UNIVERSIDAD PONTIFICIA ICAI ICADE COMILLAS MADRID

COURSE SYLLABUS 2016-2017

GENERAL INFORMATION

Course information	
Name	Advanced Computing Tools for Applied Research
Code	MRE-525
Degree	Master's Degree in Research in Engineering Systems Modeling (MRE)
Year	Even years
Semester	2nd (Spring)
ECTS credits	3 ECTS
Type	Elective
Department	
Area	
Coordinator	Rafael Palacios Hielscher

Instructor	
Name	Rafael Palacios Hielscher
Department	Telematics and Computer Sciences
Area	
Office	Dirección (Alberto Aguilera, 25)
e-mail	Rafael.Palacios@iit.comillas.edu
Phone	91.542.28.00 – Ext. 2758
Office hours	Arrange an appointment through email.

Instructor	
Name	Jaime Boal Martín-Larrauri
Department	Electronics, Control and Communications
Area	
Office	D-220 (Alberto Aguilera, 25)
e-mail	jaime.boal@iit.comillas.edu
Phone	91.542.28.00 – Ext. 2742
Office hours	Arrange an appointment through email.

DETAILED INFORMATION

Contextualization of the course

Contribution to the professional profile of the degree

Any engineering research activity involves the use of computers for running optimization, simulation or other computing algorithms. This course is not focused on any particular programming language, it provides mostly general knowledge about software development techniques and tools, and method to measure and improve performance.

Prerequisites

Previous experience in software development is required, but not in a particular programming language.

CONTENTS

Contents
Theory
Chapter 1. Introduction to software engineering
1.1 The software development process 1.2 System modeling: Unified Modeling Language (UML)
Chapter 2. Coding conventions
2.1 Naming conventions 2.2 Statements 2.3 Code layout
Chapter 3. Source code documentation
3.1 Types of comments 3.2 Commenting guidelines 3.3 Automatic documentation tools
Chapter 4. Version control
4.1 What is version control? 4.2 Types of version control systems 4.3 Git 4.4 Repository hosting services
Chapter 5. Design of user interfaces
5.1 Introduction 5.2 Interface design 5.3 Web-based applications
Chapter 6. Computers and programming languages
6.1 Brief history of computers 6.2 Storage systems 6.3 Programming approaches 6.4 Computer architectures
Chapter 7. Reliability and performance
7.1 Software Reliability 7.2 Software Performance 7.3 Performance tools 7.4 Performance examples
Chapter 8. Management of large data volumes
8.1 Introduction 8.2 Database description 8.3 Database management systems 8.4 SQL 8.5 Storage engines
Chapter 9. Inter-process communications
9.1 Introduction 9.2 Unix command line 9.3 File-based data exchange 9.4 Inter-process communications 9.5 Database connection 9.6 AJAX approach

Chapter 10. High-performance computing
10.1 High Performance Computing overview 10.2 Parallel Performance 10.3 Parallel Programming 10.4 Parallel MATLAB
Laboratory
Lab 1. Unified Modeling Language (UML)
In this first lab session, students will gain a deeper insight into system modeling using UML diagrams.
Lab 2. Coding conventions and source code documentation
The aim of this practice is to put into practice the programming style guidelines presented in the lectures.
Lab 3. Introduction to Git version control
In the third lab session students will become familiar with the basic Git workflow. They will learn to create and manage a source code repository both through a web page interface and a desktop client.
Lab 4. Interfaces
The objective of this Lab session is to use different tools for interface design
Lab 5. Performance
In this session students will measure the performance of code in different environments: C, MATLAB, web pages… Changes in the code will be made to analyze the impact in performance.
Lab 6. Databases
Students will start by creating several tables in a database and executing deletes in cascade. The impact of indexes in performance will also be analyzed.
Lab 7. Inter-process communications
Several methods of inter-process communications will be tested: MATLAB, command line, and AJAX.
Lab 8. High-performance computing
This session will start by measuring the performance a single computer, and it will finish by using a professional cloud infrastructure such as Microsoft's Azure.
Final project
In the final project students will have to apply several of the concepts covered during the course to a research problem that involves software development. A project proposal will have to be submitted and approved by the instructors.

Competences and learning outcomes

Competences[1]

Basic and general competences

CB1. Having acquired advanced knowledge and demonstrated, in the context of scientific, technological or highly specialized research, a solid and specialized understanding of theoretical and practical aspects as well as the work methods in one or more fields of study.

