VNIVERSITAT DE VALÈNCIA

Guía Docente
34661 Ingeniería del software I

FICHA IDENTIFICATIVA

Datos de la Asignatura

Código	34661
Nombre	Ingeniería del software I
Ciclo	Grado
Créditos ECTS	6.0
Curso académico	2021 - 2022

Titulación(es)

Titulación	Centro	Curso	Periodo
1400 - Grado de Ingeniería Informática	Escuela Técnica Superior de Ingeniería	2	Segundo cuatrimestre

Materias

Titulación	Materia	Caracter
1400 - Grado de Ingeniería Informática	7 - Ingeniería del Software y Gestión de Proyectos	Obligatoria

Coordinación

Nombre	Departamento
RUEDA PASCUAL, SILVIA	240 - Informática

RESUMEN

La asignatura “Ingeniería del Software” es una asignatura obligatoria que forma parte de la materia Ingeniería del Software y Gestión de Proyectos del Grado en Ingeniería Informática. Tiene asignada una dedicación de 6 ECTS que se imparten durante el 2º cuatrimestre del 2º 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 curso Informática (34653) y Programación (34656)

Estar matriculado/a o haber aprobado la asignatura Entornos de Usuario (34660)

COMPETENCIAS

1400 - Grado de Ingeniería Informática

- G1 - Capacidad para concebir, redactar, organizar, planificar, desarrollar y firmar proyectos en el ámbito de la ingeniería en informática que tengan por objeto la concepción, el desarrollo o la explotación de sistemas, servicios y aplicaciones informáticas.
- G3 - Capacidad para diseñar, desarrollar, evaluar y asegurar la accesibilidad, ergonomía, usabilidad y seguridad de los sistemas, servicios y aplicaciones informáticas, así como de la información que gestionan.
- G4 - Capacidad para definir, evaluar y seleccionar plataformas hardware y software para el desarrollo y la ejecución de sistemas, servicios y aplicaciones informáticas, de acuerdo con los conocimientos adquiridos según las competencias específicas establecidas.
- G5 - Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones informáticas empleando los métodos de la ingeniería del software como instrumento para el aseguramiento de su calidad, de acuerdo con los conocimientos adquiridos según las competencias específicas establecidas.
- G9 - Capacidad para resolver problemas con iniciativa, toma de decisiones, autonomía y creatividad. Capacidad para saber comunicar y transmitir los conocimientos, habilidades y destrezas de la profesión de Ingeniero Técnico en Informática.
- R1 - Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
- R8 - Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
- R16 - Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.
- TI2 - Capacidad para seleccionar, diseñar, desplegar, integrar, evaluar, construir, gestionar, explotar y mantener las tecnologías de hardware, software y redes, dentro de los parámetros de coste y calidad adecuados.
- SI3 - Capacidad para participar activamente en la especificación, diseño, implementación y mantenimiento de los sistemas de información y comunicación.

RESULTADOS DE APRENDIZAJE

Esta asignatura permite obtener los siguientes resultados de aprendizaje:

- Aplicar metodologías para el desarrollo, implantación y mantenimiento de sistemas de información.
- Planificar y ejecutar correctamente procesos de desarrollo de software iterativos.
- Saber aplicar patrones de diseño software en cada situación en función de las necesidades del proyecto de desarrollo software.
- Definir pruebas de validación y verificación de requisitos.
- Obtener requisitos de usuario y del software.
- Desarrollar y exponer documentación técnica de proyectos en inglés.

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.
- Diseñar una estructura de módulos, utilizando patrones de diseño, para solucionar un problema y evaluar otras alternativas.
- Implementar un módulo que se ejecute correctamente y de manera eficiente.
- Probar aplicaciones de manera sistemática definiendo casos de prueba exhaustivos.

Trabajar en un pequeño equipo, colaborando en estos aspectos del desarrollo de software intercambiando ideas de manera constructiva y organizada.

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: 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 2: Trabajando con Diagramas de Clases
Sesión 3: Trabajando con Diagramas de Secuencia
Sesión 4: 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 4: 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. Comportamiento complejo y Diagramas de Estado/Actividad

Destrezas a adquirir:
- Representar mediante Diagramas de Estado y Actividad comportamientos complejos

Contenidos:
7.1 Elementos de Actividad
7.2 Diagramas de Actividad
7.3 Elementos de Máquina de Estados
7.4 Diagramas de Estado
7.5 Ejercicios de diagramas de Actividad y EstadoComplex behavior and State/Activity Diagrams

VOLUMEN DE TRABAJO

ACTIVIDAD	Horas	% Presencial
Clases de teoría	30,00	100
Prácticas en laboratorio	20,00	100
Prácticas en aula	10,00	100
Asistencia a eventos y actividades externas	3,00	0
Elaboración de trabajos en grupo	14,00	0
Elaboración de trabajos individuales	6,00	0
Estudio y trabajo autónomo	7,00	0
Preparación de clases de teoría	20,00	0
Preparación de clases prácticas y de problemas	30,00	0
Resolución de casos prácticos	7,00	0
Resolución de cuestionarios on-line	3,00	0
TOTAL	**150,00**	

METODOLOGÍA DOCENTE

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

Actividades teóricas.

