de un Ingeniero de Software.
- Tomar conciencia de la importancia de realizar siempre un análisis y diseño previos del problema, como pasos anteriores a la implementación en un lenguaje de programación.
- Ser consciente de la necesidad del modelado y la abstracción en el desarrollo de software.
- Conocer el concepto de método de desarrollo de software y sus principales clasificaciones.
- Distinguir los conceptos de diagrama y modelo.
- Conocer los principales diagramas UML: casos de uso, clases, paquetes, objetos, interacción (secuencia y comunicación), estados y actividades, y ser capaz de aplicarlos al modelado de un proyecto de tamaño medio.
- Dada una aplicación de tamaño medio, ser capaz de abordar el análisis de requisitos centrado en casos de uso, el modelado del dominio o conceptual, el análisis de colaboraciones entre objetos con una apropiada asignación de responsabilidades y teniendo en cuenta detalles tecnológicos.
- Conocer técnicas de diseño y aplicarlas en el marco de un proceso iterativo.
- Elegir la opción de diseño conceptual de datos más adecuada entre varias alternativas posibles, justificando y argumentando la decisión tomada.
- Conocer y aplicar los patrones de diseño básicos para construcción de software y valorar su papel como forma de reutilización de la experiencia.
- Utilizar una herramienta software que permita la creación de los diferentes diagramas UML.

CONOCIMIENTOS PREVIOS

Relación con otras asignaturas de la misma titulación

No se han especificado restricciones de matrícula con otras asignaturas del plan de estudios.

Otros tipos de requisitos

Es requisito imprescindible:

Haber aprobado las asignaturas de primer y segundo curso Informática (34877) y Ampliación de Informática (34878)

Estar matriculado/a o haber aprobado la asignatura Programación (34888)

COMPETENCIAS

1403 - Grado de Ingeniería Telemática

- R1 - Capacidad para aprender de manera autónoma nuevos conocimientos y técnicas adecuados para la concepción, el desarrollo o la explotación de sistemas y servicios de telecomunicación.
- G4 - Capacidad de resolver problemas con iniciativa, toma de decisiones, creatividad, y de comunicar y transmitir conocimientos, habilidades y destrezas, comprendiendo la responsabilidad ética y profesional de la actividad del Ingeniero Técnico de Telecomunicación.
- G9 - Capacidad de trabajar en un grupo multidisciplinar y en un entorno multilingüe y de comunicar, tanto por escrito como de forma oral, conocimientos, procedimientos, resultados e ideas relacionadas con las telecomunicaciones y la electrónica.
- R7 - Conocimiento y utilización de los fundamentos de la programación en redes, sistemas y servicios de telecomunicación.
- E3 - Capacidad de construir, explotar y gestionar servicios telemáticos utilizando herramientas analíticas de planificación, de dimensionado y de análisis.
- E4 - Capacidad de describir, programar, validar y optimizar protocolos e interfaces de comunicación en los diferentes niveles de una arquitectura de redes.

RESULTADOS DE APRENDIZAJE

Esta asignatura permite obtener los siguientes resultados de aprendizaje:

- Aplicar metodologías para el desarrollo, implantación y mantenimiento de sistemas de información. **Competencias G4, G9, R1, R7, E3 y E4**
- Planificar y ejecutar correctamente procesos de desarrollo de software iterativos. **Competencias G4, G9, R1, R7, E3 y E4**
- Saber aplicar patrones de diseño software en cada situación en función de las necesidades del proyecto de desarrollo software. **Competencias G4, R1, E3 y E4**
- Definir pruebas de validación y verificación de requisitos. **Competencias G4, R1, E3 y E4**
- Obtener requisitos de usuario y del software. **Competencias G4, G9, R1, E3 y E4**
- Conocer los conceptos básicos, los procesos de desarrollo, los métodos y las herramientas de la ingeniería de software. **Competencia E3**
- Conocer y comprender los paradigmas actuales de la ingeniería del software dirigidos al desarrollo distribuido de software, la ingeniería del software libre y la ingeniería Web. **Competencias G4, R1 y E4**
- Conocer los diferentes modelos de arquitectura, las tecnologías que pueden integrarse y las soluciones comerciales para formar una solución distribuida particular. **Competencias E3 y E4**

