Software Engineering	Code & No:	CS360
	Credits:	3 (3,0,1)
	Pre-requisite:	CS 120
	Co-requisite:	None
	Level:	8 (CS), 6 (IT)

Course Description:

This course introduces concepts and techniques relevant to the production of large software systems. Students are taught a programming method based on the recognition and description of useful abstractions. Topics include modularity, specification, data abstraction, object modeling, design patterns, and testing. Students complete several programming projects of varying size, working individually and in groups. Topics to be covered :

1) **Introduction**
   - FAQs about software engineering
   - Professional and ethical responsibility

2) **Software processes**
   - Software process models
   - Process iteration
   - Process activities
   - The Rational Unified Process
   - Computer-Aided Software Engineering

3) **System Models**
   - Context models
   - Behavioural models
   - Data models
   - Object models
   - CASE workbenches

4) **Architectural Design**
   - Architectural design decisions
   - System organization
   - Modular decomposition styles
   - Control styles

5) **Object Oriented Concepts**

- Object approach

6) **Unified Modeling Language (UML)**
   - Class Diagram
   - Object Diagram
   - Use Case Diagram
   - Collaboration Diagram
   - Sequence Diagram
   - Component Diagram
   - Deployment Diagram

Course Aims:

The course introduces the principles of software engineering and detailed study of topics such as software development lifecycle, requirements gathering, program specification, design techniques, implementation guidelines, validation and verification. Software tools, team oriented processes. Different software process approaches to development, including waterfall model, prototyping, formal modeling, and spiral model. Software engineering economics.. Major topics include:

1) Knowledge of Software processes
2) Knowledge of System Models
3) Knowledge of Architectural Design
4) Knowledge of Object Oriented Concepts
5) Knowledge of Unified Modeling Language
6) Knowledge of Testing

Student Outcomes (SOs):

☒(a) An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline

☒(b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

☒(c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

☒(d) An ability to function effectively on teams to accomplish a common goal

☐(e) An understanding of professional, ethical, legal, security and social issues and responsibilities

☐(f) An ability to communicate effectively with a range of audiences

☐(g) An ability to analyze the local and global impact of computing on individuals, organizations, and society

☐(h) Recognition of the need for and an ability to engage in continuing professional development

☒(i) An ability to use current techniques, skills, and tools necessary for computing practice.

☐(j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices. [CS]

☐(k) An ability to apply design and development principles in the construction of software systems of varying complexity. [CS]

☐(j) An ability to use and apply current technical concepts and practices in the core information technologies of human computer interaction, information management, programming, networking, and web systems and technologies. [IT]

☐(k) An ability to identify and analyze user needs and take them into account in the selection, creation, evaluation, and administration of computer-based systems. [IT]

☐(l) An ability to effectively integrate IT-based solutions into the user environment. [IT]

☐(m) An understanding of best practices and standards and their application. [IT]

☐(n) An ability to assist in the creation of an effective project plan. [IT]

Course Learning Outcomes (CLOs):

1. Explain the principles of software engineering and why it is important
2. Discuss the concept of software processes and software process models
3. Apply the concepts of behavioral modeling, data modeling and object modeling in real time projects
4. Understand the decisions that have to be made about the system architecture during the architectural design process
5. Employ the Unified Modeling Language (UML) in real-time software projects

SOs and CLOs Mapping:

CLO/SO	a	b	c	d	e	f	g	h	i	j	k	l	m	n
CLO1	√													
CLO2			√											
CLO3									√					
CLO4		√												
CLO5				√										

No.	Topics	Weeks	Teaching hours
1	Introduces concepts and techniques relevant to the production of large software engineering concepts & development activities Participants and Roles - Systems and Models Work Products - Activities, Tasks and Resources - Functional and Nonfunctional Requirements - Notations, Methods, and Methodologies. Developmental Activities Requirements Elicitation, Analysis	2	6
2	The Principles of software engineering and detailed study of topics such as software development lifecycle, Software processes :Software process models-Waterfall, Iterative, Spiral, Prototyping models, Agile model	2	6
3	System Models : Context models, Interaction models, Structural models, Behavioral models, Model driven Engineering	1	3
4	Architectural Design: Architectural design decisions, System Organization Modular decomposition styles Control styles	2	6
5	Object-Oriented design using Unified Modeling Language(UML), Design Patterns, Open Source development, Case Studies	2	6
6	Software Configuration management-Version Management-Change Management	1	3
7	Project Planning, Project scheduling, Estimation techniques	2	6
8	Software Testing, Type of Testing, Unit and Integration testing, System testing, Debugging, User testing	2	6
	Total	**14**	**42**

Textbook:

- Software Engineering, Ian Sommerville, 9th edition, Addison Wesley, 2010

Essential references:

- Software Engineering: A Practitioners Approach, Pressman, 7th edition, McGraw Hill, 2010.
- Software Engineering: An Engineering Approach, Peters and Pedrycz, Wiley, 2007.
- Fundamentals of software Engineering, Ghezzi, Jazayeri, and Mandrioli, Prentice Hall, 2002.