UNIVERSITY OF NICOSIA
ΠΑΝΕΠΙΣΤΗΜΙΟ ΛΕΥΚΩΣΙΑΣ

COMP-401 Software Engineering

Course Code	**Course Title**	**Credits (ECTS)**
COMP-401	Software Engineering	6
Department	**Semester**	**Prerequisites**
Computer Science	Fall, Spring	COMP-201 Systems Analysis & Design COMP-152 Programming I and Junior Standing
Type of Course	**Field**	**Language of Instruction**
Requirement	Computer Science	English
Level of Course	**Year of Study**	**Lecturer**
1st Cycle	3 or 4	Prof. A. I. Kokkinaki
Mode of Delivery	**Work Placement**	**Office**
Face-to-face	N/A	EU-209c

Objectives of the Course:

This course aims to provide students with the application of theory, knowledge, and practice to develop software systems that satisfy the requirements of users and customers in an effective and efficient way. The main objectives of this course are to:

- Describe all phases of the life cycle of a software system, including requirements analysis and specification, design, construction, testing, deployment, and operation and maintenance.
- Demonstrate tools for managing software development; analyzing and modeling software artifacts; assessing and controlling quality; and for ensuring a disciplined, controlled approach to software evolution and reuse.
- Present the "good practice" tools, methods, and approaches that are most applicable for a given development environment.

Learning Outcomes:

Upon completion of this course students are expected to be able to:

1. Explain the concept of a software life cycle and provide an example, illustrating its phases including the deliverables that are produced.
2. Select, with justification the software development models and process elements most appropriate for the development and maintenance of a diverse range of software products.
3. Explain the role of process maturity models. Develop a medium-size software product using a software requirement specification, an accepted program design methodology (e.g., structured or object-oriented), and appropriate design notation.

4. Follow ICONIX methodology and design appropriate UML diagrams for a medium-sized software system.
5. Discuss the properties of good software design including the nature and the role of associated documentation.
6. Evaluate the quality of multiple software designs based on key design principles and concepts.
7. Distinguish between program validation and verification.
8. Describe the role that tools can play in the validation of software.
9. Distinguish between the different types and levels of testing (unit, integration, systems, and acceptance) for medium-size software products and related materials.
10. Create, evaluate, and implement a test plan for a medium-size code segment.

Course Contents:

• Software life-cycle and process models
• Software process capability maturity models
• Approaches to process improvement
• Software process measurements
• Software requirements elicitation
• Requirements analysis modeling techniques
• Functional and non-functional requirements
• Prototyping
• Basic concepts of formal specification techniques
• Fundamental design concepts and principles
• Design patterns
• Software architecture
• Structured design
• Object-oriented analysis and design
• Component-level design
• Design qualities
• Internal including low coupling, high cohesion, information hiding, efficiency
• External including reliability, maintainability, usability, performance
• Other approaches: data-structured centered, aspect oriented, function oriented, service oriented, agile
• Design for reuse
• Use of open-source materials
• Distinguishing between verification and validation
• Static approaches and dynamic approaches
• Validation planning; documentation for validation
• Different kinds of testing – human computer interface, usability, reliability, security, conformance to specification
• Testing fundamentals, including test plan creation and test case generation black-box and white-box testing techniques
• Inspections, reviews, audits

Teaching Methods:

- Faculty Lectures and Guest-Lectures Seminars
- Directed and Background Reading
- Collaborative Learning through a Group Project
- In-class Exercises
- Student-led Presentations

Assessment Methods:

Mid-Term Exam, Final Exam, Project, Homework Assignments.

Required Textbooks:

Authors	Title	Publisher	Year	ISBN
Pressman, R. S. And Maxim, B.R	Software Engineering: A Practitioner's Approach	McGraw Hill, 8th Edition	2014	978-007802216

Recommended Textbooks/Reading:

Authors	Title	Publisher	Year	ISBN
Stephens, R.	Beginning Software Engineering	Wrox, 1st Edition	2015	978-1-118-969144
Thayer R. H. and Dorfman, M.	Software Engineering, Volume 1, The Development Process, 4th Edition	Wiley-IEEE Computer Society Press	2012	978-0-985270705
Thayer, R. H. and Dorfman, M.	Software Engineering, Volume 2, The Supporting Processes, 2nd Edition	Wiley-IEEE Computer Society Press	2005	ISBN: 978-0-471-68418-3
A series of articles and best practices approaches on students' intranet.				