Como complemento a los resultados anteriores, esta asignatura también permite adquirir las siguientes destrezas y habilidades sociales:

- Analizar un problema de desarrollo de software y deducir su naturaleza de manera concreta y con precisión. **Competencias G4, G9, R1 y E3**
- Diseñar una estructura de módulos, utilizando patrones de diseño, para solucionar un problema y evaluar otras alternativas. **Competencias G4, G9, R1 y E3**
- Implementar un módulo que se ejecute correctamente y de manera eficiente. **Competencias R7, E3 y E4**
- Probar aplicaciones de manera sistemática definiendo casos de prueba exhaustivos. **Competencia E4**

- Trabajar en un pequeño equipo, colaborando en todos los aspectos del desarrollo de software intercambiando ideas de manera constructiva y organizada. **Competencias G4, G9, R1, R7, E3 y E4**

DESCRIPCIÓN DE CONTENIDOS

1. Introducción al proceso de Desarrollo de software UML

Destrezas a adquirir:
- Comprender qué es la Ingeniería del Software y su necesidad
- Conocer y comprender los conceptos fundamentales que conforman la terminología básica de la ingeniería del software
- Comprender las relaciones entre los conceptos de proceso software, ciclo de vida del software y metodología software
- Conocer las características y explicar las ventajas y desventajas de diferentes modelos de proceso del software
- Conocer los principales tipos de metodologías software
- Conocer las características básicas de un proceso de desarrollo del Software generalista
- Entender en qué consiste el modelado de software y qué beneficios aporta
- Reconocer UML como lenguaje estándar en la construcción de software

Contenidos:
1.1 Visión General de la Ingeniería del Software
1.2 Conceptos básicos de la Ingeniería del Software
1.3 Modelos de Proceso del Software
1.4 Modelado de Software
1.5 El Lenguaje Unificado de Modelado UML 2.0
1.4.1 Marco Conceptual UML
1.4.2 Vistas UML
1.6 Proceso de Desarrollo Software OO
1.6.1 Fases
1.6.2 Actividades y Artefactos

Laboratorio:
Todas las sesiones.

2. Fase de Planificación

Destrezas a adquirir:
- Comprender la importancia de obtener y gestionar los requisitos y su influencia en el éxito de un proyecto
- Entender qué son los requisitos y la complejidad de la extracción de requisitos
- Conocer las actividades de requisitos
- Conocer los distintos tipos de requisitos y ser capaz de diferenciarlos
- Conocer las diferentes técnicas existentes para capturar los requisitos de un sistema
- Conocer en qué consiste el Documento de Requisitos
- Conocer el estándar IEEE/ANSI 830-199 para SRS
- Elaborar un documento SRS para sistemas de tamaño medio
- Conocer los diferentes elementos y diagramas que UML proporcionar para representar Casos de Uso
- Representar Requisitos Funcionales mediante Casos de Uso
- Realizar especificaciones detalladas de Casos de Uso

Contenidos:
2.1. Requisitos
2.1.1 Definición y características
2.1.2 Requisitos Funcionales vs. No Funcionales
2.1.3 Documento de Requisitos del Software
2.1.4 Ejercicios sobre requisitos
2.2 Prototipo
2.3 Casos de Uso
2.3.1 Introducción.
2.3.2 Actores
2.3.3 Especificación de Casos de Uso
2.3.4 Relaciones: generalización, extensión, inclusión
2.3.5 Diagramas de Casos de Uso
2.3.6 Modelado de Casos de Uso
2.3.7 Ejercicios sobre Casos de Uso.

