Course guide
340379 - AMEP-I4O23 - Further Software Engineering

Last modified: 08/02/2024

Unit in charge: Vilanova i la Geltrú School of Engineering
Teaching unit: 747 - ESSI - Department of Service and Information System Engineering.

Degree: BACHELOR'S DEGREE IN INFORMATICS ENGINEERING (Syllabus 2018). (Compulsory subject).

Academic year: 2023 **ECTS Credits:** 6.0 **Languages:** Catalan

LECTURER

Coordinating lecturer: Xavier Franch Gutiérrez

Others: Xavier Franch Gutiérrez
Lidia López Cuesta

PRIOR SKILLS

- Knowledge of the basic principles of Software Engineering (SE)
- Ability to specify, design and implement non-trivial programs using a plan-driven methodology (UML)
- Ability to participate in a software development project in small teams

REQUIREMENTS

Introduction to Software Engineering (INEP)

DEGREE COMPETENCES TO WHICH THE SUBJECT CONTRIBUTES

Specific:
1. CEFC12. Knowledge and application of characteristics, functions and structure of data base which allow suitable use and design and analysis and implementation of applications based on them.
2. CEFC13. Knowledge and application of necessary tools for storage, processing and access to informatic systems, including the ones based on webs.
3. CEFC16. Knowledge and application of principals, methodologies, life cycles of programming engineering.
4. CEFC6. Basic knowledge and application of algorithmic processes, informatic techniques to design solutions of problems, analyzing if proposed algorisms are apt and complex.
5. CEIS1. Ability to develop, to maintain and avaluate programming services and systems which satisfy all requirements of user having a reliable and efficient behavior, being comprehensible to develop and maintain and observe to current rules, applying theory, principals, methods, practices of pragramming engineering.
6. CETI2. Ability to select, design, develop, integrate, value, construct, tmanage, exploit and maintain technologies of machines, programming and nets, keeping suitable costs and quality parameters.

Transversal:
7. SELF-DIRECTED LEARNING - Level 1. Completing set tasks within established deadlines. Working with recommended information sources according to the guidelines set by lecturers.
8. SELF-DIRECTED LEARNING - Level 2: Completing set tasks based on the guidelines set by lecturers. Devoting the time needed to complete each task, including personal contributions and expanding on the recommended information sources.
9. SELF-DIRECTED LEARNING. Detecting gaps in one's knowledge and overcoming them through critical self-appraisal. Choosing the best path for broadening one's knowledge.
10. EFFICIENT ORAL AND WRITTEN COMMUNICATION - Level 1. Planning oral communication, answering questions properly and writing straightforward texts that are spelt correctly and are grammatically coherent.
11. EFFICIENT ORAL AND WRITTEN COMMUNICATION - Level 2. Using strategies for preparing and giving oral presentations. Writing texts and documents whose content is coherent, well structured and free of spelling and grammatical errors.
12. EFFICIENT ORAL AND WRITTEN COMMUNICATION. Communicating verbally and in writing about learning outcomes, thought-building and decision-making. Taking part in debates about issues related to the own field of specialization.
13. THIRD LANGUAGE. Learning a third language, preferably English, to a degree of oral and written fluency that fits in with the future needs of the graduates of each course.
14. TEAMWORK - Level 1. Working in a team and making positive contributions once the aims and group and individual responsibilities have been defined. Reaching joint decisions on the strategy to be followed.
15. TEAMWORK - Level 2. Contributing to the consolidation of a team by planning targets and working efficiently to favor communication, task assignment and cohesion.
16. TEAMWORK. Being able to work as a team player, either as a member or as a leader. Contributing to projects pragmatically and responsibly, by reaching commitments in accordance to the resources that are available.

TEACHING METHODOLOGY

- A case study that conveys the whole course.
- Material in the form of transparencies, for the study and review of the contents.
- A software development projects in groups of six to eight people as a fundamental learning and evaluation mechanism.
- Tutoring sessions for the work done in the laboratory classes.

LEARNING OBJECTIVES OF THE SUBJECT

- Learn more about the concept and importance of software engineering.
- Learn and apply a new approach (agile methodologies) for software development.
- Be able to do an entire medium-sized development in a professional-like environment: excessive requirements, lack of time, teamwork, continuous feedback.
- Recognize the importance of testing and documentation in software development.
- Learn to use some basic software development support tools

STUDY LOAD

Type	Hours	Percentage
Hours large group	30,0	20.00
Self study	90,0	60.00
Hours small group	30,0	20.00

Total learning time: 150 h

CONTENTS

Introduction to agile software development methodologies

Description:
Presents the basic concepts of the agile methodology that will be used throughout the course

Specific objectives:
- Review the basic concepts of software engineering (SE that apply to the course
- Introduce the basic concepts of agile software development methodologies, understanding the differences with the methodology directed by a plan studied at INEP
- Analyze the impact of these changes on the various stages of the software life cycle

Related activities:
Parcial Exam (ExPartial), Final Exam (ExFinal)

Full-or-part-time: 2h
Theory classes: 2h

Basic concepts of agile methodologies. The SCRUM methodology

Description:
Know and apply the basic concepts of agile methodologies in general, and Scrum in particular

