Proyecto/Guía docente de la asignatura

Se debe indicar de forma fiel cómo va a ser desarrollada la docencia. Esta guía debe ser elaborada teniendo en cuenta a todos los profesores de la asignatura. Conocidos los espacios y profesorado disponible, se debe buscar la máxima presencialidad posible del estudiante siempre respetando las capacidades de los espacios asignados por el centro y justificando cualquier adaptación que se realice respecto a la memoria de verificación. Si la docencia de alguna asignatura fuese en parte online, deben respetarse los horarios tanto de clase como de tutorías). La planificación académica podrá sufrir modificaciones de acuerdo con la actualización de las condiciones sanitarias.

Asignatura	DISEÑO, INTEGRACIÓN Y ADAPTACIÓN DE SOFTWARE		
Materia	TECNOLOGÍAS DE LA INFORMACIÓN		
Módulo	TECNOLOGÍAS ESPECÍFICAS		
Titulación	GRADO EN INGENIERÍA INFORMÁTICA		
Plan	545	**Código**	46937
Periodo de impartición	2º CUATRIMESTRE	**Tipo/Carácter**	OBLIGATORIA (Mención TI)
Nivel/Ciclo	GRADO	**Curso**	3º
Créditos ECTS	6 ECTS		
Lengua en que se imparte	CASTELLANO		
Profesor/es responsable/s	José Manuel Marqués Corral		
Datos de contacto (E-mail, teléfono…)	TELÉFONO: 983 423000 / ext. 5638 E-MAIL: jmmc@infor.uva.es		
Departamento	Departamento de Informática.		

1. Situación / Sentido de la Asignatura

1.1 Contextualización

Esta asignatura forma parte del módulo “Tecnologías Específicas” dentro de la materia “Tecnologías de la Información” y está orientada a la formación en modelado y diseño software. El modelado y el diseño son dos de los aspectos básicos del cuerpo de conocimiento de la ingeniería del software[1] y forman parte del conjunto de materias esenciales de la formación en informática[2].

Teniendo en cuenta que esta es una asignatura de grado, los conocimientos previos de los alumnos y, sobre todo, las limitaciones temporales, se ha considerado que los contenidos han de centrarse en los aspectos fundamentales del modelado y el diseño de la arquitectura de las aplicaciones software. La intención es que el estudiante comprenda el concepto de arquitectura software, que conozca alguno de los patrones y estilos arquitectónicos de mayor utilización y difusión y, por supuesto, que sea capaz de concebir, desarrollar y documentar el diseño arquitectónico y detallado de un sistema software.

El objetivo es que los estudiantes sean capaces de adquirir las habilidades necesarias para abordar el análisis y diseño de un sistema informático aplicando principios, métodos, herramientas y prácticas propias de la ingeniería.

1.2 Relación con otras materias

Fundamentos de Ingeniería del Software, Sistemas distribuidos, Fundamentos de Programación, Programación Orientada a la Integración de Sistemas, Interacción Persona – Computador, Diseño de Bases de Datos.

1.3 Prerrequisitos

Se recomienda que los alumnos hayan cursado con aprovechamiento al menos las siguientes asignaturas del vigente plan de estudios: Fundamentos de Ingeniería del Software, Sistemas Distribuidos, Interacción Persona-Computadora y Tecnología y Diseño de Bases de Datos.

Se recomienda que el alumno tenga un nivel de inglés que le permita la comprensión de la bibliografía básica y complementaria recomendada para la asignatura.

Dadas las características inherentes a todo proyecto de diseño de un sistema software, son muy recomendables la constancia, la iniciativa personal y la predisposición al trabajo colaborativo en grupo. La asimilación de los contenidos teóricos de la asignatura implica, por una parte, la capacidad de lectura crítica de los textos básicos y complementarios puestos a su disposición, pero además la búsqueda proactiva del material complementario en la red que es publicado por los grandes fabricantes de hardware y software, y que resulta imprescindible para el desarrollo de un buen diseño.

