Subject: Software Analysis and Design Project
Code: 17833
Institution: Escuela Politécnica Superior
Degree: Computer Science Engineering
Level: Undergraduate
Type: Compulsory
ECTS: 6

1. COURSE TITLE

Software Analysis and Design Project

1.1. Course number

17833

1.2. Course area

Computer Science

1.3. Course type

Compulsory

1.4. Course level

Undergraduate

1.5. Year

2°

1.6. Semester

2°

1.7. Credit allotment

6

1.8. Prerequisites

The student should have knowledge on structural programming, acquired on the courses of the module *Data Programming and Structures*. It is recommended to have completed the courses *Programming 1*, *Programming 2* and *Programming Project*. It is also recommended to take this course **together with** the course *Software Analysis and Design* (in the same semester) as both provide complementary education and related concepts.

The student is advised to read the recommended texts of the bibliography, to use the teaching material available in Moodle (https://moodle.uam.es), as well as the active search of complementary material on the Internet. This course demands student autonomy, initiative and perseverance in the implementation of the applications, as well as predisposition for collaborative group work.

1.9. Minimum attendance requirement

The students who quit the continuous evaluation, or do not follow its requirements, can move to the non-continuous evaluation (with requirements different from those in the continuous evaluation).

ITINERARY WITH COMPULSORY CLASS ATTENDANCE
Assistance to at least 85% of lectures.

ITINERARY WITHOUT COMPULSORY CLASS ATTENDANCE
Assistance is not compulsory, but it is encouraged.

1.10. Faculty data

Coordinator:

Dr. Juan de Lara Jaramillo
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Office: B-430 Edificio B – 4ª Planta
Tel.: +34 91 497 2277
e-mail: juan.delara@uam.es
Web: http://www.ii.uam.es/~jlara

1.11. Course objectives

The student will gain practical knowledge on methods, practice, languages and tools to develop a software project of medium size using the Object Oriented paradigm. This knowledge will be acquired in practice through the realization of a project in a working team.

The skills provided by this course are the following:

Core:
C1. Ability to design, develop, select and evaluate computer applications and systems, ensuring their reliability, safety and quality in accordance with ethical principles and current legislation and standards.
C2. Ability to plan, conceive, deploy and direct projects, services and Computer Systems in any environment, leading all phases of the development, and assessing any economic and social impact.
C3. Ability to understand the importance of negotiation, effective working habits, leadership and communication skills in all areas of software development.

C4. Ability to prepare a technical specifications document for a Computer installation complying with current standards and regulations.
C5. Knowledge, administration and maintenance of Computer Systems, services and applications.
C8. Ability to analyse, design, construct and maintain applications in a robust, secure and efficient manner, choosing the most adequate paradigm and language.
C16. Knowledge and application of the principles, methodology and life cycles of software engineering.
C17. Ability to design and assess human-computer interfaces that guarantee accessibility and usability for Computer Systems, services and applications.

Specific technology:
IS1. Ability to develop, maintain and assess software systems that satisfy all user requirements and that behave reliably and efficiently, are accessible to development and maintenance, and that comply with quality standards, applying theories, principles, methods and practice in Software Engineering.

At the end of each unit, the student should be capable of:

GENERAL OBJECTIVES	
G1	Analysing, Designing, Implementing and Testing programs using Object Oriented Technologies, in order to produce maintainable, high-quality, software applications
G2	Applying Software Engineering good practices, methods, notations and tools for the development of software applications inside a working team

SPECIFIC OBJETIVES	
UNIT 1.- Requirements	
1.1.	Requirements elicitation for medium-size applications
1.2.	Requirements representation and analysis using flow-oriented and scenario-oriented notations
UNIT 2.- Design	
2.1.	Architectural high-level design of software applications
2.2.	Structural and behavioural design of software applications, using the Object Oriented paradigm
UNIT 3.- Implementation and Unit Testing	
3.1.	Implementation of a medium-size software application using Java, working in a team
3.2.	Design of test suites guaranteeing a certain level of confidence in the software quality
3.3.	Good practices in Software Engineering, and testing tools like JUnit
UNIT 4.- Testing	
5.1.	Integration testing, system testing and acceptance testing

