COURSE OUTLINE

(1) GENERAL

SCHOOL	ENGINEERING		
ACADEMIC UNIT	ELECTRICAL AND COMPUTER ENGINEERING DEPT.		
LEVEL OF STUDIES	Undergraduate		
COURSE CODE	**ECE_INF950**	**SEMESTER**	**9**
COURSE TITLE	Software Quality		

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	3	
Seminars / Practice exercises	1	
Laboratory		
Add rows if necessary. The organisation of teaching and the teaching methods used are described in detail at (4).	4	5

COURSE TYPE *general background, special background, specialised, general knowledge, skills development*	Specialised, skills development
PREREQUISITE COURSES:	
LANGUAGE OF INSTRUCTION and EXAMINATIONS:	Greek
IS THE COURSE OFFERED TO ERASMUS STUDENTS	No
COURSE WEBSITE (URL)	https://www.ece.uop.gr/

(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*

The main aim of the course is to teach the basic concepts, methods and techniques related to the management and assurance of Software Quality and how they are applied.

Keywords: Verification, Validation, Static Analysis, Software Testing, Test Case, Functional Testing, Structural Testing, Interface Testing, Debugging, Performance Assessment, Software Quality Metrics, Petri Networks.

Learning Outcomes

Upon successful completion of the course, the student will be able to:

At the knowledge level:

- know the most important techniques for Software Verification, Validation and Debugging.
- distinguish verification and validation (V&V) software activities and understand their role in the software life cycle.
- know the software testing strategies and be able to distinguish the three testing phases: unit testing, integration testing and acceptance testing.
- recognize diverse software testing techniques, such as top to bottom and bottom to top testing, alpha and beta testing, performance assesment, recovery testing, etc.
- know what internal and external software quality metrics are, how to measure them and to what extent they are interrelated.
- be aware of the fundamental problems and objectives of the Human Computer Interaction field as well as the basic theories of human interaction modeling (such as human processor model and typing model, Fitts law, Hick-Hyman law, etc.).
- know the most common Fromal Techniques for producing software specifications

At the skill level:

- recognize the basic software quality standards and use them appropriately.
- recognize and implement software quality processes in all phases of the software life cycle.
- apply the most important test cases design techniques for the functional testing of the software (black-box testing), such as: Equivalence Partitioning, Boundary Value Analysis, Cause – Effect Graph.
- apply the most important test cases design techniques for structural testing of the software (White-Box Testing), such as: Basic Paths Testing and Loops Testing.
- apply the most important test cases design techniques to test interfaces between modules that make up a software system.
- apply the debugging methods which follow a successful testing phase, in order to correct the errors that it revealed, such as brute force techniques, backtracking method, cause removal techniques.
- design and implement performance testing (load testing and stress testing) of multiuser applications, and more specifically of World Wide Web applications.
- know and be able to effectively apply the most common methods of software evaluation (heuristic evaluation, cognitive navigation, user observation, questionnaires, performance measurement).
- recognize and design Petri networks of three different categories: condition/event networks, place/transition networks, and distinct tokens.

At the level of abilities:

- select and combine the most appropriate test cases design techniques for functional testing, structural testing and interface testing, depending on software specifications.
- select and combine the most appropriate software debugging techniques.
- evaluate the overall quality of the software, taking into account quality and performance metrics.
- use effectively tools for designing Petri networks and use them for real-life problems.

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 *Project planning and management*

information, with the use of the necessary technology	*Respect for difference and multiculturalism*
Adapting to new situations	*Respect for the natural environment*
Decision-making	*Showing social, professional and ethical responsibility and*
Working independently	*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
- Production of new research ideas
- Project planning and management
- Criticism and self-criticism
- Production of free, creative and inductive thinking

