



Universitat de Lleida

# GUÍA DOCENTE **INGENIERÍA DE SOFTWARE**

Coordinación: SENDIN VELOSO, MONTSERRAT

Año académico 2023-24

## Información general de la asignatura

<b>Denominación</b>	INGENIERÍA DE SOFTWARE		
<b>Código</b>	102018		
<b>Semestre de impartición</b>	1R Q(SEMESTRE) EVALUACIÓN CONTINUADA		
<b>Carácter</b>	Grado/Máster	Curso	Carácter
	Doble titulación: Grado en Ingeniería Informática y Grado en Administración y Dirección de Empresas	3	OBLIGATORIA
	Grado en Ingeniería Informática	3	OBLIGATORIA
	Máster Universitario en Ingeniería Informática		COMPLEMENTOS DE FORMACIÓN
<b>Número de créditos de la asignatura (ECTS)</b>	6		
<b>Tipo de actividad, créditos y grupos</b>	<b>Tipo de actividad</b>	<b>PRALAB</b>	<b>TEORIA</b>
	<b>Número de créditos</b>	3	3
	<b>Número de grupos</b>	2	1
<b>Coordinación</b>	SENDIN VELOSO, MONTSERRAT		
<b>Departamento/s</b>	INGENIERÍA INFORMÁTICA Y DISEÑO DIGITAL		
<b>Distribución carga docente entre la clase presencial y el trabajo autónomo del estudiante</b>	6 ECTS = (10 h de Clase presencial + 15 h de Trabajo autónomo del estudiante) x 6 = 150 h de trabajo 40% Presencial (equivalente a 60 h) 60% Trabajo autónomo (equivalente a 90 h)		
<b>Información importante sobre tratamiento de datos</b>	Consulte <a href="#">este enlace</a> para obtener más información.		
<b>Idioma/es de impartición</b>	Preferentemente en Catalán (Castellano si algún estudiante muestra dificultades con el Catalán).		
<b>Distribución de créditos</b>	Montserrat Sendin Veloso 9		

Profesor/a (es/as)	Dirección electrónica\nprofesor/a (es/as)	Créditos impartidos por el profesorado	Horario de tutoría/lugar
SENDIN VELOSO, MONTSERRAT	montse.sendin@udl.cat	9	

## Información complementaria de la asignatura

Asignatura que se imparte durante el 2º semestre del 2º curso de la titulación Grado en Ingeniería Informática.

Corresponde a la Materia "Informática" dentro del Módulo de "Formación Básica".

Se asumen los conocimientos sobre programación orientada a objetos y estructuras de datos correspondientes a las asignaturas de Programación II y Estructuras de Datos.

## Objetivos académicos de la asignatura

- Conocer las bases conceptuales y los diferentes aspectos de la disciplina, entre ellos los modelos de proceso del ciclo de vida del software.
- Aplicar la técnica de los casos de uso.
- Especificar textualmente las necesidades funcionales y no funcionales de un determinado sistema software planteado a través de un enunciado (y/u otras informaciones procedentes del usuario).
- Desarrollar el diagrama de clases de un determinado sistema software siguiendo los principios del Modelado Orientado a Objetos.
- Utilizar una herramienta de modelado basada en UML.
- Comprender el concepto de código como algo que evoluciona en el tiempo.
- Ser capaz de programar pruebas unitarias básicas.
- Comprender los principios fundamentales del diseño orientado a objetos.
- Reconocer el concepto de responsabilidad como fundamental para plantear un diseño orientado a objetos.

## Competencias

### Competencias transversales

- **EPS-11:** Capacidad de comprender las necesidades del/de la usuario/a expresadas en un lenguaje no técnico.

### Competencias específicas

- **GII-CRI2:** Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
- **GII-CRI12:** Conocimiento y aplicación de las características, funcionalidades y estructura de las bases de datos, que permitan su adecuado uso, y el diseño y el análisis e implementación de aplicaciones basadas en ellos.
- **GII-CRI13:** Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los Sistemas de información, incluidos los basados en web.
- **GII-CRI16:** Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de

software.

- **GII-CRI17:** Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.

