FICHA IDENTIFICATIVA

DATOS DE LA ASIGNATURA

Código: 34840
Nombre: Ingeniería del software
Ciclo: Grado
Créditos ECTS: 6
Curso académico: 2025-26

TITULACIONES

Titulación	Centro	Curso	Periodo
1407 - Grado en Ingeniería Multimedia	Escola Tècnica Superior d'Enginyeria	3	Primer cuatrimestre

MATERIAS

Titulación	Materia	Carácter
1407 - Grado en Ingeniería Multimedia	Desarrollo del Software Multimedia	OBLIGATORIA

COORDINACIÓN

MARTINEZ PLUME JAVIER

RESUMEN

La asignatura "Ingeniería del Software" es una asignatura obligatoria que forma parte del Grado en Ingeniería Multimedia. 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

Sin haber requisitos previos de matrícula, se recomienda haber cursado las siguientes materias/asignaturas:
• Programación

COMPETENCIAS / RESULTADOS DE APRENDIZAJE

-

B4 - Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.

B5- Conocimiento de la estructura, organización, funcionamiento e interconexión de los sistemas informáticos, los fundamentos de su programación, y su aplicación para la resolución de problemas

propios de la ingeniería.

I9- Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.

MM21 - Comunicar de forma efectiva, tanto por escrito como oralmente, conocimientos, procedimientos, resultados e ideas relacionadas con las TIC y, concretamente de la Multimedia, conociendo su impacto socioeconómico.

MM23 - Usar de forma apropiada teorías, procedimientos y herramientas en el desarrollo profesional de la Ingeniería Multimedia en un contexto real (especificación, diseño, implementación, despliegue y evaluación de soluciones de sistemas multimedia).

MM24 - Capacidad para diseñar, desarrollar, evaluar y asegurar la accesibilidad, ergonomía, usabilidad y seguridad de los sistemas, servicios y aplicaciones multimedia, así como de la información que gestionan.

MM26 - Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones multimedia 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.

MM28 - 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 Multimedia.

MM3 - Aplicar de forma adecuada las metodologías, tecnologías, procedimientos y herramientas en el desarrollo profesional de los productos multimedia en un contexto de uso real, aplicando las soluciones adecuadas en cada entorno.

MM5 - Saber aplicar los recursos teóricos y prácticos para abordar en su globalidad una aplicación multimedia.

DESCRIPCIÓN DE CONTENIDOS

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 del Proceso Unificado de desarrollo del Software
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. Introducción al proceso de Desarrollo de software UML

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 del Proceso Unificado de desarrollo del Software1.3 Modelos y Procesos
1.3.1 Introducción al Proceso del Software
1.3.2 Modelos de Proceso del Software
1.3.3 Metodologías de desarrollo del Software
1.4 El Lenguaje Unificado de Modelado UML 2.0
1.4.1 Modelado Estructural y Dinámico
1.4.2 Vistas UML
1.5 Proceso Unificado de Desarrollo Software OO
1.5.1 Características
1.5.2 Fases
1.5.3 Actividades y Artefactos

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 de los Requisitos
2.1.2 Tipos de Requisitos
2.1.3 Actividades de Requisitos
2.1.4 Técnicas de obtención de Requisitos
2.1.5 Especificación de Requisitos del Software
2.1.6 Ejercicios sobre requisitos
2.2 Prototipo
2.3 Casos de Uso
2.3.1 Introducción.
2.3.2 Elementos: Actores, Sujetos, Eventos y Escenarios

2. Fase de Planificación y Especificación

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
2.3.3 Especificación de Casos de Uso
2.3.4 Relaciones: realización, generalización, extensión, inclusión
2.3.5 Diagramas de Casos de Uso
2.3.6 Errores típicos y Recomendaciones
2.3.7 Ejercicios sobre casos de uso.

