



Universitat de Lleida

DEGREE CURRICULUM

SOFTWARE ENGINEERING

Coordination: SENDÍN VELOSO, MONTSERRAT

Academic year 2021-22

Subject's general information

Subject name	SOFTWARE ENGINEERING			
Code	102018			
Semester	1st Q(SEMESTER) CONTINUED EVALUATION			
Typology	Degree	Course	Character	Modality
	Bachelor's Degree in Computer Engineering	3	COMPULSORY	Attendance-based
	Double bachelor's degree: Degree in Computer Engineering and Degree in Business Administration and Management	3	COMPULSORY	Attendance-based
	Master's Degree in Informatics Engineering		COMPLEMENTARY TRAINING	Attendance-based
Course number of credits (ECTS)	6			
Type of activity, credits, and groups	Activity type	PRALAB	TEORIA	
	Number of credits	3	3	
	Number of groups	2	1	
Coordination	SENDÍN VELOSO, MONTSERRAT			
Department	COMPUTER SCIENCE AND INDUSTRIAL ENGINEERING			
Teaching load distribution between lectures and independent student work	40% Presential (equivalent to 60h) 60% Autonomous work (equivalent to 90h)			
Important information on data processing	Consult this link for more information.			
Language	Preferably Catalan (Spanish if any student shows difficulties with Catalan).			
Distribution of credits	Juan Manuel Gimeno Illa 4.5 Montserrat Sendin Veloso 4.5			

Teaching staff	E-mail addresses	Credits taught by teacher	Office and hour of attention
SENDÍN VELOSO, MONTSERRAT	montse.sendin@udl.cat	9	

Subject's extra information

Compulsory subject of 3rd year (1st quarter) that belongs to the common studies in the computer science branch.

Matter: Analysis and Design of Applications.

RECOMMENDATIONS: We assume the student knows the concepts about object-oriented programming and data structures taught in Programming II and Data Structures.

Learning objectives

- Knowing the conceptual basis and the different aspects of the discipline, among other the software lifecycle process model
- Apply the Use Case technique
- Specifying in a textual way the functional and non functional needs for a certain software system planned by means of a statement and/or other inputs from the user
- Developing the classes diagram for a certain software system following the Object Oriented Modeling principles
- Be familiar with a UML-based modeling tool
- Understanding the concept of code as a something that evolves over time
- Be able to program basic unit tests
- Understanding the object oriented design fundamental principles
- Recognizing the concept of responsibility as a fundamental one when planning an object oriented design

Competences

Cross-disciplinary competences

- **EPS-11:** Capacity to understand the needs of the user expressed in a no technical language

Specific competences

- **GII-CRI2:** Capacity to plan, conceive, deploy and direct projects, services and computer systems in all the fields, leading his set up and his continuous improvement and evaluation his economic and social impact
- **GII-CRI12:** Knowledge and application of the characteristics, functionalities and structure of the databases, that allow their suitable use, and the design and the analysis and implementation of applications based in them
- **GII-CRI13:** Knowledge and application of the necessary tools for the storage, processing and access to the Systems of information, including those based in web
- **GII-CRI16:** Knowledge and application of the principles, methodologies and life cycle of the software engineering
- **GII-CRI17:** Capacity to design and evaluate person-computer interfaces that guarantee the accessibility and usability of systems, services and computer applications.

Subject contents

Theme I - *Introductory aspects*

- 1.1. Initial questions about the Software Engineering
- 1.2. A little of history
- 1.3. Software development process
- 1.4. Software process models
- 1.5. Conclusions

Theme II - *Requirements Analysis*

- 2.1. Requirements specification
- 2.2. The *Use Cases technique*
- 2.3. A step more in the specification: the System Sequence Diagram
- 2.4. Conclusions

Theme III - *Domain Analysis*

- 3.1. *Analysis Classes Diagram*
- 3.2. A step more in the domain analysis: the *Contracts of the operations*
- 3.3. Conclusions

Theme IV - *Introduction to Design and Unit Testing*

- 4.1. The need for code design
- 4.2. The JUnit framework

Tema V - *The SOLID principles*

- 5.1. Single responsibility principle
- 5.2. Open-closed principle
- 5.3. Liskov substitution principle
- 5.4. Interface segregation principle
- 5.5. Dependency inversion principle

Tema VI - *Responsibility based design*

- 6.1. The concept of responsibility
- 6.2. The GRASP patterns of responsibility assignment

Methodology

Big-size Groups: Masterly Classes (3 credits)

- Theoretical part: Supported by snapshots and/or specific notes
- Practical application part: Always working over examples. A **problems collection** is available. In class concrete problems are being solved. The solutions are being delivered along the semester
- **Methodology: Flipped classroom** (to reverse the traditional method)
 - Active participation of students**
 - We will tend towards active learning, where the student is the centre
 - Participatory and dynamic sessions (incorporation of questions / surveys, questions to be discussed, review of points explored, contribution of ideas, etc.)
 - => It requires commitment from the student
 - Previous to the class:
 - Revision of the specific material (material, links, videos) on your own
 - Beginning of the class:
 - Polls and reinforcement of revised contents

Medium-size Groups: Laboratory Classes (3 credits)

- Guided classes and personalized monitoring
- UML Modeling tool usage: **ArgoUML** and/or **Visual Paradigm**
- Control version tools with **GIT** and testing framework with **JUnit**
- Progressive work regarding a certain **practical statement**, which will simulate the software project development as practical application of the subject contents

Autonomous work (non presential):

- Practical work will be completed during **no presential** hours
- **Highly recommended** to the student: solving the problems from the **collection**, in order to get feedback from the teacher

The **avaluation system** (detailed in el corresponding section) is composed of: 1) written tests (l2 partial exams); and 2) practices (to develop in group).