2. Competencias

[1] "Swebok - IEEE Computer Society." 2015. 2 Oct. 2015 <http://www.computer.org/web/swebok>

[2] "SE2004, Software Engineering 2004 - IEEE Computer Society." 2002. 2 Oct. 2015 <http://sites.computer.org/ccse/>
"Computing Curricula 2001 Computer Science - ACM." 2015. 2 Oct. 2015 <https://www.acm.org/education/curric_vols/cc2001.pdf>

2.1 Generales

Código	Descripción
CG1	Capacidad para concebir, redactar, organizar, planificar, desarrollar y firmar proyectos en el ámbito de la ingeniería en informática que tengan por objeto, de acuerdo con los conocimientos adquiridos según lo establecido en las competencias de formación especificadas a continuación en esta sección de la memoria, la concepción, el desarrollo o la explotación de sistemas, servicios y aplicaciones informáticas.
CG3	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.
CG5	Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones informáticas empleando los métodos de la ingeniería de software como instrumento para el aseguramiento de su calidad, de acuerdo con los conocimientos adquiridos según lo establecido en las competencias deformación especificadas a continuación en esta sección de la memoria
CG6	Capacidad para concebir y desarrollar sistemas o arquitecturas informáticas centralizadas o distribuidas integrando hardware, software y redes de acuerdo con los conocimientos adquiridos según lo establecido en las competencias de formación especificadas a continuación en esta sección de la memoria.
CG9	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 en Informática.

2.2 Específicas

TI1.	Capacidad para comprender el entorno de una organización y sus necesidades en el ámbito de las tecnologías de la información y las comunicaciones.
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.
TI3.	Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.
IS3.	Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.
IS4.	Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.

3. Objetivos

- Conocer los principios y técnicas del análisis de requisitos y del modelado de sistemas software.
- Conocer los principios y conceptos fundamentales de la arquitectura y el diseño de sistemas software.
- Ser capaces de modelar y diseñar la arquitectura de un producto software ajustándose a un conjunto de requisitos funcionales y no funcionales
- Ser capaces de aplicar patrones arquitectónicos y de diseño en el proceso de desarrollo de aplicaciones informáticas.

4. Contenidos y/o bloques temáticos

Bloque 1: Modelado y análisis de requisitos

Carga de trabajo en créditos ECTS: 2

a. Contextualización y justificación

En la asignatura “Fundamentos de Ingeniería del Software” se proporciona a los estudiantes los principios, métodos y herramientas necesarios para llevar a cabo un análisis básico de requisitos. En este primer bloque se completará la formación en análisis de requisitos de forma que los estudiantes puedan abordar el análisis de sistemas de cierto nivel de complejidad. En la primera parte del bloque se presenta el marco conceptual en el que se desarrollará el resto de la materia. Esta primera parte se completa con una introducción al modelado de sistemas software y aquellos elementos del lenguaje de modelado UML no vistos en la asignatura de “Fundamentos de Ingeniería del Software”. La segunda parte del bloque se centra en el modelado de análisis, estructural y del comportamiento, de un sistema.

b. Objetivos de aprendizaje

- Conocer los fundamentos y principios del modelado en la ingeniería del software.
- Conocer los principios y técnicas avanzadas del análisis de requisitos y de la especificación de sistemas software.

c. Contenidos

PARTE TEÓRICA

1. Modelado y diseño de software.
    a. Introducción
    b. Conceptos de la arquitectura y el diseño
    c. Modelado de sistemas software
    d. El lenguaje de modelado UML 2.x
2. Especificación y análisis de requisitos.
    a. Introducción.
    b. Modelado de la estructura.
    c. Modelado del comportamiento.

PARTE PRÁCTICA. – Lab 1.

- Elaboración y especificación de modelos del dominio con UML
- Elaboración y especificación de modelos de análisis de requisitos con UML

d. Métodos docentes

Actividad	Metodología
Clase de teoría	• Clase magistral participativa • Resolución de ejercicios
Clase práctica	• Realización de ejercicios guiados por el profesor. • Realización de un proyecto guiado por el profesor.

e. Plan de trabajo

ACTIVIDADES PRESENCIALES	HORAS	ACTIVIDADES NO PRESENCIALES	HORAS
Clases teórico-prácticas (T/M)	10	Estudio y trabajo autónomo individual	20
Práctica de laboratorio	10	Estudio y trabajo autónomo grupal	10
	20		30

f. Evaluación

Ver apartado 7, Sistema y características de la evaluación.

g Material docente

g.1 Bibliografía básica

- Sommerville, Ian. Software Engineering. 9th ed., Pearson. (https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991004898409705774)
- Maciaszek, Leszek. Requirements Analysis and System Design: Developing Information Systems with UML. Addison-Wesley. (https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991006838819705774)

g.2 Bibliografía complementaria

- Arlow, Jim, and Ila Neustadt. UML 2. Anaya Multimedia. (https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991004944729705774)
- Booch, Grady, et al. The Unified Modeling Language User Guide. 2a ed., Addison Wesley. (https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991002288959705774)
- Larman, Craig. Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design and the Unified Proces . 3a ed. Prentice-Hall. (https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991001167589705774)

g.3 Otros recursos telemáticos (píldoras de conocimiento, blogs, videos, revistas digitales, cursos masivos (MOOC), …)

