Program	61CI - Bachelor of Software Engineering 61CT - Dual Bachelor Software Engineering and Information Society Technologies

Course number and name	
Number	615000253
Name	Architecture and Software Design
Semester	S6 [(February - June)]

Credits and contact hours	
ECTS Credits	6
Contact hours	60

Coordinator's name	Fernando Arroyo Montoro (fernando.arroyo@upm.es)

Specific course information

Description of course content

The software architecture translates the requirements into a more detailed model representing of the software solution, which will typically include the architecture design specifications and detailed design specifications. Alternatively, in software engineering, it is the process of defining the software architecture (the structure), components, modules, interfaces, test methods, and data for the software system to satisfy given requirements. For the design of a software architecture it will be necessary to use appropriate methods and select the architectural patterns that are considered more suitable for the resulting system such that it meets the required quality attributes.

List of topics to be covered

1. SOFTWARE ARCHITECTURE AND THE LIFE CYCLE
1.1. Introduction to the architecture concept
1.2. Historical evolution of the concept of architecture
1.3. The software architect's role
2. ARCHITECTURE DESIGN
2.1. Quality attributes of an architecture
2.2. Attribute-driven design
2.3. Architecture design flow
2.4. Design aspects, options and decisions
2.5. Types of decisions
3. ARCHITECTURAL VIEWS
3.1. Introduction to architectural views.
3.2. IEEE 1471 model
3.3. Viewpoint Specification
3.4. Kruchten's 4 + 1 model
4. ARCHITECTURE'S DOCUMENTATION

4.1. Documentation standards
4.2. Nomenclatures
4.3. Combination of views
5. ARCHITECTURAL PATTERNS
5.1. Introduction to Architectural Patterns
5.2. Description of architectural patterns
6. ARCHITECTURE EVALUATION AND REFACTORING
6.1. Introduction to Architecture Evaluation
6.2. Architectural Analysis Techniques
6.3. ATAM model of architectures evaluation
6.4. Refactoring
7. REFERENCE ARCHITECTURES
7.1. Concept of reference architecture
7.2. Examples of reference architectures
8. PRODUCT LINES
8.1. Examples of reference architectures
8.2. Feature models

Prerequisites or co-requisites

It is recommended some knowledge in Software Engineering Fundamentals, Software Engineering Processes and Requirements Engineering.

Course category in the program

61SW - R (required) **61ST - R (required)**	**____ - E (elective)**

Specific goals for the course

Specific outcomes of instruction

- Be able to know and apply current theories, models and techniques for problem identification, analysis, software design, development, implementation, verification and documentation.
- Model and design solutions according to the efficiency, modularity, and modularity tradeoffs.
- Identify and analyse problems to solve software solutions based on an adequate knowledge of theories, models and techniques.
- Be able to perform a complex task in an autonomous way, selecting the most convenient strategies to approach the study, based on the analysis and approach the study, based on the analysis of the conditions and the proposed goal.
- Evaluate the effectiveness of the planning of tasks and make the appropriate decisions to achieve their purpose.
- Be able to work as a member of a team with the purpose of contributing to the development of projects with pragmatism and sense of responsibility, assuming commitments and taking into account the available resources. available.

