Proyecto/Guía docente de la asignatura

Asignatura	Programación II		
Materia	Programación		
Módulo			
Titulación	Grado en Ingeniería en Tecnologías de la Telecomunicación		
Plan	727	**Código**	48074
Periodo de impartición	Segundo cuatrimestre	**Tipo/Carácter**	Obligatoria
Nivel/Ciclo	Grado	**Curso**	Primero
Créditos ECTS	6 ECTS		
Lengua en que se imparte	Español		
Profesor/es responsable/s	Guillermo Vega Gorgojo		
Datos de contacto (E-mail, teléfono…)	Dr. Guillermo Vega Gorgojo E-mail: guiveg@tel.uva.es, Tel: 983 185 538 Dirección: ETSIT, Paseo de Belén, 15, 47011, Valladolid Perfil investigador (Scholar): https://scholar.google.es/citations?user=jcmyWIgAAAAJ&hl=es&oi=ao Dr. Mohamed Saban E-mail: mohamed.saban@uva.es, Tel: 983 423 696 / 698 Dirección: ETSIT, Paseo de Belén, 15, 47011, Valladolid Perfil investigador (Scholar): https://scholar.google.com/citations?user=Z5sc4s8AAAAJ&hl=en&oi=sra		
Departamento	Teoría de la Señal y Comunicaciones e Ingeniería Telemática		
Fecha de revisión por el Comité de Título	8/7/2024		

1. Situación / Sentido de la Asignatura

1.1 Contextualización

Al enfrentarse a esta asignatura, puede que el alumno tenga aún la idea preconcebida de que hacer software es, fundamentalmente, escribir código. Un planteamiento inicial de ese tipo suele estar destinado al fracaso en proyectos de cierta envergadura—las recomendaciones de distribución del esfuerzo en el desarrollo del código de un proyecto software suelen ser de un 15–20% del esfuerzo global.

Tras una profunda crisis en la industria del software en los años 70 y 80, acompañada de un sustancial incremento en la complejidad del software a desarrollar, se hizo evidente la necesidad de contar con nuevas técnicas que permitieran el paso de la producción de software de manera artesanal a un proceso de ingeniería. Así, la IEEE define la ingeniería de software como la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software. Por tanto, la ingeniería de software es la disciplina que se ocupa del software, enfrentándose al mismo como un producto de ingeniería que requiere planificación, análisis, diseño, implementación, pruebas y mantenimiento.

Desarrollar software es, por tanto, mucho más que escribir código, requiriendo un esfuerzo previo para diseñarlo. Por este motivo, en *Programación II* el énfasis se pone no tanto en el producto final (el software como un producto), como en su proceso de desarrollo (el software como un proceso). El proceso del software define el enfoque que se aplica cuando el software es tratado utilizando una aproximación ingenieril, tal y como hace la ingeniería de software. El proceso que se seguirá en la asignatura es uno de los más extendidos, el denominado Proceso Unificado (*Unified Process*) de desarrollo de software. Además, se empleará como notación el Lenguaje de Modelado Unificado (UML: *Unified Modelling Language*), el lenguaje *de facto* para documentar proyectos software. En cuanto a las herramientas, se empleará una herramienta CASE (*Computer Aided Software Engineering*) que permita elaborar diagramas en UML, un entorno integrado de desarrollo (IDE: *Integrated Development Environment*) para el lenguaje de programación Java y el sistema de control de versiones Git.

El Proceso Unificado de desarrollo de software y el Lenguaje de Modelado Unificado serán el hilo conductor de buena parte de las sesiones teóricas y prácticas, en el que los estudiantes realizarán un proyecto software siguiendo este proceso y preparando entregables típicos del Proceso Unificado—en muchos casos correspondientes a un diagrama UML. Se empleará la orientación a objetos en dicho proyecto, al tratarse del paradigma de programación dominante en el panorama actual del desarrollo del software. En la asignatura se hará especial énfasis en el análisis y diseño orientado a objetos, utilizando patrones de software para realizar un diseño orientado a objetos efectivo. Se usará el lenguaje de programación Java para la implementación, al tratarse de uno de los lenguajes más empleados en la industria y de especial utilidad para la programación en redes.