- https://www.omg.org/spec/UML/2.5.1/PDF

h. Recursos necesarios

- Aula virtual – Se utilizará el campus virtual de la Uva (https://campusvirtual.uva.es/) como herramienta de gestión, comunicación y soporte para el desarrollo y el seguimiento de la asignatura.
- Laboratorio. Herramienta CASE para el desarrollo de la práctica de laboratorio.

i. Temporalización

CARGA ECTS	PERIODO PREVISTO DE DESARROLLO
2 ECTS	Semanas 1 a 5

Bloque 2: Arquitecturas y diseño de aplicaciones software.

Carga de trabajo en créditos ECTS: 4

a. Contextualización y justificación

La arquitectura y el diseño software son dos materias fundamentales en la formación en ingeniería del software de los graduados en informática. El *Computing Curriculum Software Engineering*[3] recomienda el diseño y la arquitectura software como una de sus diez áreas esenciales. El ACM Computing Curricula 2004 recomienda el diseño software tanto para la especialización en ingeniería del software como en la especialización en Computer Science.

En este bloque se estudiarán los principios, métodos y técnicas que facilitan la concepción y el desarrollo de la arquitectura y el diseño software. Los contenidos estarán enfocados a proporcionar al estudiante una base que le permita adaptarse rápidamente a cualquier plataforma tecnológica. Esta base de conocimientos se complementará con una visión de los métodos y técnicas actuales y de mayor difusión en la industria del software. Se trabajará, desde un enfoque coherente e integrado en las técnicas, métodos y herramientas relacionadas con el modelado y el diseño de aplicaciones software.

b. Objetivos de aprendizaje

- Conocer los principios y conceptos fundamentales de la arquitectura y el diseño de sistemas software.
- Ser capaces de modelar y diseñar la arquitectura de sistemas software, ajustándose a un conjunto de requisitos funcionales y no funcionales
- Ser capaces de aplicar patrones arquitectónicos y de diseño en el proceso de desarrollo de aplicaciones informáticas

c. Contenidos

PARTE TEÓRICA

1. Arquitectura Software.
    a. Introducción
    b. Estilos y patrones arquitectónicos.
    c. Arquitecturas de referencia.
2. Diseño de software.
    a. Introducción.
    b. Diseño arquitectónico.

[3] http://sites.computer.org/ccse/

c. Diseño detallado.

PARTE PRÁCTICA – Lab 2.

- Elaboración y especificación de modelos de la arquitectura software con UML
- Elaboración y especificación de modelos de diseño software con UML

d. Métodos docentes

Actividad	Metodología
Clase de teoría	• Clase magistral participativa • Resolución de ejercicios
Clase práctica	• Realización de ejercicios guiados por el profesor. • Realización de un proyecto guiado por el profesor.

e. Plan de trabajo

ACTIVIDADES PRESENCIALES	HORAS	ACTIVIDADES NO PRESENCIALES	HORAS
Clases teórico-prácticas (T/M)	18	Estudio y trabajo autónomo individual	40
Práctica de laboratorio	18	Estudio y trabajo autónomo grupal	20
Tutoría activa	4		
	40		60

El cronograma de actividades aparecerá en el Aula Virtual o página web asociada a la asignatura

f. Evaluación

Ver apartado 7, Sistema y características de la evaluación.

g Material docente

g.1 Bibliografía básica

- Gomaa, Hassan. Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures. Cambridge University Press. (https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991008118593205774)
- Larman, Craig. Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design and the Unified Proces . 3a ed. Prentice-Hall. (https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991001167589705774)

g.2 Bibliografía complementaria

- Arlow, Jim, and Ila Neustadt. UML 2. Anaya Multimedia. (https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991004944729705774)
- Booch, Grady, et al. The Unified Modeling Language User Guide. 2a ed., Addison Wesley. (https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991002288959705774)

- Buschmann, Frank. Pattern-Oriented Software Architecture: a System of Patterns. John Wiley & Sons. (https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991007057859705774)
- Fowler, Martin. Patterns of Enterprise Application Architecture . 21st. print., Addison-Wesley. (https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991001008869705774)

g.3 Otros recursos telemáticos (píldoras de conocimiento, blogs, videos, revistas digitales, cursos masivos (MOOC), …)

h. Recursos necesarios

- Aula virtual – Se utilizará el campus virtual de la Uva (https://campusvirtual.uva.es/) como herramienta de gestión, comunicación y soporte para el desarrollo y el seguimiento de la asignatura.
- Laboratorio. Herramienta CASE para el desarrollo de la práctica de laboratorio.

i. Temporalización

CARGA ECTS	PERIODO PREVISTO DE DESARROLLO
4 ECTS	Semanas 6 a 14
	La semana 15 se dedicará a la resolución de dudas, desarrollo de tutorías activas y entregas de prácticas.