Further reading and supplementary materials
• Software Architecture: Foundations, Theory, and Practice. R. N. Taylor,N. Medvidovic , E. M. Dashofy.Wiley, 2009. ISBN-10: 0470167742. • Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, Second Edition. Nick Rozanski; Eoin Woods. Addison-Wesley Professional. October 25, 2011. ISBN-10: 0-321-71833-X. ISBN-13: 978-0-321-71833-4. • Applied SOA: Service-Oriented Architecture And Design Strategies. Michael Rosen; Boris Lublinsky; Kevin T. Smith; Marc J. Balcer. John Wiley & Sons. 2008. ISBN: 978-0-470-22365-9. • Documenting Software Architectures: Views and Beyond, Second Edition. By: Paul Clements; Felix Bachmann; Len Bass; David Garlan; James Ivers; Reed Little; Paulo Merson; Robert Nord; Judith Stafford. AddisonWesley Professional. • Essential Software Architecture.Ian Gorton. Springer-Verlag Berlin and Heidelberg GmbH & Co. K; Edición: 2nd ed. 2011. ISBN-10: 3642191754. ISBN-13: 978-3642191756. • Software Architecture in Practice, Third Edition. Len Bass; Paul Clements; Rick Kazman. AddisonWesley Professional. 2015. Safari Collection, • Beyond Software Architecture: Creating and Sustaining Winning Solutions. Luke Hohmann. AddisonWesley Professional. 2003. Safari Collection. • Software Architecture: A Case Based Approach. Vasudeva Varma. Pearson Education India. 2009. ISBN-10: 81-3170-749-0. ISBN-13: 978-8-131-70749-4. • Scaling Software Agility: Best Practices for Large Enterprises. Dean Leffingwell. Addison-Wesley Professional. 2007. Safari Collection.

Teaching methodology			
X__ **lectures**	_X _ **problem solving sessions**	_X _ **collaborative actions**	_X_ **laboratory sessions**
Other:			

Program	61IW - Bachelor of Software Engineering 61ST - Dual Bachelor Software Engineering and Information Society Technologies 61SI - Bachelor of Information Systems 61TI - Bachelor of Information Society Technologies 61IC - Bachelor of Computers Engineering 61CT - Dual Bachelor Computers Engineering and Information Society Technologies

Course number and name	
Number	615000242
Name	Concurrent and Advanced Programming
Semester	S4 [(February - June)]

Credits and contact hours	
ECTS Credits	6
Contact hours	60

Coordinator's name	Jorge Enrique Pérez Martinez (jorgeenrique.perez@upm.es)

Specific course information
Description of course content
A "concurrent program" can be defined as a set of sequential programs that can be executed in parallel (M. Ben-Ari). This set of sequential programs (or processes) can be executed on several processors (true parallelism) or on a single processor (pseudo-parallelism). In either case, as soon as there is cooperation between two or more processes or shared resources, the need arises to synchronize these concurrent processes in order to achieve the expected results.
List of topics to be covered
1. Introduction to concurrent programming 1.1. Computer SW Levels 1.2. Programming languages 1.3. Memory management in the operating system 1.4. Process management in the operating system 1.5. Threads within a process 1.6. Concurrent applications 2. Abstraction of concurrent programming 2.1. The Role of Abstraction 2.2. Concurrent execution as interleaving atomic sentences 2.3. Justification of Abstraction 2,4, Arbitrary interleaving

2.5. Atomic sentences
2.6. Correctness
2.7. Fairness
2.8. Machine code instructions
2.9. Volatile and Non-Atomic Variables
3. The critical section problem
3.1. Introduction and definition of the problem
3.2. First attempt
3.3. Correctness test with state diagrams
3.4. Further attempts
3.5. Dekker's algorithm
4. Semaphores
4.1. States of a process
4.2. Semaphore Definition
4.3. Critical section problem for two processes
4.4. Invariants of the semaphores
4.5. Critical section problem for N processes
4.6. Execution order problems
4.7. Producer-consumer problem
4.8. Other semaphore definitions
4.9. Philosophers' dinner problem
5. Monitors
5.1. Introduction
5.2. Declaration and use of monitors
5.3. Condition variables
5.4. Producer-Consumer Problem
5.5. Immediate Resumption Problem
5.6. Reader-writer problem
5.7. Philosophers' Dinner Problem
5.8. Monitors in Scala
6. Channels and messages
6.1. Communication models
6.2. Channels
6.3. Client-Server Interaction
6.4. Channels and actors
7. Interlocks
7.1. System model
7.2. Characterization of Interlocks
7.3. Methods for handling interlocks
7.4. Interlock Prevention
7.5. How to avoid deadlocks
7.6. Detection of interlocks
7.7. Interlock recovery
8. Asynchronous programming with futures
8.1. Futures
8.2. Callbacks
8.3. Composition of futures
9. Advanced programming
9.1. Parallel programming
9.2. Distributed programming
9.3. Software transactional memory

Prerequisites or co-requisites

It is recommended some knowledge in Programming Fundamentals and Oriented Object

Programming.

