COURSE OUTLINE

(1) GENERAL

<table>
<thead>
<tr>
<th>SCHOOL</th>
<th>SCHOOL OF ENGINEERING</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACADEMIC UNIT</td>
<td>COMPUTER ENGINEERING AND INFORMATICS</td>
</tr>
<tr>
<td>LEVEL OF STUDIES</td>
<td>Undergraduate</td>
</tr>
<tr>
<td>COURSE CODE</td>
<td>CEID_NY232</td>
</tr>
<tr>
<td>SEMESTER</td>
<td>SPRING (8th)</td>
</tr>
<tr>
<td>COURSE TITLE</td>
<td>SOFTWARE ENGINEERING</td>
</tr>
</tbody>
</table>

INDEPENDENT TEACHING ACTIVITIES

<table>
<thead>
<tr>
<th></th>
<th>WEEKLY TEACHING HOURS</th>
<th>CREDITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Laboratory Exercises</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Recitation sections</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Add rows if necessary. The organisation of teaching and the teaching methods used are described in detail at (d).

6  6

COURSE TYPE

Specialised general knowledge

general background, special background, specialised
general knowledge, skills development

PREREQUISITE COURSES:

There are no prerequisite courses. It is however recommended that students have a good knowledge of Programming.

LANGUAGE OF INSTRUCTION and EXAMINATIONS:

Greek. Instruction may be given in English if foreign students attend the course.

IS THE COURSE OFFERED TO ERASMUS STUDENTS

YES (in English)

COURSE WEBSITE (URL)


https://eclass.upatras.gr/courses/CEID1030/

(2) LEARNING OUTCOMES

Learning outcomes

The course learning outcomes, specific knowledge, skills and competences of an appropriate level, which the students will acquire with the successful completion of the course are described.

Consult Appendix A

- Description of the level of learning outcomes for each qualifications cycle, according to the Qualifications Framework of the European Higher Education Area
- Descriptors for Levels 6, 7 & 8 of the European Qualifications Framework for Lifelong Learning and Appendix B
- Guidelines for writing Learning Outcomes

With the successful completion of the course, students will be able to:

- Recognize most common software engineering models, to combine various software engineering methods and to fit in a software engineering process.
- Identify and manage risk, to estimate effort resources and cost in small scale and to manage software engineering projects.
- Collect and analyze customer requirements, to use UML to design use cases and to perform robustness analysis.
- Work on both the dynamic and static model of a software project and to use the proper tools to work from the domain model towards coding.
• Use code repositories, collaborate in team and handle code versions.
• Collaborate productively in software development teams using Agile and iterative methods.
• Select the appropriate testing techniques and to design test cases and perform testing using the appropriate tools.
• Use debugging techniques and tools effectively.

General Competences
Taking into consideration the general competences that the degree-holder must acquire (as these appear in the Diploma Supplement and appear below), at which of the following does the course aim?

| Search for, analysis and synthesis of data and information, with the use of the necessary technology | Project planning and management |
| Adapting to new situations | Respect for difference and multiculturalism |
| Decision-making | Respect for the natural environment |
| Working independently | Showing social, professional and ethical responsibility and sensitivity to gender issues |
| Team work | Criticism and self-criticism |
| Working in an international environment | Production of free, creative and inductive thinking |
| Working in an interdisciplinary environment | Others... |
| Production of new research ideas | |

Search for, analysis and synthesis of data and information, with the use of the necessary technology
Decision-making
Working independently
Team work
Project planning and management
Showing social, professional and ethical responsibility and sensitivity to gender issues
Production of free, creative and inductive thinking

(3) SYLLABUS

1. From coding to software engineering, software engineering ethics, legacy systems.
2. Software engineering models, Waterfall, Spiral, V&V, iterative models, CASE tools.
3. Software projects management, PERT and Gantt chats, managing time, staff allocation.
4. Risk analysis and risk management, cost estimation and management, estimation models.
5. Requirements analysis, OOAD, UML, Use Cases, Domain model and Class Diagrams.
6. Software design, the ICONIX methodology, robustness diagrams, sequence diagrams.
7. Development and programming, principles of OOAD, from dynamic models to static model and code.
8. Programming in teams, code repositories, Gits and version control, code reviews and code inspections.
9. Iterative development, rapid prototyping, visual programming, RUP.
10. Agile methods, extreme programming, programming in pairs, timebox, SCRUM, Kanban.
11. Verification and Validation, units and system testing, white box testing, equivalence partitioning, boundary value analysis.
12. Interface testing, black box testing, basic path testing, condition coverage, stress testing.
13. Debugging, brute force techniques, backtracking, cause elimination, regression testing.
(4) TEACHING and LEARNING METHODS - EVALUATION

**DELIVERY**

Face-to-face, Distance learning, etc.

**USE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY**

Use of ICT in teaching, laboratory education, communication with students

Lectures will use slides that will be available through the university LMS (eClass). Content provision and communication with the professors and peers will also be through eClass (messages and e-forum).

**TEACHING METHODS**

The manner and methods of teaching are described in detail.

Lectures, seminars, laboratory practice, fieldwork, study and analysis of bibliography, tutorials, placements, clinical practice, art workshop, interactive teaching, educational visits, project, essay writing, artistic creativity, etc.

The student’s study hours for each learning activity are given as well as the hours of non-directed study according to the principles of the ECTS.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Semester workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>2 hours x 13 weeks = 26</td>
</tr>
<tr>
<td>Recitation</td>
<td>2 hours x 13 weeks = 26</td>
</tr>
<tr>
<td>Project (preparation, development, peer assessment)</td>
<td>5 projects x 12 hours = 60</td>
</tr>
<tr>
<td>Study and analysis of bibliography</td>
<td>4 hours x 13 weeks = 52</td>
</tr>
<tr>
<td>Participating in exams</td>
<td>3</td>
</tr>
<tr>
<td><strong>Course total</strong></td>
<td><strong>167</strong></td>
</tr>
</tbody>
</table>

**STUDENT PERFORMANCE EVALUATION**

Description of the evaluation procedure

Language of evaluation, methods of evaluation, summative or conclusive, multiple choice questionnaires, short-answer questions, open-ended questions, problem solving, written work, essay/report, oral examination, public presentation, laboratory work, clinical examination of patient, art interpretation, other.

Specifically-defined evaluation criteria are given, and if and where they are accessible to students.

A passing grade is required on the team project for the participation on the exams. This counts as the B-lab.

The written examination counts for 50% of the final grade, while a passing grade is also required.

Written exams deal with problem solving related to the projects but in a smaller scale. Exams are in Greek, as well as in English for ERASMUS students.

(5) ATTACHED BIBLIOGRAPHY

**Basic bibliography**

The course is based on:


**Suggested bibliography**


**Relative Scientific Journals**

- ACM Transactions on Software Engineering and Methodology (TOSEM)
- IEEE Transactions on Software Engineering