Universitat Autònoma de Barcelona

2022/2023

Software Engineering

Code: 104340
ECTS Credits: 6

Degree	Type	Year	Semester
2503758 Data Engineering	OB	3	1

Contact

Name: Xavier Otazu Porter
Email: xavier.otazu@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

Teachers

Joan Protasio Ramirez

Prerequisites

Since the subject is introductory to the block of software engineering subjects, it will be assumed that no prior knowledge of the subject is possessed. It is the responsibility of the subject itself to provide students with a means to acquire the knowledge described in the contents section of the subject (section 6 of this guide). Despite this, it is recommended to have passed the subjects of Fundamentals of Computer Science and Methodology of Programming in the first year, and the subject of Databases in the second year.

Objectives and Contextualisation

This subject is an introduction to the block of subjects on the software engineering process. The overall goal 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 entire software engineering process). The fundamental activities that make up the development process (requirements analysis, design, implementation, testing and maintenance) will be introduced, most of which will be covered in more depth in later subjects.

It is therefore a matter of the student acquiring an overview of what a software development process is, 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, the paradigms, activities, and goals of each activity.
Know what software modeling is. The importance and difficulty of specifying requirements and design in their different views.
Learn the basics of UML to work in a practical way in software modeling.
Introduce the student to the principles, concepts and techniques that are applied to administer and control the

quality of the software.
Introduce the administration techniques required to plan, organize, monitor, and control software projects.
Familiarize yourself with the SCRUM model as the most relevant evolutionary development paradigm.

Competences

- Design efficient algorithmic solutions to computational problems, implement them in the form of robust software developments which are structured and easy to maintain, and verify their validity.
- Make a critical evaluation of work carried out.
- Plan and manage the available time and resources.
- Students must have and understand knowledge of an area of study built on the basis of general secondary education, and while it relies on some advanced textbooks it also includes some aspects coming from the forefront of its field of study.

Learning Outcomes

1. Analyse applications' performance, detect bottlenecks and apply possible optimisations.
2. Analyse, design, distribute and maintain applications, ensuring their quality and maintainability.
3. Describe a software system and transform it into a design model.
4. Make a critical evaluation of work carried out.
5. Plan and manage the available time and resources.
6. Students must have and understand knowledge of an area of study built on the basis of general secondary education, and while it relies on some advanced textbooks it also includes some aspects coming from the forefront of its field of study.

Content

BLOCK 1. SOFTWARE AND SOFTWARE ENGINEERING

Topic 1. Principles of software engineering.

Definition and objectives of ES. Definition of software. Software features. Software applications. Definition of ES. Objectives of the ES.
Software evolution. 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). Prototyping model. Evolutionary model. Spiral model.

BLOCK 2. ADMINISTRATION AND DEVELOPMENT OF SOFTWARE PROJECTS

Topic 2. SCRUM.

Introduction. Agile project management methodology.
SCRUM. Characteristics. History.
SCRUM components. Roles. Artifacts. Processes.
Scalability. Scrum by Scrums.
Software tools.
Other agile models. Lean. Xtreme programming. Kanban.
A case study. Presentation of case studies to follow during the course.

[NOTE: Since it is necessary to acquire the knowledge of SCRUM for internships, this topic is addressed at this time of the course despite belonging to block 4 on project management]

BLOCK 3. MODELING OF SOFTWARE

Topic 3. Software Requirements Analysis.

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

Item 4. Software Design.

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 5. UML (Unified Modeling Language)

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

BLOCK 4. SOFTWARE QUALITY ADMINISTRATION

Topic 6. Software quality.

Introduction. Quality concepts.
Software test. Testing strategies, conventional application testing, web application testing.
Software configuration management.

BLOCK 5. ADMINISTRATION AND DEVELOPMENT OF SOFTWARE PROJECTS

Item 7. Software project management.

Project management concepts.
Planning.
Risk management.
Maintenance and reengineering.

