Universitat Autònoma de Barcelona

2021/2022

Ingeniería del Software

Código: 102743
Créditos ECTS: 6

Titulación	Tipo	Curso	Semestre
2502441 Ingeniería Informática	OB	2	2

La metodología docente y la evaluación propuestas en la guía pueden experimentar alguna modificación en función de las restricciones a la presencialidad que impongan las autoridades sanitarias.

Contacto

Nombre: Josep Lladós Canet
Correo electrónico: Josep.Llados@uab.cat

Uso de idiomas

Lengua vehicular mayoritaria: catalán (cat)
Algún grupo íntegramente en inglés: No
Algún grupo íntegramente en catalán: Sí
Algún grupo íntegramente en español: Sí

Otras observaciones sobre los idiomas

Algún grupo de prácticas puede ser en inglés.

Equipo docente

Alicia Fornes Bisquerra
Jorge Bernal del Nozal

Equipo docente externo a la UAB

Albert Berenguel
Daniel Soto Alvarez

Prerequisitos

Dado que la asignatura es de introducción al bloque de asignaturas de ingeniería del software, se asumirá que no se posee ningún tipo de conocimiento previo sobre la materia. Es responsabilidad de la propia asignatura proporcionar a los / las alumnos un medio para adquirir los conocimientos descritos en el apartado de contenidos de la asignatura (apartado 6 de esta guía). A pesar de ello es recomendable haber superado las asignaturas de Fundamentos de Informática y Metodología de la Programación de primer curso, y la asignatura de Bases de Datos de segundo curso.

Objetivos y contextualización

Esta asignatura es introductoria del bloque de asignaturas sobre el proceso de ingeniería del software. El objetivo general es dar una visión global y ordenada del proceso de desarrollo del software que no sea simplemente el de la programación (que constituye, sólo, una fase dentro de todo el proceso de la ingeniería

del software). Se introducirán las actividades fundamentales que constituyen el proceso de desarrollo (análisis de requisitos, diseño, implementación, pruebas y mantenimiento), la mayoría de las cuales se tratan en más profundidad en asignaturas posteriores.

Se trata, por tanto, que el alumno adquiera una visión general de qué es un proceso de desarrollo de software, como se modela software, qué herramientas se utilizan, cómo se gestiona la calidad y cómo se administra un proyecto. Más concretamente, los objetivos son:

- Proporcionar una visión general del proceso de desarrollo del software, los paradigmas, las actividades, y los objetivos de cada actividad.
- Conocer qué es el modelado del software. La importancia y la dificultad de la especificación de requerimientos y el diseño en sus diferentes vistas.
- Aprender los conceptos básicos de UML para trabajar de manera práctica en el modelado del software.
- Introducir al alumno / a en los principios, conceptos y técnicas que se aplican para administrar y controlar la calidad del software.
- Dar a conocer las técnicas de administración requeridas para planificar, organizar, monitorizar y controlar proyectos de software.
- Familiarizarse con el modelo Agile/SCRUM como paradigma de desarrollo evolutivo más relevante.

Competencias

- Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
- Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software.
- 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.
- Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.
- 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.
- 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.
- Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.
- Trabajar en equipo.

Resultados de aprendizaje

1. Asumir y respetar el rol de los diversos miembros del equipo, así como los distintos niveles de dependencia del mismo.
2. Comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software, a un nivel avanzado.
3. Comprender qué es el análisis de requerimientos y porqué es importante.
4. Comprender qué es el diseño y cómo conseguir un buen diseño que incluya los requerimientos de accesibilidad, ergonomía, usabilidad y seguridad de las aplicaciones informáticas.
5. Conocer el proceso de desarrollo del software RUP.
6. Conocer la notación y construcción de diagramas UML.
7. Conocer y aplicar métodos de comunicación y negociación eficaz para realizar las tareas propias de la profesión.
8. Diseñar aplicaciones sabiendo seleccionar el paradigma de desarrollo software más adecuado.
9. Diseñar y evaluar interfaces software persona computador que garanticen la accesibilidad y usabilidad de los sistemas, servicios y aplicaciones informáticas.
10. Diseñar y mantener aplicaciones siguiendo criterios de robustez y fiabilidad.