CB3. Knowing how to evaluate and select the most appropriate scientific theoretical framework and methodology from their fields of study to formulate opinions on the basis of incomplete or limited information including, when necessary and appropriate, their views on social responsibility issues or the ethical repercussions linked to the solution proposed in each case.

CB4. Being able to predict and control the evolution of complex situations by developing new and innovative work methodologies adapted to the specific scientific/research, technological or professional field, generally multidisciplinary, in which they work.

CB5. Being able to communicate findings from scientific and technological research, including those of a highly innovative nature, as well as the most relevant foundations on which these are based to a specialized or lay audience in a clear and unambiguous manner.

CB7. Being able to take responsibility for their own professional development and specialization in one or more fields of expertise.

Specific competences

CE7. Knowing how to use the essential tools to cover a research topic.

CE9. Mastering the techniques, methods and/or tools needed to cover a research topic in a specific sector or technological context.

Additional competences

CO1. Acquire the knowledge needed to design advanced computing tool efficiently.

CO2. Get a general overview of the different computing tools available, and the knowledge to select the one most suitable for each type of problem.

CO3. Learn to identify performance problems, to know strategies for improving computing times, to know how to estimate expected performance improvements, and finally to know how to present a proposal of changes.

Learning outcomes

By the end of the course students should be able to:

RA1. Enumerate the main aspects that characterize computing tools for applied research.

RA2. Conceive and design complex computing tools, able to manage large data volumes.

RA3. Design computing tools that merge different technologies and communicate with other systems.

RA4. Design data structures useful for working efficiently with large volumes of data and for implementing advanced analysis on the data.

RA5. Develop procedures to conduct systematic analysis of different implementation approaches in order to determine which one is better, being able to select the most suitable technique for each type of problem.

[1] Competences in English are a free translation of the Spanish version extracted from the degree's Official Verification Report.

TEACHING METHODOLOGY

General methodological aspects	
The best way of gaining a full understanding of any computing tool is experimenting with it. Consequently, all the proposed activities focus on providing students with the hands-on experience they require to be able to successfully develop a research oriented computing tool by the end of the term.	
In-class activities	**Competences**
▪ **Lectures (12 hours)**: The lecturer will introduce the fundamental concepts of each chapter, along with some practical recommendations, and will go through worked examples to support the explanation. Active participation will be encouraged by raising open questions to foster discussion and by proposing short application exercises to be solved in class.	CB1, CB3, CE7, CE9, CO1, CO2, CO3
▪ **Lab sessions (10 hours):** Under the instructor's supervision, students, divided in small groups, will apply the concepts and techniques covered in the lectures to real problems and will become familiar with the most widespread software tools.	CB1, CB3, CE7, CE9, CO1, CO2, CO3
▪ **Tutoring** for groups or individual students will be organized upon request.	–
Out-of-class activities	**Competences**
▪ Personal study of the course material (20 hours).	CB1, CB3, CE7, CE9, CO1, CO2, CO3
▪ Lab results analysis and report writing (10 hours).	CB1, CB3, CE7, CE9, CO1, CO2, CO3
▪ Development of a final project during the last third of the course (30 hours).	CB1, CB3, CB4, CB7, CE7, CE9, CO1, CO2, CO3

ASSESSMENT AND GRADING CRITERIA

Assessment activities	Grading criteria	Weight
Final exam	▪ Understanding of the theoretical concepts.	60%
Lab reports	▪ Application of theoretical concepts. ▪ Ability to develop robust, efficient and clear software. ▪ Written communication skills.	
Final project	▪ Problem analysis. ▪ Information search skills. ▪ Difficulty. ▪ Quality of the proposed solution. ▪ Oral presentation skills.	40%

GRADING AND COURSE RULES

Grading

Regular assessment

- Knowledge and participation will account for 60%. This includes:
  - Practical exercises
  - Active learning attitude
  - Final exam
- The remaining 40% corresponds to a final project that will be evaluated based on:
  - The quality of the work
  - The quality of the oral presentation

Course rules

- Class attendance is mandatory according to Article 93 of the General Regulations (Reglamento General) of Comillas Pontifical University and Article 6 of the Academic Rules (Normas Académicas) of the ICAI School of Engineering. Not complying with this requirement may have the following consequences:
  - Students who fail to attend more than 15% of the in-class hours may be denied the right to take the final exam.
  - Missed laboratory sessions must be made up for credit.
- Students who commit an irregularity in any graded activity will receive a mark of zero in the activity and disciplinary procedure will follow (cf. Article 168 of the General Regulations (Reglamento General) of Comillas Pontifical University).