Methodology

The subject of Software Engineering has a strong applied component, but also requires a theoretical part, the result of experience, from which practical decisions are implemented. Therefore, the assignment will not be purely ABP (Problem Based Learning) but a mixture of ABP and traditional method. The student will be proposed a practical case (generic problem) of real dimension, around which the problem sessions and the practical work will be structured. The student will have to acquire the profile of analyst / architect of the software to take the opportune decisions that conclude in the construction of a proposal of design of the proposed system. The teacher will play the role of tutor and, in certain situations, of client, encouraging simulations of developer-client dialogue.

FOLLOW-UP PRACTICAL WORK

There will be no face-to-face lab sessions. Students will solve the guiding problem autonomously in small groups, and will be monitored during class hours on a regular basis. The practice strips (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 their work (the person in charge of making the presentation will be the person who acts as the scrum master in that iteration). In the second part of the session the internship tutor will have an individual interview with each team to monitor and evaluate. A SCRUM-based development is followed, where each increment, called a sprint, corresponds to one session. Attendance at follow-up interviews will be MANDATORY for all students. The absence will have a negative impact on the evaluation of the practical work, leading to not passing the evaluation if the number of non-attendances exceeds 15% of the sessions. In the event of a justified absence, teachers will consider compensation mechanisms. The size of the internship teams will be determined at the beginning of the course. As a guide they will be teams of 5 people. Since attendance at follow-up sessionsis mandatory, and will be done during class hours, it is important that all members of an internship team are enrolled and attending the same group.

PRACTICAL WORK

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

ACHIEVEMENT OF COMPETENCES

T03 - Work in a team. We will work on practical work where students will form teams of 5-6 people, and where teamwork is considered one of the skills to work on. The students will have to distribute the roles that would be in a software project, and each week one of them will act as coordinator (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: Work will be done on specific problem sessions where requirements capture techniques will be proposed, in particular gamestorming, and also in practice, where working groups are organized following the principles of the SCRUM methodology, and in each week (session) one of the team members will be required to assume the role of SCRUM master (coordinator). It will be evaluated from the individualized follow-up of the practical work (from the acts that the Scrum master does of the follow-up meetings).
CRI08.01 - Design and maintain applications following 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 problems, and in practical work. It will be evaluated with the written test (theory exam, first control) as well as with the qualification of the practical work.
CRI16.01 - Understand what requirements analysis is and why it is important and CRI16.02 - Know the process of developing agile software: They will work on the first sessions of the theory and problems course (first four weeks), as well as in the first practice sessions (first two sessions). They will be evaluated with the written test (theory exam, second control) as well as with the qualification of the practical work.
CRI17.02 - Know the notation and construction of UML and E03.02 diagrams - Design and evaluate person-computer software interfaces that guarantee the accessibility and usability of computer systems, services and applications: The central sessions of the theory and problems course will be worked on (week 5 to week 10), as well as in the practice sessions (sessions 3 to 5). They will be evaluated with the written test (theory exam, second control) as well as with the qualification of the practical work. Students will be offered optional exercises that they can submit to have additional scores in the exam corresponding to this part.

The course consists of 4 hours per week in person. There is no distinction between theory schedules, problems, and laboratory practices. During the contact hours, theory, problem or practice classes 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), but some sessions of problems and practical cases will also be gradually interspersed. Once the theoretical contents have been taught, all the remaining sessions will be dedicated to problem-based work, identifying some compulsory attendance sessions where the practice will be carried out in teams of 5 people or problems will be delivered at the end of the session.

THEORY

It consists of master classes with multimedia material available on the UAB Virtual Campus. The main goal of these classes is to introduce the basics of processes, methods, and tools related to viewing the software lifecycle from an engineering standpoint. It will explain the principles and paradigms of software engineering, the different activities and the tasks to be performed in each activity. Theory classes should allow the student to take a real view of the rigor, planning and systematics required by the professional development of the software. It will not deepen in the concrete subjects since in asignaturas of later courses these already develop, but will explain the necessary concepts so that the student understand the process of development of complete software.

