COURSE DATA

DATA SUBJECT

Code: 34662
Name: Software engineering II
Cycle: Undergraduate Studies
ECTS Credits: 6
Academic year: 2026-27

STUDY (S)

Degree	Center	Acad. year	Period
1400 - Degree in Computer Engineering	Escola Tècnica Superior d'Enginyeria	3	First quarter
1936 - Double Degree Program in Mathematics-Telematics Engineering	Facultat de Ciències Matemàtiques	4	First quarter

SUBJECT-MATTER

Degree	Subject-matter	Character
1400 - Degree in Computer Engineering	Software engineering and project management	COMPULSORY
1936 - Double Degree Program in Mathematics-Telematics Engineering	Cuarto curso	COMPULSORY

COORDINATION

GIL PASCUAL MIRIAM

SUMMARY

The Software Engineering II course is part of the subject Software Engineering and Project Management. The overall objective of this course is to expand on the contents covered in the course Software Engineering I. Specifically, it frames these contents within the MÉTRICA version 3 methodology, introduces agile methodologies, software testing, maintenance and reengineering processes of information systems, and describes a series of organizational or support activities for the development process and products. If these activities exist in the organization, they should be applied to enrich or influence the execution of the main process activities of the software development methodology. If they do not exist, they must be carried out to complement and ensure the success of the developed project (Software configuration management, Quality management and assurance).

In general terms, the objectives of the course are:

- To provide students with a general overview of the information systems development process, explaining the characteristics of maintenance and reengineering processes of information systems.

- To present organizational or support activities associated with the software development process that should be applied to enrich or influence the execution of the main process activities of the development methodology.

- To introduce students to agile methodologies and how they are implemented in software development compared to traditional methodologies.

- To highlight the need to maintain the integrity of the products obtained throughout the system development process, ensuring that uncontrolled changes are not made and that all participants have the appropriate version of the products they handle.

- To provide students with a common reference framework for the definition and implementation of specific quality assurance plans applicable to specific projects.

- To raise students' awareness of the need to incorporate the software testing process to ensure the quality and reliability of software products in project development.

- To present students with the most common techniques for measurement and estimation in software projects during their development and maintenance.

- To emphasize the importance of software architectures as fundamental structures that provide a framework for the design and development of software systems.

From a teaching perspective, the course has a fundamentally practical approach and is focused on developing practical skills for the engineer, which they will need in their professional development as project managers or as part of a project team. To this end, they must acquire a series of skills related to the management of both material and human resources, as well as task decomposition.

PREVIOUS KNOWLEDGE

RELATIONSHIP TO OTHER SUBJECTS OF THE SAME DEGREE

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

OTHER REQUIREMENTS

As the name suggests the course Software Engineering II has a very direct connection with what is taught in Software Engineering I, being recommended to have previously studied this subject. More specifically, the course is based on the following concepts will be assumed known:

- Specification and modeling of software requirements
- System analysis (modeling use cases, class diagrams, etc.)
- System design modeling class diagram, sequence diagram, state diagrams, etc.)
- System implementation based on the design models

COMPETENCES / LEARNING OUTCOMES

1400 - Degree in Computer Engineering

G10 - Knowledge to perform measurements, calculations, assessments, appraisals, surveys, studies, reports, scheduling and other similar work in the field of computer engineering, in accordance with both the knowledge and the specific skills acquired in the degree.

G2 - Ability to lead project activities in the field of information technology, in accordance with both the knowledge and the specific skills acquired in the degree.

G3 - Ability to design, develop, evaluate and ensure the accessibility, ergonomics, usability and security of computer systems, services and applications, and of the information that these manage.

G5 - Ability to design, develop and maintain computer systems, services and applications using software engineering methods as an instrument for quality assurance, in accordance with both the knowledge and the specific skills acquired in the degree.

G6 - Ability to design and develop computer systems and centralised or distributed computer architectures which integrate hardware, software and networks, in accordance with both the knowledge and the specific skills acquired in the degree.

G9 - Ability to solve problems with initiative, decision making, autonomy and creativity. Ability to communicate and transmit the knowledge, skills and abilities of a computer engineer.

R16 - Knowledge and application of the principles, methodologies and life cycles of software engineering.

R18 - Knowledge of the rules and regulations of computer science at the national, European and international levels.

R1 - Ability to design, develop, select and evaluate computer applications and systems while ensuring their reliability, safety and quality, according to ethical principles and current legislation and regulations.