WORK PLAN AND SCHEDULE[2]

In and out-of-class activities	Date/Periodicity	Deadline
Final exam	May	
Lab sessions	Weekly	
Review and self-study of the concepts covered in the lectures	After each lesson	–
Lab report writing	–	One week after the end of each session
Final project	During the last third of the course	Last month
Final exam preparation	May	–

STUDENT WORK-TIME SUMMARY		
IN-CLASS HOURS		
Lectures	**Lab sessions**	**Assessment**
12	10	8
OUT-OF-CLASS HOURS		
Self-study	**Lab report writing**	**Final project**
20	10	30
	ECTS credits:	**3 (90 hours)**

[2] A detailed work plan of the subject can be found in the course summary sheet (see last page). Nevertheless, this schedule is tentative and may vary to accommodate the rhythm of the class.

BIBLIOGRAPHY

Basic bibliography

- Notes prepared by the instructors (available in Moodle).

Complementary bibliography

- I. Sommerville, Software Engineering, 9th Ed., Addison-Wesley, 2011. ISBN-13: 978-0137035151
- P. Roques, UML in Practice: The Art of Modeling Software Systems Demonstrated Through Worked Examples and Solutions, 1st Ed., John Wiley & Sons, 2004.
- S. McConnell, *Code Complete: A Practical Handbook of Software Construction*, 2nd Ed., Microsoft Press, 2004. ISBN-13: 079-0145196705
- A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 6th Ed., McGraw-Hill, 2011. ISBN-13: 978-0073523323
- D. B. Kirk and W. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, 2nd Ed., Morgan Kaufmann, 2013. ISBN-13: 978-0124159921
- Git documentation, [Online]. Available: http://git-scm.com/doc/
- OS X Human Interface Guidelines, Apple Inc., [Online]. Available: https://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/OSXHIGuidelines/
- iOS Human Interface Guidelines, Apple Inc., [Online]. Available: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/
- Android Design, Google Inc., [Online]. Available: https://developer.android.com/design/

	In-class activities				Out-of-class activities				Learning outcomes
Week	**Time [h]**	**Lecture & Problem-solving**	**Laboratory**	**Assessment**	**Time [h]**	**Self-study**	**Lab preparation and report writing**	**Other activities**	**Code**
1	**2**	Course presentation (0.5h) Introduction to software engineering (1.5h)			**1**	Review and self-study (1h)			RA1, RA2, RA3, RA5
2	**2**	Coding conventions (1h)	Lab 1 (1h)		**3**	Review and self-study (2h)	Report writing (1h)		RA2
3	**2**	Source code documentation (1h)	Lab 2 (1h)		**3**	Review and self-study (2h)	Report writing (1h)		RA2
4	**2**	Version control (1h)	Lab 3 (1h)		**3**	Review and self-study (2h)	Report writing (1h)		RA2, RA3
5	**2**	Design of user interfaces (1h)	Lab 4 (1h)		**3**	Review and self-study (2h)	Report writing (1h)		RA2
6	**2**	Computers and programming languages (2h)			**1**	Review and self-study (1h)			RA2, RA3
7	**2**	Reliability and performance (1h)	Lab 5 (1h)		**4**	Review and self-study (1h)	Report writing (1h)	Final project proposal (2h)	RA5
8	**2**		Lab 5 (2h)		**8**		Report writing (2h)	Final project development (6h)	RA5
9	**2**	Management of large data volumes (1h)	Lab 6 (1h)		**8**	Review and self-study (1h)	Report writing (1h)	Final project development (6h)	RA2. RA4
10	**2**	Inter-process communications (1h)	Lab 7 (1h)		**8**	Review and self-study (1h)	Report writing (1h)	Final project development (6h)	RA3
11	**2**	High-performance computing (1h)	Lab 8 (1h)		**8**	Review and self-study (1h)	Report writing (1h)	Final project development (6h)	RA5
12	**2**			Final project presentations (2h)	**4**			Final project presentation preparation (4h)	RA2, RA3, RA4, RA5
13	**2**			Final project presentations (2h)	**2**	Final exam preparation (2h)			RA2, RA3, RA4, RA5
14	**2**			Final project presentations (2h)	**2**	Final exam preparation (2h)			RA2, RA3, RA4, RA5
15	**2**			Final exam (2h)	**2**	Final exam preparation (2h)			RA1, RA2, RA3, RA4, RA5