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 las Operaciones del Sistema
Desarrollar los Contratos de las Operaciones del Sistema
Representar mediante Diagramas de Estado y Actividad comportamientos complejos
Contenidos:
3.1 Modelo Conceptual
3.2 Diagrama de clases
3.2.1 Clasificadores: Tipos y propiedades
3.2.2 Relaciones: dependencia, generalización, asociación, realización
3.3 Diagrama de objetos
3.3.1. Instancias
3.4 Ejercicios de diagramas de clases y objetos
3.5 Diagramas de Secuencia del Sistema
3.5.1 Interacciones
3.5.2 Líneas de vida
3.5.3 Mensajes
3.6 Contratos
3.7 Ejercicios de diagramas de secuencia y contratos
3.8 Comportamiento complejo y Diagramas de Estado/Actividad

3. Fase de 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 las Operaciones del Sistema
Desarrollar los Contratos de las Operaciones del Sistema
Representar mediante Diagramas de Estado y Actividad comportamientos complejos
Contenidos:
3.1 Modelo Conceptual
3.2 Diagrama de clases
3.8.1 Elementos de Actividad
3.8.2 Diagramas de Actividad
3.8.3 Elementos de Máquina de Estados
3.8.4 Diagramas de Estado
3.9 Ejercicios de diagramas de Actividad y Estado

4. Fase de 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 Modelo de Diseño
4.1.1 Responsabilidades
4.2 Diagrama de Clases de Diseño
4.3 Diagramas de Interacción
4.3.1 Diagrama de secuencia de operación
4.3.2 Diagrama de comunicación
4.4 Ejercicios sobre diagramas de interacción
4.5 Patrones para la asignación de responsabilidades.
4.5.1 Patrones GRASP
4.5.2 Patrones GoF
4.6 Ejemplo: Diseño del TPV

5. Arquitectura del sistema

Destrezas a adquirir:
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 patrones sobre capas de software
Contenidos:
5.1 Arquitectura multicapa y UML
5.1.1 Capas y particiones
5.1.2 Paquetes
5.1.3 Diagrama de Paquetes
5.2 Patrones de conexión entre paquetes
5.2.1 Patrones GRASP
5.2.2 Patrones GoF

6. Fase de 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:
6.1 Decisiones previas
6.2 Tipos de transformación
6.2.1 Transformaciones del modelo
6.2.2 Transformaciones del código
6.2.3 Transformaciones del modelo al código: Ingeniería directa
6.2.4 Transformaciones del código al modelo: Ingeniería inversa
6.3 Ingeniería directa
6.3.1 Mapeo de Clases
6.3.2 Mapeo de Relaciones
6.3.3 Mapeo de Herencia
6.3.4 Creación de Métodos
6.3.5 Mapeo de Contratos

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. 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
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 Aceptación
7.3.4 Pruebas de Sistema
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 (HORAS)

ACTIVIDADES PRESENCIALES

Actividad	Horas
Teoría	30,00
Prácticas en aula	10,00
Laboratorio	20,00
Total horas	**60,00**

ACTIVIDADES NO PRESENCIALES

Actividad	Horas
Asistencia a otras actividades	0,00
Elaboración de trabajos individuales o en grupo	4,00
Estudio y trabajo autónomo	4,00
Preparación de clases	73,00
Preparación de actividades de evaluación	0,00
Resolución de casos prácticos	9,00
Total horas	**90,00**

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 estudiante.

- **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. Comprenden los siguientes tipos de actividades:

- Clases de problemas y cuestiones en aula
- Sesiones de discusión y resolución de problemas y ejercicios previamente trabajados por los estudiantes
- 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)

Para la ejecución de estas actividades, el grupo teórico se subdividirá en subgrupos de menor tamaño (20 alumnos como máximo) de acuerdo a la necesidad.

La metodología docente de la asignatura seguirá el modelo docente aprobado por la Comisión Académica de los grados GII/GIM (https://go.uv.es/catinfmult/ModeloDocenciaGIIGIM). En caso de que se produzca un cierre de las instalaciones debido a la situación sanitaria, y si eso afectara total o parcialmente a las clases de la asignatura, éstas serán sustituidas por clases donde la presencialidad física será sustituida por clases síncronas online siguiendo los horarios establecidos.

En caso de que se produzca un cierre de las instalaciones debido a la situación sanitaria, y si eso afectara a alguna de las pruebas presenciales de la asignatura, éstas serán sustituidas por pruebas de naturaleza similar pero en modalidad virtual a través de las herramientas informáticas soportadas por la Universitat de València. Los porcentajes de evaluación permanecerán igual que los establecidos en la guía.

- **Trabajo personal del estudiantado**.

Preparación de clases y exámenes (estudio). Esta tarea se realizará de manera individual e intenta potenciar el trabajo autónomo.

