COURSE DATA

Data Subject	
Code	34886
Name	Software Engineering
Cycle	Grade
ECTS Credits	6.0
Academic year	2020 - 2021

Study (s)

Degree	Center	Acad. year	Period
1403 - Degree in Telematics Engineering	School of Engineering	3	First term

Subject-matter

Degree	Subject-matter	Character
1403 - Degree in Telematics Engineering	11 - Software engineering	Obligatory

Coordination

Name	Department
RUEDA PASCUAL, SILVIA	240 - Computer Science

SUMMARY

The course "Software Engineering" is a core course as part of the field "Software Engineering and Project Management" of the Telematics Engineering Degree. The course workload is 6 ECTSand it’s offered in the 3rd semester of 3rd 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.

In regard to 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 an essential requirement:

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)

OUTCOMES

1403 - Degree in Telematics Engineering

- R1 - Ability for self-learning of new knowledge and techniques appropriate for the conception, development and exploitation of telecommunications systems and services.
- 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.
- 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.

LEARNING OUTCOMES

Learning goals of the course:

- Apply methods for developing, implementing and maintaining information systems. **G4, G9, R1, R7, E3 and E4 skills**
- Successfully plan and execute software development process iterative. **G4, G9, R1, R7, E3 and E4 skills**
- Know how to apply software design patterns in each situation depending on the needs of the software development project. **G4, R1, E3 and E4 skills**
- Define testing requirements validation and verification. **G4, R1, E3 and E4 skills**
- Obtain user and software requirements. **G4, G9, R1, E3 and E4 skills**
- Knowing the basics, development processes, methods and tools of software engineering. **G3 skill**
- Know and understand the current paradigms of software engineering aimed at developing distributed software, free software engineering and Web engineering. **G4, R1 and E4 skills**
- Know the different architecture models, which can be integrated technologies and business solutions to form a distributed solution particular. **E3 and E4 skills**

It is also pretended in this course to further develop the following skills:

- Analyze a software development problem and derive its nature specifically and accurately. **G4, G9, R1 and E3 skills**
- Design a structure of modules, using design patterns to solve problems and evaluate alternatives. **G4, G9, R1 and E3 skills.**

- Implement a module to run properly and efficiently. **R7, E3 and E4 skills**
- Test applications systematically defining comprehensive test cases. **E4 skill**
- Work in a small team, collaborating on the issues of software development, exchanging ideas constructively and organized. **G4, G9, R1, R7, E3 and E4 skills**

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: Prototyping UGI
Session 2: 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 Interacciones
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 3: Working on Class Diagrams
Session 4: Working on Sequence Diagrams
Session 5: 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 5: 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. Testing

Skills to be acquired:
- Understand and differentiate the key issues related to software testing
- Understand the need for testing as an essential part of software system development
- Distinguish the different testing levels according on the purpose
- Learn different software testing techniques

Contents:
7.1 Basics: Errors, Defects, Failures, Test Cases
7.2 Verification and Validation
7.2.1 Software Inspections
7.2.2 Software Testing
7.2.3 Debugging
7.3 Software Testing Levels
7.3.1 Unit Testing
7.3.2 Integration Testing
7.3.3 System Tests
7.3.4 Validation Testing
7.4 Software Testing Techniques
7.4.1 Black-Box Testing
7.4.2 White-Box Testing
7.5 Test Plan

WORKLOAD

ACTIVITY	Hours	% To be attended
Theory classes	30,00	100
Laboratory practices	20,00	100
Classroom practices	10,00	100
Attendance at events and external activities	3,00	0
Development of group work	14,00	0
Development of individual work	6,00	0
Study and independent work	7,00	0
Preparing lectures	20,00	0
Preparation of practical classes and problem	30,00	0
Resolution of case studies	7,00	0
Resolution of online questionnaires	3,00	0
TOTAL	**150,00**	

TEACHING METHODOLOGY

The formative activities will be developed according to the following distribution:

- **Theoretical activities.**

Using the flipped classroom, the theoretical contents of the subject will be provided through online materials, consisting on presentations and videos, that students must consult individually prior to synchronous sessions. During the theoretical synchronous sessions, students doubts and common mistakes will be resolved, providing a global and integrative vision.