R2 - Ability to plan, design, implement and manage computer projects, services and systems in all areas, leading their implementation and continuous improvement by assessing their economic and social impact.

R3 - Ability to understand the importance of negotiation, effective work habits, leadership and communication skills in all software development environments.

SI3 - Ability to actively participate in the specification, design, implementation and maintenance of information and communication systems.

TI2 - Ability to select, design, implement, integrate, evaluate, build, manage, exploit and maintain hardware, software and network technologies, within adequate cost and quality thresholds.

DESCRIPTION OF CONTENTS

1. Software methodologies: MÉTRICA version 3
1.1. Introduction to MÉTRICA version 3
1.2. Main processes
1.3. Process activities
1.4. Software development interfaces
1.5. Stakeholders

2. Agile methodologies: user stories
2.1. Introduction to SCRUM
2.2. Requirements definition: user stories
2.2.1. Methods for elicitation and representation
2.2.2. Required information
2.2.3. Description of the story
2.2.4. Estimation and evaluation of a story
2.2.5. Prioritization of stories
2.2.6. Validation and quality criteria
2.2.7. Technical stories

3. Software configuration management
3.1. Introduction
3.2. Elements of configuration management
3.3. Configuration management activities
3.3.1. Change control
3.3.2. Version control
3.3.3. System build
3.3.4. Delivery management
3.3.5. Incident and problem management
3.4. Tools

4. Software architectures
4.1. Introduction to software architectures
4.2. Architectural styles
4.3. Design principles in software architectures
4.4. Modeling tools and techniques

5. Software testing
5.1. Introduction
5.2. Testing techniques
5.3. Software testing strategies
5.3.1. Unit testing
5.3.2. Integration testing
5.3.3. System testing
5.3.4. Acceptance testing
5.4. Debugging

6. Software measurement
6.1. Software measurement
6.2. Product metrics

6.3. Process and resource metrics
6.4. Methodologies and standards for measurement

7. Software evolution
7.1. Software evolution
7.2. Evolution dynamics
7.3. Software maintenance
7.4. Types of maintenance
7.5. Maintenance costs
7.6. Maintenance forecasting
7.7. Maintenance metrics
7.8. Maintenance solutions
7.8.1. Main activities and costs
7.8.2. Reverse engineering, reengineering and refactoring

8. Software quality
8.1. Introduction
8.2. Quality standards
8.3. Product quality
8.4. Process quality
8.4.1. Quality assurance
8.4.2. The CMMI model
8.4.3. SPICE model: ISO/IEC 15504 standard
8.4.4. ISO 9000
8.5. Inspections and audits

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	0,00
Individual or group project	30,00
Independent study and work	20,00
Preparation of lessons	25,00
Preparation for assessment activities	15,00
Resolution of case studies	0,00
Total hours	**90,00**

TEACHING METHODOLOGY

The development of the course is structured around the following areas: theoretical lessons, problem sessions and practical exercises, laboratory sessions and activities and practical work to be performed by the student individually and in small groups in order to translate the knowledge acquired during the lessons of theory and problems.

- Theoretical lessons. In the lectures will develop the fundamental contents of the subject providing a global and inclusive vision, analyzing in detail the key issues and more complex. This media is used (such as presentations, transparency, chalkboard, etc.) at all times encouraging student participation.

- Practical exercises and problems. The lectures will be complemented by practical activities and problem solving in the classroom in order to verify and reinforce concepts presented earlier. He planned several types of activities and problems:

- **Individual Activities:** What will be done individually by students in order to assess individual understanding of matter. Encourage the involvement of students who will be responsible for his explanation and correction on the board and will be responsible to meet the concerns with the other partners.
- **Activities in small groups:** The problems will be solved by groups of students together to encourage teamwork and collective analysis of different approaches.

- Laboratory sessions. Consist in resolving problems related to the theoretical by software so that the student knows a practical way of implementing CASE tools development methodologies and software maintenance. The laboratory sessions will be organized around working groups of a maximum of five people.

- Individual student. Consist of performing work, issues and problems posed by the teacher to outside the classroom and class preparation in advance, reading recommended texts and exam preparation (study). This will be done individually and try to promote self-employment. Some of these activities shall be voluntary and additional training of the subject.

- Work in small groups. At the beginning of the course is set up small groups of 3 or 4 students who will remain fixed throughout the course. These groups will have to perform work or solve problems outside the classroom, to supplement individual work, building the capacity of integration into working groups. The nature of this work will be diverse, but include the submission of a report of work performed, the exposure of such work and class participation as reviewers of the papers presented by other groups.