Laboratorio:
Sesión 1: Desarrollo del Prototipo del GUI
Sesión 2: Trabajando con Diagramas de Casos de Uso

3. Análisis

Destrezas a adquirir:
- Conocer los pasos a seguir para completar la etapa de análisis en el primer ciclo de desarrollo y los artefactos a crear
- Ser capaces de elaborar el Diccionario de Datos
- Ser capaces de abstraer los Conceptos relevantes para elaborar un Modelo Conceptual
- Representar mediante Diagramas de Clases el Modelo Conceptual de un sistema
- Identificar los eventos del sistema en las descripciones de los Casos de Uso para determinar las

Operaciones del Sistema
- Elaborar los Diagramas de Secuencia del Sistema de los diferentes Casos de Uso a partir de su especificación expandida
- Desarrollar los Contratos de las Operaciones del Sistema

Contenidos:
Parte I:
3.1 Introducción
3.2 Diagrama de clases
3.2.1 Clasificadores
3.2.2 Clases
3.2.3 Interfaces
3.2.4 Relaciones: dependencia, generalización, asociación, realización
3.3 Modelo Conceptual
3.4 Ejercicios de diagramas de clases y objetos

Parte II:
3.5 Interacciones
3.6 Diagramas de Secuencia
3.5.1 Elementos
3.5.2 Modelado Diagramas de Secuencia
3.5.3 Aplicación en el ciclo de vida
3.7 Diagramas de Secuencia Generales del Sistema
3.8 Contratos
3.9 Ejercicios de diagramas de secuencia y contratos

Laboratorio:
Sesión 3: Trabajando con Diagramas de Clases
Sesión 4: Trabajando con Diagramas de Secuencia
Sesión 5: Trabajando el Diseño del Ciclo 1
Sesión 7: Trabajando el Análisis y el Diseño del Ciclo 2

4. Diseño

Destrezas a adquirir:
- Conocer los pasos a seguir para completar la etapa de diseño en el primer ciclo de desarrollo y los artefactos a crear
- Conocer el concepto de responsabilidad
- Conocer y saber cómo aplicar una serie de patrones a la hora de decidir la asignación de responsabilidades a clases
- Ser capaces de elaborar los Diagramas de Interacción de cada una de las operaciones del sistema a partir de sus Contratos
- Elaborar el Diagrama de Clases de Diseño a partir del Modelo Conceptual

Contenidos:

4.1 Diseño del Sistema
4.1.1 Responsabilidades
4.1.2 Diagramas de Secuencia de Diseño
4.1.3 Diagramas de Clases de Diseño
4.1.4 Patrones para la asignación de responsabilidades.
4.2 Ejercicios

Laboratorio:
Sesión 5: Trabajando el Diseño del Ciclo 1
Sesión 7: Trabajando el Análisis y el Diseño del Ciclo 2

5. Implementación

Destrezas a adquirir:
- Conocer las decisiones previas a tomar antes de implementar
- Conocer los tipos de transformación espacio del modelo-espacio del código
- Ser capaces de transformar los artefactos del diseño en código
- Ser capaces de determinar la necesidad de modificar los modelos para introducir optimizaciones

Contenidos:
5.1 Decisiones previas
5.2 Tipos de transformación
5.2.1 Transformaciones del modelo
5.2.2 Transformaciones del código
5.2.3 Transformaciones del modelo al código: Ingeniería directa
5.2.4 Transformaciones del código al modelo: Ingeniería inversa
5.3 Ingeniería directa
5.3.1 Mapeo de Clases
5.3.2 Mapeo de Relaciones
5.3.3 Mapeo de Herencia
5.3.4 Creación de Métodos
5.3.5 Mapeo de Contratos
5.4 Ejercicios de Implementación

Laboratorio:
Sesión 6: Trabajando la Implementación del Ciclo 1
Sesión 8: Trabajando la Implementación del Ciclo 2

6. Arquitectura del Sistema

Destrezas a adquirir:
- Conocer los pasos a seguir para completar la etapa de diseño en el segundo ciclo de desarrollo y los artefactos a crear
- Conocer los conceptos de capas, paquetes y particiones y como utilizarlos en la organización de la arquitectura del sistema.

