Universitat Autònoma de Barcelona

2019/2020

Software Development Management

Code: 102789
ECTS Credits: 3

Degree	Type	Year	Semester
2502441 Computer Engineering	OB	3	2
2502441 Computer Engineering	OT	4	2

Errata

"The contact of the subject is Xavier Otazu Porter (Xavier.Otazu@uab.cat)"

Contact

Name: Francisco Javier Sánchez Pujadas
Email: Javier.Sanchez.Pujadas@uab.cat

Use of Languages

Principal working language: catalan (cat)
Some groups entirely in English: No
Some groups entirely in Catalan: Yes
Some groups entirely in Spanish: No

Prerequisites

There is no prerequisite to take this course. But it is also advisable to study the subjects:

- Architecture and software technology
- Integrated software laboratory

Objectives and Contextualisation

In the software engineering subject of the second course you have received an overview of the software engineering process. You have learned the basics and, for those who have studied the subjects of the first semester of the mention of software engineering, you would have deepened into some of these concepts, such as the analysis of requirements, design and test and control techniques of quality

In this subject you will learn how to deal with the changes. Every software development process is subject to changes of all kinds. Changes might be caused by the need for improvement of the developed product. Changes in initial requirements. Changes in terms or work team, budget, etc. In short, at the end of this course you will have to be able to manage any change that supposes a deviation from the initial planning.

More specifically, the objectives of this course are:

- Deepen in software control and development techniques.
- Deepening in the knowledge of the administration techniques necessary to plan, monitor and control software projects.
- Learn to use some of the existing tools to carry out the necessary control of the software development.

Finally, for the correct follow-up of this course, the teaching team assumes that all students have achieved the skills explained on software engineering.

Competences

Computer Engineering

- Acquire personal work habits.
- Acquire thinking habits.
- Capacity to design, develop, evaluate and ensure the accessibility, ergonomics, usability and security of computer systems, services and applications, as well as of the information that they manage.
- Have the capacity to solve integration problems in accordance with available strategies, standards and technologies.
- Have the capacity to solve problems with initiative, decision making, autonomy and creativity. Have the capacity to know how to communicate and transmit the knowledge and skills of the IT engineering profession.

Learning Outcomes

1. Define and manage the documentation generated during the development of a software application.
2. Develop a capacity for analysis, synthesis and prospection.
3. Develop a mode of thought and critical reasoning.
4. Plan the integration of the different components developed in the coding process.
5. With initiative and autonomy, resolve problems with the integration of software.
6. Work independently.

Content

BLOCK 1: Configuration Management (CG)

1. Introduction. Concepts of the CG
GC configuration and aggregate items. Versions and configurations. Requests for changes. Promotions and releases. Deposits and work spaces. Version identification schemes. GC tools.

2. Activities of the CG
Configuration and identification items of the aggregates of the GC. Management of the promotion. Release management. Management of the branches. Management of variants. Change management

3. Managing the CG
Documentation of the GC. Assignment of responsibilities in the GC. Organize the activities of the GC.

BLOCK 2: Project Management (PG)

4. Introduction. Concepts of the GP
Tasks and activities. Products, packages and roles. Task model. Profiles array. Organization structures: type and visualization.

5. PG activities
Planning, organization, control and completion

6. PG methodologies
Royce Iterative Methodology and incremental development. Agile development

Methodology

The student's learning process will be based on these two types of activities:

Theory and Problems. With the case study approach, students will practice in class the theoretical concepts that will be introduced at the beginning of the class through documentary and whiteboard support. Next, the students will work in small groups and will discuss among them the possible solutions until concluding and choosing one of them. In the final section of the class, all the solutions proposed will be shared.

Practices. The practical classes will work in groups of two people and the use of the necessary configuration management tools in all software development projects will be practiced.

All the documentation and material needed to track the course will be available on the Virtual Campus.