Course category in the program	
61SW - R (required) **61ST - R (required)** **61IC - R (required)** **61SI - R (required)** **61CT - R (required)**	**____ - E (elective)**

Specific goals for the course

Specific outcomes of instruction

- Explain the usefulness of concurrent programming in the contexts of parallel programming and real-time systems.
- Use the different message passing models to solve concurrent programming problems, selecting the most appropriate one in each situation.
- Use monitors as a high-level tool to solve concurrent programming problems.
- Given a short-term learning objective, autonomously identifies the necessary knowledge and skills and establishes a plan to achieve it. The plan integrates the selection of information sources, methods of searching for timely information and criteria for synthesizing the information gathered.
- Apply the theories and techniques necessary to correctly develop concurrent programs.
- Detect deadlock situations and apply mechanisms to cope with such situations.
- Be able to develop applications that run concurrent threads using different tools
- Identify the elementary algorithms to solve basic concurrent programming problems and the limitations of such solutions to solve concurrent programming problems.
- Be able to use semaphores as a low-level tool to solve concurrent programming problems.

Further reading and supplementary materials

- Principles of Concurrent and Distributed Programming (2nd. Ed.). Ben-Ari (2006); Addison-Wesley.
- Programming in Scala (3rd. Ed.). Odersky, Spoon, Venners, 2016; Ed. Artima.
- Learning concurrent programming in Scala, second editionz. Prokopec, 2017; Ed. PACKT.
- The art of multiprocessor programming. Herlihy, Shavitt, 2012; Ed. Morgan-Kaufmann.
- Concurrent programming. Raynal, 2013; Ed. Springer.

Teaching methodology			
__ lectures	_X _ problem solving sessions	_X _ collaborative actions	X laboratory sessions
Other:			

Program	61TI - Bachelor of Information Society Technologies

Course number and name	
Number	615000735
Name	Distributed Systems
Semester	S7 [(September - January)]

Credits and contact hours	
ECTS Credits	6
Contact hours	60

Coordinator's name	Sergio Arévalo Viñuales (sergio.arevalo@upm.es)

Specific course information

Description of course content

This subject considers a distributed system such as a collection of autonomous programmable entities. Normally, these entities communicate with each other in an asynchronous way through a communication channel. Both, the channel and the entities can present failures that compromise the reliability and coherence of the system. The main goal of this subject is the identification of the basic concepts and models underlying all distributed systems in order to apply the most appropriate algorithms for achieving robust and coherent applications, according to the model considered.

List of topics to be covered

1. Characterization of Distributed Systems
   - 1.1. Introduction
   - 1.2. Examples of distributed systems
   - 1.3. Trends in distributed systems
   - 1.4. Challenges
2. 2. System Models
   - 2.1. Introduction
   - 2.2. Architectural Models
   - 2.3. Fundamental Models
3. 3. Synchronization
   - 3.1. Introduction
   - 3.2. Synchronizing Physical Clocks
   - 3.3. Logical Time and Logical Clocks
   - 3.4. Global States
4. 4. Coordination and Agreement
   - 4.1. Introduction
   - 4.2. Distributed Mutual Exclusion

4.3. Election
4.4. Multicast Communication
4.5. Consensus
5. 5. Transactions
5.1. Introduction
5.2. Flat and Nested Distributed Transactions
5.3. Atomic Commit Protocols
5.4. Transaction Recovery Protocols
6. Lab 1. Java Review
7. Lab 2. Operating System Support
8. Lab 3. Interprocess Communication
9. Lab 4. Remote Invocation. RMI10.
10. Lab 5. ORB
11. Skills. Critical Thinking

Prerequisites or co-requisites

It is recommended some knowledge in Operating Systems, Concurrent Programming and Computer Networks.