- Representar paquetes y sus relaciones en Diagramas de Paquetes
- Decidir la arquitectura a emplear y representarla mediante Diagramas de Paquetes
- Conocer y saber aplicar otros patrones

Contenidos:
6.1 Arquitectura multicapa y UML
6.2 Patrones de conexión entre paquetes
6.3 Ejercicios

Laboratorio:
Sesión 7: Trabajando el Análisis y el Diseño del Ciclo 2

7. Pruebas

Destrezas a adquirir:
- Conocer y diferenciar los aspectos fundamentales relacionados con las pruebas de software
- Comprender la necesidad de las pruebas como parte esencial del desarrollo de un sistema software
- Saber diferenciar los distintos niveles de prueba en función del objetivo de la misma
- Conocer las diferentes técnicas de prueba del software

Contenidos:
7.1 Conceptos Básicos: Errores, Defectos, Fallos, Casos de Prueba
7.2 Verificación y Validación
7.2.1 Inspecciones del software
7.2.2 Pruebas del software
7.2.3 Depuración
7.3 Niveles de Prueba del Software
7.3.1 Pruebas Unitarias
7.3.2 Pruebas de Integración
7.3.3 Pruebas de Sistema
7.3.4 Pruebas de Validación
7.4 Técnicas de Prueba del Software
7.4.1 Pruebas de Caja Negra
7.4.2 Pruebas de Caja Blanca
7.5 Plan de Pruebas

VOLUMEN DE TRABAJO

ACTIVIDAD	Horas	% Presencial
Clases de teoría	30,00	100
Prácticas en laboratorio	20,00	100
Prácticas en aula	10,00	100
Elaboración de trabajos en grupo	4,00	0
Estudio y trabajo autónomo	4,00	0
Preparación de clases de teoría	23,00	0
Preparación de clases prácticas y de problemas	50,00	0
Resolución de casos prácticos	9,00	0
TOTAL	**150,00**	

METODOLOGÍA DOCENTE

Las actividades formativas se desarrollarán de acuerdo con la siguiente distribución:

- **Actividades teóricas**.

En las clases teóricas se desarrollarán los temas proporcionando una visión global e integradora, analizando con mayor detalle los aspectos clave y de mayor complejidad, fomentando, en todo momento, la participación del estudiantado. Para conseguirlo se hará uso de la clase invertida.

- **Actividades prácticas**.

Complementan las actividades teóricas con el objetivo de aplicar los conceptos básicos y ampliarlos con el conocimiento y la experiencia que vayan adquiriendo durante la realización de los trabajos propuestos. Comprendenlos siguientes tipos de actividades:

- Clases de problemas y cuestiones en aula
- Sesiones de discusión y resolución de problemas y ejercicios previamente trabajados por el estudiantado
- Prácticas y seminarios en aula informática
- Trabajos en grupo para planificación y desarrollo de proyectos software y generación de dinámicas de grupo.
- Tutorías programadas (individualizadas)

- **Trabajo personal**.

Preparación de clases y exámenes (estudio). El estudiantado tendrá que trabajar previamente los materiales para la clase invertida en casa. Esta tarea se realizará de manera individual e intenta potenciar el trabajo autónomo.

- **Trabajo en pequeños grupos**.

Realización, por parte de pequeños grupos de estudiantado (5-6) de trabajos, cuestiones, problemas fuera del aula. Esta tarea complementa el trabajo individual y las actividades prácticas y fomenta la capacidad de integración en grupos de trabajo. Comprende la realización de las siguientes actividades:

- Trabajo en grupo práctico de extracción de requisitos con diferentes técnicas y elaboración de un plan de pruebas.
- Presentación del trabajo en grupo (en castellano).
- Trabajo en grupo para planificación y desarrollo de un proyecto software y generación de dinámicas de grupo, cuya documentación deberá presentarse por escrito.
- Presentación del proyecto software.
- Tutorías programadas (en grupos).