Specific objectives:
- Know the concept and typology of backlogs in agile methodologies
- Know the concept of user story and similar (epic, ...) and its relationship with the classic concept of requirement
- Learn the properties of user stories, as well as the techniques for their prioritization
- Introduce the basic principles of Scrum
- Introduce roles in Scrum projects
- Present the key events of the agile ceremony in Scrum

Related activities:
Parcial Exam (ExPartial), Final Exam (ExFinal)

Full-or-part-time: 15h
Theory classes: 10h
Self study : 5h

Advanced techniques to support software development

Description:
Learn some techniques that can speed up the software development process and improve its quality

Specific objectives:
- Understand the concepts of continuous integration and continuous deployment and motivate their importance in the software development process
- Show a couple of tools to support these activities
- Discuss the limitations and challenges inherent in the adoption of these automation tools
- Present some basic support tools for software quality management
- Learn to make good use of code generation tools

Related activities:
Final Exam (ExFinal)

Full-or-part-time: 10h
Theory classes: 6h
Self study : 4h

Software testing. Documentation

Description:
Understand and apply basic code documentation and testing techniques

Specific objectives:
- Motivate and justify the need for software testing
- Enter the various types of tests that exist
- Visualize the relationship between software testing and the other stages of software engineering, in the context of agile methodologies. Definition of Done
- Recognize good practices associated with design and code quality
- Give basic guidelines for code documentation

Related activities:
Partial Exam (ExPartial), Final Exam (ExFinal)

Full-or-part-time: 14h
Theory classes: 8h
Self study : 6h

Conclusions

Description:
Summarize the knowledge and practices taught in the subject and advances the challenges that will be studied in the AMEP subject

Specific objectives:
- Summarize and relate the knowledge and practices taught in the subject
- List the limitations of the techniques seen
- Advancing some solutions to these limitations, and possible alternative techniques
- Give an overview of advanced methods not presented in the subject

Related activities:
Final Exam (ExFinal)

Full-or-part-time: 2h
Theory classes: 2h

Project development

Description:
Apply the knowledge introduced during the course in a laboratory project developed as a team

Specific objectives:
- Learn to effectively use a set of software development support tools suitable for the course objectives
- Learn to develop a software system incrementally and with a life cycle by iterations
- Apply the theoretical notions introduced in the theory class
- Face the typical challenges of programming a system by applying the principles and methods of software engineering with an agile approach

Related activities:
Project (Proj). Partial Exam (ExPartial), Final Exam (ExFinal)

Full-or-part-time: 105h
Laboratory classes: 30h
Self study : 75h

GRADING SYSTEM

The final grade of the subject is obtained applying the following formula:

Final Grade = 0.1*ExPartial+0.2*ExFinal+0.7*Proj,

where:
*ExPartial and ExFinal are the marks obtained in the two corresponding tests. Both exams will combine purely theoretical questions, and questions related to the development of project. In this second case, the aim will be to solve, in a limited time and individually, some of the aspects that, in a more complex way or with some nuance of difference, have already been solved during the realization of the project as a team.
* Proj is the project note. The project will be organized in four iterations, with one deliverable for each. The first iteration corresponds to the inception of the problem, and the other three correspond to various stages in the development of the software. Instructors will correct assignments and provide feedback to students, but will only grade the project at the end of the fourth assignment (which will include a presentation). Teachers can decide to assign different grades to the members of the same project development team based on various sources of information (reports at each iteration, traces in the development tools used, participation in the class dynamics, etc.).

Not Presented: any student who does not appear in the final exam and who does not appear as a member of the team in the final deliverable will be assessed as Not Presented

EXAMINATION RULES.

To pass the project, students must regularly attend laboratory classes except for justified reasons, and during the course they must demonstrate their individual contribution to the development of the project.

BIBLIOGRAPHY

Basic:
- Pressman, Roger S. Ingeniería del software : un enfoque práctico [on line]. 9a ed. México [etc.]: McGraw Hill, 2021 [Consultation: 16/02/2024]. Available on: https://www-ingebook-com.recursos.biblioteca.upc.edu/ib/NPcd/IB_BooksVis?cod_primaria=1000187&codigo_libro=4272. ISBN 9781456287726.
- Cohn, Mike. Agile estimating and planning [on line]. Upper Saddle River, NJ [etc.]: Prentice Hall Professional Technical Reference, 2006 [Consultation: 05/03/2024]. Available on: https://ebookcentral-proquest-com.recursos.biblioteca.upc.edu/lib/upcatalunya-ebooks/detail.action?pq-origsite=primo&docID=7114929. ISBN 0131479415.
- Rasmusson, Jonathan. The Agile samurai : how agile masters deliver great software. Raleigh ; Dallas: The Pragmatic Bookshelf, 2010. ISBN 9781934356586.
- Cohn, Mike. User stories applied : for agile software development [on line]. Boston, MA: Addison-Wesley, 2004 [Consultation: 01/03/2024]. Available on: https://ebookcentral-proquest-com.recursos.biblioteca.upc.edu/lib/upcatalunya-ebooks/detail.action?pq-origsite=primo&docID=7116223. ISBN 9780321205681.
- Larman, Craig. Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development. 3th ed. Upper Saddle River, N.J: Prentice Hall PTR, 2005. ISBN 0131489062.

RESOURCES

Other resources:
- Scrum Guide: https://www.scrum.org/resources/scrum-guide />
- Agile Manifesto: http://agilemanifesto.org/