Course category in the program

_X_____(required)	**____ - E (elective)**

Specific goals for the course

Specific outcomes of instruction

The student will be able to:

- Define the fundamental model and the architectural model for a distributed system.
- Identify the most common problems occurring in a distributed system due to the lack of shared memory, shared clock, synchrony and the occurrence of errors both network and processes.
- Select the most appropriate algorithms for solving the above problems given a fundamental model.
- Identify the elementary concepts and abstractions underlying all distributed systems.
- Build distributed applications using well-known and proprietary algorithms as well as off-the-shelf middleware platforms for distributed computing.
- Make judgments and decisions in a well-judged way.
- Analyze, interpret and evaluate information and arguments from different points of view. Synthesizes and relates information and draws conclusions in a well-judged way.

Further reading and supplementary materials

- George Coulouris, Jean Dollimore,Tim Kindberg, and Gordon Blair.2011. Distributed Systems: Concepts and Design (5th ed.). Addison-Wesley Publishing Company, USA.
- Andrew S. Tanenbaum and Maartenvan Steen. 2006. Distributed Systems: Principles and Paradigms (2nd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
- Nancy A. Lynch. 1996. Distributed Algorithms. Morgan Kaufmann Publishers Inc., San Francisco, CA,USA.
- Hagit Attiya and Jennifer Welch.2004. Distributed Computing: Fundamentals, Simulations and Advanced Topics. John Wiley &Sons.
- Wan Fokkink, Distributed Algorithms: An Intuitive Approach, MIT Press,2013

Teaching methodology			
_X_ **lectures**	_X _ **problem solving sessions**	_X _ **collaborative actions**	_X_ **laboratory sessions**
Other:			

Program	61IW – Bachelor of Science in Software Engineering 61CI – Bachelor of Science in Computer Engineering 61SI – Bachelor of Science in Information Systems 61TI – Bachelor of Science in Information Society Technologies

Course number and name	
Number	615000332
Name	English for professional and academic communication
Semester	S1 [(September-January)]

Credits and contact hours	
ECTS Credits	6
Contact hours	60

Coordinator's name	Carola Álvarez-Bolado Sánchez (carola.alvarez-bolado@upm.es)

Specific course information	
Description of course content	
The course is intended to deal with the different communicative situations both in professional and academic settings. It aims at practising and eventually getting a good command of written and oral communication skills in those environments. The course will use authentic material as models which the students will produce later on their own. After this course the students are able to understand and write different kind of professional and academic documents, participate in meetings and prepare subject-content oral presentations.	
List of topics to be covered	
1. Terminology in academic environments and TICs. 2. General characteristics of written professional communication. 3. Types of professional communication: letters, e-mail, report, memorandum and other types of documents. 4. General characteristics of written academic communication. 5. Types of academic communication: abstracts, papers, technical reports, diploma projects and other types of documents. 6. Oral professional communication. 7. Facts intervening in oral presentations: context, partners, formality degree, body language and others.	
Prerequisites or co-requisites	
Certification of B2 level of English according to the Council of Europe's Common European Framework of Reference for Languages (CEFRL)	
Course category in the program	
R (required)	

Specific goals for the course
Specific outcomes of instruction
RA493 - The student is able to write a logically organized and coherent document on a wide variety of topics and support his/her views RA492 - The student is able to write specialized-content documents RA495 - The student is able to collect information from different sources, i.e. lecturers and bibliographic resources RA523 - The student is able to understand complex and abstract ideas RA496 - The student is able to communicate fluently and accurately in written and oral English in professional and academic environments

Further reading and supplementary materials
- Bombardo, C.; Aguilar, M.; Barahona, C. (2007); "Technical Writing. A Guide for Effective Communication". Barcelona, Ediciones UPC. - Downer, C. (2013); "Job Hunting". Cambridge. Cambridge University Press. - Dochao Moreno, L. y Llavona Arregui, J.L. (2002);"Manual de Correspondencia Comercial y Profesional en Inglés Orientada a Alumnos de Inglés para Entornos de Ingeniería". - Lowe, S., Pile, L. (2007); "Negotiating". London: Delta Publishing. Taylor, S. (2004); "Model Business Letters, E-mails and Other Busines Documents". London: Pearson Education Limited. - Thompson, K. (2007); "English for Meetings". Oxford: Oxford University Press Bibliography Textbook for meetings. - Godwin, J., Strutt, L. (2008); "Test Your Business Vocabulary in Use. Advanced". Cambridge: Cambridge University Press.

