COURSE DATA

DATA SUBJECT

Code: 34886
Name: Software engineering
Cycle: Undergraduate Studies
ECTS Credits: 6
Academic year: 2026-27

STUDY (S)

Degree	Center	Acad. year	Period
1403 - Degree in Telematics Engineering	Escola Tècnica Superior d'Enginyeria	3	First quarter
1935 - Double Degree Program in Mathematics-Telematics Engineering	Facultat de Ciències Matemàtiques	4	First quarter

SUBJECT-MATTER

Degree	Subject-matter	Character
1403 - Degree in Telematics Engineering	Software engineering	COMPULSORY
1935 - Double Degree Program in Mathematics-Telematics Engineering	Cuarto curso	COMPULSORY

COORDINATION

ZARAGOZA ALVAREZ IRENE

SUMMARY

Software Engineering is a compulsory course that is part of the Software Engineering and Project Management subject area within the Bachelor's Degree in Telecommunication Engineering. It is allocated 6 ECTS credits and is taught during the first semester of the third academic year.
This course aims to teach students how to develop software projects by following a systematic process and using tools that help improve software quality in production environments.
Students will be introduced to the knowledge and use of different information systems development methodologies.
The course seeks to provide sufficient knowledge of the software process so that students are able to elicit requirements, analyse, design, implement, test, and deploy software projects accurately and effectively.
With regard to the practical part, the course aims for students to use the UML modelling language and the Java programming language to put into practice the knowledge acquired in the theoretical part of the course.
The main objective of the course is to introduce students to the development of software projects, from requirements analysis to product deployment and customer verification, so that they are able to:

- Understand the basic concepts, development processes, methods, and tools of Software

Engineering.
- Understand and comprehend the current Software Engineering paradigms aimed at distributed software development, open-source software engineering, and Web Engineering.
- Understand the different architectural models, the technologies that can be integrated, and the commercial solutions available to build a specific distributed solution.
- Apply the concepts and foundations of Software Engineering, software architecture, and related techniques in software production.
- Apply methodologies for the development, deployment, and maintenance of information systems.
- Correctly plan and execute iterative software development processes.
- Be able to apply software design patterns in each situation according to the needs of the software development project.
- Select the most appropriate architectural style for each project.

PREVIOUS KNOWLEDGE

RELATIONSHIP TO OTHER SUBJECTS OF THE SAME DEGREE

There are no specified enrollment restrictions with other subjects of the curriculum.

OTHER REQUIREMENTS

It is recommended:
-Have passed the subjects of the first and second course Informatics I (34877) and Informatics II (34878) .
-Be enrolled or have passed the subject Programming (34888)

COMPETENCES / LEARNING OUTCOMES

1403 - Degree in Telematics Engineering

Ability for self-learning of new knowledge and techniques appropriate for the conception, development and exploitation of telecommunications systems and services.

E3 - Ability to construct, operate and manage telematic services using analytical tools for planning, dimensioning and analysis.

E4 - Ability to describe, program, validate and optimize communication protocols and interfaces at different levels of a network architecture.

G4 - Ability to solve problems with initiative, decision-making and creativity, and to communicate and transmit knowledge, abilities and skills, understanding the ethical and professional responsibility of the activity of a telecommunications technical engineer.

G9 - Ability to work in a multidisciplinary environment and in a multilingual group and to communicate, in writing and orally, knowledge, procedures, results and ideas related to telecommunications and electronics.

R7 - Understand and use the basic principles of programming for telecommunication networks, systems and services.

DESCRIPTION OF CONTENTS

This topic introduces the fundamentals of Software Engineering, analysing its origins and the need to apply systematic approaches to the development of quality software. The so-called software crisis, the basic terminology of the discipline, and the relationships between the concepts of software process, life cycle, and methodology are studied. In addition, the main process models and types of development methodologies are reviewed, analysing their characteristics, advantages, limitations, and areas of application.

This topic addresses requirements engineering as one of the fundamental activities for the success of a software project. The basic concepts of requirements, the different existing types, and the difficulties associated with their elicitation, analysis, and management are studied. Furthermore, the main requirements elicitation techniques are presented, along with the preparation of Software Requirements Specification (SRS) documents according to the IEEE/ANSI 830 standard, and the modelling of functional requirements through use cases and their detailed specifications.

This topic addresses the analysis phase aimed at transforming requirements into models that describe the system from a conceptual perspective independent of implementation. The activities and artifacts involved in analysis are studied, including the development of the data dictionary, the identification of relevant domain concepts, and their representation through conceptual class diagrams. Dynamic modelling techniques using UML diagrams are also covered, such as system sequence diagrams, operation contracts, activity diagrams, and state diagrams, with the aim of describing the behaviour and interactions of the system.

This topic focuses on transforming analysis models into a technical solution that can be implemented. The fundamental principles of object-oriented design are studied, including the assignment of responsibilities to classes through basic design patterns and the development of the main design artifacts, such as interaction diagrams and design class diagrams. Quality criteria and best practices are also analysed to obtain modular, maintainable, and reusable solutions.