Mediante el uso de la clase invertida, los contenidos teóricos de la asignatura serán proporcionados a través de material online en formato presentaciones y videos que el alumnado deberá consultar de manera individual previamente a las sesiones síncronas. Durante las sesiones síncronas de teoría, se resolverán las dudas y revisarán los errores comunes del alumnado, proporcionando una visión global e integradora.

Actividades prácticas

Las actividades prácticas complementan los contenidos teóricos y permiten al alumnado poner en práctica los conceptos adquiridos y mejorar su comprensión. Incluyen los siguientes tipos de actividades:

- Clases de problemas síncronas
- Sesiones de discusión y resolución de problemas y ejercicios
- Prácticas y seminarios síncronos
- Trabajos en grupo para planificación y desarrollo de proyectos software y generación de dinámicas de grupo
- Tutorías grupales e individuales programadas y bajo demanda

Trabajo personal

Preparación de clases y trabajos (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 grupos

Realización, por parte de grupos de estudiantado (5-6) de vídeos, trabajos, cuestiones, problemas principalmente dentro 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 para el desarrollo de un proyecto de software, desde la captura de los requisitos hasta su implementación, que incluirá la planificación, la generación de dinámicas de grupo y la documentación (que tendrá que presentarse en castellano, valenciano y/o en inglés).
- Presentación del proyecto software (en castellano, valenciano y/o en inglés).
- Tutorías grupales programadas y bajo demanda.

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 evaluarán del siguiente modo:

Se evaluará de forma regular su participación en las diferentes actividades, el grado de aprovechamiento obtenido en las diferentes actividades formativas y la implicación mostrada hacia su propio proceso de aprendizaje. Para ello se valorarán los siguientes aspectos:

- **Evaluación continua:** se valorará la participación y aprovechamiento de las actividades asíncronas propuestas (consulta de materiales, cuestionarios formativos, …) y la participación e implicación en el desarrollo y resolución de las actividades síncronas (presenciales u online) previstas (*N_Continua*) tanto en las sesiones teórico-prácticas como en las de laboratorio.
- **Proyecto:** se valorará tanto la calidad de la solución como la documentación y defensa de este (*N_Proyecto*).
- **Actividad individual:** consistente en diversas pruebas individuales durante el curso y una actividad final individual sobre el proyecto desarrollado (*N_Individual*).

La nota final se obtendrá de aplicar la siguiente fórmula:

Nota = 25% N_Continua + 65% N_Proyecto + 10% N_Individual

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.

Sólo se considerarán los trabajos entregados en la fecha estipulada por el profesorado. Esto incluye todas y cada una de las actividades, cuestionarios y ejercicios propuestos, el proyecto software y, en general cualquier tarea que se le encargue al alumnado.

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_Continua, N_Proyecto, N_Individual*.

Dada las características de las actividades que dan lugar a la nota de *N_Continua*, esta parte de la nota no es recuperable, manteniéndose la nota obtenida para la 2ª convocatoria, pero eliminándose el requisito de nota mínima en esta parte. Es decir, en 2ª convocatoria la nota *N_Continua* podrá ser inferior a 4,5 pero no el resto de las notas *N_Proyecto* y *N_Individual*.

Para poder obtener una calificación en la nota de evaluación continua (*N_Continua*) será necesario un índice de asistencia a las sesiones síncronas de prácticas y de laboratorio superior al 75%. Si por un motivo razonado y admitido por el profesorado, un alumno o alumna no pudiera asistir con regularidad a dichas sesiones síncronas, y, siempre y cuando participe de las tareas asíncronas de evaluación continua, podrá eliminarse este requisito.

Dado que la calificación de la parte de evaluación continua no es recuperable, para poder solicitar adelanto de convocatoria, el alumnado deberá haber cursado previamente la asignatura y haber superado la evaluación continua. En este caso, el alumnado deberá realizar el proyecto de la asignatura y la actividad final. La calificación final se obtendrá del siguiente modo:

Nota = 25% N_Continua + 65% N_Proyecto + 10% N_ Individual

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 máster 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 con la normativa vigente.

REFERENCIAS

Básicas

- Apuntes de la asignatura
- [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]
- [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]

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

Si la situación sanitaria lo requiere, la Comisión Académica de la Titulación aprobará un Modelo Docente de la Titulación y su adaptación a cada asignatura, estableciéndose en dicho modelo las condiciones concretas en las que se desarrollará la docencia de la asignatura, teniendo en cuenta los datos reales de matrícula y la disponibilidad de espacios.