11. Entender qué es el software, cuáles son los objetivos que nos planteamos al desarrollar un software y la problemática asociada a su desarrollo.
12. Identificar, gestionar y resolver conflictos.
13. Trabajar cooperativamente.

Contenido

BLOQUE 1. EL SOFTWARE Y LA INGENIERÍA DEL SOFTWARE

Tema 1. Principios de la ingeniería del software.

- Definición y objetivos de la ES. Definición de software. Características del software. Aplicaciones del software. Definición de ES. Objetivos del SE.
- Evolución del software. Etapas. Crisis del software: problemas y causas.
- Proceso, método y herramienta. Definiciones. Actividades en el proceso de desarrollo del software.
- Paradigmas del desarrollo del software. Modelo lineal secuencial (ciclo de vida clásico). Modelo de prototipaje. Modelo evolutivo. Modelo en espiral.

BLOQUE 2. MODELADO DE SOFTWARE

Tema 2. Análisis de Requerimientos del Software.

- Introducción. Tipo de requerimientos. Tareas a realizar.
- Comprensión del problema. Técnicas de comunicación. Problemas asociados. Principios del análisis.
- Especificación de requerimientos. Propiedades deseables de una ER. Estándares de ES. Revisión y validación de la especificación.

Tema 3. Diseño del Software.

- Introducción. Proceso de diseño. Diseño de datos, diseño arquitectónico, diseño de la interfaz, diseño procedimental. Principios (objetivos) del diseño.
- Conceptos del diseño. Abstracción. Modularidad. Refinamiento.
- Diseño modular efectivo. Independencia funcional. Cohesión. Acoplamiento. Heurísticas para un diseño modular efectivo.
- Diseño de Interfaces de usuario.

Tema 4. UML (Unified Modeling Language)

- Evolución e historia de UML.
- Vistas de UML. Vista de casos de uso, vista lógica, vista de componentes, vista de desarrollo.

BLOQUE 3. ADMINISTRACIÓN DE LA CALIDAD DEL SOFTWARE

Tema 5. Calidad del software.

- Introducción. Conceptos de calidad.
- Prueba del software. Modelos de prueba del software. Estrategias de prueba. Los problemas del mal diseño de software. Principios básicos del diseño de software OO. TDD: Test-Driven Development.
- Administración de la configuración del software.

BLOQUE 4. ADMINISTRACIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE

Tema 6. SCRUM.

- Introducción. Metodología Ágil de gestión de proyectos.
- SCRUM. Características. Historia.
- Componentes de SCRUM. Roles. Artefactos. Procesos.
- Escalabilidad. Scrum de Scrums.
- Herramientas Software.
- Otros modelos ágiles. Lean. Xtreme programming. Kanban.

- Un caso de estudio. Presentación de caso de estudio a seguir durante el curso.

Metodología

La asignatura de Ingeniería del software tiene una fuerte componente aplicada. Por este motivo, después de introducir los conceptos teóricos fundamentales, la asignatura se desarrolla según una metodología ABP (Aprendizaje Basado en Problemas). De esta manera, el alumno adquirirá los conocimientos a partir de un caso práctico (problema guía) de dimensión real. El problema guía dirigirá el proceso de aprendizaje a lo largo del curso, de manera que el trabajo práctico o los seminarios se estructurarán en torno a este caso. El alumno deberá adquirir el perfil de analista / arquitecto del software para tomar las decisiones oportunas que concluyan en la construcción de una propuesta de diseño del sistema propuesto. El profesor hará el papel de tutor y, en situaciones determinadas, de cliente, fomentando simulaciones de diálogo desarrollador - cliente.