Cross-curricular competences: The cross-curricular competences will be worked out and evaluated at various moments throughout the course. Some examples of how they will work are the following:

- T01.01 - Develop a way of thinking and critical reasoning:
  The theory classes explain examples of problems that appear in the management of software development. They explain the solutions that historically have been applied and the pros, cons and errors are valued.
  In practice, students must build a software development model within a project and be able to find a strategy to manage it.
  In the theory test there are questions that force the student to make arguments about the subject to find the answer.
- T02.03 - Manage time and available resources . Work in an organized way:
  This course deals with how to manage time, resources and organize work within a software development project.
  In the second practice, the students must specify the resources, the temporary planning and what will be the methodology used of a software project.
- T02.04 - Preventing and solving problems:
  One of the subjects of the subject is about how to control a project and prevent its risks.
  In the first practice students must design a repository thinking about the problems that may arise during the development of a specific software.
  In the second practice students must manage a project including the possible changes in the specifications that will be made during the development.
- T02.05 - Take own decisions:
  In practice, students must make decisions about how to organize a software repository and how to organize a project.

Activities

Title	Hours	ECTS	Learning Outcomes
Type: Directed			
Practical sessions	6	0.24	1, 2, 3, 4
Problem classes	13	0.52	5
Theory classes	13	0.52	1, 2, 3, 4, 5
Type: Supervised			
Tutorials	9	0.36	1, 4, 5
Type: Autonomous			
Preparation and completion of practical exercises	15	0.6	1, 4, 5, 6
Preparation of exercises and resolution of theoretical assumptions	15	0.6	3, 5, 6

Assessment

The subject consists of 2 types of assessments: theory (NTeo) and practices (NLab). The weight of each of these evaluations is as follows:

NFinal = 0.5 * NTeo + 0.5 * NLab

NLab = 0.5 * NLab1 (SPMP) + 0.5 * NLab2

To calculate the NFinal note, the marks of each one of the different evaluations (NTeo and NLab) should be equal or greater than 5. Otherwise, the subject is not passed and the final grade of the subject will be the minimum mark between NTeo and NLab.

If NTeo is less than 5 in the final examen, the student can have a second chance in the recovery exam.

To calculate the NLab note, each and every one of the NLab (i) marks, that is, each and every one of the practice deliveries, must be higher than 4. Otherwise, NLab is the minimum note of the notes NLab (i). The evaluations through practical cases and the writing of practical reports are part of the practice evaluation block (NLab). The delivery dates and the delivery method of the different case studies will be recorded in the same statements. There is no second call for the NLab note or any of the NLab (i) notes, that is, there is no second call for the laboratory sessions.

For each assessment activity, a place, date and time of revision will be indicated in which the student will be able to review the activity with the teacher. In this context, claims can be made about the activity note, which will be evaluated by the teachers responsible for the subject. If the student does not submit to this review, this activity will not be reviewed later.

A student is considered Non-Appraising only if the assessment activities he has undertaken are approved (but he has not done all the activities of the course evaluation). If an evaluation activity is not passed, the subject is not passed (regardless of the number of activities that have been presented).

The honor grade can be obtained with a mark average superior or equal to 9.0. Due to the fact that there is a limited number of honor grades that can be given by group, they will be awarded in order of note from major to minor.

Only the practice note (NLab) of previous years will be kept between years.

We will notify the students using the Virtual Campus where they will always be informed about possible changes in the course's development since it is understood to be the usual mechanism for exchanging information between teacher and students.

Notwithstanding other disciplinary measures deemed appropriate, and in accordance with the current academic regulations, irregularities committed by a student that may lead to a variation of the qualification will be classified by zero (0). For example, plagiarizing, copying, copying, ... will involve suspending this evaluation activity with zero (0). Assessment activities qualified in this way and by this procedure will not be recoverable. If it is necessary to pass any of these assessment activities to pass the subject, this subject will be suspended directly, without opportunity to recover it in the same course.

Assessment Activities

Title	Weighting	Hours	ECTS	Learning Outcomes
Exam of Theory	50%	3	0.12	1, 2, 3, 4, 5, 6
Reports / work practices	50%	1	0.04	1, 2, 3, 4, 5, 6

Bibliography

- Bruegge, B. I Dutoit A.H. Object-Oriented Software Engineering. Pearson, Prentice-Hall, 2004.
- Berczuk, S. and Appleton B.Software Configuration Management Patterns: Effective Teamwork, Practical Integration. Addison-Wesley, 2002.