270015 - IES - Introduction to Software Engineering

Coordinating unit: 270 - FIB - Barcelona School of Informatics
Teaching unit: 747 - ESSI - Department of Service and Information System Engineering
Academic year: 2018
Degree: BACHELOR'S DEGREE IN INFORMATICS ENGINEERING (Syllabus 2010). (Teaching unit Compulsory)
ECTS credits: 6  
Teaching languages: Catalan

Teaching staff

Coordinator: - Ernest Teniente Lopez (teniente@essi.upc.edu)
Others: - Claudia Patricia Ayala Martinez (cayala@essi.upc.edu)
- Dolors Costal Costa (dolors@essi.upc.edu)
- Manel Rello Saltor (rello@essi.upc.edu)
- Montserrat Estañol Lamarca (estanyol@essi.upc.edu)
- Raimon Lapuente Ferran (raimon.lapuente@upc.edu)
- Sergio Morales Garcia (smorales@essi.upc.edu)
- Xavier Oriol Hilari (xoriol@essi.upc.edu)

Prior skills

Students are expected to be sufficiently knowledgeable about data structures, algorithms and programming to study software engineering.

Requirements

- Prerequisite BD
- Pre-Corequisite EDA

Degree competences to which the subject contributes

Specific:
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.
CT2.4. To demonstrate knowledge and capacity to apply the needed tools for storage, processing and access to the information system, even if they are web-based systems.
CT3.3. To be able to find and interpret basic information for evaluating the economic environment of the organization.
CT5.5. To use the tools of a software development environment to create and develop applications.
CT6.1. To demonstrate knowledge and capacity to manage and maintain computer systems, services and applications.
CT7.1. To demonstrate knowledge about metrics of quality and be able to use them.
CT8.1. To identify current and emerging technologies and evaluate if they are applicable, to satisfy the users needs.
CT8.2. To assume the roles and functions of the project manager and apply, in the organizations field, the techniques for managing the timing, cost, financial aspects, human resources and risk.
CT8.3. To demonstrate knowledge and be able to apply appropriate techniques for modelling and analysing different kinds of decisions.
CT8.4. To elaborate the list of technical conditions for a computer systems installation fulfilling all the current standards and normative.

CT8.5. To manage and solve problems and conflicts using the capacity to generate alternatives or future scenarios analysed properly, integrating the uncertainty aspects and the multiple objectives to consider.

CT8.6. To demonstrate the comprehension of the importance of the negotiation, effective working habits, leadership and communication skills in all the software development environments.

CT8.7. To control project versions and configurations.

**General:**

G4. EFFECTIVE ORAL AND WRITTEN communication: To communicate with other people knowledge, procedures, results and ideas orally and in a written way. To participate in discussions about topics related to the activity of a technical informatics engineer.

**Teaching methodology**

The subject is structured around theory and problem-solving classes.

In the theory classes the lecturer will explain the main subject content. Lecturers typically use slides that students should obtain before class.

In problem-solving classes, course content (whether presented in class or studied independently) will be studied by completing problems. This will sometimes require problems to be resolved (or at least attempted) before class, so that the best solutions can be collectively analysed and discussed in class. On other occasions, the problem will be both set and resolved in class.

**Learning objectives of the subject**

1. Students should be able to provide an overview of the software engineering process
2. Students should be able to understand the requirements of a software system and relate these to the different parts of the specifications
3. Students should be able to write specifications for a UML software system
4. Students should be able to understand the desirable properties of specifications for a software system.
5. Students should be able to analyse the completeness and consistency of the specifications.
6. Students should be able to understand the general principles of software architecture and object-oriented design in UML
7. Students should be able to understand the logic structure in layers of an information system: presentation, domain and data management layers
8. Students should be able to transform a UML model into a design specification
9. Students should be able to understand the concept of design template and use some of the better known templates.
10. Students should be able to understand the basic concepts of software testing
## Study load

<table>
<thead>
<tr>
<th>Total learning time: 150h</th>
<th>Theory classes:</th>
<th>30h</th>
<th>20.00%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Practical classes:</td>
<td>30h</td>
<td>20.00%</td>
</tr>
<tr>
<td></td>
<td>Laboratory classes:</td>
<td>0h</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>Guided activities:</td>
<td>6h</td>
<td>4.00%</td>
</tr>
<tr>
<td></td>
<td>Self study:</td>
<td>84h</td>
<td>56.00%</td>
</tr>
</tbody>
</table>
## Content

