UNIVERSIDAD AUTÓNOMA DE MADRID

17833 - SOFTWARE ANALYSIS AND DESIGN PROJECT

Information of the subject

Code - Course title: 17833 - SOFTWARE ANALYSIS AND DESIGN PROJECT

Degree: 473 - Graduado/a en Ingeniería Informática
474 - Graduado/a en Ingeniería Informática y Matemáticas
722 - Graduado/a en Ingeniería Informática
734 - Graduado/a en Ingeniería Informática y Matemáticas (2019)

Faculty: 350 - Escuela Politécnica Superior

Academic year: 2020/21

1. Course details

1.1. Content area

Ingeniería del software

1.2. Course nature

Compulsory

1.3. Course level

Grado (EQF/MECU 6)

1.4. Year of study

2

1.5. Semester

Second semester

1.6. ECTS Credit allotment

6.0

1.7. Language of instruction

Español, English

1.9. Recommendations

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.

1.10. Minimum attendance requirement

In both itineraries of evaluation, described below, it is mandatory to attend all class sessions during the first two weeks of classes. This is justified because in that period, students must participate in the requirements elicitation phase of the project to be developed during the semester. Such requirements elicitation phase is, in itself, an evaluable part of the development of the project and without participating in it is not possible to develop the project successfully. The score corresponding to the requirements elicitation phase will be penalized by a reduction of 25% for each elicitation session that the student has not attended to.

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---see Section on Evaluation Procedures below).

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. Final individual exam weighs higher than in the other itinerary (see Section on Evaluation Procedures).

1.11. Subject coordinator/s

Esther Guerra Sanchez

https://autoservicio.uam.es/paginas-blancas/

1.12. Competences and learning outcomes

1.12.1. Competences

C1 Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
C2 Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
C3 Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software.
C4 Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.
C5 Conocimiento, administración y mantenimiento sistemas, servicios y aplicaciones informáticas.
C8 Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
C16 Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.
C17 Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.

1.12.2. Learning outcomes

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.

In addition to the competences listed above, this course will provided the following competence
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.

1.12.3. Course objectives

At the end of each unit, the student should attain the following objectives:

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
3.4.	User interfaces and event-driven programming
UNIT 4.- Testing	
5.1.	Integration testing, system testing and acceptance testing

1.13. Course contents

Summary

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

Syllabus

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.
   3.4. User interfaces and event-driven programming
4. **Testing**
   4.1. Integration Testing.
   4.2. System Testing.
   4.3. Acceptance Testing.

1.14. 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. Available in Spanish.
2. Software engineering, 9ª ed. Addison Wesley. Ian Sommerville. Available in Spanish.
3. Software requirements styles and techniques. Lauesen, Soren. Addison-Wesley, 2002.

Unit 2:

4. Ingeniería de software clásica y orientada a objetos, Sexta Edición. Stephen Schach. McGraw-Hill.
5. El lenguaje unificado de modelado manual de referencia. Rumbaugh, James. Pearson Addison Wesley. 2007.
6. Patrones de diseño elementos de software orientado a objetos reutilizable. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Addison-Wesley, 2003.
7. Software Architecture in Practice (2nd Edition). Bass, Clements, Kazman. Addison-Wesley Professional, 2003.
8. Software Architecture: Foundations, Theory, and Practice. R. N. Taylor, N. Medvidovic, E. M. Dashofy, E. M. Dashofy. Wiley, 2010.
9. Designing the User Interface Strategies for effective human-computer interaction. Shneiderman, Ben. Pearson Education, 2005.

Unit 3:

10. Core Java 2 Vol. 1 Fundamentos, Horstmann, Cay S. Prentice Hall, 2006. Available in Spanish.
11. Core Java 2 Vol. 2 Características avanzadas, Horstmann, Cay S. Prentice Hall, 2006. Available in Spanish.
12. Pruebas de software y JUnit: un análisis en profundidad y ejemplos prácticos. Bolaños, Sierra, Alarcón. Prentice-Hall, 2008.
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, 7ªed. Roger Pressman. McGraw Hill Higher Education, 2010. 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).

2. Teaching-and-learning methodologies and student workload

2.1. Contact hours

	#horas
Contact hours (minimum 33%)	51
Independent study time	99

2.2. List of training activities

Activity		Nb. hours	Percentage
Face-to-face	Classes	42 h (28%)	51 h (34%)
	Programmed tutorials	6 h (4%)	

Activity		Nb. hours	Percentage
	Final exam	3h (2%)	
Home activities	Autonomous work	49 h (32.7%)	99 h (66%)
	Practical activities	50h (33.3%)	
Total workload: 25 hours x 6 ECTS		**150 h**	

This course is made of practical face-to-face sessions, as well as work to be done at home. Face-to-face sessions are divided in two parts:

A. Explanation of techniques, methods, notations and tools needed to perform some phase of the development.
B. Practical application of the explained techniques in the context of a software project.

The teaching methodology includes the following activities:

***Practical work:**

Activities of the professor:
Explanation of the techniques, methods, notations and tools needed to perform some phase of the development.
Assign a project/practical work to each work group, and explain the work to be done. Supervise the work of the work groups.
The tools to be used are the development environments (typically under Eclipse) and the computers of the lab for the modeling, execution, testing and analysis of the applications.

