COURSE OUTLINE

(1) GENERAL

SCHOOL	SCHOOL OF ENGINEERING		
ACADEMIC UNIT	COMPUTER ENGINEERING AND INFORMATICS		
LEVEL OF STUDIES	Undergraduate		
COURSE CODE	**CEID_NE5577**	**SEMESTER**	**FALL (elective course)**
COURSE TITLE	SOFTWARE QUALITY ASSURANCE AND STANDARDS		

INDEPENDENT TEACHING ACTIVITIES *if credits are awarded for separate components of the course, e.g. lectures, laboratory exercises, etc. If the credits are awarded for the whole of the course, give the weekly teaching hours and the total credits*	**WEEKLY TEACHING HOURS**	**CREDITS**
Lectures, Laboratory Exercises, Recitation sections	2,2,1	5
Add rows if necessary. The organisation of teaching and the teaching methods used are described in detail at (d).	**TOTAL**	**5**

COURSE TYPE *general background, special background, specialised general knowledge, skills development*	Specialised general knowledge
PREREQUISITE COURSES:	There are no prerequisite courses.
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://www.ceid.upatras.gr/en/undergraduate/courses/quality-assurance-and-standards 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 basic software quality standards and to use them effectively.
- Know and to apply software quality methods throughout the software engineering cycle, and in particular:
    - Quality processes on requirements analysis (such as formal specifications, Petri Nets)
    - Quality processes on software design (such as the LUCID methodology, usability design).
    - Quality processes on software development (by using software metrics).
    - Quality processes on software testing (such as the basic path method).
- Evaluate software usability using methods:
    - Analytical methods (such as heuristic evaluation, KLM model).
    - Experimental methods (such as the thinking aloud protocol).
    - Inquiry methods (such as questionnaires, focus groups, standard usability scale).

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	……
Production of new research ideas	*Others…*
	…….

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. Definition of Quality, differences of software quality and products quality, total quality management.
2. Statistical quality control, quality standards, CMM and CMMI, ISO standards, IEEE and ACM standards.
3. Software process quality, FCM model, ISO9126 standard, quality in all software engineering phases (from requirements to testing).
4. Quality on requirements analysis, formal specifications, Petri Nets.
5. Quality on software design, usability, ISO9241 standard, LUCID methodology, usability evaluation.
6. Analytical methods, the KLM model, Fitts law, heuristic evaluation.
7. Experimental methods, thinking aloud protocol.
8. inquiry methods, questionnaires, focus groups, standard usability scale.
9. Quality on software development, software metrics and measurements.
10. Size structure and data metrics, LOC and Halstead metrics.
11. Complexity metrics, McCabe metric.
12. Quality on testing, basic path testing, cause and effect diagram.
13. Quality on maintenance, cost of quality.

(4) TEACHING and LEARNING METHODS - EVALUATION

DELIVERY *Face-to-face, Distance learning, etc.*	Face-to-face
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*	

Activity	***Semester workload***
Lectures	2 hours x 13 weeks = **26**
Recitation	1 hour x 13 weeks = **26**
Project (preparation, development, peer assessment)	4 projects x 12 hours = **48**
Study and analysis of bibliography	4 hours x 13 weeks = **52**
Participating in exams	3
Course total	***142***

hours of non-directed study according to the principles of the ECTS	
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 grade higher to 4.0 is required on the 4+1 projects (the peer assessment counts as one project) for the participation on the exams. This counts as the B-lab. Any student with grade 7.5 or higher on the B-lab may use this grade as the course grade or choose to participate in the exams (in this case B-lab counts for 50% of the final grade). 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:

- Xenos, M. (2003). Software quality assurance. Filomathia Publications (in Greek).

Suggested bibliography

- April, A., & Laporte, C. Y. (2018). Software Quality Assurance. John Wiley & Sons.
- Fenton, N., & Bieman, J. (2014). Software metrics: a rigorous and practical approach. CRC Press.
- Davis, C. W. (2015). Agile metrics in action: Measuring and enhancing the performance of agile teams.
- Jones, C., & Bonsignour, O. (2011). The economics of software quality. Addison-Wesley Professional.
- Lazar, J., Feng, J. H., & Hochheiser, H. (2017). Research methods in human-computer interaction. Morgan Kaufmann.

Relative Scientific Journals

- Software Quality Journal, Springer
- ACM Transactions on Computer-Human Interaction (TOCHI)