La asignatura consta de 4 horas semanales presenciales. No se distingue entre horarios de teoría, problemas y prácticas de laboratorio. Durante las horas presenciales se alternarán clases de teoría, problemas o prácticas según la planificación que se indicará el día de la planificación de la asignatura. De manera general, se concentrará la explicación de los contenidos teóricos fundamentales durante las primeras sesiones del curso (aproximadamente las 6 primeras semanas). Una vez impartidos los contenidos teóricos, se dedicarán todas las sesiones a trabajo basado en problemas, identificando algunas sesiones de asistencia obligatoria donde se llevará a cabo la práctica en equipos de 5 personas o se harán problemas a entregar al final de la sesión.

TEORÍA

Consiste en clases magistrales con material multimedia disponibleen el Campus Virtual de la UAB. El objetivo principal de estas clases es introducir las nociones básicas sobre los procesos, métodos y herramientas relativos a la visión del ciclo de vida del software desde un punto de vista de ingeniería. Se expondrá cuáles son los principios y paradigmas de la ingeniería del software, las diferentes actividades y las tareas a llevar a cabo en cada actividad. Las clases de teoría deben permitir al alumno/a adquirir una visión real del rigor, planificación y la sistemática que requiere el desarrollo profesional del software. No se profundizará en los temas ya que en asignaturas de cursos posteriores estos ya se desarrollan, pero se explicarán los conceptos necesarios para que el alumno/a entienda el proceso de desarrollo de software completo. Las sesiones de teoría asumirán que el alumnado sigue los materiales distribuidos a través del campus virtual, por lo que se orientarán a explicar los conceptos clave, y fomentar el debate crítico sobre los contenidos. Para hacer las clases más dinámicas, se suele distribuir materiales previos al alumnado para autoaprendizaje (videos, enlaces web, presentaciones, etc.) y se pide que se trabajen antes de la clase, por lo que la clase se dirige a debatir los conocimientos, resolver dudas, incidir en lo más esencial y trabajar ejemplos prácticos.

SEMINARIOS / PROBLEMAS

Los seminarios son clases con trabajo participativo del alumnado. Según el objetivo de los seminarios, se dividirá el alumnado del grupo en subgrupos más reducidos. Tienen una doble función, enlazando por un lado con las clases magistrales y por la otra con las sesiones prácticas sobre el problema a resolver en equipos. Por una parte, los seminarios complementarán los contenidos tecnológicos expuestos en las clases magistrales, completando su comprensión. Por la otra serán el marco de trabajo donde debatir, establecer las bases y desarrollar el problema guía. Se le darán al estudiante los conocimientos necesarios para llevar adelante el trabajo práctico o se le indicará dónde conseguirlos. En los seminarios se promueve fundamentalmente la capacidad de análisis y síntesis, así como el razonamiento crítico y la toma de decisiones del alumno frente a la resolución del problema guía. Dado que se trata de una asignatura muy práctica y donde los contenidos teóricos son de baja complejidad, se refuerza a menudo el trabajo práctico con resolución de problemas o el trabajo de actividades participativas. Se coordinarán los contenidos de los seminarios con las etapas a resolver en el trabajo de la práctica. Dado que se alternarán sesiones de teoría, problemas y prácticas a lo largo del curso, para una mejor organización, el alumnado dispondrá a principio de curso a través del campus virtual de la programación de todas las sesiones.