FICHA TÉCNICA DE LA ASIGNATURA

Datos de la asignatura	
Nombre	Herramientas computacionales avanzadas para la investigación aplicada
Código	MRE-525
Titulación	Master's Degree in Research in Engineering Systems Modeling (MRE)
Curso	Años pares
Cuatrimestre	2º
Créditos ECTS	3 ECTS
Carácter	Optativa
Departamento	
Área	
Coordinador	Rafael Palacios Hielscher

Profesor	
Nombre	Rafael Palacios Hielscher
Departamento	Telemática y Computación
Área	
Despacho	Dirección (Alberto Aguilera, 25)
e-mail	Rafael.Palacios@iit.comillas.edu
Teléfono	91.542.28.00 – Ext. 2758
Tutorías	Concertar cita por correo electrónico.

Profesor	
Nombre	Jaime Boal Martín-Larrauri
Departamento	Electrónica, Automática y Comunicaciones
Área	
Despacho	D-220 (Alberto Aguilera, 25)
e-mail	jaime.boal@iit.comillas.edu
Teléfono	91.542.28.00 – Ext. 2742
Tutorías	Concertar cita por correo electrónico.

DATOS ESPECÍFICOS DE LA ASIGNATURA

Contextualización de la asignatura

Aportación al perfil profesional de la titulación

Cualquier actividad de investigación en ingeniería necesita de ordenadores donde ejecutar programas de optimización, simulaciones y otros algoritmos. Este curso no se centra en un lenguaje de programación en concreto, sino que proporciona nociones generales sobre técnicas y herramientas de uso común en el desarrollo de aplicaciones, así como métodos para medir y mejorar su rendimiento.

Prerrequisitos

Se requiere experiencia previa en el desarrollo de aplicaciones en cualquier lenguaje de programación.

BLOQUES TEMÁTICOS Y CONTENIDOS

Contenidos
Teoría
Tema 1. Introducción a la ingeniería de software
1.1 El proceso de desarrollo de una aplicación 1.2 Modelado de sistemas: Lenguaje de modelado unificado (UML)
Tema 2. Convenciones de código
2.1 Nomenclatura 2.2 Sentencias 2.3 Organización del código
Tema 3. Documentación de código fuente
3.1 Tipos de comentarios 3.2 Criterios para comentar adecuadamente 3.3 Herramientas de documentación automática
Tema 4. Control de versiones
4.1 ¿Qué es el control de versiones? 4.2 Sistemas de control de versiones 4.3 Git 4.4 Servicios de alojamiento de repositorios
Tema 5. Diseño de interfaces de usuario
5.1 Introducción 5.2 Diseño de interfaces 5.3 Aplicaciones web
Tema 6. Ordenadores y lenguajes de programación
6.1 Breve historia de los ordenadores 6.2 Sistemas de almacenamiento 6.3 Enfoques de programación 6.4 Arquitecturas de ordenadores
Tema 7. Fiabilidad y rendimiento
7.1 Fiabilidad del software 7.2 Rendimiento del software 7.3 Herramientas para monitorizar el rendimiento 7.4 Ejemplos de rendimiento
Tema 8. Gestión de grandes volúmenes de datos
8.1 Introducción 8.2 Descripción de bases de datos 8.3 Sistemas de gestión de bases de datos 8.4 SQL 8.5 Motores de almacenamiento
Tema 9. Comunicación entre procesos
9.1 Introducción 9.2 La línea de comandos de Unix 9.3 Intercambio de datos basado en archivos 9.4 Comunicación entre procesos 9.5 Conexión mediante bases de datos 9.6 AJAX

Tema 10. Computación de alto rendimiento

10.1 Visión general de la computación de alto rendimiento
10.2 Rendimiento de la computación en paralelo
10.3 Programación en paralelo
10.4 Paralelización en MATLAB

Laboratorio

Práctica 1. Lenguaje de modelado unificado (UML)

En esta primera sesión de laboratorio, los estudiantes comprenderán mejor cómo se modelan sistemas usando diagramas UML.

