Course guide
230719 - ARQSOFT - Software Architecture

Last modified: 14/05/2026

Unit in charge: Barcelona School of Telecommunications Engineering
Teaching unit: 701 - DAC - Department of Computer Architecture.

Degree: MASTER'S DEGREE IN TELECOMMUNICATIONS ENGINEERING (Syllabus 2013). (Optional subject).
MASTER'S DEGREE IN AERONAUTICAL ENGINEERING (Syllabus 2014). (Optional subject).
MASTER'S DEGREE IN ADVANCED TELECOMMUNICATION TECHNOLOGIES (Syllabus 2019). (Compulsory subject).

Academic year: 2026 **ECTS Credits:** 5.0 **Languages:** English

LECTURER

Coordinating lecturer: JUAN CARLOS CRUELLAS IBARZ

Others: Primer quadrimestre:
JUAN CARLOS CRUELLAS IBARZ - 11

PRIOR SKILLS

Solid conceptual bases of object-oriented programming: object, attributes, methods, class, exception, data encapsulation, inheritance and polymorphism. Solid bases of Java programming: class definition syntax, inheritance management syntax, polymorphism. Management of exceptions in Java. Basic entry / exit in Java. Management of most common containers: sets, lists, maps, graphs.

TEACHING METHODOLOGY

There will be theoretical lectures, where the most relevant theoretical concepts will be presented, which will be accompanied by examples of their application. Lab sessions will be intermixed throughout the course, which will serve to consolidate the assimilation of the theoretical contents and put them into practice.
A program of moderate / high complexity degree must be developed by groups, where the theoretical elements presented during the course will be put into practice.

LEARNING OBJECTIVES OF THE SUBJECT

To achieve that whoever studies this subject is capable of:
1. Facing successfully the work in a team that confronts the accomplishment of a program of moderate / high complexity;
2. Carry out object-oriented analysis tasks, object-oriented design using correctly software design patterns;
3. Develop automated program testing environments.
4. Add to the programs the level of persistence of data using relational databases. Be able to successfully face work within a team that confronts the completion of a program of moderate / high complexity. Be able to carry out tasks of: object-oriented analysis, object-oriented design using correctly software design patterns. Be able to develop automated testing environments for programs. Be able to implement a level of data persistence using relational databases.

STUDY LOAD

Type	Hours	Percentage
Self study	86,0	68.80
Hours large group	26,0	20.80
Hours small group	13,0	10.40

Total learning time: 125 h

CONTENTS

Introduction to software tests. JUnit.

Description:
Testable software design.
Unit tests: JUnit
Simulation of external components ("Fakes" and "Mocks"). Mockito Tool.

Specific objectives:
Introduce the code testing activities in the context of the software development process. Introduce tools that facilitate these activities. Use JUnit as a tool that allows you to automate software tests

Related activities:
Lab session. Generation of tests for checking correctness of the code of a class.

Full-or-part-time: 8h
Theory classes: 3h
Laboratory classes: 3h
Self study : 2h

Iterative software development processes. Introduction to Scrum

Description:
Concept of software development process. Main characteristics of iterative processes. Comparison with cascade processes. Introduction to Scrum.

Specific objectives:
Briefly introduce the concept of software development process as a set of activities carried out by a team whose objective is the development of a program. Show the main characteristics of iterative processes as an alternative to cascade processes, highlighting the strong temporal overlap of analysis, design, coding and testing. Show details of one of the existing iterative processes: Scrum.

Related activities:
Development of a program of moderate complexity within a team.

Full-or-part-time: 3h
Theory classes: 2h
Self study : 1h

Introduction to object oriented analysis

Description:
Object oriented analysis artifacts: repertoire of use cases and conceptual model of the problem.
Requirements concept. Types of requirements. Requirements capture techniques. Use cases as a method for capturing requirements. Concept and parts of a use case.
Building the conceptual model taking as basis the use cases. Identification of the classes. Identification of the attributes of the classes. Identification and complete definition of the relationships between classes.
Examples

Specific objectives:
Categorize the different types of requirements of a program. Be able to identify use cases. Be able to fully develop the most relevant use cases.
Be able to construct the conceptual model from the use cases: concepts / classes, attributes and relationships.

