Universitat Autònoma de Barcelona

2020/2021

Software Engineering

Code: 102743
ECTS Credits: 6

Degree	Type	Year	Semester
2502441 Computer Engineering	OB	2	2

The proposed teaching and assessment methodology that appear in the guide may be subject to changes as a result of the restrictions to face-to-face class attendance imposed by the health authorities.

Contact

Name: Josep Lladós Canet
Email: Josep.Llados@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: Yes

Other comments on languages

Some of the practicals groups are in English.

Teachers

Alicia Fornes Bisquerra

Prerequisites

Since the subject is an introduction to the block of software engineering subjects, it will be assumed that there is no prior knowledge about the subject. It is the responsibility of the subject to provide the students with a means to acquire the knowledge described in the contents section of the subject (section 6 of this guide). In spite of this it is advisable to have passed the subjects of Foundations of Computing and of Methodology of the Programming of first course, and the subject of Databases of second course.

Objectives and Contextualisation

This subject introduces the block of subjects on the software engineering process. The general objective is to give a global and orderly view of the software development process that is not simply that of programming (which is, only, a phase within the whole process of software engineering). The fundamental activities that constitute the development process will be introduced (requirements analysis, design, implementation, testing and maintenance), most of which are treated in more depth in subsequent subjects.

It is, therefore, that the student acquires an overview of what is a software development process, how software is modeled, what tools are used, how quality is managed and how a project is managed. More specifically, the objectives are:

- Provide an overview of the software development process, paradigms, activities, and objectives of each activity.

- Know what software modeling is The importance and difficulty of the requirements specification and the design in its different views.
- Learn the basic concepts of UML to work in a practical way in software modeling.
- Introduce the student to the principles, concepts and techniques that are applied to manage and control the quality of the software.
- Be familiar with the management techniques required to plan, organize, monitor and control software projects.
- Become familiar with the SCRUM model as a paradigm of more relevant evolutionary development.

Competences

- Analyse, design, build and maintain robust, safe and efficient applications, and select the most suitable programming paradigms and languages.
- 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 conceive, develop and maintain computer systems, services and applications employing the methods of software engineering as an instrument to ensure quality.
- Have the capacity to design and evaluate person-computer interfaces that guarantee the accessibility and usability of computer systems, services and applications.
- 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.
- Have the capacity to understand the importance of negotiation, effective work habits, leadership and communication skills in all environments of software development.
- Know and apply the principles, methodologies and lifecycles of software engineering.
- Work in teams.

Learning Outcomes

1. Accept and respect the role of the various team members, and its different levels of dependence.
2. Design and evaluate person-computer software interfaces that guarantee the accessibility and usability of computer systems, services and applications.
3. Design and maintain applications observing robustness and reliability criteria.
4. Design applications while selecting the most suitable software development paradigm.
5. Identify, manage and resolve conflicts.
6. Know and apply effective communication and negotiation methods when performing ones tasks in the profession.
7. Know how to notate and build UML diagrams.
8. Understand requirement analysis and why it is important.
9. Understand the importance of the negotiation, effective work habits, leadership and communication skills in all environments of software development, to an advanced level.
10. Understand the process of developing RUP software.
11. Understand what design is and how to achieve a good design that includes the accessibility, ergonomic, usability and security requirements of computer applications.
12. Understand what software , what are the goals that are set to develop a software and problems associated with their development.
13. Work cooperatively.

Content

BLOCK 1. THE SOFTWARE AND SOFTWARE ENGINEERING

Topic 1. Principles of software engineering.

- Definition and objectives of the SE. Definition of software. Characteristics of the software. Software applications. Definition of SE. Objectives of the SE.
- Evolution of the software. Stages. Software crisis: problems and causes.
- Process, method and tool. Definitions. Activities in the software development process.
- Software development paradigms. Sequential linear model (classical life cycle). Prototype model. Evolutionary model. Spiral model.

BLOCK 2. SOFTWARE MODELING

Topic 2. Software Requirements Analysis.