## Contenidos fundamentales de la asignatura

### Tema I - Aspectos introductorios

- 1.1. Cuestiones iniciales sobre la Ingeniería del Software
- 1.2. Un poco de historia
- 1.3. Proceso de desarrollo de software
- 1.4. Modelos de proceso de software
- 1.5. Conclusiones

### Tema II - Análisis de requisitos

- 2.1. Especificación de requisitos
- 2.2. La técnica de los Casos de uso
- 2.3. Un paso más en la especificación: Los *Diagramas de Secuencia del Sistema*
- 2.4. Conclusiones

### Tema III - Análisis del Dominio

- 3.1. *Diagramas de Clases del Análisis*
- 3.2. Un paso más en el análisis del dominio: los *Contratos de las operaciones*
- 3.3. Conclusiones

### Tema IV - Introducción al Diseño y Pruebas unitarias

- 4.1. La necesidad del diseño de código
- 4.2. El framework para pruebas unitarias JUnit 5
- 4.3. El sistema de control de versiones GIT

### Tema V - Los principios SOLID

- 5.1. Principio de responsabilidad única
- 5.2. Principio abierto-cerrado
- 5.3. Principio de sustitución de Liskov
- 5.4. Principio de segregación de interfaces
- 5.5. Principio de inversión de dependencias

## Tema VI - Diseño basado en responsabilidades

- 6.1. El concepto de responsabilidad
- 6.2. Patrones GRASP de asignación de responsabilidades
- 6.3. Realización de un caso de uso de ejemplo

## Ejes metodológicos de la asignatura

### Grupos Grandes: Clases Teoría (3 créditos)

- Parte teórica: clases soportadas con transparencias y/o apuntes
- Parte de aplicación práctica: se trabaja siempre con ejemplos. Se dispone de una **colección de problemas**, y en clase se trabaja la resolución de problemas concretos. Se van proporcionando las soluciones a lo largo del cuatrimestre.
- Conseguir que las sesiones sean participativas y dinámicas **requiere el compromiso por parte del estudiante**.

### Grupos Medianos: Clases Laboratorio (3 créditos)

- Clases dirigidas y seguimiento personalizado por equipos de trabajo en cada grupo de prácticas.
- **Aprendizaje cooperativo**: Equipos de trabajo de 3 personas con quien colaborar en la resolución de la Práctica de curso.
- Uso de la herramienta de Modelado en UML: **ArgoUML** y/o Visual Paradigm, y también otras online.
- Herramientas de control de versiones con **GIT** y framework de testing con **JUnit**.
- Trabajo continuado alrededor de un cierto **enunciado de práctica**, que simulará el desarrollo de un proyecto de software, como aplicación práctica de los contenidos de la asignatura.

### Trabajo Autónomo (no presencial):

- La práctica se completará en horas no presenciales.
- **Se recomienda** que el alumno resuelva por cuenta propia los problemas de la **colección de problemas**, con objeto de obtener feedback por parte del profesor.

El **sistema de evaluación** (detallado en el apartado correspondiente) consta de: **1**) pruebas escritas (los 2 exámenes parciales); y **2**) prácticas (a desarrollar en equipos preferentemente de dos personas).

### SOFTWARE a emplear:

- **Herramienta de Modelado**. Cualquier herramienta de modelado UML online, que incluya los diagramas trabajados en la asignatura (diagrama de casos de uso, diagrama de clases y diagrama de secuencia). En el laboratorio se dispone de las herramientas offline **Argo UML** y **Visual Paradigm** (Community Edition).
- **IntelliJ Idea** (Community Edition). Con los plugins de **JUnit 5**.

## Plan de desarrollo de la asignatura

Semana	Teoría (GG)	Laboratorio (GM)	Trabajo autónomo
1	Presentación Asignatura T1: Aspectos introdutorios	T1: Aspectos introductorios	Estudio