Práctica 2. Convenciones de código y documentación de código fuente

El objetivo de esta práctica es poner en práctica las guías de estilo presentadas en las clases teóricas.

Práctica 3. Introducción al control de versiones con Git

En la tercera sesión de laboratorio, los estudiantes se familiarizarán con el flujo de trabajo básico de Git. Aprenderán a crear y gestionar un repositorio de código Fuente tanto a través de la interfaz web como usando un cliente de escritorio.

Práctica 4. Interfaces

El objetivo de esta sesión de laboratorio es que el alumno conozca diferentes herramientas para el diseño de interfaces.

Práctica 5. Rendimiento

En esta sesión, los estudiantes medirán el rendimiento de código en distintos entornos: C, MATLAB, páginas web… Se realizarán modificaciones en el código para analizar su influencia en el rendimiento.

Práctica 6. Bases de datos

Los estudiantes empezarán creando varias tablas en una base de dato y ejecutarán borrados en cascada. También se analizará el impacto de los índices en el rendimiento.

Práctica 7. Comunicación entre procesos

Se probarán varios métodos de comunicación entre procesos: MATLAB, línea de comandos y AJAX.

Práctica 8. Computación de alto rendimiento

Esta sesión empezará midiendo el rendimiento de un único ordenador y terminará usando una infraestructura profesional de computación en la nube como Microsoft Azure.

Proyecto final

En el proyecto final, los estudiantes tendrán que aplicar varios de los conceptos cubiertos durante el curso a un problema de investigación que involucre el desarrollo de una aplicación. Una propuesta de proyecto deberá ser remitida y aprobada por los profesores.

Competencias y resultados de aprendizaje

Competencias

Básicas y generales

CB1.	Haber adquirido conocimientos avanzados y demostrado, en un contexto de investigación científica y tecnológica o altamente especializado, una comprensión detallada y fundamentada de los aspectos teóricos y prácticos y de la metodología de trabajo en uno o más campos de estudio.
CB3.	Saber evaluar y seleccionar la teoría científica adecuada y la metodología precisa de sus campos de estudio para formular juicios a partir de información incompleta o limitada incluyendo, cuando sea preciso y pertinente, una reflexión sobre la responsabilidad social o ética ligada a la solución que se proponga en cada caso.
CB4.	Ser capaces de predecir y controlar la evolución de situaciones complejas mediante el desarrollo de nuevas e innovadoras metodologías de trabajo adaptadas al ámbito científico/investigador, tecnológico o profesional concreto, en general multidisciplinar, en el que se desarrolle su actividad.
CB5.	Saber transmitir de un modo claro y sin ambigüedades a un público especializado o no, resultados procedentes de la investigación científica y tecnológica o del ámbito de la innovación más avanzada, así como los fundamentos más relevantes sobre los que se sustentan.
CB7.	Ser capaces de asumir la responsabilidad de su propio desarrollo profesional y de su especialización en uno o más campos de estudio.

Competencias específicas

CE7.	Conocer las herramientas imprescindibles para abordar un tema de investigación.
CE9.	Conocer las técnicas, métodos y/o herramientas necesarias para abordar un tema de investigación específico en un sector o contexto tecnológico determinado.

Competencias adicionales

CO1.	Adquirir los conocimientos para diseñar eficientemente herramientas computacionales avanzadas.
CO2.	Obtener una visión general de las distintas herramientas informáticas disponibles y saber elegir aquellas que resulten más adecuadas para resolver cada tipo de problema.
CO3.	Aprender a identificar problemas de rendimiento, conocer estrategias para mejorar los tiempos de cálculo, saber estimar las mejoras esperadas y finalmente saber justificar la propuesta de cambios.

Resultados de aprendizaje

Al finalizar el curso los alumnos deben ser capaces de:

RA1.	Conocer los principales aspectos que caracterizan las herramientas para la investigación aplicada.
RA2.	Saber concebir y diseñar herramientas computacionales de alto nivel de complejidad y con capacidad de trabajar con grandes volúmenes de datos.
RA3.	Saber diseñar herramientas que integren diferentes tecnologías y se comuniquen con otros sistemas.
RA4.	Saber concebir estructuras de la información para trabajar eficientemente con grandes volúmenes de datos y poder realizar un análisis avanzado de los mismos.
RA5.	Ser capaz de realizar un análisis sistemático de varias soluciones para determinar cuál es la mejor, sabiendo elegir la técnica más adecuada para cada tipo de problema.