SEMINARS / PROBLEMS

Seminars are classes with student participatory work. Depending on the purpose of the seminars, the students of the group will be divided into smaller subgroups. They have a dual function, linking on the one hand 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 in the master classes, completing their understanding. On the other hand they will be the framework in which to discuss, lay the groundwork and develop the guiding problem. The student will be given the knowledge necessary to carry out the practical work or will be told where to get it. In the seminars the capacity of analysis and synthesis is promoted fundamentally, as well as the critical reasoning and the decision-making of the student in front of the resolution of the guide problem. Since this is a very practical subject and where the theoretical contents are of low complexity, the practical work with problem solving or the work of participatory activities is often reinforced. The contents of the seminars will be coordinated with the stages to be solved in the work of the practice.

E03.01 - Understand what design is and how to achieve a good design that includes the requirements of accessibility, ergonomics, usability and security of computer applications and E03.02 - Design and evaluate human-computer software interfaces that guarantee accessibility and usability of computer systems, services and applications: Like the previous competence, they will work in the central sessions of the course of theory and problems (week 5 to week 10), as well as in the sessions of practices (sessions 3 to 5). They will be evaluated with the written test (theory exam, second control) as well as with the qualification of the practical work.
E05.01 - Understand what software is, what are the objectives of developing software and the problems associated with its development. Work will be done in the final sessions of theory and problems (weeks 11 and 12), as well as in the practice sessions (session 6). It will be evaluated with the written test (theory exam, second control) as well as with the qualification of the practical work.
E09.04 - Know and apply methods of communication and effective negotiation in carrying out the tasks of the profession. Basically, work will be done in the practice sessions where students will have to defend in front of the teachers decisions taken in their project. The teacher will play the role of client to simulate what a real project would be. It will be evaluated in the qualification of the practical work.

Annotation: Within the schedule set by the centre or degree programme, 15 minutes of one class will be reserved for students to evaluate their lecturers and their courses or modules through questionnaires.

Activities

Title	Hours	ECTS	Learning Outcomes
Type: Directed			

Preparation and study	30	1.2	4, 5, 6
Seminars/problems	15	0.6	1, 2, 4, 3
Theory sessions	30	1.2	3, 6
Type: Supervised			
Interviews and support for practicum assignment	15	0.6	1, 2, 4, 3, 5
Supervision	15	0.6	4, 5
Type: Autonomous			
Practicum assignemt	38	1.52	1, 2, 4, 3, 5, 6

Assessment

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

Individual assessment (AI). Face-to-face written test to assess the degree of knowledge achieved by the student at the individual level. There will be two partial tests during the course that will allow to release matter as long as the mark obtained is equal or superior to 5. The note corresponding to each partial has a weight of 50% in the final note of the individual evaluation.

Therefore, AI = 0.5 Partial_Note1 + 0.5 Partial_Note2.

Collective evaluation of work teams (AC). Test of the joint performance of the different working groups to solve the guide problem and the different activities presented in the seminars. It consists of two instruments:
AC1 (technical evaluation). Documentation delivered by students of their practical work. The technical resolution of the guide problem will be evaluated.
AC2 (assessment of progress and competencies). The defense of the work by the students in the deliveries, the follow-up of the evolution of the work and the active participation in the seminars. It will be evaluated from the interviews with the teacher, short written tests, deliveries or expositions of problems and cases raised in the activities. Although this instrument is for collective assessment, at the discretion of the tutor, it can be corrected individually in cases where the student's participation stands out (due to excess or defect) with respect to their group.

It should be noted that while the AI and AC1 instrumentsassess the student's performance and the degree of knowledge achievement, the AC2 instrument assesses the student's effort.