Related activities:
Examples and project developed within a team

Full-or-part-time: 16h 30m
Theory classes: 4h 30m
Self study : 12h

Introduction to object-oriented design

Description:
Object-oriented design artifacts: class diagrams, interaction diagrams, documentation of designed classes.
Introduction to the interaction diagrams in UML: sequence diagrams and collaboration diagrams.
System sequence diagram
Techniques to identify and specify the methods of the classes of the program. GRASP patterns to identify methods and assign them to the relevant class.
Use of GRASP patterns to construct interaction diagrams from the use cases.

Specific objectives:
Be able to build UML sequence diagrams. Be able to build the sequence diagram of a system. Be able to correctly use the GRASP patterns of assignment of responsibilities to the classes starting at the use cases for identifying the methods of the classes that are designed.

Related activities:
Examples and project carried out within a team

Full-or-part-time: 16h 30m
Theory classes: 4h 30m
Self study : 12h

Introduction to databases

Description:
Concept of databases. Brief historical review. Types of databases. Relational databases. Brief introduction to SQL. Use of a relational database in Java.

Specific objectives:
To know what a relational database is. Be able to design simple tables of relational databases. Understand the basic elements of an Entity-Relationship diagram. Know introductory elements of the SQL language. Connect a Java program with a SQL database

Related activities:
Specific lab session and project carried out within a team

Full-or-part-time: 6h
Theory classes: 2h
Laboratory classes: 2h
Self study : 2h

Completing object-oriented design with design patterns

Description:
Introduction of software design pattern concept. GoF patterns. Classification.

Specific objectives:
Understand the concept of software design pattern, its types and the benefits that its use entails in a design.

Related activities:
Theory session

Full-or-part-time: 2h
Theory classes: 1h
Self study : 1h

Study of some relevant design patterns

Description:
Study of some of the most relevant design patterns: Strategy, Decorator, Factory Method, Abstract Factory, Compound, Observer

Specific objectives:
To know the details of the studied patterns. Be able to correctly postulate the use of software design patterns in the design phase of a program. Be able to correctly implement the studied patterns.

Related activities:
Theory sessions
Lab sessions for some patterns
Project carried out in a group

Full-or-part-time: 29h
Theory classes: 8h
Laboratory classes: 8h
Self study : 13h

Pattern aggregation: the model-view-controller pattern

Description:
Study of the model-view-controller pattern as an aggregation of several of the design patterns studied above.

Specific objectives:
Learn the details of the model-view-controller as the result of adding several of the design patterns studied in the previous topic.
Implement this pattern aggregation correctly.

Related activities:
Theory session and project developed within a team

Full-or-part-time: 3h
Theory classes: 2h
Self study : 1h

Project

Description:
content english

Full-or-part-time: 41h
Self study : 41h

GRADING SYSTEM

Any academic fraud, plagiarism or use or possession of unauthorised means in an assessment activity leads to a mark of zero (0) for the affected test or assignment. Furthermore, in accordance with the University's regulations, if disciplinary proceedings are initiated the subject is recorded as "Pending Assessment" until a decision is reached. These incidents are managed in accordance with the **Action Framework for Academic Integrity in Assessment at the UPC**.

Project: 100% (of which 20% is a subjective evaluation of the solution delivered)

EXAMINATION RULES.

The teaching staff will define the partial deliveries of the project, evaluate each delivery, and return the review and comments on said deliveries.

BIBLIOGRAPHY

Basic:
- Larman, C. Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development. 3rd ed. Upper Saddle River, N.J.: Prentice Hall PTR, 2005. ISBN 0-13-148906-2.
- Freeman, E.; Freeman, E.; Sierra, K.; Bates, B. Head First design patterns. 1st ed. Sebastopol, CA: O'Reily, 2004. ISBN 0-596-00712-4.
- Gamma, E.; Helm, R.; Johnson, R.; Vissides, J. Patrones de diseño: elementos de software orientado a objetos reutilizable. 1st ed. Madrid: Addison Wesley, 2003. ISBN 84-7829-059-1.

RESOURCES

Hyperlink:

- Writing effective use cases / Alistair Cockburn. Boston : Addison-Wesley, cop. 2001. ISBN: 0201702258. https://www.infor.uva.es/mlaguna/is1/materiales/BookDraft1.pdf.

https://discovery.upc.edu/discovery/fulldisplay?docid=alma991002517209706711&context=L&vid=34CSUC_UPC:VU1&lang=ca- Design patterns. https://www.oodesign.com/- Java design patterns - example tutorial. https://www.journaldev.com/1827/java-design-patterns-example-tutorial