5. Métodos docentes y principios metodológicos

Actividad	Metodología
Clase de teoría	• Clase magistral participativa • Estudio de casos en aula • Resolución de ejercicios
Clase práctica	• Realización de ejercicios y pequeños supuestos prácticos. • Realización de un proyecto guiado por el profesor. La realización de una parte del proyecto se llevará a cabo en grupos de 2/3 alumnos siguiendo un enfoque colaborativo; el resto del proyecto será desarrollado por cada alumno de forma individual.
Tutoría	• Evaluación de los contenidos teóricos y de los proyectos

6. Tabla de dedicación del estudiante a la asignatura

ACTIVIDADES PRESENCIALES o PRESENCIALES A DISTANCIA(1)	HORAS	ACTIVIDADES NO PRESENCIALES	HORAS
Clases teórico-prácticas (T/M)	28	Estudio y trabajo autónomo individual	60
Clases prácticas de aula (A)		Estudio y trabajo autónomo grupal	30
Laboratorios (L)	28		
Prácticas externas, clínicas o de campo			
Tutorías grupales (TG)	4		
Evaluación			
Total presencial	**60**	Total no presencial	**90**
		TOTAL presencial + no presencial	**150**

(1) Actividad presencial a distancia es cuando un grupo sigue una videoconferencia de forma síncrona a la clase impartida por el profesor.

7. Sistema y características de la evaluación

INSTRUMENTO/PROCEDIMIENTO	PESO EN LA NOTA FINAL	OBSERVACIONES
Trabajo evaluable de laboratorio	40%	Varias entregas en evaluación continua a lo largo del curso.
Examen final	60%	Periodo de exámenes

CRITERIOS DE CALIFICACIÓN

Convocatorias ordinaria y extraordinaria:

- El examen final se realizará mediante examen escrito sobre las materias incluidas en el programa de la asignatura. Incluirá la elaboración de modelos de análisis y diseño correspondientes a supuestos prácticos. Se valorará el ajuste de la solución elaborada al problema planteado y la correcta utilización de los principios y notaciones de modelado y diseño.
- La evaluación de la parte práctica se realizará sobre el trabajo de laboratorio realizado en grupo a lo largo del curso académico. La evaluación de la parte práctica, al tener carácter de evaluación continua, solamente se puede realizar durante el periodo de actividad docente.
    - En la **convocatoria extraordinaria** se podrá entregar una nueva versión de aquellos trabajos ya presentados a la convocatoria ordinaria y que no hayan obtenido la calificación mínima establecida. La entrega y posible defensa de la nueva versión del trabajo desarrollado se realizará antes de la fecha de examen.
    - La calificación de estas nuevas versiones de los trabajos elaborados no estará condicionada por la calificación que se hubiese obtenido en la versión anterior.
- Para aprobar la asignatura es necesario haber aprobado la práctica de laboratorio y el examen final.
    - Para la práctica de laboratorio, 2,0 puntos sobre 4.
    - Para el examen final de teoría, 3,0 puntos sobre 6.
- Para establecer la calificación final de un alumno se tiene en cuenta la calificación obtenida en la práctica de laboratorio y la obtenida en el examen escrito de teoría.
    - Sea NTotal = NPL+ NEF;
        - NPL es la nota de la práctica de laboratorio.
        - NEF la calificación del examen final.

La nota final en la asignatura, NFinal, para un alumno será:

- NFinal=NTotal, si se cumple la condición 3, ha obtenido la calificación mínima en la práctica de laboratorio y en el examen final.
- NFinal = min (4,0, NTotal), en caso contrario.
- Si algún trabajo no ha sido entregado y/o el alumno no se ha presentado al examen final, la calificación será “No Presentado”.

8. Consideraciones finales

La realización fraudulenta cualquiera de las pruebas de evaluación o de los trabajos de laboratorio (copia o trabajos no originales), supondrá automáticamente una calificación de SUSPENSO con una nota de 0.0 puntos en el acta de la asignatura.