CPCS-351 Syllabus

Catalog Description

CPCS-351 Software Engineering (I)
Credit: 3 (Theory: 3, Lab: 0, Practical: 1)
Prerequisite: CPCS-204
Classification: Department Required

The objective of this course to introduce students to the basic concepts of software engineering as they relate to the development of medium to large scale software systems. Topics include the software life cycle, requirements development, object orientation, software architecture & design, and software testing. Students are expected to learn how to apply such principles to a real world problem. A term project of a medium-size is required.

Class Schedule

Lab/Tutorial 90 minutes 1 times/week

Meet 50 minutes 3 times/week or 80 minutes 2 times/week

Textbook

Timothy Lethbridge, Langaniere, Robert, , "Object-oriented software engineering", McGraw-Hill; 2 edition (2005)

ISBN-13 9780077109080 **ISBN-10** 0077109082

Grade Distribution

Week	Assessment	Grade %
5	Graded Lab Work 1	2.5
5	Homework Assignments 1	5
6	Exam 1	10
8	Graded Lab Work 2	2.5
10	Graded Lab Work 3	2.5
11	Homework Assignments 2	5
12	Exam 2	15
13	Graded Lab Work 4	2.5
15	Group Project	25
16	Comprehensive Final Exam	30

Last Articulated

April 14, 2018

Relationship to Student Outcomes

a	b	c	d	e	f	g	h	i	j	k
x	x							x		x

Course Learning Outcomes (CLO)

By completion of the course the students should be able to

1. Explain the concept of a software lifecycle and provide an example, illustrating its phases including the deliverables that are produced. (b)
2. **Define software quality and the role of quality assurance activities in the software process. (b)**
3. **Differentiate among the phases of software development (b)**
4. Explain the different software development models (traditional, iterative, incremental and agile) and their processes correctly. (k)
5. **Justify both functional and non-functional requirements in a given requirements specification for a software system (i)**
6. Apply key elements and common methods for elicitation and analysis to produce a set of software requirements for a medium-sized software system. (i)
7. **Articulate design principles including separation of concerns, information hiding, coupling and cohesion, and encapsulation. (i)**
8. **Create appropriate design models for the structure and behavior of software products from their requirements specifications. (i)**
9. Learn UML Use Case diagram notation and style to depict high-level user requirements and be able to write up Use Case Centric Functional Requirements Specifications in a typical commercial system. (i)
10. Depict UML diagrams that illustrate the structure and the dynamic behavior of a system (Class diagrams, sequence diagrams, activity diagrams and state diagrams). (k)
11. Identify popular software architectures (such as3-tier, pipe-and-filter, and client server) and the influence of an architecture on the design of a software system. (b)
12. Apply design patterns to enhance class diagram in the design model (i)
13. Communicate orally and in writing the results of applying the acquired knowledge of software engineering phases of analysis and design and their relevant techniques to a small sized group project. (a)
14. Describe and distinguish among the different types and levels of testing (unit, integration, systems, and acceptance). (a)

CPCS-351 Syllabus

Topics Coverage Durations

Topics	Weeks
Introduction to the course.	1
Software Quality Assurance and System Engineering.	1
System Engineering and Process and Methodology.	1
Software Requirements Elicitation	2
Applying Software Design Principles	1
Domain Modeling	2
Modelling Interactions and Behaviour	2
Architectural Design	1
Architectural Design and Applying Responsibility Assignment Patterns	1
Applying Responsibility Assignment Patterns	1
Software Testing	1

Coordinator(s)

Dr. Reem Alnanih, Associate Professor

Dr. Asif Irshad Khan, Assistant Professor