Además, y puesto que la creación de un producto software de cierta relevancia es siempre una tarea en equipo, se trabajará bajo dicha perspectiva. Así, la asignatura está diseñada siguiendo el método de aprendizaje por proyectos, de modo que los alumnos trabajarán en grupo para llevar a cabo un proyecto de un sistema software. Se trata de sentar las bases para abordar proyectos complejos de desarrollo software en equipo, lo cual podrá ser útil al alumno tanto en otras asignaturas de la titulación como en su Trabajo Fin de Grado o en su carrera profesional posterior. En este sentido no debe perderse de vista que el ingeniero de software es un perfil profesional ampliamente demandado por el mercado laboral tanto en España como fuera de nuestro país y que

la industria del software tiene una gran importancia, conviviendo gigantes tecnológicos como Google o Microsoft con numerosas microempresas de menos de 10 trabajadores.

1.2 Relación con otras materias

Esta asignatura se apoya en la asignatura *Programación I* de la materia *Programación* que se imparte en el primer cuatrimestre del primer curso de esta titulación.

Por otra parte, dado que *Programación II* sienta las bases de la ingeniería de software y se lleva a cabo un proyecto de un sistema software, se recomienda fuertemente haberla cursado antes de abordar las asignaturas posteriores del plan de estudios que están relacionadas con la construcción de software—especialmente si se sigue el paradigma de programación orientada a objetos y se utiliza el lenguaje de programación Java, tratados en *Programación II*.

1.3 Prerrequisitos

Aunque no existen requisitos previos para matricularse en *Programación II*, se recomienda haber cursado previamente la asignatura *Programación I* de la materia *Programación* que se imparte en el primer cuatrimestre del primer curso de esta titulación. En dicha asignatura se introducen conceptos básicos de programación y algorítmica utilizando el lenguaje de programación C.

2. Resultados del proceso de formación y de aprendizaje

2.1 Conocimientos o contenidos

Los conocimientos o contenidos que se adquirirán con esta asignatura son los siguientes:

C3 Conocer y aplicar lenguajes de programación para desarrollar aplicaciones software y conocer alguna metodología de ingeniería de software relevante.

2.2 Habilidades o destrezas

Las habilidades o destrezas que se adquirirán con esta asignatura son las siguientes:

HD6 Capacidad de razonamiento, análisis y síntesis.

HD7 Capacidad para relacionar conceptos y adquirir una visión integrada, evitando enfoques fragmentarios.

HD8 Capacidad de toma de decisiones en la resolución de problemas básicos de ingeniería de telecomunicación, así como identificación y formulación de los mismos.

HD9 Capacidad para trabajar en grupo, participando de forma activa, colaborando con sus compañeros y trabajando de forma orientada al resultado conjunto, y en un entorno multilingüe.

HD10 Conocimiento de materias básicas, científicas y tecnologías, que le capacite para el aprendizaje de nuevos métodos y tecnologías.

HD15 Capacidad para resolver problemas con iniciativa, creatividad y razonamiento crítico.

HD24 Capacidad de organización, planificación y gestión del tiempo.

HD25 Capacidad para comunicar, tanto por escrito como de forma oral, conocimientos, procedimientos, resultados e ideas relacionadas con las telecomunicaciones y la electrónica.

HD26 Capacidad para trabajar en cualquier contexto, individual o en grupo, de aprendizaje o profesional,

local o internacional, desde el respeto a los derechos fundamentales, de igualdad de sexo, raza o religión y los principios de accesibilidad universal, así como la cultura de paz.

2.3 Competencias

Las competencias que se adquirirán con esta asignatura son las siguientes:

B2 Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.

3. Objetivos

El objetivo general de la asignatura puede formularse de la siguiente manera:

OG Comprender los principios, conceptos y métodos de ingeniería de software, y en particular del Proceso Unificado, para organizar, diseñar, desarrollar y documentar un proyecto de un sistema software orientado a objetos de tamaño mediano.

Tras definir el objetivo general de *Programación II*, se especifican los siguientes objetivos específicos:

OE1 Aplicar la metodología del Proceso Unificado para planificar y organizar proyectos de software.
OE2 Analizar y formalizar los requisitos de software de un proyecto.
OE3 Realizar el análisis y el diseño orientado a objetos de un sistema software.
OE4 Aplicar patrones de software como instrumento principal en el diseño orientado a objetos.
OE5 Programar en el lenguaje Java un sistema software a partir de un diseño orientado a objetos.
OE6 Documentar un sistema software utilizando el lenguaje UML como notación fundamental.
OE7 Trabajar en equipo de manera efectiva para la construcción de un sistema software.
OE8 Utilizar el sistema de control de versiones Git para desarrollar software de manera colaborativa.