Development plan

Week	Theory (GG)	Laboratory (LG)	Autonomous Work
1	Subject presentation T1: Introductory aspects	T1: Introductory aspects	Study
2	T1: Introductory aspects	T1: Introductory aspects	Study
3	T2: Requirements analysis Requirements specification	T2: Requirements analysis Requirements specification	Study and problems solving (Analysis problems collection)

4	T2: Requirements analysis. The Use Cases technique. Problems	UML Modeling usage Use Cases technique practical application	Study, problems solving (Analysis problems collection) and Analysis practice development
5	T2: Requirements analysis Use Case Specification Problems	Application of the Use Cases technique to the practice drafting T2: Requirements analysis. System Sequence Diagrams <i>Requirements analysis (1st part) Delivery</i>	
6	T3: Domain analysis Object Oriented Modeling technique	SSD application to the practice drafting	Study, problems solving (Analysis problems collection) and Analysis practice development
7	T3: Domain analysis Object Oriented Modeling technique Problems	UML Modeling usage Practical application of the Object Oriented Modeling technique	Study, problems solving (Analysis problems collection) and practice development <i>Requirements analysis (2nd part) Delivery</i>
8	T3: Domain analysis Object Oriented Modeling technique Problems	Construction of the Domain Model for the practice drafting	Study, problems solving (Analysis problems collection) and Domain analysis practice development
9	First midterm		Development of the Domain analysis practice
10	T3: Domain analysis Contracts of operations T4: Introduction to Design Test concept	Simple testing problems	Study, problems solving (Testing problems collection) and Domain analysis practice development
11	T4: JUnit Substitute objects	Testing with substitutions problems	Study and problems solving (Testing problems collection) <i>Domain Analysis and Contracts Delivery</i>
12	T5: Principis SOLID Intro, OCP & LSP	Testing problems. Advanced aspects of JUnit	Study, problems solving (Testing problems collection) and testing practice development
13	T5: Principis SOLID SRP, ISP & DIP	Git usage	Study, problems solving (Testing problems collection) and testing practice development
14	T6: Patrons GRASP Responsability concept	Git usage	Study, problems solving (Testing problems collection) and testing practice development
15	T6: GRASP Patrons Expert, Creator, Low Coupling	T6: GRASP Patrons High cohesion, Controller	Study and testing practice development
16	Second midterm		Testing practice development
17	Second midterm		<i>Testing Delivery</i>
18	Tutorization		
19	Recovery		

Activt.	Description	Weight	Mínimum Grade	In group	Presential	Mandatory	Recoverable
Part1	First midterm	25%	3,0	No	Yes	Yes	Sí
Part2	Second midterm	25%	3,0	No	Yes	Yes	Sí
Actv1	Requirements Analysis	20%	No	Yes	No	Yes	No
Actv2	Domain Model and Contracts	10%	No	Yes	No	Yes	No
Actv3	Unitary testing	20%	No	Yes	No	Yes	No
Particip	Participation	10%	No	No	Yes	No	No

Final grade = 0,25 * Part1 + 0,25 * Part2 + 0,20 * Actv1 + 0,10* Actv2 + 0,20 * Actv3 + 0,10 * Particip

- Subject is passed if **final grade** is greater or equal than **5,0** and all midterms are above the minimum required.
- All the **compulsory activities** are required for succeeding the subject.
- The **optional part** is valorated in **1 point**. In case of taking part of the activities proposed in class sessions, subject is scored over **11** (1 *extra point*).

Other considerations and criteria:

- Type of exams: concept fixation and problems solving.
- For all activities: programmed deliveries, unmovable dates.
- **When any of the partial exams is below the minimum grade**, the student must do the recovery exam of the corresponding midterm/s.
 - If even though, one or both midterms are below the minimum required, or below 5 in weighing, **the final grade will be 4,5 at most**.
- The recovery exam is a chance to obtain the subject success (current grade < 5), and also to improve the mark (taking into account that the mark obtained in it is the one that prevails).
 - In any case, in the recovery exam must be examined, at least, the midterm exam with the lower grade.
- *Particip*: The participation of the student in activities proposed additionally, to be developed during the theoretical class sessions, will be scored **up to 1 extra point in the mark**. They can consist of: the synthesis of the provided material, the contribution of ideas and intervention in debates, preparation of micro-themes agreeded together, etc.. They will serve to dynamize the classes and encourage an active participation, such as it is mentioned in the methodology.

Bibliography

Basic bibliography

- Craig Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice-Hall, 2005 (3^a ed.)
- Boni García: Mastering Software Testing with Junit 5. Packt, 2017

Complementary bibliography

- Gerald Kotonya, Ian Sommerville: Requirements Engineering: Processes and Techniques. Wiley, 1998
- Robert Martin: Agile Software Development: Principles, Patterns, and Practices, Prentice-Hall, 2002
- Lasse Koskela, Effective Unit Testing. A guide for Java developers. Manning, 2013