Course Title	Software Engineering				
Course Code	ACSC383				
Course Type	BSc Computer Science: Required Course				
Level	BSc (Level 1)				
Year / Semester	3rd year / 5th semester				
Teacher's Name	Dr. Achilleas Achilleos				
ECTS	6	Lectures / week	3	Laboratories/week	1*
Course Purpose	The aim of this course is to help students understand the basic concepts of software engineering and enable them to critically think and decide when to apply the different software process models, techniques and tools to solve real world problems. The content of the course covers principally the software engineering process models (waterfall model, rapid prototyping model, agile development model) and the key phases of the software engineering lifecycle (requirements analysis, system design, implementation, validation, evolution). The Unified Modelling Language is taught and applied as the standard that supports modelling for the analysis and design of object-oriented software systems.				
Learning Outcomes	Upon successful completion of the course, students will be able to: • Describe the notion of software engineering and explain the need for a systematic approach to development of software as a product. • Compare different software engineering lifecycle models: waterfall model, rapid prototyping model, agile development model. • Outline the key phases of the software development lifecycle: i.e., requirements analysis, design, implementation, validation and evolution. • Outline the steps and methods involved in requirements analysis, specify and validate the needs in a given problem domain. • Define the notion of system modelling and compare data-driven modelling with event driven modelling. • Outline and apply UML as de-facto standard for CASE working with Use Case, Class, State, Collaboration and Sequence diagrams for analysis and design of object-oriented software systems. • Describe and explain the nature of design as continuation of analysis and apply specific methods and techniques to system design, including architectural patterns and design patterns. • Explain the direct relationship between design and implementation and use patterns for the development of complex software systems. • Describe and outline the implementation and validation methods and tools, as well as the different reuse levels: software reuse and				

* The lecture hours become two and one hour is devoted for laboratories on specific weeks.

	configuration management. • Describe the concept of open source software development and argue on the importance of software licensing.		
Prerequisites	**ACSC223, ACSC382.**	Corequisites	**None.**
Course Content	**1. The Nature of Software Engineering (3 Weeks)** - Technology and Business processes, modelling software, complexity of software, estimation of risks, roles and responsibilities. Software Development Life Cycle: Waterfall, Rational Unified Process, Agile process. Prototypes. CASE tools and reverse engineering. **2. Requirements Engineering (2 Weeks)** - Requirements Definition. User and System Requirements. Functional and non-functional requirements. Requirements engineering processes. Requirements elicitation. Requirements specification. Requirements validation. Requirements change. **3. System Modelling (3 Weeks)** - System Modelling. UML: visual modelling language. UML Diagram Types. Context models: operational context of a system. Process Models: UML Activity Diagrams. Interaction models: UML Use case and sequence diagrams. Structural models: organization of a system. UML Class Diagrams. Behavioral models: Data-Driven and Event-Driven Modelling. UML Activity and State Diagrams. Model-driven engineering: from models to code. **4. Architectural Design and Patterns (3 Weeks)** - Definition of Architectural design. Agility and Architecture. Architectural design decisions. Architectural views. Architectural patterns: MVC (Model-View-Controller) patterns, Layered architecture Pattern, Repository Pattern, Client Server Pattern, Pipe and Filter Pattern. Application architectures. Application Types: Data processing, Transaction processing, Event processing, Language processing. **5. Design and Implementation (2 Weeks)** - Object-oriented design using the UML. Design Patterns: The Observer Pattern. Implementation Issues. Validation and Testing. Reuse Levels. Software Reuse. Configuration Management. Host-Target Development. Integrated Development Environments (IDEs). Open Source Development. Software Licensing.		
Teaching Methodology	The course is structured principally around lectures that are delivered to the students with the help of computer presentations. Furthermore, the ArgoUML CASE tool is used to demonstrate to the students how to engage with software engineering at a practical level though the use of UML diagrams: class, use case, activity, statechart, sequence and collaboration. This helps to motivate and help students to engage in solving software engineering problems. The material of the course		

* The lecture hours become two and one hour is devoted for laboratories on specific weeks.

	is Lecture notes in the form of presentations and UML-based diagrams that are available through the e-learning system. This material is the main resource for students to use in their study, in combination with the recommended textbooks and references. The assessment of the course includes initially a midterm test and assignments. Finally, the course assessment is completed through a three-hours final exam at the end of the semester.
Bibliography	**Textbooks:** 1. Software Engineering, 10th Edition, by Ian Sommerville (Author), 816 pages, Publisher: Pearson; 10 edition (April 3, 2015), Language: English, ISBN-10: 0133943038, ISBN-13: 978-0133943030. **References:** 1. John Vlissides, Ralph Johnson, Richard Helm, Erich Gamma, “Design Patterns: Elements of Reusable Object-Oriented Software”, Publisher: Addison-Wesley Professional, Release Date: October 1994, ISBN: 0201633612 2. M. Seidl, M. Scholz, C. Huemer, G. Kappel, “UML@Classroom: An Introduction to Object-Oriented Modeling” (Undergraduate Topics in Computer Science) 2015 Edition, Online Available: https://link.springer.com/content/pdf/10.1007%2F978-3-319-12742-2.pdf 3. Textbook Resources: https://iansommerville.com/software-engineering-book/. 4. UML.org Resources page – https://www.uml.org/resource-hub.htm. 5. Homepage of ArgoUML – http://argouml.tigris.org/.
Assessment	• Midterm Test: 20% • Assignments: 20% • Final Exam: 60%
Language	English.

* The lecture hours become two and one hour is devoted for laboratories on specific weeks.