# 270080 - AS - Software Architecture

**Coordinating unit:** 270 - FIB - Barcelona School of Informatics  
**Teaching unit:** 747 - ESSI - Department of Service and Information System Engineering  
**Academic year:** 2019  
**Degree:** BACHELOR'S DEGREE IN INFORMATICS ENGINEERING (Syllabus 2010). (Teaching unit Optional)  
**ECTS credits:** 6  
**Teaching languages:** Catalan

## Teaching staff

- **Coordinator:** Cristina Gómez Seoane (cristina@essi.upc.edu)
- **Others:** Carme Quer Bosor (cquer@essi.upc.edu)  
  Lidia López Cuesta (llopez@essi.upc.edu)

## Prior skills

- Knowing what is a UML/OCL formal specification of functional and nonfunctional requirements of a system.  
- Knowing the context in which the design of a system may be performed and, in particular, what are the previous and following activities.  
- Knowing the object-oriented programming and (at least) an object-oriented programming language.

## Requirements

- Prerequisite IES

## Degree competences to which the subject contributes

### Specific:

- CES1.1. To develop, maintain and evaluate complex and/or critical software systems and services.  
- CES1.4. To develop, maintain and evaluate distributed services and applications with network support.

- CES2.2. To design adequate solutions in one or more application domains, using software engineering methods which integrate ethical, social, legal and economical aspects.  
- CT2.1. To demonstrate knowledge and capacity to apply the principles, methodologies and life cycles of software engineering.  
- CT2.3. To design, develop, select and evaluate computer applications, systems and services and, at the same time, ensure its reliability, security and quality in function of ethical principles and the current legislation and normative.

### Generical:

G9. PROPER THINKING HABITS: capacity of critical, logical and mathematical reasoning. Capacity to solve problems in her study area. Abstraction capacity: capacity to create and use models that reflect real situations. Capacity to design and perform simple experiments and analyse and interpret its results. Analysis, synthesis and evaluation capacity.
270080 - AS - Software Architecture

Teaching methodology

The teaching of the course is structured in theory, exercises and laboratory sessions.

In the theory sessions teachers present the essential content of the course. Typically teachers use slides that students have available before the session. Other contents of the subject will be studied autonomously.

In the exercises sessions the contents of the subject are practiced through the exercises. In the exercises sessions proposed exercises will be solved and several solutions will be discussed. In some sessions, students deliver in advance the exercises to be solved. Also, in some sessions, at the beginning of the session, the students answer a questionnaire about the contents presented and practiced the week before and/or about the exercises proposed in the previous session.

In the laboratory classes the students work in groups of two (pair programming). During the session, the students solve the laboratory exercises proposed by the teacher. At the beginning of the session, the students answer a questionnaire about the contents presented and practiced the week before and/or about the exercises proposed in the previous session.

The students must attend to the group sessions where they are enrolled.

Learning objectives of the subject

1. To explain what is the logical and physical architecture of a software system and relations between them.
2. To explain the characteristics of a layered architecture and the design principles that govern the construction of software systems with this type of architecture.
3. To explain the characteristics of an object-oriented architecture and the design principles that govern the construction of software systems with this type of architecture.
4. To design software systems using a layered and object-oriented software using software services from the specification of requirements (functional and nonfunctional) using traditional and agile methodologies.
5. To represent the design of software systems using UML.
6. To correctly apply design principles when designing software systems.
7. To design the domain (business) layer of a software system using traditional and agile methodologies.
8. To design the presentation layer of a software system by identifying the design patterns to apply and applying them correctly.
9. To explain the characteristics of the different persistence strategies.
10. To design the data management layer of a software system using access or data mapper patterns.
11. To identify and correctly use existing software services when designing software systems.
12. To evaluate and critique the design of software systems and to propose alternative designs to improve their quality.
13. To explain what is software architecture and design and what are the different views of software architecture.
270080 - AS - Software Architecture

Study load

<table>
<thead>
<tr>
<th>Total learning time: 150h</th>
<th>Hours large group: 30h</th>
<th>20.00%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hours medium group: 0h</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>Hours small group: 30h</td>
<td>20.00%</td>
</tr>
<tr>
<td></td>
<td>Guided activities: 6h</td>
<td>4.00%</td>
</tr>
<tr>
<td></td>
<td>Self study: 84h</td>
<td>56.00%</td>
</tr>
</tbody>
</table>

Content

Introduction to the software architecture and design

Degree competences to which the content contributes:

Description:
Software development methodologies. Software architecture and design. Logical Architecture and Physical Architecture. Role of design patterns in software design. Software architecture and design in traditional and agile methodologies.

