GUÍA DOCENTE

INGENIERÍA DE SOFTWARE

Coordinación: SENDIN VELOSO, MONTSERRAT

Año académico 2022-23

INGENIERÍA DE SOFTWARE 2022-23

Información general de la asignatura

<table>
<tr><td>Denominación</td><td>INGENIERÍA DE SOFTWARE</td></tr>
<tr><td>Código</td><td>102018</td></tr>
<tr><td>Semestre de impartición</td><td>1R Q(SEMESTRE) EVALUACIÓN CONTINUADA</td></tr>
<tr><td>Carácter</td><td>
<table>
<tr><th>Grado/Máster</th><th>Curso</th><th>Carácter</th><th>Modalidad</th></tr>
<tr><td>Doble titulación: Grado en Ingeniería Informática y Grado en Administración y Dirección de Empresas</td><td>3</td><td>OBLIGATORIA</td><td>Presencial</td></tr>
<tr><td>Grado en Ingeniería Informática</td><td>3</td><td>OBLIGATORIA</td><td>Presencial</td></tr>
<tr><td>Máster Universitario en Ingeniería Informática</td><td></td><td>COMPLEMENTOS DE FORMACIÓN</td><td>Presencial</td></tr>
</table>
</td></tr>
<tr><td>Número de créditos de la asignatura (ECTS)</td><td>6</td></tr>
<tr><td>Tipo de actividad, créditos y grupos</td><td>
<table>
<tr><th>Tipo de actividad</th><th>PRALAB</th><th>TEORIA</th></tr>
<tr><td>Número de créditos</td><td>3</td><td>3</td></tr>
<tr><td>Número de grupos</td><td>2</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Coordinación</td><td>SENDIN VELOSO, MONTSERRAT</td></tr>
<tr><td>Departamento/s</td><td>INFORMATICA E INGENIERIA INDUSTRIAL</td></tr>
<tr><td>Distribución carga docente entre la clase presencial y el trabajo autónomo del estudiante</td><td>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)</td></tr>
<tr><td>Información importante sobre tratamiento de datos</td><td>Consulte este enlace para obtener más información.</td></tr>
<tr><td>Idioma/es de impartición</td><td>Preferentemente en Catalán (Castellano si algún estudiante muestra dificultades con el Catalán).</td></tr>
<tr><td>Distribución de créditos</td><td>Montserrat Sendin Veloso 9</td></tr>
</table>

INGENIERÍA DE SOFTWARE 2022-23

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 2 º semestre del 2 º 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 Programaciín 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 responsabilitat única

5.2. Principio abierto-cerrado

5.3. Principio de substitución de Liskov

5.4. Principio de segregación de interficies

5.5. Principio de inversión de dependencias

Tema VI - ***Diseño basado en responsabilidades***

6.1. El concepto de responsabilidad

6.2. Patrones GRASP de assignación 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 **colecció 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.**
- 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).

Plan de desarrollo de la asignatura

Semana	Teoría (GG)	Laboratorio (GM)	Trabajo autónomo
1	Presentación Assignatura T1: Aspectos introductorios	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 técnica 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 *Entrega Análisis de Requisitos* (1ª parte)
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 práctica Tècnica de Modelado Orientado a Objetos	Estudio, resolució problemes (Colección de problemas Análisis) y desarrollo pràctica análisis *Entrega Análisis de Requisitos* (2ª parte)
8	T3: Análisis del Dominio Tècnica de Modelado Orientado a Objetos Problemas	Construcción Modelo del Dominio en enunciado práctica en enunciado de práctica	Estudio, resolució problemes (Colección de problemas Análisis) y desarrollo práctica análisis dominio
9	Primer parcial		Desarrollo práctica análisis dominio
10	T3: Análisis del Dominio Contratos de las operaciones T4: Introducció al disseny Concepte de proves	Problemas simples testing	Estudio, resolución de problemas (Colección de problemas testing) y desarrollo práctica análisis dominio
11	T4: Junit Objectes substituts	Problemas testing con sustituciones	Estudio y resolución de problemas (Colección de problemas testing) *Entrega Análisis Dominio y Contratos*
12	T5: Principios SOLID Intro, OCP & LSP	Problemas testing Aspectos avanzados de JUnit	Estudio, resolución de problemas (Colección de problemas testing) y desarrollo práctica testing
13	T5: Principios SOLID SRP, ISP & DIP	Uso de Git	Estudio, resolución de problemas (Colección de problemas testing) y desarrollo práctica testing
14	T6: Patrones GRASP Concepto de responsabilidad	Uso de Git	Estudio, resolución de problemas (Colección de problemas testing) y desarrollo práctica testing
15	T6: Patrones GRASP Experto, Creador, Bajo acoplamiento	Alta cohesión, Controlador	Estudio y desarrollo práctica testing
16	Semana segundo parcial		Desarrollo práctica testing
17	Semana segundo parcial		*Entrega Pruebas Unitarias*

18	Tutorías	
19	Recuperación	

Sistema de evaluación

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
Actv1	Análisis de Requisitos	20%	No	Sí	No	No	No
Actv2	Modelo del Dominio y Contratos	10%	No	Sí	No	No	No
Actv3	Pruebas unitarias	20%	No	Sí	Sí	No	No

Nota final = 0,30 * Parc1 + 0,20 * Parc2 + 0,20 * Actv1 + 0,10 * Actv2 + 0,20 * Actv3

- 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.
- *Examen de RECuperación:*
  1. **Caso de no llegar a la Nota mínima en el examen 1r Parcial,** se debe ir al examen de RECuperación.
  2. **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.
  3. 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 1r Parcial, **la nota final será un 4,5 como máximo**.

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