SEGUIMIENTO TRABAJO PRÁCTICO

No habrá sesiones presenciales de laboratorio. Los/las estudiantes resolverán el problema guía de manera autónoma en grupos reducidos, y el seguimiento se hará durante las horas de clase en el horario regular. Las sesiones obligatorias de prácticas (6 sesiones de 2 horas durante el curso) consistirán en dos partes. Primero, cada equipo de prácticas presentará al resto de la clase el progreso de su trabajo (el encargado de hacer la presentación será la persona que haga de scrum master en aquella iteración). En la segunda parte de la sesión el/la tutor/a de prácticas tendrá una entrevista individual con cada equipo para hacer el seguimiento y la evaluación. Se sigue un desarrollo basado en SCRUM, donde cada incremento, llamado sprint, corresponde a una sesión. La asistencia a las entrevistas de seguimiento será OBLIGATORIA para todos los alumnos/as. La no presencia incidirá negativamente en la evaluación del trabajo práctico, comportando no superar la evaluación si el número de no presencias supera el 15% de las sesiones. En caso de ausencia justificada, el profesorado considerara mecanismos de compensación. El tamaño de los equipos de prácticas se determinará a principio de curso. De manera orientativa serán equipos de 5 personas. Dado que la asistencia a las sesiones de seguimiento es obligatoria, y se hará durante los horarios presenciales, es importante que todas las personas miembros de un equipo deprácticas estén matriculados y asistan al mismo grupo.

TRABAJO PRÁCTICO

Se entregará a los alumnos al inicio de curso un problema guía. Este problema tendrá dos funciones. En primer lugar utilizarlo como ejemplo en las clases de teoría y seminarios, resolviendo partes adecuadas a los contenidos de cada momento. En segundo lugar se puede entender como el enunciado de una práctica que se va resolviendo por etapas a lo largo del curso. Los alumnos/as se organizarán en equipos de trabajo de 5 personas (según número de matriculados). La resolución del trabajo la harán de manera autónoma, con el seguimiento semanal antes descrito.

LOGRO DE COMPETENCIAS

- T03 - Trabajar en equipo. Se trabajará en el trabajo práctico donde los alumnos / as formarán equipos de 5-6 personas, y donde se considera el trabajo en equipo como una de las competencias a trabajar. Los alumnos / as deberán distribuirse los roles que habría en un proyecto de software, y cada semana uno de ellos hará las funciones de coordinador (scrum master). Se evaluará como uno de los indicadores del trabajo práctico.
- CRI03.03 - Comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo software, a un nivel avanzado: Se trabajará a sesiones específicas de problemas donde se propondrán técnicas de captura de requerimientos, en particular de Gamestorming, y también en la práctica, donde los grupos de trabajo se organizan siguiendo los principios de la metodología SCRUM, y en cada semana (sesión) se requerirá que uno de los miembros del equipo asuma el papel de SCRUM master (coordinador). Se evaluará a partir del seguimiento individualizado del trabajo práctico (a partir de las actas que el Scrum master haga de las reuniones de seguimiento).
- CRI08.01 - Diseñar y mantener aplicaciones siguiendo criterios de robustez y fiabilidad y CRI08.02 - Diseñar aplicaciones sabiendo seleccionar el paradigma de desarrollo software más adecuado: Se trabajarán tanto en las sesiones presenciales, en particular de problemas, como el trabajo práctico. Se evaluará con la prueba escrita (examen de teoría, primer control) como con la calificación del trabajo práctico.
- CRI16.01 - Comprender qué es el análisis de requerimientos y porque es importante y CRI16.02 - Conocer el proceso de desarrollo del software ágil: Se trabajarán en las primeras sesiones del curso de teoría y problemas (primeras cuatro semanas), así como las primeras sesiones de prácticas (primeras dos sesiones). Se evaluarán con la prueba escrita (examen de teoría, segundo control) como con la calificación del trabajo práctico.
- CRI17.02 - Conocer la notación y construcción de diagramas UML y E03.02 - Diseñar y evaluar interfases software persona-computador que garanticen la accesibilidad yusabilidad de los sistemas, servicios y aplicaciones informáticas: Se trabajarán a las sesiones centrales del curso de teoría y problemas (semana 5 a semana 10), así como a las sesiones de prácticas (sesiones 3 a 5). Se evaluarán con la prueba escrita (examen de teoría, segundo control) como con la calificación del trabajo práctico. Se propondrá a los alumnos ejercicios opcionales que podrán entregar por tener puntuaciones adicionales en el examen correspondiente a esta parte.