- **Practical activities**

The practical activities complement the theoretical classes and allow the students to put into practice the contents and improve the understanding of the course concepts. They include the following types of activities:

- Synchronous problem sessions
- Discussion sessions and problem solving and exercises
- ynchronous practices and seminars
- Group work for planning and developing software projects and generating group dynamics
- Support tutorial sessions in group and individual, scheduled and on demand

- **Personal work**

Preparation of classes and works (study). Students should work at home with the material in order to be prepare for the flipped classroom. This work should be done individually and tries to promote autonomous work habits.

- **Teamwork in groups**

It will be carried out by groups of students (5-6). It consists of work to be done along the class timetable in form of exercises and problems. This work complements individual work and practical activities and fosters integration capabilities in working groups and leadership skills. They include the following types of activities:

- Group work for software project development, from requirements elicitation to software implementation. The work will include planning, generating group dynamics and documentation (which will have to be presented in Spanish, Valencian and/or English)
- Presentation of the software project (in Spanish, Valencian and/or English)
- Support tutorial sessions in groups, scheduled and on demand

During the course the e-learning platform (Aula Virtual) of the University of Valencia will be used to support the teaching activities. This platform allows the access to the course materials used in the classes as well as additional documents, solved problems and exercises.

EVALUATION

The knowledge acquired by the students 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 regularly evaluated. For this, the following aspects will be assessed:

- **Continuous evaluation**: the participation and use of the proposed asynchronous activities (consulting materials, training questionnaires, ...) and the participation and involvement in the development and resolution of the planned synchronous activities (face-to-face or online) (*N_Continuous*) both in theoretical-practical sessions as in laboratory sessions. **G4, G9, R1, R7, E3 and E4 skills**
- **Project**: both the quality of the solution and the documentation and defense of this (*N_Project*) will be assessed. **G4, G9, R1, R7, E3 and E4 skills**
- **Individual final activity**: consisting of an individual final test about the developed project (*N_Activity*). **G4, R1 and E3 skills**

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

Note = 25% N_Continuous + 65% N_Project + 10% N_Activity

The marks of 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 works submitted before the date stipulated by the teaching staff will be considered. This includes every single one of the proposed activities, questionnaires and exercises, the software project and, in general, any task assigned to the students.

A minimum mark of 4,5 points (out of 10) for each part (*N_Continuous*, *N_Project*, *N_Activity*) is required to obtain a final average mark.

Due to the characteristics of the activities involved in *N_Continuous* mark, this part of the grade is not recoverable, keeping the grade obtained for the 2nd call, but eliminating the minimum grade requirement in this part. In other words, on 2nd call the grade *N_Continuous* may be less than 4.5 but not the rest of the grades *N_Project* and *N_Activity*.

To obtain a grade in the continuous evaluation grade (*N_Continuous*), an attendance rate for synchronous practical and laboratory sessions higher to 75% will be necessary. If, for a justified and admitted by the teaching staff reason, a student could not attend to the synchronous sessions regularly, and, as long as they participate in the asynchronous tasks of continuous evaluation, this requirement may be eliminated.

Given that the qualification of the continuous assessment part is not recoverable, to apply for an advance call, students must have previously taken the course and have passed the continuous assessment. In this case, students must complete the project for the course and the final activity. The final grade will be obtained as follows:

Note = 25% N_Continuous + 65% N_Project + 10% N_Activity

In any case, the evaluation of the subject will be done in accordance with the Regulation of evaluation and qualification of the University of Valencia for the degree and master degrees approved by the Governing Council of May 30, 2017 (ACGUV 108/2017).

In accordance with the regulations of the Universitat de València, the carrying out of fraudulent actions in a test or part of it will result in a grade of zero in it, regardless of the disciplinary procedure that may be opened and the sanction that may be coming in accordance with current regulations.

REFERENCES

Basic

- Apuntes de la asignatura
- [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]

Additional

- [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)

ADDENDUM COVID-19

This addendum will only be activated if the health situation requires so and with the prior agreement of the Governing Council

In the event of a closure of the facilities due to the health situation that affects all or part of the classes of the subject, these will be replaced by non-presential sessions following the established schedules. If the closure affects a course evaluation test, it will be replaced by a test of a similar nature that will be carried out in virtual mode through the University of Valencia's institutional support tools. The percentages of each evaluation test will remain unchanged, as established by this guide