<table>
<thead>
<tr>
<th>Section</th>
<th>Degree competences to which the content contributes:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction to software engineering</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Software requirements and software specifications</strong></td>
<td></td>
</tr>
<tr>
<td><strong>UML use case models</strong></td>
<td></td>
</tr>
<tr>
<td><strong>UML software system specification</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>Class diagrams, behaviour diagrams and state diagrams.</td>
</tr>
<tr>
<td><strong>Introduction to software design</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Introduction to object-oriented design with UML</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Test design in an object-oriented context</strong></td>
<td></td>
</tr>
</tbody>
</table>
# Planning of activities

| **Introduction to software engineering** | **Hours**: 4h  
Theory classes: 2h  
Practical classes: 0h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 2h |
| **Specific objectives:**  
1 |

| **Software requirements and software specifications** | **Hours**: 4h  
Theory classes: 2h  
Practical classes: 0h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 2h |
| **Specific objectives:**  
1, 2 |

| **UML use case models** | **Hours**: 6h  
Theory classes: 2h  
Practical classes: 0h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 4h |
| **Specific objectives:**  
2, 3 |

| **UML data structural diagram** | **Hours**: 34h  
Theory classes: 4h  
Practical classes: 10h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 20h |
| **Specific objectives:**  
2, 3, 4, 5 |
## UML behaviour diagram

**Hours:** 28h  
Theory classes: 4h  
Practical classes: 8h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 16h

**Specific objectives:**  
2, 3, 4, 5

## State diagrams in UML

**Hours:** 4h  
Theory classes: 2h  
Practical classes: 0h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 2h

**Specific objectives:**  
3, 4, 5

## Introduction to software design

**Hours:** 4h  
Theory classes: 2h  
Practical classes: 0h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 2h

**Specific objectives:**  
6, 7

## Introduction to software design in UML

**Hours:** 48h  
Theory classes: 8h  
Practical classes: 12h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 28h

**Specific objectives:**  
6, 8, 9
### Design testing in the object-oriented context

**Hours:** 12h  
Theory classes: 4h  
Practical classes: 0h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 8h

**Description:**  
Parts of learning objectives 3 to 5 will be assessed

**Specific objectives:**  
3, 4, 5

### UML structural model exercise

**Hours:** 2h  
Guided activities: 2h  
Self study: 0h

**Description:**  
Parts of learning objectives 3 to 5 will be assessed

**Specific objectives:**  
3, 4, 5

### UML structural model test

**Hours:** 2h  
Guided activities: 2h  
Self study: 0h

**Description:**  
Parts of learning objectives 1 to 5 will be assessed

**Specific objectives:**  
1, 2, 3, 4, 5

### UML behavioural model test

**Hours:** 2h  
Guided activities: 2h  
Self study: 0h

**Description:**  
Parts of learning objectives 1 to 5 will be assessed

**Specific objectives:**  
2, 3, 4, 5

### Software design exercise

**Hours:** 2h  
Guided activities: 2h  
Self study: 0h

**Description:**  
Part of learning objectives 8 and 9 will be assessed
Specific objectives:
8, 9

Software design test

<table>
<thead>
<tr>
<th>Hours</th>
<th>Guided activities: 2h</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Self study: 0h</td>
</tr>
</tbody>
</table>

Description:
Part of learning objectives 6 to 10 will be assessed

Specific objectives:
6, 7, 8, 9, 10

Qualification system

The final mark will be based on five assessment activities (C1, C2, FHC1 to FHC3) issued throughout the course and on participation in course activities. The final mark is calculated as:

Final mark = 10% C1 + 25% FHC1 + 15% FHC2 + 15% C2 + 25% FHC3 + 10% participation.

taking into account that:
Final Mark = NP if the student does not show to FHC3 and his/her mark according to the previous formula is <4.

There will be no final exam.

In addition to a subject mark, a generic competency mark will be awarded with the score A (excellent), B (good), C (satisfactory), D (fail) or NA (Not evaluated).
270015 - IES - Introduction to Software Engineering

Bibliography

Basic:


Complementary:


Others resources:

Hyperlink

http://www.omg.org/

http://www.uml.org/

http://hillside.net/patterns/