- E03.01 - Comprender qué es el diseño y cómo conseguir un buen diseño que incluya los requerimientos de accesibilidad, ergonomía, usabilidad y seguridad de las aplicaciones informáticas y E03.02 - Diseñar y evaluar interfases software persona-computador que garanticen la accesibilidad y usabilidad de los sistemas, servicios y aplicaciones informáticas: al igual que la competencia anterior, se trabajarán en las sesiones centrales del curso de teoría y problemas (semana 5 a semana 10), así como las sesiones de prácticas (sesiones 3 a 5) . Se evaluarán con la prueba escrita (examen de teoría, segundo control) como con la calificación del trabajo práctico.
- E05.01 - Entender qué es el software, cuáles son los objetivos que se plantean en desarrollar un software y la problemática asociada a su desarrollo. Se trabajará a las sesiones finales de teoría y problemas (semanas 11 y 12), así como las sesiones de prácticas (sesión 6). Se evaluará con la prueba escrita (examen de teoría, segundo control) como con la calificación del trabajo práctico.
- E09.04 - Conocer y aplicar métodos de comunicación y negociación eficaz en la realización de las tareas propias de la profesión. Fundamentalmente se trabajará en las sesiones de prácticas donde los alumnos deberán defender ante el profesor las decisiones tomadas en su proyecto. El/la profesor/a hará el papel de cliente para simular lo que sería un proyecto real. Se evaluará en la calificación del trabajo práctico.

CLASES ONLINE

Aunque s'espeera que toda la docencia se pueda desarrollar en modalidad presencial, las sesiones de clase pueden tener que adaptarse a modalidad on-line a través de plataformas de videoconferencia. Cualquier modificación que afecte el formato y horarios, se informará a través del campus virtual y al inicio de curso.

Nota: se reservarán 15 minutos de una clase dentro del calendario establecido por el centro o por la titulación para que el alumnado rellene las encuestas de evaluación de la actuación del profesorado y de evaluación de la asignatura o módulo.

Actividades

Título	Horas	ECTS	Resultados de aprendizaje
Tipo: Dirigidas			
Clases de teoria	30	1,2	4, 3, 5, 6, 11
Preparación y estudio	30	1,2	4, 3, 5, 6, 8, 11
Seminarios / Problemas	15	0,6	3, 7, 6, 8
Tipo: Supervisadas			
Entrevistas de seguimiento de la práctica	15	0,6	1, 7, 12, 13
Tutorias	15	0,6	4, 3, 5, 6, 11
Tipo: Autónomas			
Trabajo práctico	38	1,52	1, 2, 3, 7, 6, 12, 13

Evaluación

La evaluación se llevará a cabo en base al trabajo práctico desarrollado por el alumno e incluido en su portfolio (resolución final de problema guía, material generado en las actividades de los seminarios), el grado de implicación en los seminarios y los conocimientos tecnológicos alcanzados de la materia a estudio. Se contemplarán los siguientes instrumentos de evaluación:

- Evaluación individual (AI). Prueba escrita presencial para valorar el grado de conocimientos alcanzados por el alumno a nivel individual. Habrá dos pruebas parciales durante el curso que permitirán liberar

materia siempre y cuando la nota obtenida sea igual o superior a 5. La nota correspondiente a cada parcial tiene un peso del 50% en la nota final de la evaluación indiv idual.

Por lo tanto, AI = 50% Nota_Parcial1 + 50% Nota_Parcial2.

- Evaluación colectiva los equipos de trabajo (AC). Prueba del rendimiento conjunto de los diferentes grupos de trabajo para resolver el problema guía y las diferentes actividades presentadas en los seminarios. Consta de dos instrumentos:
  AC1 (evaluación técnica). Documentación entregada por el alumnado de su trabajo práctico. Se evaluará la resolución técnica del problema guía.
  AC2 (evaluación de progreso y competencias). La defensa del trabajo por parte de los alumnos en las entregas, el seguimiento de la evolución del trabajo y la participación activa en los seminarios. Se evaluará a partir de las entrevistas con el profesor, pruebas escritas de corta duración, entregas o exposiciones de problemas y casos planteados en las actividades. Aunque este instrumento es de evaluación colectiva, a criterio del tutor / a, puede ser corregido individualmente en casos en que la participación del alumno / a destaque (exceso o defecto) respecto a su grupo.