Se utilizará la plataforma de e-learning (Aula Virtual) de la Universitat de València como soporte de comunicación con el estudiantado. A través de ella se tendrá acceso al material didáctico utilizado en clase, así como los problemas y ejercicios a resolver.

EVALUACIÓN

Los conocimientos adquiridos por el estudiantado se podrán evaluar de las dos formas siguientes:

- Sistema de evaluación continua
- Sistema de evaluación única.

Sistema de Evaluación Continua

Este es el método que se recomendará al alumnado. Mediante este sistema se evaluará de forma regular la participación del alumnado en el aprovechamiento de actividades formativas y la participación del alumnado en el proceso de aprendizaje.

Se valorarán los siguientes aspectos:

- **Sesiones Teórico-Prácticas:**se valorará la implicación, teniendo en cuenta la asistencia regular a las actividades presenciales previstas, la entrega de los ejercicios propuestos y la participación en la resolución de los mismos (*N_Teoría*). **Competencias G4, G9 y R1**
- **Sesiones de Laboratorio:**se valorará la implicación, teniendo en cuenta la asistencia regular a las actividades presenciales previstas, la entrega de los ejercicios propuestos y de los cuestionarios de evaluación (*N_Laboratorio*).**CompetenciasG4, G9, R1, R7, E3 y E4**
- **Trabajo:**se valorará tanto la documentación como la defensa del trabajo en pequeños grupos (*N_Trabajo*).**Competencias G4, G9 y R1**
- **Proyecto:**se valorará tanto la calidad de la solución como la documentación y defensa oral del mismo (*N_Proyecto*).**Competencias G4, G9, R1, R7, E3 y E4**
- **Pruebas objetivas individuales:**consistente en unos exámenes parciales y en un examen final (*N_Exámenes*).**Competencias G4, R1 y E3**

Para poder aplicar este tipo de evaluación será necesario un índice de asistencia a las sesiones de laboratorio superior al 75%.

La nota final se obtendrá de aplicar la siguiente fórmula:

Nota = 20% N_Continua + 60% N_TrabProy + 20% N_Exámenes

N_Continua = 50% N_Teoría + 50%N_Laboratorio

N_TrabProy = 10% N_Trabajo + 90% N_Proyecto

N_Exámenes = 20% N_Parciales + 80% N_ExFinal

Para poder presentarse al examen final, las notas de *N_Continua*y*N_Proyecto* tendrán que ser superiores o iguales a 4,5.

Sistema de Evaluación Única

Este método se aplicará a cualquier alumno/a que, por un motivo razonado y admitido por el profesorado, no pueda asistir con regularidad a las clases o bien no haya superado la evaluación continua en primera convocatoria.

Se valorarán los mismos aspectos que con la evaluación continua, pero cambian los pesos de cada una de las partes, de tal manera que la nota final se obtendrá de aplicar la siguiente fórmula:

Nota = 10% N_Continua + 60% N_TrabProy + 30% N_Exámenes

N_Contínua =50% N_Teoría + 50% N_Laboratorio

N_TrabProy = 10% N_Trabajo + 90% N_Proyecto

N_Exámenes = 20% N_Parciales + 80% N_ExFinal

Para poder presentarse al examen final, la nota de *N_ Proyecto* tendrá que ser superior o igual a 4,5.

La asistencia a las sesiones de laboratorio seguirá siendo del 75% mínimo.

En ambos métodos, sólo se considerarán los trabajos entregados en la fecha estipulada por el profesorado. Esto incluye los ejercicios propuestos en clase (teoría y prácticas), los ejercicios y cuestionarios de laboratorio, el trabajo y el proyecto software.