This topic addresses the construction of software based on the models obtained during the analysis and design phases. It examines decisions prior to implementation, the transformation of design artifacts into code, and different strategies for maintaining consistency between models and the implemented system. In addition, programming best practices, code management, and criteria for identifying when it is necessary to adapt or refine models in order to improve the quality, performance, or maintainability of the developed solution are presented.

Topic 1. Introduction to Software Engineering and the Software Development Process

This topic introduces the fundamentals of Software Engineering, analysing its origins and the need to apply systematic approaches to the development of quality software. The so-called software crisis, the basic terminology of the discipline, and the relationships between the concepts of software process, life cycle, and methodology are studied. In addition, the main process models and types of development methodologies are reviewed, analysing their characteristics, advantages, limitations, and areas of application.

Topic 2. Planning: Requirements and System Scope

This topic addresses requirements engineering as one of the fundamental activities for the success of a software project. The basic concepts of requirements, the different existing types, and the difficulties associated with their elicitation, analysis, and management are studied. Furthermore, the main requirements elicitation techniques are presented, along with the preparation of Software Requirements Specification (SRS) documents according to the IEEE/ANSI 830 standard, and the modelling of functional requirements through use cases and their detailed specifications.

Topic 3. Analysis: Software Modelling

This topic addresses the analysis phase aimed at transforming requirements into models that describe the system from a conceptual perspective independent of implementation. The activities and artifacts involved in analysis are studied, including the development of the data dictionary, the identification of relevant domain concepts, and their representation through conceptual class diagrams. Dynamic modelling techniques using UML diagrams are also covered, such as system sequence diagrams, operation contracts, activity diagrams, and state diagrams, with the aim of describing the behaviour and interactions of the system.

Topic 4. Software Design

This topic focuses on transforming analysis models into a technical solution that can be implemented. The fundamental principles of object-oriented design are studied, including the assignment of responsibilities to classes through basic design patterns and the development of the main design artifacts, such as interaction diagrams and design class diagrams. Quality criteria and best practices are also analysed to obtain modular, maintainable, and reusable solutions.

Topic 5. Implementation

This topic addresses the construction of software based on the models obtained during the analysis and design phases. It examines decisions prior to implementation, the transformation of design artifacts into code, and different strategies for maintaining consistency between models and the implemented system. In addition, programming best practices, code management, and criteria for identifying when it is necessary to adapt or refine models in order to improve the quality, performance, or maintainability of the developed solution are presented.

Topic 6. Software Architectures

This topic studies the fundamental principles of software architecture and its role in the construction of complex systems. The main classical architectural styles are analysed, such as layered architecture, client-

server architecture, and component-based architecture, as well as more modern approaches based on services, microservices, and event-driven architectures. Furthermore, architectural decisions and their influence on quality attributes such as maintainability, scalability, performance, security, and system evolvability are addressed.

Topic 7. Software Testing

This topic introduces the fundamentals of software verification and validation through testing techniques. Different levels of testing are studied, including unit, integration, system, and acceptance testing, as well as the main strategies for designing test cases based on analysis, design, and implementation artifacts. Different testing approaches and their application throughout the development process are also analysed in order to detect defects and ensure the correct functioning of the system.

WORKLOAD

PRESENCIAL ACTIVITIES

Activity	Hours
Theory	30,00
Laboratory	20,00
Classroom practices	10,00
Total hours	**60,00**

NON PRESENCIAL ACTIVITIES

Activity	Hours
Attendance at other activities	3,00
Individual or group project	20,00
Independent study and work	7,00
Preparation of lessons	50,00
Preparation for assessment activities	3,00
Resolution of case studies	7,00
Total hours	**90,00**

TEACHING METHODOLOGY

LECTURES:
The lectures will be based on active lectures where every 20/25 minutes will be introduced in any activity that requires the involvement of students, so that 1) they can do an activity based on the content they have just learnt, 2) recover the level of attention to the next block.

LECTURES PREPARATION:
Students have to prepare the lecture content, following the plan of the course. To do this they will use the literature suggested by the lecturer as well as the materials provided him or/and any other directions provided by the lecturer.

PREPARATION OF PRACTICAL WORK:
To better assimilate the contents of the lectures, practical sessions are conducted in the laboratories. Attendance to practical sessions is mandatory and will be verified by the lecturer in charge of the group. Students who are working and cannot attend practical sessions should contact the lecturer before the beginning of the first session. The results of these activities must be submitted to the lecturer in charge of the group during the course and in the terms established by the lecturer. Students are expected to

do/prepare some of these activities at home.

TEAM WORK:

A set of problems will be proposed that should be solved in teams of 5 to 6 persons. Each member of the group will be graded both the joint mark of the group as the individual mark of each member.

The e-learning platform (Aula Virtual) will be used as communication tool between the lecturer and the student. The student will access to all the material used in the lectures, through Aula Virtual, as well as all the problems and exercise that needs to solve.

EVALUATION

Students' achievement of the intended learning outcomes will be assessed as follows.