4. Contenidos y/o bloques temáticos

Bloque 1: Programación II

Carga de trabajo en créditos ECTS: 6

a. Contextualización y justificación

Véase el apartado 1.1 (justificación de la asignatura).

b. Objetivos de aprendizaje

Véase el apartado 3 (objetivos de la asignatura).

c. Contenidos

Tema 1: Introducción a Git

1.1 Sistemas de control de versiones
1.2 Aspectos básicos de Git
1.3 Comandos de Git

Tema 2: Introducción a la ingeniería de software

2.1 El software como producto
2.2 El proceso de creación del software
2.3 El Proceso Unificado (UP)
2.4 El Lenguaje Unificado de Modelado (UML)

Tema 3: Principios de orientación a objetos

3.1 Introducción a la orientación a objetos
3.2 Clases y objetos
3.3 Encapsulamiento
3.4 Herencia y polimorfismo

Tema 4: El lenguaje de programación Java

4.1 Fundamentos del lenguaje y de la plataforma
4.2 Elementos del lenguaje Java
4.3 Clases y objetos en Java
4.4 Tratamiento de excepciones
4.5 Colecciones y mapas

Tema 5: Análisis de requisitos

5.1 Requisitos de un proyecto software
5.2 Los casos de uso
5.3 Modelos de dominio
5.4 Prototipos de la interfaz de usuario

Tema 6: Diseño orientado a objetos

6.1 Diseño arquitectónico de un sistema software
6.2 Diagrama de clases de diseño
6.3 Diagramas de interacción
6.4 Patrones de software

d. Métodos docentes

- Clase magistral participativa.
- Aprendizaje colaborativo.
- Aprendizaje basado en proyectos.

e. Plan de trabajo

Plan detallado en el Anexo I. Se presenta aquí una planificación orientativa de la asignatura.

- **Clases teóricas**

Las clases teóricas (20 horas) se distribuyen a lo largo del curso según la siguiente tabla.

Temas	Duración aproximada (horas presenciales)	Periodo previsto de desarrollo
Tema 1: Introducción a Git	2 horas	Semana 1
Tema 2: Introducción a la ingeniería de software	1 hora	Semana 2
Tema 3: Principios de orientación a objetos	3 horas	Semanas 2–3
Tema 4: El lenguaje de programación Java	4 horas	Semanas 4–5
Tema 5: Análisis de requisitos	2 horas	Semana 6
Tema 6: Diseño orientado a objetos	6 horas	Semanas 7–9
Repaso de la asignatura	2 horas	Semana 15

- **Laboratorios**

Las sesiones de laboratorio (40 horas) se distribuyen a lo largo del curso, como se describe en la siguiente tabla. Las prácticas se abordan en el marco de un proyecto que se va a desarrollar a lo largo de todo el curso.

Prácticas	Duración aproximada (horas presenciales)	Periodo previsto de desarrollo
Práctica 0: Introducción a Git y Eclipse	4 horas	Semanas 2–3
Práctica 1: Primeras clases en Java y modelo de dominio	12 horas	Semanas 4–7
Práctica 2: Iteración I de diseño e implementación	12 horas	Semanas 8–11
Práctica 3: Iteración II de diseño e implementación	12 horas	Semanas 12–15

f. Evaluación

La evaluación de los resultados de aprendizaje se basará en:

- Las entregas de las prácticas realizadas en grupo por los alumnos.
- El examen final de la asignatura.

g Material docente

g.1 Bibliografía básica

- Craig Larman. Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design

and Iterative Development (tercera edición). Prentice Hall, Upper Saddle River, NJ, EEUU, 2005.

También disponible en la plataforma Leganto.

g.2 Bibliografía complementaria

- Roger S. Pressman y Bruce R. Maxim. Software engineering: a practitioner's approach (novena edición). McGraw-Hill, Columbus, OH, EEUU, 2019.
- Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Design patterns: elements of reusable object-oriented software. Prentice Hall, Upper Saddle River, NJ, EEUU, 1995.
- Robert Nystrom. Game programming patterns. Genever Benning, Exeter, Reino Unido, 2014.
- Martin Fowler. UML distilled: a brief guide to the standard object modeling language (tercera edición). Addison-Wesley, Boston, MA, EEUU, 2003.
- Timothy Budd. An introduction to object-oriented programming (tercera edición). Addison Wesley, Boston, MA, EEUU, 2002.
- Benjamin J. Evans y David Flanagan. Java in a Nutshell: A Desktop Quick Reference (séptima edición). O'Reilly, Sebastopol, CA, EEUU, 2019.
- Paul Deitel y Harvey Deitel. Java How to Program, Early Objects (undécima edición). Pearson, NY, NY, EEUU, 2017.
- Scott Chacon y Ben Straub. Pro Git (segunda edición). Apress, 2014.
- Mike Cohn. Succeeding with agile: software development using Scrum. Addison-Wesley, Boston, MA, EEUU, 2009.