Cabe señalar que mientras los instrumentos AI y AC1 valoran el rendimiento del alumno / ay el grado de consecución de los conocimientos, el instrumento AC2 valora el esfuerzo del alumno / a.

INDICADORES Y VALORACIÓN:

Lacalificación final se obtendrá según el siguiente baremo:

QF = 50% AI + 50% AC

Para aprobar la asignatura es necesario haber conseguido una puntuación mínima de 5 en las dos calificaciones (AI y AC), así como en las pruebas parciales para liberar materia establecidos a lo largo del curso. A criterio del profesor / a podrá, sin embargo, establecer compensaciones entre las notas de las entregas parciales de las actividades (correspondientes AC).

La asignatura será evaluada como no Evaluable sólo en el caso que el alumno no se haya presentado a ninguna de las pruebas de evaluación contempladas ni haya entregado total o parcialmente los trabajos.

En caso de no superar la asignatura debido a que alguna de las actividades de evaluación no alcanza la nota mínima requerida, la nota numérica del expediente será el valor menor entre 4.5 y la media ponderada de las notas. Con las excepciones de que se otorgará la calificación de "no evaluable" a los / as estudiantes que no participen en ninguna de las actividades de evaluación, y de que la nota numérica del expediente será elvalor menor entre 3.0yla media ponderada de las notas en caso de que el estudiante haya cometido irregularidades en un acto de evaluación (y por tanto no será posible el aprobado por compensación).

Se otorgarán las Matrículas de Honor dentro de los máximos admitidos por la normativa de la UAB (en función del número de matriculados) a las notas más altas iguales o superiores a 9.

Para cada actividad de evaluación, se indicará un lugar, fecha y hora de revisión en la que el estudiante podrá revisar la actividad con el profesor. En este contexto, se podrán hacer reclamaciones sobre la nota de la actividad, que serán evaluadas por el profesorado responsable de la asignatura. Si el estudiante no se presenta en esta revisión, no se revisará posteriormente esta actividad.

Ver apartado "PLAGIO" sobre medidas en casos de irregularidades por plagio en las actividades de evaluación.

ACTIVIDADES VOLUNTARIAS DE EVALUACIÖN:

Durante el curso se propondrán al alumnado actividades de entrega opcional que servirán para complementar las notas de las actividades obligatorias.

RECUPERACIONES:

- Evaluación individual (AI). Se harán dos exámenes parciales de teoría liberatorios durante horas lectivas. Los alumnos que no superen esta prueba (con nota igual o superior a 5), dispondrán de un examen de recuperación en la fecha de evaluación final programada para la titulación.
- Evaluación colectiva (AC). El trabajo práctico se evalúa en forma de evaluación continua en las sesiones de seguimiento. Por lo tanto no habrá ninguna actividad de recuperación al final del curso. Sin embargo y con respecto a la resolución del problema guía, el hecho de no entregar a tiempo o no superar una de las entregas intermedios puede ser recuperadoreprogramando la planificación del trabajo de las sesiones posteriores durante el curso, de acuerdo con el profesor.

FECHAS DE EVALUACIÓN:

Las fechas de evaluación continua y entrega de trabajos se publicarán en el campus virtual y pueden estar sujetos a cambios de programación por motivos de adaptación a posibles incidencias. siempre informará al campus virtual sobre estos cambios ya que se entiende que es el mecanismo habitual de intercambio de información entre profesor y estudiantes.

