Course Title	Software Engineering				
Course Code	CSC650				
Course Type	Compulsory				
Level	Master (2nd Cycle)				
Year / Semester	2nd year / 1st semester				
Teacher's Name	TBA				
ECTS	10	Lectures / week	3 Hours / 14 week	Laboratories / week	N/A
Course Purpose and Objectives	A first objective of this course is to introduce students to the principles of Information Systems (IS). Traditional and novel systems' development methodologies are described and their basic characteristics are compared. Students learn how to apply the modeling tools of systems' development methodologies in realistic development cases. The course continues with discussion on the underlying process of the issues involved in the analysis of a system, the identification of the problem areas and the development of alternative solutions. A key objective of the first part of the course is the production of the Software Requirements Specification document which will be used in a later course as the base of the design and development of a software system. The second part of the course concerns the design, programming and implementation of a software system. It completes the software life cycle and provides the student with practical experience in large systems development. The student will also gain practical experience in developing technical manuals to be used by systems administrators, and user manuals to be utilized in training sessions and as references by potential users of the system. It also introduce the students to the Project Management: how to manage teams; PM and team roles; requirement specification; task allocation; scheduling and planning. Finally, the course also aims to encourage students to think critically about the applicability of existing and emerging technologies and research on a number advanced topics in software engineering.				
Learning Outcomes	Upon successful completion of this course student will be able to: • Describe the principles of Software Engineering and the main software development process models • Elicit and analyze requirements for a software development project • Construct the software requirements specification document • Create design model representations of software data, architectures, components and interfaces • Construct the software design specification document				

	• Carry out coding based on the design specification document • Describe any apply non-functional requirements modelling and validation • Demonstrate knowledge of contemporary software engineering methods and the relationship between software and systems engineering. • Describe, discuss and compare various architectural models and contemporary software engineering methods • Describe and apply component-based and service-oriented architectures as platforms for software reuse. • Describe the fundamental software testing concepts • Design and execute effective software test cases • Construct user and technical manuals • Plan, schedule and control a software development project • Measure software at different dimensions		
Prerequisites	CSC635	Co-requisites	None
Course Content	**System Analysis and Design:** Data Modeling: Introduction to data modeling, entities, attributes, relationships, the process of logical data modeling, analyzing the data model, normalization. Process Modeling: Process concepts, data flows, external agents, data stores, the process of logical process modeling, how to construct process models, the context data flow diagram, the functional decomposition diagram, the event response list, system and primitive diagrams, synchronizing of system models. Systems Design: Modern Structured Design, Information Engineering (IE), Prototyping, Joint Application Development (JAD), Rapid Application Development (RAD), Object-Oriented Design (OOD), FAST Systems Design Methods. **Software Engineering:** What is Software Engineering? The need for software engineering. Software characteristics, components and applications. Software reliability, software reuse, Software process models: waterfall model, incremental model, prototyping, RAD model, spiral model, Rational Unified Process. Systems concepts, boundaries, environment, inputs, outputs, characteristics of systems. From software to Systems engineering. Socio-technical and safety critical systems. Security in systems engineering, project and risk management in systems engineering Software Requirements engineering:		

	Problem definition, feasibility study, requirements elicitation, requirements analysis, requirements negotiation, requirements specification, requirements management Software Requirements elicitation: Requirements elicitation techniques: interviews, scenarios, use cases. Software Analysis models: Scenario-based models, object models, data models, information flow models, behavioral models. The need for better nonfunctional requirements management Elaborate and demonstrate benefits of the goal oriented approaches, the agent oriented approaches. Methods for validating NFRs Software Architecture: Software Analysis models: Scenario-based models, object models, data models, information flow models, behavioral models. Distributed systems architectures, Real-time software design, Concurrency modeling. The model driven architecture Object-Oriented analysis: Unified Modeling Language. UML diagrams: class/object diagrams, activity diagrams, swimlane diagrams, sequence diagrams, state diagrams. Service Oriented Software Engineering: Composition of reusable services provided by service providers. Service orchestration Pattern oriented software engineering: Demonstrate the use of high–level architectural patterns and medium–level design patterns to low–level idioms. Patterns and reusability. Aspect oriented software engineering: Identification, modularisation, representation and composition of crosscutting concerns (the aspects) throughout the software life cycle Coding: Mapping design models to code, refactoring, forward engineering, reverse engineering, using API's to increase coding performance and software reliability. Software Testing: Testing concepts: errors, bugs, defects. Types of testing: white-box testing, black-box testing, unit testing, integration testing, regression testing, acceptance testing. Test cases. Stubs, drivers. Equivalence partitioning. Debugging. Documentation: Writing user and technical manuals. Software Measurement Software measurement in theory and practice, Software costing, software quality assurance. Project Management:

	Management activities, project planning, project scheduling, Managing teams, the team leader. Task definition, work allocation. PERT diagrams, GANTT diagrams, the Critical Path Method (CPM). Risk management, quality management, configuration management, process improvement activities Estimation: Estimating effort, time and cost. Human, Hardware and Software resources, Software productivity metrics. Cost estimation techniques. Recent developments and contemporary issues pertaining to the subject-matter of the course.
Teaching Methodology	Face-to-Face
Bibliography	Sommerville, Ian, SOFTWARE ENGINEERING, Addison-Wesley Whitten and Bentley, SYSTEMS ANALYSIS AND DESIGN METHODS, McGraw Hill. Pressman, Roger, SOFTWARE ENGINEERING (A Practitioners Approach), Prentice - Hall Bruegge, B. and Dutoit, A.H., OBJECT-ORIENTED SOFTWARE ENGINEERING USING UML, PATTERNS AND JAVA, Pearson Prentice Hall Maciaszek, L.A. and Liong, B.L., PRACTICAL SOFTWARE ENGINEERING, A CASE STUDY APPROACH, Addison-Wesley Rumbaugh J., Jacobson, I., and Booch, G., THE UNIFIED MODELING LANGUAGE REFERENCE MANUAL
Assessment	Final examination 50% Project 40% Class participation and Attendance 10% 100%
Language	English