Teaching methodology			
≈ 12 hours **lectures**	≈ 39 hours **collaborative actions**	≈ 9 hours **Tests and exams**	≥20 hours **Non-contact activities**

Program	61IW – Bachelor of Science in Software Engineering 61CI – Bachelor of Science in Computer Engineering 61SI – Bachelor of Science in Information Systems 61TI – Bachelor of Science in Information Society Technologies

Course number and name	
Number	615000520, 615000532, 615000740, 615000547
Name	Information Coding
Semester	S7 [(September-January)]

Credits and contact hours	
ECTS Credits	6
Contact hours	60

Coordinator's name	Lias Quintero, Ana Isabel (anaisabel.lias@upm.es)

Specific course information
Description of course content
The subject of this course is the study of the different possibilities to encode the information numerically, depending on the intended goal: conciseness (data compression), integrity (error detection codes) or security (cryptography). The general objectives are: a) To understand the different mathematical concepts and tools underlying the models under consideration; and b) To implement these models, with special attention to efficiency and security issues.
List of topics to be covered
1. Introduction to Information Coding. Cryptology 1.1. Trasmission of Information 1.2. Types of codes 1.3. Compression with variable-length codes: Huffman codification 1.3.1. Introduction to information theory 1.3.2. Huffman codification 1.3.3. Minimal variance Huffman codification 1.4 Error detection codes 1.4.1 Cyclic Redundancy Codes (CRC). 1.5. Cryptography and cryptosystems 1.5.1. Private key cryptosystems 1.5.2. Cryptanalysis 2. Computational complexity 2.1. Problems and algorithms 2.2. Complexity of elemental arithmetic operations

2.3. Classification of problems regarding its complexity

3. Number theory
   - 3.1. The multiplicative group of integers mod n
   - 3.2. Euler's totient function
   - 3.3. Euler and Fermat Theorems
   - 3.4. Order of an element. Primitive root
   - 3.5. Discrete logarithm

4. Public key cryptosystems
   - 4.1. Diffie- Hellman key exchange protocol
   - 4.2. RSA cryptosystem
   - 4.3. ElGamal cryptosystem
   - 4.4. Digital signature
   - 4.5. Other applications

5. Primality tests
   - 5.1. Deterministic tests: Erathostenes' sieve and trial division
   - 5.2. Probabilistic tests: Fermat, Miller and Miller-Rabin

Prerequisites or co-requisites

None.

Course category in the program

__ R (required)	**_X_ E (elective)** *(elective courses may not be offered every year)*

Specific goals for the course

Specific outcomes of instruction

- RA295 - Be able to determine the computational complexity of simple algorithms involving elementary arithmetic operations
- RA297 - Be able to use software properly to solve information coding problems, accurately describing the protocols used
- RA291 - Be able to use different information coding types depending on the goal (error correction, information encryption or information compression)
- RA299 - Be able to compress files using the right compressor codes
- RA290 - Know and apply authentication protocols (digital signature) and key exchange protocols based on public key cryptosystems.
- RA292 - Know and apply deterministic and probabilistic primality tests
- RA294 - Know the difference between public and private key cryptosystems and be able to encrypt and decrypt using translation, affine and affine matrix cryptosystems
- RA298 - Be able to use linear codes for error encoding, detection and correction
- RA293 - Be able to solve open problems, assessing several possible alternatives and arguing for the selected option according to the specified problem-solving criteria. Be able to identify, and design and develop an effective strategy to locate, the information required for the chosen option, and clearly report the result and respective findings.
- RA296 - Be able to apply key results in number theory to cryptology, using the RSA and El-Gamal cryptosystems for encryption and decryption.