Students will be assessed on a continuous basis through their participation in the different learning activities, the extent to which they benefit from the proposed training activities, and their level of engagement in their own learning process. The following assessment components will be considered:

- **C) Continuous Assessment and Practical Activities**, based on participation and engagement throughout the teaching and learning process, taking into account regular attendance at the scheduled face-to-face sessions and the completion of the assigned coursework. As part of the continuous assessment, students will individually complete a set of practical exercises and/or theoretical assignments, which must be submitted through the Virtual Classroom within the established deadlines. In addition, students will take one midterm test covering part of the course content. Finally, throughout the semester, students will develop a software project, which will be presented in class at the end of the course. The grade for this assessment component will be calculated as follows:

C (Continuous Assessment and Practical Activities Grade) = 0.4 * Test + 0.4 *Assignments + 0.2 * Project Presentation

Late submissions will not be accepted, and missed continuous assessment activities cannot be retaken. Any form of academic dishonesty or plagiarism in any of these activities will be strictly penalized by invalidating all continuous assessment grades obtained by the student. Competences G4, G9, R1, R7, E3 and E4.

- **(P) Laboratory Activities**. They will be done in groups of 5 or 6 people and both the quality of the solution and the documentation in each of the practices will be assessed. Competences G4, G9, R1, R7, E3 and E4
- **(E) Individual objective tests**, consisting of one or several exams, or knowledge tests, which will consist of both theoretical-practical questions and problems. It will be necessary to pass each of these tests or exams in order to pass the subject. Competences G4, R1 and E3.

The final grade for both, the first and second assessment periods will be calculated according to the following formula:

Final Grade = 0.35*C + 0.35*E+0.3*P

Grades awarded for group activities will not necessarily be identical for all group members and may vary depending on each student's individual contribution and level of involvement.

Only coursework submitted by the deadlines established by the teaching staff will be considered for assessment. This applies to all assignments, quizzes, exercises, the software project, and any other task assigned during the course.

The C part is not recoverable on second assessment period. The P and E parts will only be combined in the final grade if a minimum score of 5.0/10 has been obtained in each component.

If either E or the P is not passed with a grade of at least 5.0/10, and the student has taken the examination (E), the official final grade recorded will be:

Final Grade = minimum(E,P,4)

Students who do not sit the examination (E) will receive the official grade of Absent

Since the Continuous Assessment and Practical Activities (C) component cannot be retaken, students wishing to apply for an early examination session must have previously enrolled in the course and successfully completed this component.

In any case, the evaluation of the subject will be done in accordance with the Evaluation and Qualification Regulations of the University of Valencia for undergraduate and master's degrees approved by the Governing Council on May 30, 2017 (**ACGUV 108/2017**).
Copying or plagiarism of any activity that is part of the evaluation will result in the impossibility of passing the course, and the student will then be subject to the appropriate disciplinary procedures indicated in the ACTION PROTOCOL FOR FRAUDULENT PRACTICES AT THE UNIVERSITY OF VALENCIA. (ACGUV 123/2020).

REFERENCES

- Apuntes de la asignatura
- [Kenneth E. Kendall, Julie E Kendall (2010)] Systems Analysis and Design, 8th Edition (Prentice Hall)
- [Frank Tsui, Orlando Karam, and Barbara Bernal(2016)] Essentials of Software Engineering. Jones & Bartlett Learning, LLC [Recurs Electrònic: https://ebookcentral.proquest.com/lib/univalencia/detail.action?docID=4826428]
- [Martina Seidl, Marion Scholz, Christian Huemer, Gerti Kappel] UML @ Classroom. An Introduction to Object-Oriented Modeling [Recurs Electrònic: https://link.springer.com/book/10.1007%2F978-3-319-12742-2]
- [C. Larman (2004)] Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd (Edition Prentice Hall)[Recurs electronic: http://proquest.safaribooksonline.com/0131489062?uicode=valencia]
- [Grady Booch, James Rumbaugh, Ivar Jacobson (2005)] The Unified Modeling Language User Guide (2nd Rev. Edition) (Addison-Wesley) [Recurs electronic: http://proquest.safaribooksonline.com/0321267974]
- [Michael R. Blaha, James R Rumbaugh (2005)] Object-Oriented Modeling and Design with UML (2nd Edition) (Prentice Hall)

- [A. Weitzenfeld (2004)] Ingeniería de software orientada a objetos con UML, Java e Internet (Thomson)
- [Robert C. Martin (2003)] UML for Java programmers (Prentice Hall)[Recurs electronic: http: //proquest.safaribooksonline.com/0131428489?uicode=valencia]
- [Roger S. Pressman (2009)] Software Engineering: A Practitioner's Approach, 7th Edition (Mc Graw Hill)
- [I. Sommerville (2011)] Software Engineering, 9th Edition (Addison-Wesley)
- [S. Sánchez Alonso, M. A. Sicilia Urbán, D. Rodríguez García (2011)] Ingeniería de software: un enfoque desde la guía SWEBOK (Garceta)
- [Bernd Bruegge, Allen H. Dutoit] Object-Oriented Software Engineering Using UML, Patterns, and Java, 3rd Edition (Edition Prentice Hall)