2	T1: Aspectos introductorios	T1: Aspectos introductorios	Estudio
3	T2: Análisis de requisitos Especificación de requisitos	T2: Análisis de requisitos Especificación de requisitos	Estudio y resolución problemas (Colección de problemas Análisis)
4	T2: Análisis de requisitos La tècnica de Casos de Uso. Problemas	Uso herramienta modelado UML Aplicación tècnica de Casos de Uso	Estudio, resolución problemas (Colección de problemas Análisis) y desarrollo práctica análisis
5	T2: Análisis de requisitos Especificación de Casos de Uso Problemas	Aplicación tècnica de Casos de Uso en enunciado práctica T2: Análisis de requisitos Diagramas de Secuencia del Sistema.	Estudio, resolución problemas (Colección de problemas Análisis) y desarrollo práctica análisis <i>Entrega Análisis de Requisitos (1ª parte)</i>
6	T3: Análisis del Dominio Tècnica de Modelado Orientado a Objetos	Aplicación DSS en enunciat de pràctica	Estudio, resolució problemes (Colección de problemas Análisis) y desarrollo pràctica análisis
7	T3: Análisis del Dominio Tècnica de Modelado Orientado a Objetos Problemas	Uso herramienta modelado UML Aplicación pràctica Tècnica de Modelado Orientado a Objetos	Estudio, resolució problemes (Colección de problemas Análisis) y desarrollo pràctica análisis <i>Entrega Análisis de Requisitos (2ª parte)</i>
8	T3: Análisis del Dominio Tècnica de Modelado Orientado a Objetos Problemas	Construcción Modelo del Dominio en ejercicios de la colección de problemas	Estudio, resolució problemes (Colección de problemas Análisis)
9	Primer parcial		
10	T4: Introducción al diseño	T4: JUnit 5- Concepto de pruebas Problemas simples de testing	Estudio, resolución de problemas (Colección de problemas testing)
11	T5: Principios SOLID Intro, OCP & LSP	T4: JUnit 5 - Objectes substituïts Problemas de testing con sustituciones	Estudio y resolución de problemas (Colección de problemas testing)
12	Sesión de CODE SMELLS	T4: JUnit 5 - Aspectos avanzados de JUnit 5 Problemas de testing	Estudio, resolución de problemas (Colección de problemas testing) y desarrollo práctica testing
13	T5: Principios SOLID SRP, ISP & DIP	Desarrollo práctica de testing	Estudio, resolución de problemas (Colección de problemas testing) y desarrollo práctica testing
14	T6: Patrones GRASP Concepto de responsabilidad Experto, Creador, Bajo acoplamiento	Uso de Git	Estudio, resolución de problemas (Colección de problemas testing) y desarrollo práctica testing

15	T6: Patrones GRASP Alta cohesión, Controlador Un Caso de Estudio: TPV	Uso de Git	Estudio y desarrollo práctica testing
16	Semana segundo parcial		Desarrollo práctica testing
17	Semana segundo parcial		<i>Entrega Pruebas Unitarias</i>
18	Tutorías		
19	Recuperación		

## Sistema de evaluación

Bloq. Activd.	Descripción	Ponderación	Nota mínima	En grupo	Presencial	Obligatoria	Recuperable
Parc1	Primer parcial	30%	3,0	No	Sí	Sí	Sí
Parc2	Segundo parcial	20%	No	No	Sí	No	Sí
BloqActvI	Análisis de Requisitos	25%	No	Sí	No	No	No
BloqActvII	Pruebas unitarias	25%	No	Sí	Sí	No	No

Nota final =  $0,30 * \text{Parc1} + 0,20 * \text{Parc2} + 0,25 * \text{BloqActvI} + 0,25 * \text{BloqActvII}$

- La asignatura se aprueba si la **nota final** es superior a **5** y se llega a la nota mínima en el primer parcial.

### Otras consideraciones y criterios:

- Tipología de los exámenes parciales: fijación de conceptos y resolución de problemas.
- Para todas las actividades evaluables: Entregas programadas, fechas no prorrogables.
- BloqActvI está compuesto por dos actividades de evaluación: *Análisis de Requisitos 1 – Iteraciones iniciales* (15%) y *Análisis de Requisitos 2 – Iteraciones avanzadas de la fase de Elaboración* (10%).
- Examen de RECuperación:
  - Caso de no llegar a la Nota mínima en el examen 1r Parcial**, se debe ir al examen de RECuperación.
  - Si la Nota Final < 5**, a pesar de llegar a la **Nota mínima** en el 1r Parcial, también se debe ir al examen de RECuperación.
  - Además, ofrece una oportunidad para mejorar la nota de la asignatura o convertir un compensable en aprobado (teniendo en cuenta que prevalece la nota obtenida en la RECuperación).
    - En estos dos últimos casos se debe examinar, al menos, del parcial con nota más baja.
    - Si aún así no se llega a la **Nota mínima** en el examen de RECuperación, **la nota final será un 4,9 como máximo** (dependiendo de la nota final ponderada).
- Evaluación alternativa**:
  - Entrega *BloqActvI* en fecha del 1r Parcial (*Semana 9ª*).
  - Entrega *BloqActvII* en fecha del 2do Parcial (*Semana 15ª*).
  - Indispensable realización de los dos exámenes parciales.

## Bibliografía y recursos de información

### Bibliografía básica

- Craig Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice-Hall, 2005 (3ª ed.)
- Boni García: Mastering Software Testing with Junit 5. Packt, 2017

## **Bibliografía complementaria**

- Gerald Kotonya, Ian Sommerville: Requirements Engineering: Processes and Techniques. Wiley, 1998
- Robert Martin: Agile Software Development: Principles, Patterns, and Practices, Prentice-Hall, 2002
- Lasse Koskela, Effective Unit Testing. A guide for Java developers. Manning, 2013