Las notas de las actividades en grupo no serán las mismas necesariamente para todos los miembros del grupo, pudiendo variar en función de la implicación de cada alumno/a.

Para poder promediar las notas de las diferentes categorías se les pedirá **una nota mínima de 4,5**puntos (sobre 10) en cada una de ellas (*N_Teoría, N_Laboratorio, N_ Trabajo, N_Proyecto, N_Exámenes y N_ExFinal)*.

La nota de N_Continua no es recuperable, manteniéndose para la 2ª convocatoria.

En cualquier caso, la evaluación de la asignatura se hará de acuerdo con el Reglamento de evaluación

y calificación de la Universitat de València para los títulos de grado y master aprobado por Consejo de Gobierno de 30 de mayo de 2017 (ACGUV 108/2017).

De acuerdo con el reglamento de la Universitat de València, la realización de actuaciones fraudulentas en una prueba o parte de ella dará lugar a la calificación de un cero en la misma, con independencia del procedimiento disciplinario que se pueda abrir y de la sanción que sea procedente de acuerdo a la normativa vigente.

Para poder solicitar adelanto de convocatoria, el estudiantado deberá haber cursado previamente la asignatura. De esta forma se trata de conciliar el derecho del estudiantado a dicho adelanto con la metodología docente y el mecanismo de evaluación de la asignatura. Adicionalmente al examen, el alumnado deberá demostrar la realización de un trabajo equivalente al proyecto de la asignatura.

REFERENCIAS

Básicas

- Apuntes de la asignatura
- [Frank Tsui, Orlando Karam, and Barbara Bernal(2016)] Essentials of Software Engineering. Jones & Bartlett Learning, LLC
  [Recurs Electrònic:
  https://ebookcentral.proquest.com/lib/univalencia/detail.action?docID=4826428]
- [Martina Seidl, Marion Scholz, Christian Huemer, Gerti Kappel] UML @ Classroom. An Introduction to Object-Oriented Modeling
  [Recurs Electrònic: https://link.springer.com/book/10.1007%2F978-3-319-12742-2]
- [C. Larman (2004)] Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd (Edition Prentice Hall)[Recurs electronic:
  http://proquest.safaribooksonline.com/0131489062?uicode=valencia]
- [Grady Booch, James Rumbaugh, Ivar Jacobson (2005)] The Unified Modeling Language User Guide (2nd Rev. Edition) (Addison-Wesley) [Recurs electronic:
  http://proquest.safaribooksonline.com/0321267974]

Complementarias

- [Kenneth E. Kendall, Julie E Kendall (2010)] Systems Analysis and Design, 8th Edition (Prentice Hall)
- [Michael R. Blaha, James R Rumbaugh (2005)] Object-Oriented Modeling and Design with UML (2nd Edition) (Prentice Hall)
- [A. Weitzenfeld (2004)] Ingeniería de software orientada a objetos con UML, Java e Internet (Thomson)
- [Robert C. Martin (2003)] UML for Java programmers (Prentice Hall)[Recurs electronic: http://proquest.safaribooksonline.com/0131428489?uicode=valencia]
- [Roger S. Pressman (2009)] Software Engineering: A Practitioner's Approach, 7th Edition (Mc Graw Hill)
- [I. Sommerville (2011)] Software Engineering, 9th Edition (Addison-Wesley)
- [S. Sánchez Alonso, M. A. Sicilia Urbán, D. Rodríguez García (2011)] Ingeniería de software: un enfoque desde la guía SWEBOK (Garceta)
- [Bernd Bruegge, Allen H. Dutoit] Object-Oriented Software Engineering Using UML, Patterns, and Java, 3rd Edition (Edition Prentice Hall)

ADENDA COVID-19

Esta adenda solo se activará si la situación sanitaria lo requiere y previo acuerdo del Consejo de Gobierno