Esta lista también está disponible en la plataforma Leganto.

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

- Materiales de la asignatura preparados por el profesor y publicados en el Campus Virtual.
- Tutoriales de Java ofrecidos por Oracle.
- Foro de preguntas para programadores StackOverflow.
- Java Notes for Professionals.

h. Recursos necesarios

Se proporcionarán los siguientes recursos para llevar a cabo la asignatura:

- Aula con proyector multimedia y pizarra para las clases magistrales participativas.
- Campus Virtual de la Universidad de Valladolid para distribuir materiales, realizar entregas y servir de canal de comunicación, principalmente a través de foros.
- Laboratorio de prácticas, con un ordenador para cada alumno, para las sesiones de laboratorio. Cada ordenador contará con una herramienta CASE que permita construir artefactos UML, con el IDE Eclipse y con el sistema de control de versiones Git.
- Documentación de apoyo.

i. Temporalización

CARGA ECTS	PERIODO PREVISTO DE DESARROLLO
6 ECTS	Semanas 1 a 15

5. Métodos docentes y principios metodológicos

- Clase magistral participativa.
- Aprendizaje colaborativo.
- Aprendizaje basado en proyectos.

6. Tabla de dedicación del estudiante a la asignatura

ACTIVIDADES PRESENCIALES o PRESENCIALES A DISTANCIA(1)	HORAS	ACTIVIDADES NO PRESENCIALES	HORAS
Clases teóricas	20	Estudio y trabajo autónomo individual	45
Laboratorios	40	Estudio y trabajo autónomo grupal	45
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
Informes del proyecto	60%	Es condición necesaria (pero no suficiente) para superar la asignatura entregar todos los informes de laboratorio y alcanzar una calificación igual o superior a 5 puntos sobre 10. Si los alumnos no entregan un informe de los solicitados, la calificación correspondiente será de 0 y el proyecto se considerará no superado.
Examen final	40%	Es condición necesaria (pero no suficiente) para superar la asignatura alcanzar una calificación igual o superior a 5 puntos sobre 10 en este examen. En caso de no realizarse este examen, la calificación de esta prueba será 0.

CRITERIOS DE CALIFICACIÓN

- **Convocatoria ordinaria:**
  - La nota final para superar la asignatura en la convocatoria ordinaria deberá ser al menos de 5,0 sobre 10,0.
  - Si un alumno no alcanza los requisitos mínimos descritos en la tabla anterior, su calificación final en la asignatura será el mínimo entre el valor calculado según la ponderación de la tabla anterior y 4,5.
- **Convocatoria extraordinaria(*):**
  - Los alumnos que no hayan superado el proyecto (primer instrumento de la tabla anterior) tendrán que realizar un nuevo proyecto de similares características. Los alumnos deberán contactar con los profesores para obtener el nuevo enunciado.
  - Los alumnos que no hayan superado el examen final (segundo instrumento de la tabla anterior) tendrán que realizar el examen de la convocatoria extraordinaria.

- Al igual que en la convocatoria ordinaria, si un alumno no alcanza los requisitos mínimos establecidos, su calificación final en la asignatura será el mínimo entre el valor calculado según la ponderación descrita y 4,5.

(*) Se entiende por convocatoria extraordinaria la segunda convocatoria.
Art 35.4 del ROA 35.4. La participación en la convocatoria extraordinaria no quedará sujeta a la asistencia a clase ni a la presencia en pruebas anteriores, salvo en los casos de prácticas externas, laboratorios u otras actividades cuya evaluación no fuera posible sin la previa realización de las mencionadas pruebas.
https://secretariageneral.uva.es/wp-content/uploads/VII.2.-Reglamento-de-Ordenacion-Academica.pdf

8. Consideraciones finales

El Anexo I (plan de trabajo detallado) mencionado en la guía se publicará al comienzo de la asignatura.