- Introduction. Type of requirements. Tasks to perform.
- Understanding the problem. Communication techniques. Associated problems. Principles of analysis.
- Specification of requirements. Desirable properties of an ER. SE standards. Review and validation of the specification.

Topic 3. Design of the Software.

- Introduction. Design process. Data design, architectural design, interface design, procedural design. Principles (objectives) of design.
- Design concepts. Abstraction. Modularity. Refinement.
- Effective modular design. Functional independence. Cohesion. Coupling. Heuristics for effective modular design.
- User Interface Design.

Topic 4. UML (Unified Modeling Language)

- Evolution and history of UML.
- UML views. Use case view, logical view, component view, deployment view.

BLOCK 3. ADMINISTRATION OF THE QUALITY OF THE SOFTWARE

Topic 5. Software quality.

- Introduction. Quality concepts
- Software testing. Test models. Test strategies. Problems of a bad software design. Basic principles of OO testing. TDD: Test-Driven Development.
- Administration of the software configuration.

BLOCK 4. ADMINISTRATION AND DEVELOPMENT OF SOFTWARE PROJECTS

Topic 6. SCRUM.

- Introduction. Agile Methodology for Project Management.
- SCRUM. Characteristics. History
- SCRUM components. Roles Artefacts Processes
- Scalability Scrum of Scrums.
- Software Tools.
- Other agile models. Lean. Xtreme programming. Kanban
- A case study. Presentation of case study to follow during the course.

Methodology

The course of Software Engineering has a strong applied component. For this reason, after introducing the fundamental theoretical concepts, the subject is developed according to an ABP (Problem Based Learning) methodology. In this way, the student will acquire the knowledge from a practical case (problem guide) of real dimension. The guide problem will guide the learning process throughout the course, so that the

practical work or the seminars will be structured around this case. The student must acquire the analyst / architect profile of the software to make the necessary decisions to conclude in the construction of a proposed design of the proposed system. The teacher will take the role of tutor and, in certain situations, of client, fomenting simulations of developer-client dialogue.

The subject consists of 4 hours per week. There is no distinction between theory hours, problems and laboratory practices. During class hours, theory classes, problems or practices will be alternated according to the planning that will be indicated on the day of the planning of the subject. In general, the explanation of the fundamental theoretical contents will be concentrated during the first sessions of the course (approximately the first 6 weeks). Once the theoretical contents are distributed, all sessions will be devoted to work based on problems, identifying some compulsory attendance sessions where the practice will be carried out on teams of 5 people or they will be made problems at the end of the session.

THEORY

It consists of master classes with multimedia material available at the Virtual Campus of the UAB. The main objective of these classes is tointroduce the basic notions about the processes, methods and tools related to the vision of the software life cycle from an engineering point of view. The principles and paradigms of software engineering, the different activities and the tasks to be carried out in each activity will be exposed. The theory classes must allow the student to take a real vision of rigor, planning and systematic that requires the professional development of the software. It will not be deepened in the subjects since in subjects of later courses these already are developed, but the necessary concepts will be explained so that the student understands the process of development of complete software. The theory sessions will assume that the students follow the materials distributed through the virtual campus, so that they will be oriented to explain the key concepts, and to foment the critical debate on the contents.

SEMINARS/PROBLEMS

The seminars are classes with participatory work of the students. According to the objective of the seminars, the group's students will be divided into smaller subgroups. They have a dual function, linking on one side with the master classes and on the other with the practical sessions on the problem to be solved in teams. On the one hand, the seminars will complement the technological contents exposed to the master classes, completing their understanding. On the other hand, they will be the framework for discussing, laying the foundations and developing the guide problem. The student will be given the necessary knowledge to carry out the practical work or will be indicated where to obtain them. The seminars essentially promote the capacity for analysis and synthesis, as well as the critical reasoning and decision making of the student in the resolution of the guide problem. Since itis a very practical subject and where the theoretical content is of low complexity, practical work is often reinforced with solving problems or the work of participatory activities. The contents of the seminars will be coordinated with the stages to be resolved in the work of the practice. Since theory sessions, problems and practices will be alternated throughout the course, for a better organization, the students will have at the beginning of the course through the virtual campus of the programming of all the sessions.