Further reading and supplementary materials
– Buchmann, Johannes A: "Introduction to Cryptography". Second Edition. Springer-Verlag. 2004. – Koblitz, Neal: "A Course in Number Theory and Cryptography". Second Edition. Springer-Verlag. 1994 – Trappe, Wade; Washington, Lawrence C.: "Introduction to Crytography with Coding Theory". Prentice-Hall.

Teaching methodology			
_X_ **lectures**	__ **problem solving sessions**	__ **collaborative actions**	_X_ **laboratory sessions**
Other:			

Program	61CI - Bachelor of Science in Computer Engineering 61CT - Dual Bachelor Science Computer Engineering and Information Society Technologies

Course number and name	
Number	615000492
Name	Real-Time Systems
Semester	S7 [(September-January)]

Credits and contact hours	
ECTS Credits	6
Contact hours	60

Coordinator's name	Javier García Martín (javier.garciam@upm.es)

Specific course information
Description of course content
In Real-Time Systems (RTS), unlike in most computer systems, a delay in the generation of an output can have serious consequences on the environment in which it works, and therefore, the system is considered to have had an unacceptable failure. *This type of system requires specific tools, methodologies and analyses to ensure that the system is able to meet the time requirements under all circumstances. In addition, RTS are directly related to embedded systems, industrial systems, multimedia systems and high performance systems, where security and reliability are very important aspects.* *This subject addresses with all issues related to the development of RTS: specification, design, analysis, technologies and implementation. The course is taught using Case-Based Learning and Project-Based Learning. In this way, the students solve practical examples, in order to acquire the knowledge and skills necessary to develop this type of systems.*
List of topics to be covered
1. INTRODUCTION 1.1. RTS characteristics; 1.2. Critical RTS; 1.3. Development environments and execution environments; 1.4. Introduction to the Ada language; 1.5. Concurrent programming in Ada language 2. PROGRAMMING OF REAL TIME ASPECTS 2.1. Time representation and data types; 2.2. Delays, Time-Outs, Asynchronous transfer of control; 2.3. Periodic processes;

2.4. Introduction to process priorities;
2.5. Non-periodic processes and sporadic servers.
3. RTS SCHEDULING
3.1. Introduction to the scheduling problem;
3.2. Cyclic executive approach;
3.3. Scheduling with fixed priorities;
3.4. Resource management protocols;
3.5. Response time analysis;
3.6. Other scheduling methods:
4. RTS AND EMBEDDED SYSTEMS MODELING
4.1. RTS requirements analysis
4.2. Architectural and detailed design of a real-time embedded system
4.3. A case study
5. FAULT TOLERANCE
5.1. Basic concepts;
5.2. Error recovery methods;
5.3. Exceptions in support of fault tolerance;
5.4. Fault tolerance under time constraints;
6. INTRODUCTION TO HIGH INTEGRITY SYSTEMS
6.1. Basic concepts of High Integrity Systems;
6.2. Introduction to Security Standards;
6.3. Verification Techniques;
6.4. ADA Language in High Integrity Systems;

Prerequisites or co-requisites

It is recommended some knowledge in Operating Systems and Programming, especially in Concurrent Programming.

Course category in the program

61CI - R (required) **61CT - R (required)**	**____ - E (elective)**

Specific goals for the course

Specific outcomes of instruction

Develop the hardware and software components for an embedded system.
Apply methodologies and techniques to improve the reliability and security of a system
Develop real time systems using the appropriate techniques to guarantee the defined constraints
Use the appropriate programming tools to implement multitasking systems following the structure of a critical real time system.
Use modelling languages to specify and design an embedded system with real-time constraints.
Write texts with complex structure to document some work or study performed. Organize the different parts of the text following a scientific text structure: hypothesis formulation, context, objectives, justification of the work, context, objectives, justification of the work, method used, results obtained, conclusions and bibliographical references. The student expresses him/herself effectively through writing and graphic supports.