INDICATORS AND EVALUATION:

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

QF = 0.4 AI + 0.6 AC

To pass the course 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 subject that are established throughout the course. At the discretion of the teacher, however, it will be possible to establish compensations between the notes of the partial deliveries of the activities (corresponding to AC).

The subject will be evaluated as Non-Evaluable only in the case that the student has not presented to any of

the proofs of evaluation contemplated nor has delivered totally or partially the works.

In case of not passing the subject due to the fact that some of the evaluation activities do not reach the required minimum mark, the numerical mark of the transcript will be the lower value between 4.5 and the weighted average of the marks. With the exceptions that the grade of "non-assessable" will be awarded to students who do not participate in any of the assessment activities, and that the numerical mark of the transcript will be zero (0) in case the the student has committed irregularities in an assessment act (and therefore the one approved for compensation will not be possible).

Honorary Enrollments will be awarded within the maximum allowed by UAB regulations (depending on the number of enrolled) to the highest marks equal to or greater than 9.

For each assessment activity, a place, date and time of review will be indicated in which the student will be able to review the activity with the teacher. In this context, claims may be made on the grade of the activity, which will be evaluated by the teacher responsible for the subject. If the student does not appear for this review,this activity will not be reviewed later.

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

RECOVERIES:

Individual assessment (AI). There will be two partial liberatory theory exams during school hours. Students who do not pass this test (with a grade equal to or higher than 5), will have a resit exam on the final assessment date scheduled by the degree. In this resit exam, the maximum grade that can be obtained is 6.
Collective evaluation (AC). The practical work is evaluated in the form of continuous assessment in the follow-up sessions. Therefore there will be no recovery activity at the end of the course. However, with respect to problem-solving guidance, failure to deliver on time or not to pass one of the intermediate deliveries can be recovered by rescheduling work planning for subsequent sessions during the course, according to the teacher.

EVALUATION DATES:

The dates of continuous evaluation and delivery of works will be published on the virtual campus and may be subject to scheduling changes for reasons of adaptation to possible incidents. He will always inform the virtual campus about these changes as it is understood to be the usual mechanism for exchanging information between teacher and students.

REPEATING STUDENTS:

No partial notes (theory or practice) are saved from one course to another. However, at the discretion of the teacher and based on previous course evaluations, compensations may be established.

PESTS:

Without prejudice to other disciplinary measures deemed appropriate, and in accordance with current academic regulations, irregularities committed by a student that may lead to a variation in the grade will be graded with a zero (0). For example, plagiarizing, copying, letting copy, ..., an evaluation activity, will imply suspending this evaluation activity with a 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 test the subject, this subject will be suspended directly with a grade of zero (0), without the opportunity to retake it in the same course. These irregularities include, but are not limited to:

the total or partial copy of a practice, report, or any other assessment activity
let copy

present group work not done entirely by group members
present as their own materials prepared by a third party, even if they are translations or adaptations, and in general works with non-original and exclusive elements of the student
have communication devices (such as mobile phones, smart watches, etc.) accessible during individual theoretical-practical assessment tests (exams).

A student who has committed irregularities in an assessment act will not be validated any of the assessment activities performed. In short: copying, copying or plagiarizing (or attempting to) in any of the assessment activities is equivalent to a SUSPENSION, with a non-compensable zero (0) and no validations of parts of the subject in later courses.

FINAL CLARIFICATION:

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

Assessment Activities

Title	Weighting	Hours	ECTS	Learning Outcomes
Groupal evaluation	60	4	0.16	1, 2, 4, 3, 5, 6
Individual evaluation	40	3	0.12	3, 6

Bibliography

- 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.

Atès que avui en dia molts materials estan en línia, molt més actualitzats que les fonts bibliogràfiques, durant el curs es proporcionen enllaços a documentació lliure a la xarxa, video-lectures, etc.

Software

ArgoUML (http://argouml.tigris.org/)

Eclipse (https://www.eclipse.org/ide/)