FOLLOW UP OF PRACTICAL WORK

There will be no face-to-face sessions of the laboratory. Students will solve the guide problem autonomously in small groups, and the monitoring will be done during class hours during regular hours. The practice bands (6 sessions of 2 hours during the course) will consist of two parts. First, each practice team will present to the rest of the class the progress of its work (the one in charge of presenting will be the person that does of scrum master in that iteration). In the second part of the session, the practice tutor will have an individual interview with each team to follow up and evaluate. A development based on SCRUM is followed, where each increment, called sprint, corresponds to a session. Assistance to the follow-up interviews will be COMPULSORY for all students. The non-presence will adversely affect the evaluation of the practical work, entailing not exceeding the evaluation if the number of non-presences exceeds 15% of the sessions. In case of justified absence, the teaching staff will consider mechanisms of compensation. The size of the practice teams will be determined at the beginning ofthe course. As an indication they will be teams of 5 people. Since attendance to the follow-up sessions is mandatory, and it will be done during the attendance hours, it is important that all the members of a team of internships are enrolled and attend the same group.

PRACTICAL WORK

Students will be given a guiding problem at the beginning of the course. This problem will have two functions. First, use it as an example in theory classes and seminars, solving appropriate parts for the contents of each moment. Secondly, it can be understood as the statement of a practice that is resolved by stages throughout the course. The students will be organized in work teams of 5 people (according to number of students enrolled). The resolution of the work will be carried out autonomously, with the weekly follow-up described above.

ACHIEVEMENT OF COMPETENCES

- **T03 - Work as a team.** Work will be done in the practical work where students will train teams of 5-6 people, where teamwork is considered one of the competencies to work. Students will have to distribute the roles they would have in a software project, and each week one of them will perform the scrum master. It will be evaluated as one of the indicators of practical work.
- **CRI03.03 - Understand the importance of negotiation, effective work habits, leadership and communication skills in all software development environments, at an advanced level:** Specific problem-specific sessions will be worked on where technical solutions will be proposed. Catch requirements, in particular gamestorming, and also in practice, where the work groups are organized according to the principles of the SCRUM methodology, and every week (session) will require one of the team members to assume the SCRUM master role (coordinator). It will be evaluated based on the individualized monitoring of the practical work (from the actions that the Scrum master makes of the follow-up meetings).
- **CRI08.01 - Design and maintain applications according to criteria of robustness and reliability and CRI08.02 - Design applications knowing how to select the most appropriate software development paradigm:** They will work both in face-to-face sessions, in particular of problems, as well as practical work. It will be assessed with the written test (theory exam, first control) and with the qualification of the practical work.
- **CRI16.01 - Understand what is the analysis of requirements and why it is important and CRI16.02 - Understand the process of agile software development:** The first sessions of the theory and problems course (first four weeks) will be worked out as well as in the first practicesessions (first two sessions). They will be assessed with the written test (theory exam, second control) and with the qualification of the practical work.
- **CRI17.02 - Understand the notation and construction of diagrams UML and E03.02 - Design and evaluate person-computer software interfaces that guarantee the accessibility and usability of the systems, services and computer applications:** They will work in the central sessions of the course of theory and problems (week 5 to week 10), as well as in practice sessions (sessions 3 to 5). They will be assessed with the written test (theory exam, second control) and with the qualification of the practical work. Optional exercises will be offered to students to be given to have additional scores in the exam corresponding to this part.
- **E03.01 - Understand what design is and how to achieve a good design that includes the accessibility, ergonomics, usability and security requirements of computer applications and E03.02 - Design and evaluate person-computer software interfaces that guarantee the design, Accessibility and usability of the systems, services and computer applications:** As in the previous competition, the central sessions of the theory and problems course (week 5 to week 10), as well as in the practical sessions (sessions 3 to 5) . They will be assessed with the written test (theory exam, second control) and with the qualification of the practical work.
- **E05.01 - Understand what the software is, what are the objectives that are posed when developingsoftware and the problems associated with its development.** The final sessions of theory and problems will be worked (weeks 11 and 12), as well as in the practical sessions (session 6). It will be evaluated with the written test (theory exam, second control) and with the qualification of the practical work.
- **E09.04 - Know and apply methods of communication and effective negotiation in the accomplishment of the tasks typical of the profession.** Basically, it will work in practice sessions where students will have to defend in front of the teacher the decisions taken in their project. The teacher will do the role of client to simulate what would be a real project. It will be evaluated in the qualification of the practical work.