- **Trabajo en pequeños grupos**.

Realización, por parte de pequeños grupos de estudiantes (3-4) 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 de investigación y recopilación de información sobre conceptos básicos de la Ingeniería del Software, ciclos de vida del software, metodologías ágiles, modelos de proceso del software, historia de UML.
- Presentación del trabajo en grupo.
- Trabajos en grupo para planificación y desarrollo de proyectos software y generación de dinámicas de grupo.
- Desarrollo de proyectos de software, cuya documentación deberá 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 los estudiantes. A través de ella se tendrá acceso al material didáctico utilizado en clase, así como los problemas y ejercicios a resolver.

esolver.

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 de los alumnos en el proceso de aprendizaje.

Se valorarán los siguientes aspectos:

- **Sesiones de Teoría:** 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, incluido el trabajo sobre los temas 1 y 2 (N_Teoría).
- **Sesiones de Problemas:** se valorará la implicación, teniendo en cuenta la asistencia regular a las actividades presenciales previstas, la entrega de los ejercicios propuestos, la participación en la resolución de los ejercicios durante las clases y la participación activa en los foros (N_Problemas).
- **Sesiones de Laboratorio:** se valorará la implicación, teniendo en cuenta la asistencia regular a las actividades presenciales previstas y la entrega de los ejercicios propuestos (N_Laboratorio).
- **Prueba objetiva individual:** consistente en uno o varios controles, o prueba de conocimiento, que

constarán tanto de cuestiones teórico-prácticas como de problemas (N_Examen).

Para poder aplicar este tipo de evaluación será necesario un índice de asistencia a las clases superior al 75%. Este porcentaje se aplicará de forma separada a cada bloque. Es decir, será necesario asistir a más del 75% de las sesiones de teoría, a más del 75% de las sesiones de prácticas y a más del 75% de las sesiones de laboratorio.

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 de laboratorio, el trabajo de los dos primeros temas y el proyecto software.

Para poder promediar las notas de las diferentes categorías se les pedirá **una nota mínima de 4,0** puntos (sobre 10) en cada una de ellas (*N_Teoría, N_Problemas, N_Laboratorio, N_Examen)*.

La nota final se obtendrá de aplicar la siguiente fórmula:

Nota=20%N_Continua+30%N_Laboratorio+50%N_Examen

N_Continua=50%N_Teoría+50%N_Problemas

Sistema de Evaluación Única

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

Las notas de las sesiones de teoría (N_Teoría), de problemas (N_Problemas) y de laboratorio (*N_Laboratorio*) no se recuperarán con ninguna otra actividad.

En ambos métodos, la evaluación se ajustará a la Normativa de Calificaciones de la Universitat de València. En el momento de redacción de la presente guía docente, la normativa vigente es la aprobada por el Consejo de Gobierno de la UVEG de 27 de enero de 2004, que se ajusta a lo establecido a tal efecto por los Reales Decretos 1044/2003 y 1125/2003. En ella se establece básicamente que las calificaciones serán numéricas de 0 a 10 con expresión de un decimal y a las que se debe añadir la calificación cualitativa correspondiente a la escala siguiente:

De 0 a 4,9: "Suspenso"

De 5 a 6,9: "Aprobado"

De 7 a 8,9: "Notable"

De 9 a 10: "Sobresaliente" o "Sobresaliente con Matrícula de Honor"

La copia o plagio manifiesto de cualquier actividad que forma parte de la evaluación supondrá la imposibilidad de superar la asignatura, sometiéndose seguidamente a los procedimientos disciplinarios oportunos indicados en el *PROTOCOLO DE ACTUACIÓN ANTE PRÁCTICAS FRAUDULENTAS EN LA UNIVERSITAT DE VALÈNCIA* (ACGUV 123/2020).

BIBLIOGRAFÍA

- Apuntes de la asignatura
- [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)
- [Grady Booch, James Rumbaugh, Ivar Jacobson (2005)] The Unified Modeling Language User Guide (2nd Rev. Edition) (Addison-Wesley)
- [C. Larman (2004)] Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd (Edition Prentice Hall)
- [Bernd Bruegge, Allen H. Dutoit] Object-Oriented Software Engineering Using UML, Patterns, and Java, 3rd Edition (Edition Prentice Hall)
- [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)