COURSE DATA

DATA SUBJECT

Code: 34661
Name: Software engineering I
Cycle: Undergraduate Studies
ECTS Credits: 6
Academic year: 2025-26

STUDY (S)

Degree	Center	Acad. year	Period
1400 - Degree in Computer Engineering	Escola Tècnica Superior d'Enginyeria	2	Second quarter
1936 - Double Degree Program in Mathematics-Telematics Engineering	Facultat de Ciències Matemàtiques	3	Second 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	Tercer curso	COMPULSORY

COORDINATION

RIERA LOPEZ JOSE VICENTE

SUMMARY

The course "Software Engineering" is a core course as part of the field "Software Engineering and Project Management" of the Computer Engineering Degree. The course workload is 6 ECTS and it's offered in the 2nd semester of 2nd year.

The aim of the course is to introduce students in the development of software projects by following a systematic process and relying on tools to improve software quality in production environments.

It will introduce students to the knowledge and use of different methodologies for developing information systems.

We seek to provide sufficient knowledge of the software process, so that students, using the Unified Process, will be able to capture requirements, analyze, design, implement, test and deploy software projects in a concrete and accuracy way.

Concerning the practical part, in this course students will be able to implement the knowledge acquired in the theoretical part using UML modeling language and Java programming language.

The main objective of this course is to introduce students to the development of software projects from requirements analysis to implementation and verification of the product by the customer, relating to the following points:

- Understand the origin and meaning of the term "Software Engineering", its historical development and current challenges (with attention to the sociocultural context of development), and be aware of the ethical and professional responsibility of a Software Engineer.
- Become aware of the importance of always performing the analysis and design of the problem, as prior tasks to implementation in a programming language.
- Be aware of the need of modeling and abstraction in software development.
- Understand the concept of software development method and its main classifications.
- Distinguish the concepts of diagram and model.
- Know the main UML diagrams: use cases, classes, packages, objects, interaction (sequence and communication), states and activities, and be able to apply them in order to model a medium sized project.
- Given an application of medium size, be able to address the requirements analysis focused on use cases, the conceptual or domain modeling, and the analysis of collaborations between objects with appropriate allocation of responsibilities, specially taking into account technological details.
- Understand and apply design techniques within the framework of an iterative process.
- Choose the best option between different data conceptual designs, justifying and arguing the decision.
- Know and apply basic design patterns for building software and evaluate its role as a way of reuse of experience.
- Use software tools that allow the creation of different UML diagrams.

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 highly recommended:

- To have passed the subjects of the first course Informatics (34653) and Programming (34656)
- To be enrolled or to have passed the subject User Environments (34660)

COMPETENCES / LEARNING OUTCOMES

-

G1 - Ability to design, write, organise, plan, develop and sign projects in the field of computer engineering aimed at the design, development or exploitation of computer systems, services and applications.

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.

G4 - Ability to define, evaluate and select hardware and software platforms for the development and implementation of computer systems, services and applications, in accordance with both the knowledge and the specific skills acquired in the degree.

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.

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.

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.

R8 - Ability to analyse, design, build and maintain applications in a robust, secure and efficient manner by choosing the most suitable paradigm and programming languages.

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. Introduction to Software Development Process UML

Skills to be acquired:

- Understand what is software engineering and its need
- Know and understand the fundamental concepts that comprise the basic terminology of software engineering
- Understanding the relationships between the concepts of software process, software lifecycle and software methodology
- Knowing the characteristics and explain the advantages and disadvantages of different software process models
- Know the main types of software methodologies

- Know the basic features of a general process software development
- Understand what is software modeling and its benefits
- Recognize UML as a standard language to build software

Contents:

1.1 Overview of Software Engineering
1.2 Basics of Software Engineering
1.3 Software Process Models
1.4 Software Modeling
1.5 The Unified Modeling Language UML 2.0
1.5.1 UML Framework
1.5.2 UML Views
1.6 A Process of OO Software Development
1.6.1 Phases
1.6.2 Activities and Artifacts

Laboratory:

All sessions

2. Planning Phase

Skills to be acquired:

- Understand the value of acquiring and managing requirements and their influence on the success of a project
- Understand what requirements are and the complexity of requirements extraction
- Learn the requirements activities
- Identify the different types of requirements and be able to discern between them
- Learn about diverse elicitation techniques to capture system requirements
- Understand what is the Requirements Document
- Know the IEEE / ANSI 830-199 for SRS
- Develop a SRS document for medium size systems
- Learn the different elements and diagrams that UML provides to represent Use Cases
- Represent Functional Requirements with Use Cases
- Accomplish detailed specification of Use Cases

Contents:

2.1. Requirements
2.1.1 Definition and characteristics of the Requirements
2.1.2 Functional vs. Non Functional Requirements
2.1.3 Software Requirements Document

2.1.4 Exercises on Requirements
2.2 Prototype
2.3 Use Cases
2.3.1 Introduction.
2.3.2 Actors
2.3.3 Use Case Specification
2.3.4 Relations: generalization, extension, including
2.3.5 Use Case Diagrams
2.3.6 Standard errors and Recommendations
2.3.7 Exercises on use cases.