1.12. Course contents

Summary

UNIT 1. Requirements
UNIT 2. Design
UNIT 3. Implementation and Unit Testing
UNIT 4. Testing

<u>**Syllabus**</u>

1. **Requirements**
   1.1. Elicitation.
   1.2. Notations.
      1.2.1. Flow-oriented.
      1.2.2. Scenario-oriented.
      1.2.3. Mockups.
2. **Design**
   2.1. Architectural.
   2.2. Detailed.
3. **Implementation and Unit Testing**
   3.1. Java programming techniques.
   3.2. Unit Testing. JUnit.
   3.3. Increments and regression testing.
4. **Testing**
   4.1. Integration Testing.
   4.2. System Testing.
   4.3. Acceptance Testing.

1.13. Course bibliography

Note: This course does not follow a particular book. The recommended bibliography is listed according to the course contents.

Unit 1:
1. Software engineering a practitioner's approach, 7ªed. Roger Pressman. McGraw Hill Higher Education, 2010. INF/681.3.06/PRE. Available in Spanish.
2. Software engineering, 9ª ed. Addison Wesley. Ian Sommerville. INF/681.3.06/SOM. Available in Spanish.
3. Software requirements styles and techniques. Lauesen, Soren. Addison-Wesley, 2002. INF/C6000/LAU.

Unit 2:
4. Ingeniería de software clásica y orientada a objetos, Sexta Edición. Stephen Schach. McGraw-Hill. INF/681.3.06/SCH.
5. El lenguaje unificado de modelado manual de referencia. Rumbaugh, James. Pearson Addison Wesley. 2007. INF/681.3.062-U/RUM.
6. Patrones de diseño elementos de software orientado a objetos reutilizable. Gamma, E., Helm, R., Johnson, R., Vlissides, J. INF/681.3.06/PAT. Addison-Wesley, 2003.

7. Software Architecture in Practice (2nd Edition). Bass, Clements, Kazman. Addison-Wesley Professional, 2003. INF/C5220/BAS.
8. Software Architecture: Foundations, Theory, and Practice. R. N. Taylor, N. Medvidovic, E. M. Dashofy, E. M. Dashofy. Wiley, 2010. INF/681.3.06/TAY
9. Designing the User Interface Strategies for effective human-computer interaction. Shneiderman, Ben. Pearson Education, 2005. INF/C5610/SHN.

Unit 3:

10. Core Java 2 Vol. 1 Fundamentos, Horstmann, Cay S. Prentice Hall, 2006. INF/681.3.062-J/HOR Vol. 1. Available in Spanish.
11. Core Java 2 Vol. 2 Características avanzadas, Horstmann, Cay S. Prentice Hall, 2006. INF/681.3.062-J/HOR Vol. 2. Available in Spanish.
12. Pruebas de software y JUnit: un análisis en profundidad y ejemplos prácticos. Bolaños, Sierra, Alarcón. Prentice-Hall, 2008. INF/681.3.06/BOL
13. Unit Testing in Java: How Tests Drive the Code. Link. Morgan Kaufmann; 1 edition, 2003.
14. Test Driven: TDD and Acceptance TDD for Java Developers. Koskela. Manning Publications, 2007.

Unit 4:

15. Test Driven: TDD and Acceptance TDD for Java Developers. Koskela. Manning Publications, 2007.
16. Pruebas de software y JUnit: un análisis en profundidad y ejemplos prácticos. Bolaños, Sierra, Alarcón. Prentice-Hall, 2008.
17. Software engineering a practitioner's approach, 7^{a}ed. Roger Pressman. McGraw Hill Higher Education, 2010. INF/681.3.06/PRE. Available in Spanish.

Note: purchasing any book is not recommended until having compared its content with the course programme and having consulted on the library.

Electronic resources: any electronic resource used in the course (recommendations on how to prepare documents, building diagrams, recommendations on programming style) will be published in the PADS section in the Moodle platform (https://moodle.uam.es).