- Tutoring. The students have a schedule of tutorials aimed at solving the problems, doubts, work orientation, etc. The schedule of these tutorials will be indicated at the beginning of the academic year. They will also have the opportunity to clarify some questions via email or discussion forums by using the tool "Virtual Classroom", which gives the University of Valencia.

EVALUATION

Knowledge assessment will be done in two ways:

1) CONTINUOUS EVALUATION

Recommended method for students. The following factors are evaluated to obtain the final mark:

- 60% theoretical knowledge and problems (TEO).

- 25% of the laboratory (LAB)

- 15% of additional work (ART)

It will be necessary that the final mark is equal to or superior to 5 to pass.

a) Theoretical knowledge and problems (TEO).

- The note of theoretical knowledge and problems are assessed according to the following factors: 80% OF INDIVIDUAL TESTS OBJECTIVES. During the course there will be different written tests on theoretical knowledge and problems. It will be necessary to get a grade of 5 or higher in each test so that you can compensate. In the final examination of the first call, those parts that have not been passed in the partial tests will have to be recovered.
- 20% OF PROBLEMS. We will evaluate the different problems that are proposed to the students, either to perform in class or at home. This activity is not recoverable.

b) Laboratory (LAB).

The laboratory grade will be obtained by averaging the grade obtained in the N practical sessions. In order to obtain the grade of the laboratory it will be necessary to have presented all the practices and to have attended a minimum 80% of the classes. It will be necessary to obtain a mark of 4,5 or more in each laboratory session so that this part can be compensated.

c) Additional works (TRA). The note of additional works will be obtained by averaging the grades obtained in each of the works by the weight assigned to each one. The note of each work will be obtained in function of the written memory, and optionally it will be possible to value the public exhibition of the work done.

2) SINGLE EVALUATION SYSTEM AND SECOND CALL

This method will apply to any student who, for a reason reasoned and admitted by the teacher, can not attend regularly to classes and in the second call. The following factors are evaluated to obtain the final mark:

- 60% theoretical knowledge and problems (TEO).

- 25% of the laboratory (LAB)

- 15% of additional work (ART)

It will be necessary that the final mark is equal to or superior to 5 to pass.

a) Theoretical knowledge and problems (TEO).

The note of theoretical knowledge and problems are assessed by a single examination, not taking into account other factors such as attendance or problem exercises performed during the course.

b) Laboratory (LAB).

The laboratory grade will be obtained by averaging the grade obtained in the practical sessions, which must have been delivered, even if you have not attended the laboratory sessions. It will be necessary to obtain a mark of 4,5 or more in each laboratory session so that this part can be compensated.

c) Additional works (TRA). The note of additional works will be obtained by averaging the grades obtained in each of the works by the weight assigned to each one. The note of each work will be obtained in function of the written memory, and optionally it will be possible to value the public exhibition of the work done.

In any case, the evaluation of this subject will be done in compliance with the University Regulations in this regard, approved by the Governing Council on 30th May 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

- Sommerville, I. (2016). Software Engineering (10th ed.). Pearson.
- Project Management Institute. A Guide to the Project Management Body of Knowledge. 4th edition, Project Management Institute (2008), ISBN: 19-33890517
- Sanchez, S; Sicilia, M.A; Rodriguez, D. Ingeniería del Software un enfoque desde la guía SWEBOK. Gaceta grupo editorial; ISBN: 978-8492812400
- Ministerio de Administraciones Públicas. (2001). Métrica versión 3. Madrid: Secretaría de Estado para la Administración Pública.
- Menzinsky, A.; López, G.; Palacio, J.; Scrum Master: Temario troncal 1. Versión 3.052 (junio 2021) Lubaris Info 4 Media S.L (https://www.scrummanager.net/bok/)
- Menzinsky, A.; López, G.; Palacio, J.; Historias de usuario: Ingeniería de requisitos ágiles. Versión 3.0 (Septiembre 2020) Lubaris Info 4 Media S.L (https://www.scrummanager.net/bok/)

- Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice (3rd ed.). Addison-Wesley
- Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach (8th Edition). McGraw-Hill Education.
- Piattini, M., García, F., Garzas, J., Genero, M. (2008). Medición y Estimación del Software: Técnicas y Métodos para Mejorar la Calidad y Productividad. Ra-Ma.
- Piattini, M., García, F., Caballero, I. (2006). Calidad de los Sistemas Informáticos. Ra-Ma.