Laboratory:

Session 1: Working on Use Case Diagrams

3. Analysis

Skills to be acquired:

- Know the steps required to accomplish the analysis phase in the first cycle of development and the artifacts to be generated
- Be able to develop the Data Dictionary
- Be able to abstract the relevant concepts to develop a Conceptual Model
- Develop using Class Diagrams the conceptual model of a system
- Identify system events in Use Cases descriptions to extract System Operations
- Develop System Sequence Diagrams for Use Cases starting from Use Case Expanded Specification
- Develop Contracts for System Operations

Contents:

Part I:

3.1 Introduction
3.2 Class Diagram
3.2.1 Classifiers
3.2.2 Classes
3.2.3 Interfaces
3.2.4 Relations dependency, generalization, association, realization
3.3 Conceptual Model
3.4 Exercises on class and object diagrams

Part II:

3.5 Interactions
3.6 Sequence Diagrams
3.5.1 Elements
3.5.2 Modeling Interaction Diagrams
3.5.3 Lifecycle Application
3.7 System General Sequence Diagrams
3.8 Contracts
3.9 Sequence diagrams and contracts Exercises

Laboratory:

Session 2: Working on Class Diagrams
Session 3: Working on Sequence Diagrams
Session 4: Working on Life Cycle 1 Design
Session 7: Working on Life Cycle 2 Analysis & Design

4. Design

Skills to be acquired:

- Know the steps required to accomplish the design phase in the first cycle of development and the artifacts to be generated
- Understand the concept of responsibility
- Understand and know how to apply a set of patterns when deciding responsibilities assignment to classes
- Be able to develop interaction diagrams for each system operation following its contract
- Develop the Design Class Diagram from the Conceptual Model

Contents:

4.1 System Design
4.1.1 Responsibilities
4.1.2 Design Sequence Diagrams
4.1.3 Design Class Diagrams
4.1.4 Patterns for the allocation of responsibilities
4.3 Exercises

Laboratory:

Session 4: Working on Life Cycle 1 Design
Session 7: Working on Life Cycle 2 Analysis & Design

5. Implementation

Skills to be acquired:

- Learn prior decisions before implementing
- Know the types of transformation from model space to code space
- Transform design artifacts into code
- Detect models modification need for system optimization

Contents:

5.1 Prior Decisions
5.2 Types of transformation
5.2.1 Model Transformations
5.2.2 Code Transformations
5.2.3 Model to Code Transformations: direct Engineering
5.2.4 Code to Model Transformations: reverse Engineering
5.3 Direct Engineering
5.3.1 Mapping Classes
5.3.2 Mapping Relations
5.3.3 Mapping Inheritance
5.3.4 Methods Creation
5.3.5 Mapping Contracts
5.4 Implementation Exercises

Laboratory:

Session 6: Working on Life Cycle 1 Implementation
Session 8: Working on Life Cycle 2 Implementation

6. System Architecture

Skills to be acquired:

- Know the steps required to accomplish the design phase in the second cycle of development and the artifacts to be generated
- Understand the concepts of layers, packages and partitions and how to use in organizing the system architecture
- Represent packages and their relationships in Package Diagrams
- Choose the architecture to be used and model it using Packages Diagrams
- Knowing and applying other patterns

Contents:

6.1 Multilayer Architecture & UML
6.2 Patterns for connecting packages
6.3 Exercises

Laboratory:

Session 7: Working on Life Cycle 2 Analysis & Design

7. Complex behavior and State/Activity Diagrams

Skills to be acquired:

- Represent complex behaviors using State and Activity Diagrams

Contents:

7.1 Activity Elements
7.2 Activity Diagrams
7.3 State Machine Elements
7.4 State Diagrams
7.5 Activity Diagrams and State Exercises

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

The knowledge acquired by the student body will be evaluated as follows:

Their participation in the different activities, the degree of achievement obtained in the different training activities and the involvement shown towards their own learning process will be evaluated on a regular basis. For this, the following aspects will be assessed:

- (C) Continuous evaluation, based on the participation and degree of involvement in the teaching-learning process, taking into account regular attendance at the planned face-to-face activities

and completion of the work. As activities within the continuous evaluation, the students will carry out individually a set of bulletins of practical exercises or theoretical development that will be delivered through the virtual classroom within the established period for it. In addition, test-type controls or short questions of a part of the subject will be carried out. Finally, throughout the course a software project will be developed that will be presented at the end of the course in class. All these activities will give rise to the continuous evaluation mark as follows:

C (Continuous Assessment Grade) = 0.4 * Controls + 0.4 *Newsletters + 0.2 * Project Presentation

Activities delivered after the deadline will not be taken into account, nor can activities not carried out be recovered. Cheating in any of the activities will be strictly penalized, canceling all the student's continuous assessment notes.

- (P) Evaluation of practices. 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.
- (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.

The final grade will be obtained by applying the following formula:

Note = 0.35*C + 0.35*E+0.3*P

The marks of the group activities will not necessarily be the same for all the members of the group, and may vary depending on the involvement of each student.

Only work delivered on the date stipulated by the faculty will be considered. This includes each and every one of the activities, questionnaires and exercises proposed, the software project and, in general, any task that is assigned to the students.

The continuous part is not recoverable on second call. The P and E parts only average if a minimum of 5 is reached in each of them.

The final grade in the second call is calculated in the same way (0,35*C + 0,35*E+0,3*P).

In case of not having passed E or P with a mark higher than 5 and having taken the E test, the mark in the minutes will be computed as:

Final Grade = minimum(E,P,4)

In case of not showing up for E, the final grade is Not Shown.

None of the notes will be saved from one year to the next.

The teachers of the subject will have the right at any time to request an oral explanation of the work

presented, whether in the continuous evaluation or in the practices, and may ask the students to make small modifications in order to check the correctness. acquisition of valued knowledge.

Given that the grade for the continuous assessment part is not recoverable, in order to request an early call, students must have previously completed the subject and have passed the continuous assessment.

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).

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).

REFERENCES

- Apuntes de la asignatura
- [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]
- [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]
- [Kenneth E. Kendall, Julie E Kendall (2010)] Systems Analysis and Design, 8th Edition (Prentice Hall)
- [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)