COURSE DATA

Data Subject

Code	34661
Name	Software Engineering I
Cycle	Grade
ECTS Credits	6.0
Academic year	2017 - 2018

Study (s)

Degree	Center	Acad. year	Period
1400 - Degree in Computer Engineering	School of Engineering	2	Second term

Subject-matter

Degree	Subject-matter	Character
1400 - Degree in Computer Engineering	7 - Software engineering and project management	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 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.

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

Without prerequisites for enrollment, it is recommended to have completed the following courses / fields:

Programming (each course the subjects taught to the previous year)
User Environments

OUTCOMES

1400 - Degree in Computer Engineering

- 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.
- 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.
- R16 - Knowledge and application of the principles, methodologies and life cycles of software engineering.
- TI2 - Ability to select, design, implement, integrate, evaluate, build, manage, exploit and maintain hardware, software and network technologies, within adequate cost and quality thresholds.
- SI3 - Ability to actively participate in the specification, design, implementation and maintenance of information and communication systems.

LEARNING OUTCOMES

Learning goals of the course:

- Apply methods for developing, implementing and maintaining information systems.
- Successfully plan and execute software development process iterative.
- Know how to apply software design patterns in each situation depending on the needs of the software development project
- Define testing requirements validation and verification
- Obtain user and software requirements
- Develop and expose technical documentation of projects in English

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.
- Design a structure of modules, using design patterns to solve problems and evaluate alternatives.
    - Implement a module to run properly and efficiently.
    - Test applications systematically defining comprehensive test cases.

Work in a small team, collaborating on the issues of software development, exchanging ideas constructively and organized.

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

ACTIVITY	Hours	% To be attended
Theory classes	30,00	100
Laboratory practices	20,00	100
Classroom practices	10,00	100
Development of group work	4,00	0
Study and independent work	4,00	0
Preparing lectures	23,00	0
Preparation of practical classes and problem	50,00	0
Resolution of case studies	9,00	0
TOTAL	**150,00**	

TEACHING METHODOLOGY

Teaching activities will be conducted in accordance with the following distribution:

- **Theoretical activities.**

The lectures will present the course contents providing a global vision, a detailed analysis of the key concepts and encouraging the student participation.

- **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 classroom activities:

- Solving problems in class.
- Regular discussion of exercises and problems that the students have previously tried to work out.
- Workshops and seminars in computer lab.
- Group work for project planning and software development and generation of group dynamics.
- Support tutorial sessions (individualized).

To carry out these activities, the general group will be subdivided into smaller subgroups (20 students maximum) according to need.

- **Personal work.**

Preparation of classes and exams (study). This is done individually and tries to promote autonomous work habits.

- **Teamwork in small groups.**

It will be carried out by small groups of students (3-4). It consists of work to be done out of the class timetable in form of exercises and problems. This work tries to improve the teamwork and leadership skills. They include the following types of activities:

- Practical work in group on requirements elicitation with different techniques.
- Presentation of group work (in Castilian).
- Group work for project planning and software development and generation of group dynamics, which documentation must be submitted in Castilian and/or English.
- Presentation of the software project (in Castilian and / or English).
- Support tutorial sessions (groups).

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

Students can choose between two different assessments:

- Continuous assessment system
- Single assessment system.

Continuous assessment system

This will be the method recommended to students. This system allows regular assessment of students' participation, their exploit of training activities and their participation in the learning process.

Following aspects will be valued:

- **Theory-practical sessions:** involvement will be assessed, taking into account regular attendance to planned classroom activities, delivery of the exercises and participation in their resolution (Theory_M).
- **Teamwork:** the quality of the documentation and oral defense carried out by the small teamwork will be valued (Teamwork_M).
- **Laboratory sessions:** involvement will be assessed, taking into account regular attendance to planned classroom activities, delivery of the proposed exercises and evaluation questionnaires (Laboratory _M).
- **Project:** the quality of the solution developed and the documentation and oral defense carried out will be valued (Project_M).
- **Individual examinations:** consisting of partial examinations and a final exam (Exam_M).

To apply such an assessment a class attendance rate above 75% will be required. This percentage will be applied separately to each block. That is, students should attend to more than 75% of theory-practice sessions and more than 75% of the laboratory sessions.

The final mark will be obtained applying the following formula:

Final Mark=20% Continuous_M +40% Project_M +40% Exam_M

Continuous_M =37.5% Theory_M +25% Teamwork *M +37.5% Laboratory_M*

Exam_M =20% partials + 80% final

In order to take the final exam, notes *Continuous_M* and Teamwork *M* will have to be greater than or equal to 4,5.

Single assessment system

This method applies to any student who cannot attend classes regularly due to a reasonable and supported by professor cause or who already fail the first session continuous assessment.

Following aspects will be valued:

- **Theory-practical sessions:** taking into account delivery of the exercises (Theory_M).
- **Teamwork:** the quality of the documentation and oral defense carried out by the small teamwork will be valued (Teamwork_M).
- **Laboratory sessions:** involvement will be assessed, taking into account regular attendance to planned classroom activities, delivery of the proposed exercises and evaluation questionnaires (Laboratory _M).
- **Project:** the quality of the solution developed and the documentation and oral defense carried out will be valued (Project_M).
- **Individual examinations:** consisting of partial examinations and a final exam (Exam_M).

To apply such an assessment a laboratory sessionsa class attendance rate above 75% will be required.

The final mark will be obtained applying the following formula:

Final Mark=10% Continuous_M +40% Project_M +50% Exam_M

Continuous_M =37.5% Theory_M +25% Teamwork *M +37.5% Laboratory_M*

Exam_M =20% partials + 80% final

In order to take the final exam, *Project_M* note will have to be greater than or equal to 4,5.

In both methods, only works submitted before the date stipulated by the teacher will be considered. This includes exercises in class (theory and practical sessions), laboratory exercises and questionnaires, the work in small groups and the software project.

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

A minimum mark of 4,5 (out of 10) for each part (Theory_M, Teamwork _M, Laboratory _M, Project_M and Exam_M) is required to obtain a final average mark.

The grade of N_Continua is not recoverable. The grade is maintained in 2º call.

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).

According to the regulations of the University of Valencia, the performance of fraudulent actions in a test or part of it will result in the qualification of a zero in the same, irrespective of the disciplinary procedure that can be opened and of the sanction that is according to the current legislation.

To apply for an advance call, students must have previously taken the course. In this way, it is attempted to reconcile the right of students to an advance call with the subject's teaching methodology and evaluation criteria. In addition to the individual examination, students must demonstrate the accomplishment of a work equivalent to the course project.

REFERENCES

Basic

- Apuntes de la asignatura
- [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)
- [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)
- [Bernd Bruegge, Allen H. Dutoit] Object-Oriented Software Engineering Using UML, Patterns, and Java, 3rd Edition (Edition Prentice Hall) [Recurs electronic: http://proquest.safaribooksonline.com/0131489062?uicode=valencia]

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]