Student activities:
Face-to-face activities: Depending on the development phase of the session, students will: i) understand the explanations on the techniques to be employed, ii) work in groups to apply those techniques to the project development, iii) make reports on the obtained results. In some sessions, it may be required to execute the application and discuss the results with the professor, who may ask individual questions to grade the work.
Home activities: Work group meetings to finish the work, and writing project reports.

***Tutorials in the classroom:**

Activities of the professor:
Solving general questions about the project.

Student activities:
Face-to-face activities: Asking individual or group questions, or discussion of different alternative solutions to the tasks to be performed.
Home activities: Study the tasks to be performed, and discussion of the different alternative solutions to the tasks

***Mandatory readings and personal study:**

Student activities:
Home activities: Autonomous learning, guided by the professor by means of the tasks published in Moodle.

3. Evaluation procedures and weight of components in the final grade

3.1. Regular assessment

- The Project will be evaluated in 4 milestones, where the student is required to submit the expected deliverables.
- The course also includes a final exam, related to the techniques used during the Project development.
- The final grade is obtained using the following equation:

Project mark: 0.15*P1 + 0.15*P2 + 0.60*P3 + 0.10*P4

Final mark in **“continuous” mode**: 0.80*Project + 0.20*Final exam

Final mark in **“non-continuous” mode**: 0.70*Project + 0.30*Final exam

- Depending on the characteristics of the Project, it is possible to split milestone P3 in two, with the lumped weight equal to 60%.
- In order to pass the course, it is mandatory to obtain a grade equal or higher tan 5, in every deliverable of P3, in the whole project, and in the final exam.
- To develop their project, students are not allowed to use tools for automated generation of UML class diagrams or partial coding of the graphical user interface.
- The final mark of the project will take into consideration the quality of the work, as well as its completeness. In particular, to pass the course it is necessary to implement every requirement of the application, and to follow a correct design using the principles of object-orientation.
- In case of assistance to less than 85% of the classes, the student will be evaluated in **“non-continuous” mode**, which means that the individual final exam will have a higher weigh in the computation of the final mark (see formulae above), and which will still require the student to submit all partial deliverables by same deadlines established both for continuous and non-continuous evaluation.
- The final grade in the extraordinary call is calculated as in the ordinary call. The weight of the project in the extraordinary call will be 70%, that is, the same as in the case of “non-continuous” mode. The project in the extraordinary call will be an extension to the project made during the course. The marks of the final exam and the project are maintained for the extraordinary call, but not for following years.
- The minimum number of submitted deliverables needed to receive a mark is 2. Under this number, the student will be graded as “not evaluated”. Even if the student does not attend the final exam, if he has submitted at least 2 deliverables, he will receive a numerical mark.

WARNING: Students are assumed to perform and submit original work. Any copy discovered, in any of the deliverables, will be severely punished, following the current regulations.

3.1.1. List of evaluation activities

Evaluatory activity	%
Final exam	between 20% and 30% (depending on evaluation itinerary)
Continuous assessment	between 70% and 80% (depending on evaluation itinerary)

3.2. Resit

- The final grade in the extraordinary call is calculated as in the ordinary call. The weight of the project in the extraordinary call will be 70%, that is, the same as in the case of “non-continuous” mode. The project in the extraordinary call will be an extension to the project made during the course. The marks of the final exam and the project are maintained for the extraordinary call, but not for following years.

3.2.1. List of evaluation activities

Evaluatory activity	%
Final exam	30%
Continuous assessment	70%

4. Proposed workplan

Week	Contents	Face-to-face hours	Autonomous work
1	- Presentation of the course, syllabus, regulations and evaluation methods. - Presentation of the Project to be performed. - **Unit 1. Requirements**	3	5 Working on the requirements phase.
2	- **Unit 1 Requirements**	3	7 Working on the requirements phase.
2	**Tutorial 1 on requirements**	1	
3	- **Unit 1 Requirements** - **Unit 2 Design**	3	7 Submission of requirements deliverables. Working on the design phase.
4	- **Unit 2 Design**	3	6 Working on the design phase.
5	- **Unit 2 Design**	3	7 Working on the design phase.
5	Tutorial 2 on design	2	
6	- **Unit 3 Coding and unit testing**	3	7 Submission of design deliverables. Working on the implementation phase
7	- **Unit 3 Coding and unit testing**	3	6 Working on the implementation phase.
8	- **Unit 3 Coding and unit testing**	3	6 Working on the implementation phase.
9	- **Unit 3 Coding and unit testing**	3	6 Working on the implementation phase.
9	Tutorial 3 on coding and unit testing	1	
10	- **Unit 3 Graphical User Interface development**	3	6 Working on the implementation phase.
11	- **Unit 3 Graphical User Interface development**	3	6 Working on the implementation phase.
12	- **Unit 3 Graphical User Interface development**	3	7 Working on the implementation phase.
13	- **Unit 4 Testing**	3	7 Submission of implementation deliverables. Working on the testing phase.
14	- **Unit 4 Testing**	3	7 Submission of testing deliverables.
	Tutorial 4: preparation for final exam	2	
	Final exam	3	10h