METODOLOGÍA DOCENTE

Aspectos metodológicos generales de la asignatura	
La mejor forma de comprender cualquier herramienta de software es experimentar con ella. En consecuencia, todas las actividades propuestas se centran en proporcionar a los alumnos la experiencia práctica que necesitan para poder desarrollar una herramienta computacional orientada a la investigación al final del cuatrimestre.	
Metodología presencial: Actividades	**Competencias**
▪ **Clases magistrales (12 horas):** El profesor introducirá los conceptos fundamentales de cada capítulo, acompañados de algunas recomendaciones prácticas, y resolverá ejemplos para apoyar la explicación. Se fomentará la participación active mediante preguntas abiertas que generen debate y mediante la proposición de breves ejercicios de aplicación para realizar en clase.	CB1, CB3, CE7, CE9, CO1, CO2, CO3
▪ **Sesiones de laboratorio (10 horas):** Bajo la supervisión del profesor, los estudiantes, divididos en pequeños grupos, aplicarán los conceptos y técnicas cubiertos en las clases magistrales a problemas reales y se familiarizarán con las herramientas computacionales más ampliamente utilizadas.	CB1, CB3, CE7, CE9, CO1, CO2, CO3
▪ **Tutorías:** Se organizarán bajo petición para grupos o de forma individual.	–
Metodología no presencial: Actividades	**Competencias**
▪ Estudio individual del material (20 horas).	CB1, CB3, CE7, CE9, CO1, CO2, CO3
▪ Análisis de los resultados de laboratorio y elaboración de informes (10 horas).	CB1, CB3, CE7, CE9, CO1, CO2, CO3
▪ Desarrollo de un proyecto final durante el último tercio del curso (30 horas).	CB1, CB3, CB4, CB7, CE7, CE9, CO1, CO2, CO3

EVALUACIÓN Y CRITERIOS DE EVALUACIÓN

Actividades de evaluación	Criterios de evaluación	Peso
Examen final	▪ Comprensión de los conceptos teóricos.	60%
Informes de laboratorio	▪ Aplicación de los conceptos teóricos. ▪ Habilidad para desarrollar código robusto, eficiente y claro. ▪ Comunicación escrita.	
Proyecto final	▪ Análisis del problema. ▪ Habilidades de búsqueda de información. ▪ Dificultad. ▪ Calidad de la solución propuesta. ▪ Comunicación oral.	40%

CALIFICACIONES Y NORMAS DE LA ASIGNATURA

Criterios de calificación

Periodo ordinario

- Los conocimientos y la participación ascenderán al 60%. Esto incluye:
  - Ejercicios prácticos
  - Actitud de aprendizaje activa
  - Examen final
- El 40% restante corresponde al proyecto final que será evaluado en base a:
  - La calidad del trabajo
  - La calidad de la presentación oral

Normas

- La asistencia a clase es obligatoria según el Artículo 93º del Reglamento General de la Universidad Pontificia Comillas y el Artículo 6 de las Normas Académicas de la Escuela Técnica Superior de Ingeniería (ICAI). El incumplimiento de esta norma puede acarrear las siguientes consecuencias:
  - La inasistencia a más del 15% de las horas presenciales puede hacer que el alumno pierda el derecho a presentarse al examen final.
  - Las sesiones de laboratorio perdidas deberán recuperarse para que puedan ser calificadas.
- Los estudiantes que comentan una irregularidad en cualquier actividad evaluada recibirán una calificación de cero en la misma y se les abrirá un proceso disciplinario (véase el Artículo 168º del Reglamento General de la Universidad Pontificia Comillas).

PLAN DE TRABAJO Y CRONOGRAMA[1]

Actividades presenciales y no presenciales	Fecha de realización	Fecha de entrega
Examen final	Mayo	
Sesiones de laboratorio	Semanalmente	
Repaso y estudio de los contenidos teóricos	Después de cada clase	–
Redacción de informes de laboratorio	–	Una semana después de cada sesión
Proyecto final	Durante el último tercio del curso	Último mes
Preparación del examen final	Mayo	–

RESUMEN DE HORAS DE TRABAJO DEL ALUMNO		
HORAS PRESENCIALES		
Lecciones magistrales	**Sesiones de laboratorio**	**Evaluación**
12	10	8
HORAS NO PRESENCIALES		
Estudio autónomo	**Informes de laboratorio**	**Proyecto final**
20	10	30
	Créditos ECTS:	**3 (90 horas)**

[1] En la ficha resumen se encuentra una planificación detallada de las asignatura (ver última página). No obstante, esta planificación tiene carácter orientativo y puede modificarse para adaptarse al ritmo de la clase.