ONLINE LECTURES

Given the situation of uncertainty caused by Covid-19, and the possibility that not all teaching can be carried out in person, the lectures may have to be adapted to online mode through videoconferencing platforms. Any modification that affects the format and schedules will be reported through the virtual campus and at the beginning of the course.

Activities

Title	Hours	ECTS	Learning Outcomes
Type: Directed			
Preparation and study	30	1.2	11, 8, 10, 7, 4, 12
Seminars / Problems	15	0.6	8, 6, 7, 4
Theory lectures	30	1.2	11, 8, 10, 7, 12

Type: Supervised			
Follow up meetings for the practical work	15	0.6	1, 6, 5, 13
Tutoring	15	0.6	11, 8, 10, 7, 12
Type: Autonomous			
Practical work	38	1.52	1, 9, 8, 6, 7, 5, 13

Assessment

The evaluation will be carried out based on the practical work developed by the student and included in his portfolio (final resolution of the guide problem, material generated in the seminar activities), the degree of involvement in seminars and knowledge Technological achievements from the subject to study. The following evaluation instruments will be contemplated:

- Individual assessment (AI). On-the-spot written test to assess the degree of knowledge acquired by the individual student. There will be two partial tests during the course that will allow the release of matter as long as the mark obtained is equal to or greater than 5. The note corresponding to each partial has a weight of 50% in the final mark of the individualindivual assessment .

So, AI = 50% Note_Partial1 + 50% Note_Partial2.

- Collective evaluation of the work teams (AC). Proof of the joint performance of the different workinggroups to solve the guide problem andthe different activities presented at the seminars. It consists of two instruments:
  AC1 (technical evaluation). Documentation delivered by students for their practical work. The technical resolution of the guide problem will be evaluated.
  AC2 (evaluation of progress and competencies). The defense of work on the part of the students in the deliveries, the monitoring of the evolution of the work and the active participation in the seminars. It will be evaluated based on interviews with the teacher, short written tests, deliveries or exhibitions of problems and cases raised in the activities.Although this instrument is a collective evaluation, at the discretion of the tutor, it can be corrected individually in cases where the student's participation (high or low) exceeds his / her group.

It should be noted that while the AI and AC1 instruments value the student's performance and the level of achievement of the knowledge, the AC2 instrument values the student's effort.

INDICATORS AND VALUATION:

The final grade will be obtained according to the following scale:

QF = 50% AI + 50% AC

To pass the subject it is necessary to have achieved a minimum score of 5 in the two grades (AI and AC), as well as in the partial tests to release material that is established throughout the course. At the discretion of the teacher, however, it may be possible to establish compensation between the notes of the partial deliveries of the activities (corresponding to AC).

The subject will be evaluated as a Not evaluated only in the case that the student has not been submitted to any of the assessment tests contemplated nor has he delivered all or part of the work.

In case of do not passing the subject because any of the assessment activities do not reach the minimum grade required, the numerical note of the file will be the lowest value between 4.5 and the weighted average of the notes. With the exceptions that the "non-evaluable" qualification will be awarded to students who do not

participate in any of the assessment activities, and that the numerical note of the file will be the lowest value between 3.0 and the average Weighted notes in case the student has committed irregularities in an evaluation act (and therefore not approved for compensation).

They will be awarded with Honor mark within the maximum levels admitted by the UAB regulations (depending on the number of students enrolled) at higher grades equal to or greaterthan 9.

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.

See section "PLAGIARISM" in cases of irregularities due to plagiarism in the evaluation activities.