Further reading and supplementary materials
• "Real-Time Systems and their programming languages", 4ª Edition. BURNS AND WELLINGS. AddisonWesley, 2009. • "Concurrency in Ada", 2ª Edition. ALAN BURNS AND ANDY WELLINGS. Cambridge University Press, 1998. • "Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE". Bran Selic, Sebastian Gérard.Ámsterdam, Elsevier. 2014 • "Ada95 Reference Manual". S. TUCKER TAFT, ROBERT A. DUFF. Lectures Notes in Computer Science. Springer, 1997. • "Real-Time Systems" Jane W.S. Liu.Prentice Hall, 2000. • "A Practitioners Handbook for RealTime Analysis: Guide to rate monotonic analysis for real-time systems". M.H. KLEIN, T. RALYA, B.POLLAK, R. OBENZA and M. GONZALEZ. KAP, 1993. • "Real-Time Systems". C.M. KRISHNA, KANG G. SHIN. McGrawHill, 1997. • "Programming in Ada95", 2ª Edición. JOHN BARNES. Addison-Wesley,

Teaching methodology			
X__ **lectures**	X_ _ **problem solving sessions**	X_ _ **collaborative actions**	_X_ **laboratory sessions**
Other:			

Program	61IW – Bachelor of Science in Software Engineering

Course number and name	
Number	615000259
Name	Software Quality
Semester	S7 [(September-January)]

Credits and contact hours	
ECTS Credits	6
Contact hours	60

Coordinator's name	Gallardo Perez, Carolina (carolina.gallardop@upm.es)

Specific course information	
Description of course content	
• To learn and apply techniques for measuring software quality. • To learn processes for quality management.	
List of topics to be covered	
1. Introduction 2. Quality techniques 3. Measuring software quality 4. Models and characteristics of software quality 5. Cost and ethics considerations 6. System quality requirements 7. Error characterization 8. Processes in quality management 9. Quality management regulation 10.Quality improvement 11.Tools for quality support	
Prerequisites or co-requisites	
None.	
Course category in the program	
_X_ R (required)	**__ E (elective)** *(elective courses may not be offered every year)*

Specific goals for the course
Specific outcomes of instruction
• RA113 - Be able to select and evaluate software quality control metrics and indicators.

- RA109 - Be versed in quality management processes and standards.
- RA83 - Be able to perform a complex task independently, selecting the best strategies for studying the problem by analyzing the specified conditions and goal. Be able to analyse and interpret information, manage information and communication technologies (ICT), have communication and interaction skills for collaborative learning. Be able to assess the effectiveness of task planning and make the right decision making to achieve a goal.
- RA114 - Be able to solve open problems, weighing up more than one alternative and critically appraising the decisions made, precisely expressing the required arguments and findings. Be able to identify and solve scenarios where the stated problem requires an approach that does not account for different levels of abstraction or the construction of a hierarchy.

Further reading and supplementary materials
– "Total Quality Management". Dale H. Besterfield, Carol Besterfield-Michna, Glen H. Besterfield, Mary Besterfield-Sacre. Prentice Hall – http://www.lpsi.eui.upm.es/webcalisoft/ – OCW

Teaching methodology			
_X_ **lectures**	__ **problem solving sessions**	__ **collaborative actions**	_X_ **laboratory sessions**
Other:			

Program	61TI – Bachelor of Science in Information Society Technologies

Course number and name	
Number	615000733
Name	System Administration
Semester	S7 [(September-January)]

Credits and contact hours	
ECTS Credits	6
Contact hours	60

Coordinator's name	Pilar Manzano (pilar.manzano@upm.es)

Specific course information	
Description of course content	
Understanding and doing some of the most important tasks of Linux system administration.	
List of topics to be covered	
1. Introduction 2. The BASH language 3. Virtualization 4. System startup and shutdown 5. Installing and updating software 6. User management 7. Managing system resources 8. System security 9. Automating tasks with cron 10. Filesystems and backups 11. RAID systems 12. Printer management	
Prerequisites or co-requisites	
None.	
Course category in the program	
_X_ **R (required)**	__ **E (elective)** *(elective courses may not be offered every year)*

Specific goals for the course
Specific outcomes of instruction
• RA143 – To be able to write BASH programs to automate tasks

