Communications Software

Academic Year: (2019 / 2020) **Review date: 24/04/2019 09:17:47**

Department assigned to the subject: Telematic Engineering Department

Coordinating teacher: IBAÑEZ ESPIGA, MARIA BLANCA

Type: Compulsory ECTS Credits : 6.0

Year : 4 Semester : 1

REQUIREMENTS (SUBJECTS THAT ARE ASSUMED TO BE KNOWN)

Programming; Systems Programming

OBJECTIVES

By the end of the course, the student will be able to understand principles, concepts, methods, and techniques of the software engineering approach. The student will apply his/her knowledge and skills to organize and develop a medium-sized software project that satisfy requirements, is reliable, easy to understand, modify and maintain.
To achieve these objectives, the student must acquire a body of knowledge, skills and attitudes.
As regards of knowledge, at the end of the course the student will know:

- What is software engineering.
- The set of software processes and software processes models.
- Different ways of expressing software requirements.
- System models that may be developed during the requirements engineering process.
- Models of software architectures.
- Main software design patterns.
- Techniques used to test software.
- How to estimate quality and costs of software

In terms of capabilities, it can be classified into two groups: specific and generic skills.
As for specific skills after the course, students will be able to:

- Planning a complete software project.
- Analyze and formalize the software requirements of a project.
- Develop and analyze models of software.
- Design the software architecture of a project.
- Using the data structures and programming techniques suitable for programming tasks that the software requires.
- Encourage the maintenance of the system at all stages of development through proper documentation.
- Verify and validate the software developed.
- Ensure software quality.

In terms of general skills, in the course we will work:

- The ability to use software engineering techniques to create effective software products.
- The ability to cope with restrictions arising from human resources, time and costs.
- The ability to work in a software development team.
- The ability to public present and defend the developed solutions.

In terms of attitudes, the student after completing the course should have:

- A critical attitude regarding the suitability of different techniques and tools that can be applied to development of a software system.
- Ethical values to his work as a software developer.
- A proactive attitude towards work.
- An attitude of cooperation and teamwork.

DESCRIPTION OF CONTENTS: PROGRAMME

1. Introduction
   1.1 Introduction to Software Engineering
   1.2 Software processes
2. Requirements analysis
   2.1 Software requirements
   2.2 System models
3. Software architecture design.
   3.1 Models of software architectures.
   3.2 Software patterns
4. Software testing. Techniques
   4.1 Introduction to software testing
   4.2 Types of software testing

LEARNING ACTIVITIES AND METHODOLOGY

The course comprises the following teaching methodology: lectures, practical sessions and laboratories supervised by teachers. Activities to be carried out:
. Lectures, where professors present the main topics listed in the syllabi. At these sessions, the teacher will encourage interaction and discussion of key issues raised.
. Practical sessions, where professors will propose real problems to be analyzed and solved by students.
. Programming laboratories, where students implement their project under teaching staff supervision. These sessions are extra work carried out by students for a period of several days. Students are assisted by the teaching staff through multiple channels to resolve their doubts.

ASSESSMENT SYSTEM

% end-of-term-examination/test:	40
% of continuous assessment (assigments, laboratory, practicals…):	60

The work of the students will be evaluated by using continuous evaluation. The following scoring will be used:
* There will a written midterm evaluation (20% of student course grade). The exam will contain study questions provided by the instructor, and "practical" problems, based on project problems.
* Evaluation of a team project (40% of student course grade). The team will be evaluated by the teacher and their peers considering:
  o The project plan presented in written form by the team.
  o Team partial project advances presented in written form
  o Student involvemente into the project
  o Oral presentation of team work.

The contribution of each team member will be valued separately.
* The final exam (40%) will contain study questions provided by the instructor, and "practical" problems, based on project problems. This exam is mandatory for all students. Students must earn a grade of at least 4 (4/10) in order to pass the subject. The final grade must be higher than 5 (5/10).
* If a student chooses not to follow the continuous evaluation, he/she must take the final exam. In this case, the grade obtained in the exam is 100% of the final grade.

BASIC BIBLIOGRAPHY

- E. Gamma, R. Helm, R. Johnson, J. Vlissides Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.

- Ian Sommerville Software Engineering, Addison-Wesley.

- J. Rumbaugh, I. Jacobson, G. Booch The Unified Modeling Language Reference Manual, Addisson-Wesley.

ADDITIONAL BIBLIOGRAPHY

- ACM Ingeniería de Software. Código Ética y Práctica Profesional 5.2, ., .

- Bruegge, B; Dutoit, A Object Oriented Software Engineering Using UML, Patterns and Java, Pearson Education Internationsl, 2004

- C. Ebert, et al. Best Practices in Software Measurement. How to Use Metrics to Improve Project and Process Performance, Springer.

- Dashofy, E; Madyidovic, N; Taylor R, Software Architecture: Foundations, Theory and Practice, John Wiley & Sons, 2009

- Fowler, M; Beck, K.; Opdyke, W; Roberts; D. Refactoring Improving the Design of Existing Code, Addison-Wesley Professional, 1999

- Fox, A; Patterson D. Engineering Software as a Service: An Agile Approach Using Cloud Computing, Strawberry Canyon, 2014

- Jorgesen, P Software Testing. A Craftsman´s Approach, CRC Press, 2013

- Martin, R. Clean COde, Prentice Hall, 2008

- W. Perry Effective Methods for Software Testing, Wiley.