VOLUNTARY EVALUATION ACTIVITIES:

During the course, students will be offered optional delivery activities that will serve to complement the notes of the mandatory activities.

REVISIONS:

- Individual evaluation (AI). There will be two partial examinations of liberation theory during teaching hours. Students who do not pass this test (with a mark equal to or greater than 5), will have a revision test on the final evaluation date programmed for the degree.
- Collective evaluation (AC). The practical work is evaluated in the form of a continuous evaluation in the follow-up sessions. Therefore there will be no revision activity at the end of the course. However, with respect to the resolution of the guide problem, the fact of not delivering on time or not exceeding one of the intermediate deliveries can be recovered reprogramming the planning of the work of the subsequent sessions during the course, according to the professor.

EVALUATION DATES:

The dates of continuous assessment and work delivery will be published in Virtual campus and may be subject to changes of programming for reasons of adaptation to possible incidents. Always the virtual campus will be informed about these changes as it is understood to be the usual mechanism for exchanging information between teacher and students.

REPEATING STUDENTS:

Partial notes (theory or practices) are not saved from one course to the next. However, according to the professor's criteria and in accordance with previous course evaluations, compensation may be established. This information will be announced on the day of the presentation of the subject, and on the virtual campus.

PLAGIARISM:

Notwithstanding other disciplinary measures that are considered opportune, and in accordance with the current academic regulations, the irregularities committed by a student who can lead to a variation of the qualification will be qualified with zero (0). Les Assessment activities qualified in this way and by this procedurewill 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 opportunityof recover it in the same course. These irregularities include, among others:
- the total or partial copy of a practice, report, or any other evaluation activity;
- let copy;
- present a group work not done entirely by the members of the group;
- Present as own materials prepared by a third party, though are translations or adaptations, and generally works with non-original and exclusive elements of the student;
- Have communication devices (such as mobile phones, smartwatches , etc.) accessible during the theoretical evaluation tests or the individual practices (exams).

In summary: copying or plagiarizing in any of the assessment activities is equivalent to a FAIL with a score of less than 3.0.

FINAL DISPOSAL:

For any doubt or discrepancy, the most up-to-date information that will be communicatedon the day of the presentation of the subject will be prevailed and will be published on the virtual campus.

Assessment Activities

Title	Weighting	Hours	ECTS	Learning Outcomes
Individual evaluation (theory exam)	50	3	0.12	1, 11, 8, 10, 6, 7, 2, 12, 5, 13
Joint evaluation (practical work)	50	4	0.16	9, 8, 10, 6, 7, 4, 3

Bibliography

The contents of the subject explained in the lectures are extracted from different sources. Many materials are online, taken from multimedia materials, videos, and so on. During the course, the materials of own preparation necessary for the follow-up of the subject will be provided through the virtual campus. Links to online documentation, software for practical exercises, etc. will also be provided. in free format. Listed below are some classic software engineering books. However, students are recommended to follow the subject with the updated materials that will be recommended in class.

- G. Booch, J. Rumbaugh, I. Jacobson. El lenguaje unificado de modelado. Addison-Wesley, 2001.
- I. Jacobson, G. Booch, J. Rumbaugh. El proceso unificado de desarrollo de software. Addison-Wesley, 2000.
- J. Rumbaugh , I. Jacobson, G. Booch. El lenguaje unificado de modelado: manual de referencia. Addison-Wesley, 2007.
- T. Quatrani. Visual Modeling with Rational Rose 2000 and UML. Addison-Wesley, 2000.
- P. Krutchen. The Rational Unified Process. An Introduction. Addison-Wesley, 2000.
- Roger S. Pressman, Ingeniería del software, un enfoque práctico. Mc Grah-Hill, 7a. edició , 2010.
- A. Álvarez García, R. de las Heras del Dedo, C. Lasa Gómez, Métodos Ágiles y Scrum. Anaya Multimedia, 2012.
- K. Pohl, C. Rupp. Requeriments Engineering Fundamentals. Rocky Nook Inc. 2011.