(3) SYLLABUS

The course is developed in the following 13 sections / lectures:

i. Definition of quality, differences in software quality with the production of material goods, quality standards (ISO, IEEE and ACM standards).
ii. Quality of software processes, quality at all phases of software engineering (from specifications to testing).
iii. Software Verification and validation (V&V). Static V&V techniques (static analysis, navigation, review, etc.). Dynamic V&V techniques (symbolic execution, simulation, sensitivity analysis, software testing).
iv. Test cases design techniques for structural testing of software (black-box testing), such as: Equivalence Partitioning, Boundary Value Analysis, Cause – Effect Graph.
v. Test cases design techniques for structural testing of software (white-box testing), such as: Basic Paths Testing and Loops Testing.
vi. Test cases design techniques for interface testing among the modules of a software system. Software testing techniques, such as top to bottom and bottom to top testing, alpha and beta testing, performance assesment, recovery testing, etc.
vii. Detailed applied examples of test cases design techniques.
viii. Debugging methods, such as: brute force techniques, backtracking method, cause removal techniques.
ix. Performance evaluation (load testing and stress testing) of multi-user applications, and more specifically of World Wide Web applications.
x. ISO9126 standard. Internal and external software quality metrics, how they are measured and to what extent they are interrelated.
xi. Quality in design, ease of use, heuristic evaluation, basic theories of human interaction modeling (such as human processor model and typing model, Fitts law, Hick-Hyman law, etc.).
xii. Size, structure and data Metrics. Object-oriented metrics and Halstead metrics. Complexity Metrics, McCabe Metric.
xiii. Quality in the specification analysis phase, formal specifications, Petri networks.

(4) TEACHING and LEARNING METHODS - EVALUATION

<table>
<tr><td>DELIVERY
Face-to-face, Distance learning, etc.</td><td>Face to face in class.
Distance learning support via e-Class system.</td></tr>
<tr><td>USE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
Use of ICT in teaching, laboratory education, communication with students</td><td>• Supporting the learning process through the e-Class platform (for notification of the course regulations, for distribution of slides, projects, supplementary material, announcements, links, bibliography, etc.).
• During the lectures of the theoretical part, a projector and presentations in electronic form are used, which are also posted on the eclass from the beginning of the semester.
• During the lectures and seminars, a computer is used to write and execute code.
• Use of specialized software tools for measuring quality metrics of software.</td></tr>
<tr><td>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</td><td>
<table>
<tr><th>Activity</th><th>Semester workload</th></tr>
<tr><td>Theory Lectures</td><td>39</td></tr>
<tr><td>Seminars</td><td>13</td></tr>
<tr><td>Project elaboration</td><td>25</td></tr>
<tr><td>Independent study of lectures and bibliography</td><td>48</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td>Course Total</td><td>125 hours (5 ECTS)</td></tr>
</table>
</td></tr>
<tr><td>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.</td><td>A. Written final exam that includes:
• Solving exercises
• Multiple choice questions
• Short answer questions
B. Project elaboration.

<u>Remarks:</u>
• The final grade results from the weighting of the theory and project grades with coefficients determined at the beginning of the semester and announced to the students via e-class. Indicatively it will be about 60% - 40%
• Projects are submitted electronically and students are asked to take an oral exam on them.
• The exam material and the evaluation process are communicated to the students in the lecture hall and in the e-class.</td></tr>
</table>

(5) ATTACHED BIBLIOGRAPHY

- Suggested bibliography:

- Pressman R., «Τεχνολογία Λογισμικού: Μια πρακτική προσέγγιση (8η έκδοση)», Εκδόσεις Τζιόλα, ISBN: 978-960-418-720-1, 2018
- Sommerville I., «Βασικές Αρχές Τεχνολογίας Λογισμικού (8η έκδοση)», Εκδόσεις Κλειδάριθμος ΕΠΕ, ISBN: 978-960-461-220-8, 2009
- Ξένος Μ., «Ποιότητα Λογισμικού (2η έκδοση)». Εκδόσεις ΓΚΟΤΣΗΣ ΚΩΝ/ΝΟΣ & ΣΙΑ Ε.Ε., ISBN: 9789609840019, 2009
- Σπινέλλης Δ., «Ποιότητα κώδικα: Η προοπτική του ανοικτού λογισμικού (1η έκδοση)», Εκδόσεις Κλειδάριθμος ΕΠΕ, ISBN: 978-960-461-123-2, 2008

- Related academic journals:

- Software Quality Journal, Springer
- ACM Transactions on Programming Languages and Systems
- ACM Transactions on Software Engineering and Methodology
- IEEE Transactions on Software Engineering