ALUMNOS REPETIDORES:

No se guardan notas parciales (teoría o prácticas) de un curso para otro. Sin embargo, a criterio del profesor y en función de las evaluaciones de cursos previos, se podrán establecer compensaciones. Esta información se anunciará el día de la presentación de la asignatura, y elcampus virtual.

PLAGIO:

Sin perjuicio de otras medidas disciplinarias que se estimen oportunas, y de acuerdo con la normativa académica vigente, las irregularidades cometidas por un estudiante que puedan conducir a una variación de la calificación se calificarán con un cero (0). Las actividades de evaluación calificadas de esta forma y por este procedimiento no serán recuperables. Si es necesario superar cualquiera de estas actividades de evaluación para aprobar la asignatura, esta asignatura quedará suspendida directamente, sin oportunidad de recuperarla en el mismo curso. Estas irregularidades incluyen, entre otros:
- la copia total o parcial de una práctica, informe, o cualquier otra actividad de evaluación;
- dejar copiar;
- presentar un trabajo de grupo no hecho íntegramente por los miembros del grupo;
- presentar como propios materiales elaborados por un tercero, aunque sean traducciones o adaptaciones, y en general trabajos con elementos no originales y exclusivos del estudiante;
- tener dispositivos de comunicación (como teléfonos móviles, SmartWatch, etc.) accesibles durante las pruebas de evaluación teórico o Prácticas individuales (exámenes).

En resumen: copiar, dejar copiar o plagiar en cualquiera de las actividades de evaluación equivale a un SUSPENSO con nota inferior a 3,0.

ACLARACIÓN FINAL:

Para cualquier duda o discrepancia, prevalecerá la información más actualizada que se comunicará el día de la presentación de la asignatura y que se publicará en el campus virtual.

Actividades de evaluación

Título	Peso	Horas	ECTS	Resultados de aprendizaje
Evaluación colectiva (trabajo pràctico)	50	4	0,16	2, 3, 5, 7, 6, 8, 10
Evaluación individual (examen de teoria)	50	3	0,12	1, 4, 3, 5, 7, 6, 9, 11, 12, 13

Bibliografía

Los contenidos de la asignatura explicados en clase se extraen de diferentes fuentes. Muchos materiales son en línea, extraídos de materiales multimedia, vídeos, etc. Durante el curso, se proporcionarán los materiales de confección propia necesarios para el seguimiento de la asignatura a través del campus virtual. También se proporcionarán enlaces a documentación en línea, software para los ejercicios prácticos, etc. en formato libre. A continuación se referencian algunos libros clásicos de ingeniería del software. Se recomienda al alumnado seguir la asignatura con los materiales actualizados que se recomendarán a clase.

- G. Booch, J. Rumbaugh, I. Jacobson. El lenguaje unificado de modelado. Addison-Wesley, 2001.
- I. Jacobson, G. Booch, J. Rumbaugh. El proceso unificado de desarrollo de software. Addison-Wesley, 2000.
- J. Rumbaugh , I. Jacobson, G. Booch. El lenguaje unificado de modelado: manual de referencia. Addison-Wesley, 2007.
- T. Quatrani. Visual Modeling with Rational Rose 2000 and UML. Addison-Wesley, 2000.
- P. Krutchen. The Rational Unified Process. An Introduction. Addison-Wesley, 2000.
- Roger S. Pressman, Ingeniería del software, un enfoque práctico. Mc Grah-Hill, 7a. edició , 2010.
- A. Álvarez García, R. de las Heras del Dedo, C. Lasa Gómez, Métodos Ágiles. Scrum, Kanban, Lean. Anaya Multimedia, 2017.
- K. Pohl, C. Rupp. Requeriments Engineering Fundamentals. Rocky Nook Inc. 2011.
- Alexey Krivitsky. Lego4Scrum. https://www.lego4scrum.com/

Software

- Scrum. Azure DevOps / Jira
- UML. PlantUML
- Gestión de la configuración. GitHub
- TDD. Cualquier IDE de Python.