Layered and object oriented architectures

Degree competences to which the content contributes:

Description:
Layered architectural pattern. Application of layered architectural pattern to software systems and services. Layered architecture design principles. Object oriented architectural pattern. Application of object oriented architectural pattern to software systems. Object oriented architecture design principles.

Software design using traditional methodologies

Degree competences to which the content contributes:

Description:

Software design using agile methodologies

Degree competences to which the content contributes:

Description:
## Planning of activities

### C2

**Hours:** 12h  
Guided activities: 2h  
Self study: 10h

**Description:**  
Test 2 (theory and exercises) where part of the objectives of chapter 3 are evaluated

**Specific objectives:**  
4, 6, 8, 9, 10, 11

### C1

**Hours:** 12h  
Guided activities: 2h  
Self study: 10h

**Description:**  
Test 1 (theory and exercises) where all objectives of chapters from 1 to part of 3 are evaluated

**Specific objectives:**  
1, 2, 3, 4, 5, 6, 7, 13

### Introduction to software architecture

**Hours:** 3h  
Theory classes: 2h  
Practical classes: 0h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 1h

**Description:**  
Listen carefully, take notes and ask questions if necessary. In addition, the student studies the contents that the teacher indicates

**Specific objectives:**  
1, 13

### Studying the layered architecture

**Hours:** 3h  
Theory classes: 0h  
Practical classes: 1h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 2h

**Description:**  
Listen carefully, take notes and ask questions if necessary. In addition, the student studies the contents that the teacher indicates

**Specific objectives:**  
2
### Studying the object-oriented architecture

**Hours:** 14h  
Theory classes: 0h  
Practical classes: 6h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 8h

**Description:**  
Listen carefully, take notes and ask questions if necessary. In addition, the student studies the contents and does the exercises that the teacher indicates

**Specific objectives:**  
3, 5, 12

### Studying software design using traditional methodologies

**Hours:** 59h  
Theory classes: 14h  
Practical classes: 14h  
Laboratory classes: 0h  
Guided activities: 3h  
Self study: 28h

**Description:**  
Listen carefully, take notes and ask questions if necessary. In addition, the student studies the contents and does the exercises that the teacher indicates

**Specific objectives:**  
4, 5, 6, 7, 8, 9, 10, 11

### Studying software design using agile methodologies

**Hours:** 35h  
Theory classes: 10h  
Practical classes: 0h  
Laboratory classes: 7h  
Guided activities: 3h  
Self study: 15h

**Description:**  
Listen carefully, take notes and ask questions if necessary. In addition, the student studies the contents and does the exercises that the teacher indicates

**Specific objectives:**  
4, 6, 7, 12

### C3

**Hours:** 12h  
Guided activities: 2h  
Self study: 10h
The final grade (NF) of the technical skills of the course is the following:

\[ NF = 0.3 \times C1 + 0.3 \times C2 + 0.3 \times C3 + 0.1 \times NT \]

C1 is the first test mark
C2 is the second test mark
C3 is the third test mark
NT is the mark of the tasks done (delivery of exercises before the class and questionnaires made at the beginning of the classes of exercises and laboratory).

Assessment of NT:
- During the course the students may take 6 tasks.
- If the student delivers 5 or less tasks, NT will be equal to the sum of the marks of all tasks delivered divided by 5.
- If the student delivers the 6 tasks, NT will be equal to the sum of the 5 better marks divided by 5.

Students will be only marked if they attend to the group sessions where they are enrolled.

Final mark of transversal skill is evaluated by one question in two exams (C1 and C3) where the students must demonstrate their critical thinking evaluating the proposed designs. Final mark is the average of the marks obtained in these questions and have an assessment of:
A is the mark is between 8.5 and 10
B is the mark is between 7 and 8.4
C is the mark is between 5 and 6.9
D is the mark is between 0 and 4.9
Bibliography

Basic:


Complementary:


Others resources:

Hyperlink


http://www.uml.org/

http://refactoring.com/catalog/

https://industriallogic.com/xp/refactoring/catalog.html
http://martinfowler.com/eaaCatalog/