BIBLIOGRAFÍA Y RECURSOS

Bibliografía básica

- Apuntes preparados por los profesores (disponibles en Moodle).

Bibliografía complementaria

- I. Sommerville, Software Engineering, 9th Ed., Addison-Wesley, 2011. ISBN-13: 978-0137035151
- P. Roques, UML in Practice: The Art of Modeling Software Systems Demonstrated Through Worked Examples and Solutions, 1ª Ed., John Wiley & Sons, 2004.
- S. McConnell, *Code Complete: A Practical Handbook of Software Construction*, 2ª Ed., Microsoft Press, 2004. ISBN-13: 079-0145196705
- A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 6ª Ed., McGraw-Hill, 2011. ISBN-13: 978-0073523323
- D. B. Kirk and W. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, 2ª Ed., Morgan Kaufmann, 2013. ISBN-13: 978-0124159921
- Documentación de Git, [Online]. Disponible: http://git-scm.com/doc/
- OS X Human Interface Guidelines, Apple Inc., [Online]. Disponible: https://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/OSXHIGuidelines/
- iOS Human Interface Guidelines, Apple Inc., [Online]. Disponible: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/
- Android Design, Google Inc., [Online]. Disponible: https://developer.android.com/design/

	Actividades presenciales				Actividades no presenciales				Resultados de aprendizaje
Sem.	**Horas**	**Clase de teoría/problemas**	**Laboratorio**	**Evaluación**	**Horas**	**Estudio individual**	**Preparación e informes de prácticas**	**Otras actividades**	**Código**
1	**2**	Presentación del curso (0,5h) Introducción a la ingeniería de software (1,5h)			**1**	Repaso y estudio individual (1h)			RA1, RA2, RA3, RA5
2	**2**	Convenciones de programación (1h)	Práctica 1 (1h)		**3**	Repaso y estudio individual (2h)	Informe (1h)		RA2
3	**2**	Documentación de código fuente (1h)	Práctica 2 (1h)		**3**	Repaso y estudio individual (2h)	Informe (1h)		RA2
4	**2**	Control de versiones (1h)	Práctica 3 (1h)		**3**	Repaso y estudio individual (2h)	Informe (1h)		RA2, RA3
5	**2**	Diseño de interfaces de usuario (1h)	Práctica 4 (1h)		**3**	Repaso y estudio individual (2h)	Informe (1h)		RA2
6	**2**	Ordenadores y lenguajes de programación (2h)			**1**	Repaso y estudio individual (1h)			RA2, RA3
7	**2**	Fiabilidad y rendimiento (1h)	Práctica 5 (1h)		**4**	Repaso y estudio individual (1h)	Informe (1h)	Propuesta de proyecto (2h)	RA5
8	**2**		Práctica 5 (2h)		**8**		Informe (2h)	Desarrollo del proyecto final (6h)	RA5
9	**2**	Gestión de grandes volúmenes de datos (1h)	Práctica 6 (1h)		**8**	Repaso y estudio individual (1h)	Informe (1h)	Desarrollo del proyecto final (6h)	RA2. RA4
10	**2**	Comunicación entre procesos (1h)	Práctica 7 (1h)		**8**	Repaso y estudio individual (1h)	Informe (1h)	Desarrollo del proyecto final (6h)	RA3
11	**2**	Computación de alto rendimiento (1h)	Práctica 8 (1h)		**8**	Repaso y estudio individual (1h)	Informe (1h)	Desarrollo del proyecto final (6h)	RA5
12	**2**			Presentaciones de proyecto (2h)	**4**			Preparación de la presentación del proyecto (4h)	RA2, RA3, RA4, RA5
13	**2**			Presentaciones de proyecto (2h)	**2**	Preparación del examen final (2h)			RA2, RA3, RA4, RA5
14	**2**			Presentaciones de proyecto (2h)	**2**	Preparación del examen final (2h)			RA2, RA3, RA4, RA5
15	**2**			Examen final (2h)	**2**	Preparación del examen final (2h)			RA1, RA2, RA3, RA4, RA5