Devi Ahilya University, Indore, India Institute of Engineering & Technology				III Year B.E. (Computer Engg.) (Full Time)			
Subject Code & Name	**Instructions Hours per Week**			**Credits**			
6CERE1	**L**	**T**	**P**	**L**	**T**	**P**	**Total**
SOFTWARE ENGINEERING PRACTICES	**3**	**1**	**1**	**3**	**1**	**1**	**5**
Duration of Theory Paper: 3 Hours							

Prerequisite:

Software engineering practices require a foundation in algorithms and data structures, software design principles, programming proficiency, skills in testing/debugging. A process-driven mind-set, familiarity with Agile/Scrum methodologies, and strong communication/collaboration skills for teamwork.

Learning Objectives

1. To introduce the various aspects of Software Engineering applied for the software development.
2. Compare and contrast different software development methodologies and identify appropriate contexts for their use.
3. Learning techniques for gathering and analysing software requirements, including stakeholder interviews and requirements documentation.
4. Identify and apply key design principles and architectural patterns in software design
5. Develop and execute test plans, including unit testing, integration testing, and system testing, to ensure software quality**.**

Course Objectives –

1. This course delves into the principles and advanced practices vital for the creation of sophisticated, software-driven systems.
2. To demonstrate a practical understanding and application of the theoretical frameworks.
3. Developing and demonstrating the arrangement of design elements to form a cohesive whole.
4. Illustrating when adhering to consistent coding conventions, procedures helps deliver high-quality software development.
5. To explain the role of software maintenance, software quality management and metrics in software development

Course Outcome:

Students earned credits will develop ability to

CO. No.	CO.	PO.
CO1.	Enact the carefully crafted protocols that are crucial to the software development life-cycle models.	PO1, PO2
CO2.	Foster robust communication, modeling, construction, and deployment practices within software development	PO1,PO2,PO3,PO10
CO3.	Conduct thorough analysis and design of software models employing the Unified Modeling Language (UML).	PO4,PO5
CO4.	Demystify the array of software testing methodologies and master the application of suitable testing strategies for software development.	PO4,PO5
CO5.	Illuminate the software maintenance, quality management, metrics, utilized in software development	PO3,PO4,PO5,PO9,PO11

CO-PO Relationship

CO	PO-1	PO-2	PO-3	PO-4	PO-5	PO-6	PO-7	PO-8	PO-9	PO-10	PO-11
CO1	2	2									
CO2	2	2	3							3	
CO3				2	3						
CO4				3	3						
CO5			2	3	3				3		3

* CO (rows) mention nil/very small/insignificant contribution to the PO(column)
1→ relevant and small significance 2 → medium or moderate and 3 →strong

UNIT - I

Introduction to Software Engineering: Software Process, Process Models, CMMI, Agile Process, Software Characteristics, Unified Process, Requirement Engineering, SRS, Context models, Behavioral and Structural Models,

UNIT – II

Design Engineering: Design concepts, process, quality, patterns, software architecture, styles and patterns, Design modeling and creation of Class, Object, Components, constraint language, designing conventional components, User Interface Design, Design evaluation.

UNIT – III

Software Development: Programming standards and procedures, Programming guidelines, Algorithm, Data structures and Programs, Design Patterns, Rewriting, Revising, Refactor, Reuse, Code documentation

UNIT - IV

Software Testing: Testing Principles, Design, Unit Test for Components, Integration strategy for builds, smoke test, regression test, validation strategy, acceptance test with customers, System testing, Bug, Error, Fault and Defect, Bug life cycle.

UNIT- V

Software Maintenance: Need for maintenance of software systems, Types of maintenance, process, Maintenance cost, Maintenance Challenges and Benefits, Maintenance tools and techniques, Reverse engineering.

Learning Outcomes:

1. The ability to elicit, model, and document requirements (functional and non-functional) and apply architectural design principles and patterns to create scalable, maintainable system blueprints.

2. The skill to operate within structured development cycles, effectively applying Agile methodologies (Scrum, Kanban) for iterative planning, tracking, and delivery.

3. The competency to implement comprehensive testing strategies (unit, integration, and system), conduct code reviews, and use V&V methods to ensure the software is reliable and correct.

4. Ability to apply systematic engineering principles to the continuous maintenance and evolution of deployed software, including corrective, adaptive, and perfective changes, to ensure sustained quality, reliability, and relevance in dynamic operational environments.

5. Professional Conduct and Teamwork: The capacity to communicate effectively, collaborate productively within a team, and adhere to ethical and professional standards in software development.

Books

1. Software Engineering: A practitioner's Approach, Roger S Pressman, sixth edition. McGraw Hill International Edition, 2005
2. Software Engineering, Ian Sommerville, seventh edition, Pearson education, 2004.
3. Best Practices for Applying UML, Part I Darius Šilingas, Ph.D. Principal Trainer for MagicDraw UML
4. Krutchen, P. (2003). *The Rational Unified Process – An Introduction (3rd edition)*. Reading, MA: Addison-Wesley.
5. Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies, Capers Jones, The McGraw-Hill Companies, 2010.