- RA142 – To know and use Unix administration commands
- RA145 – To control the use of system resources
- RA144 – To know and modify Unix configuration files
- RA146 – To schedule periodical administration tasks

Further reading and supplementary materials
– "Unix and Linux System Administration Handbook". 4th. Ed. Evi Nemeth. Prentice-Hall, 2011 – "Essential System Administration". 3rd. Edition. Aeleen Frisch. O'Reilly & Associates, 2002 – "Learning the Bash shell". 3rd. Ed. Cameron Newham. O'Reilly, 2005 – "Managing RAID on Linux". Derek Vadala. O'Reilly, 2003

Teaching methodology			
_X_ **lectures**	_X_ **problem solving sessions**	__ **collaborative actions**	_X_ **laboratory sessions**
Other:			

Program	61IW – Bachelor of Software Engineering 61IC – Bachelor of Computers Engineering 61SI – Bachelor of Information Systems 61TI – Bachelor of Information Technologies

Course number and name	
Number	615000248
Name	Translators of Programming Language
Semester	S6 [(February-June)]

Credits and contact hours	
ECTS Credits	6
Contact hours	60

Coordinator's name	Sánchez Couso, José Ramón (joseramon.sanchez.couso@upm.es)

Specific course information
Description of course content
Theory of formal languages and its applications in the parsing and translation of programming languages
List of topics to be covered
1. Formal and programming languages 1.1. Basic concepts 1.2. Operations with formal languages 1.3. Specification of formal languages 1.4. Programming languages and their relation with formal languages 2. Regular languages 2.1. Regular expressions 2.2. Finite automata 2.3. Regular grammars 2.4. Transformations between regular mechanisms 2.5. Properties of regular languages 3. Lexicographic aspects 3.1. Formal specification of lexicography 3.2. Fundamentals of a lexical analyzer 4. Context-free languages 4.1. Context-free grammars 4.2. Transformations of context-free grammars 4.3. Chomsky normal form: CYK algorithm 4.4. Properties of free context languages

4.5. Pushdown automata 5. Syntactic aspects 5.1. Formal specification of syntax 5.2. Top-down syntactic parsers 5.3. Bottom-up syntactic parsers 5.4. LL(k) and LR(k) grammars 5.5. Applications of LL(1) grammars in the syntactic analysis 6. Context-dependent and recursively enumerable languages. 6.1. Turing Machines 6.2. Arbitrary Generative Grammars 6.3. Basic aspects of computability and complexity
Prerequisites or co-requisites
It is recommended some knowledge in Programming, Data Structure, Mathematical Logic, and Discrete Mathematics
Course category in the program

_X_ R (required)	**__ E (elective)** *(elective courses may not be offered every year)*

Specific goals for the course

Specific outcomes of instruction

- Be able to solve problems by expressing with precision the necessary arguments and conclusions.
- Be able to apply the fundamentals of regular mechanisms for constructing a lexical analyzer.
- Be able to apply the fundamentals of context-free grammars and pushdown automata for constructing a syntactic analyzer.
- Evaluate features of programming languages in accordance with their implementation
- Be able to know the fundamentals of formal languages
- Implement concrete or abstract syntax tree generating lexical-syntactic parsers.
- Analyze and writes type 2 grammars.
- Be able to use regular specification mechanisms. Be able to select and evaluate software quality control metrics and indicators.
- Be able to solve open problems, weighing up more than one alternative and critically appraising the decisions made, precisely expressing the required arguments and findings.
- Be able to identify and solve scenarios where the stated problem requires an approach that does not account for different levels of abstraction or the construction of a hierarchy.

Further reading and supplementary materials

– The Theory of Parsing, Translation and Compiling. Volume I. Alfred V. Aho, Jeffrey D. Ullman. Prentice Hall,1972.

– Concepts of Programming Languages. Robert W. Sebesta. Pearson International,

2008.

- Compilers: Principles, Techniques, & Tools. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. Addison Wesley 2007.

Teaching methodology			
_X_ **lectures**	__X **problem solving sessions**	_X_ **collaborative actions**	__ **laboratory sessions**
Other:			