NIRMA UNIVERSITY

Institute:	Institute of Technology, School of Technology
Name of Programme:	MTech CSE, MTech CSE (Data Science), MTech CSE (Cyber Security)
Course Code:	6CS280ME25
Course Title:	Advanced Software Engineering
Course Type:	Department Elective-III
Year of Introduction:	2025-26

L	T	Practical Component				C
		LPW	PW	W	S	
3	0	2	-	-	-	4

Course Learning Outcomes (CLO):

At the end of the course, the students will be able to:

1. apply agile methodologies, test-driven development, and software testing techniques (BL3)
2. analyze software development paradigms, process models, cost estimation of software products, and risk management strategies (BL4)
3. evaluate software security principles, quality assurance models, and reliability engineering frameworks (BL5)
4. design software systems using formal methods, component-based design, and WebApp design techniques. (BL6)

Unit	Contents	Teaching Hours (Total 45)
Unit-I	**Software Engineering Paradigms and SDLC Models:** Overview of Traditional vs. Agile methodologies, Waterfall, Spiral, V-Model, and Rapid Application Development (RAD), component-based Software Engineering, Capability Maturity Model Integration. **Software Engineering Economics:** Software cost estimation models (COCOMO, size-oriented, function-oriented, use case-oriented), Budgeting for software projects, Return on investment (ROI) in software development **Risk Management in Software Engineering**: Identification, Analysis, and Mitigation Strategies, Risk Management Frameworks, Software Failure Analysis	10
Unit-II	**Agile Development:** Agility, cost of change, agile process, extreme programming (XP), adaptive software development (ASD), scrum, dynamic systems development method (DSDM), feature driven development (FDD), lean software development (LSD), agile modeling (AM), agile unified process (AUP), test driven development (TDD), xUnit framework and tools for TDD, behavior driven development (BDD), exploratory testing, acceptance test-driven development (ATDD), crystal methodologies	11

[illegible]

Unit-III	**Overview of Software Security Principles:** Secure SDLC & Secure Coding Best Practices, Threat Modeling & Attack Surface Reduction, OWASP Top 10 Security Risks **Overview of Software Quality Assurance & Testing**: Software Quality, SQA Tasks, Goals and Metrics, Formal Approaches to SQA, Statistical SQA, McCall's and ISO 9126 Quality Factors, ISO 9000 Quality Standards, Software Testing Lifecycle (STLC), ISO/IEC 25010 Software Quality Model, Dynamic & Static Code Analysis **Software Reliability Engineering (SRE) & Chaos Engineering:** Principles of SRE and Site Reliability, Fault Tolerance and Self-Healing Systems	12
Unit-IV	**Introduction to Formal Methods in Software Engineering:** The cleanroom strategy, functional specification, cleanroom design, cleanroom testing, formal specification languages (OCL, Z-specification language) **Design Techniques:** Component-level Design, Pattern-based Design, WebApp Design	12

Self-Study:
The self-study contents will be declared at the commencement of the semester. Around 10% of the questions will be asked from self-study contents

Suggested Readings/ References:

1. Roger S. Pressman, Bruce R. Maxim, Software Engineering: A Practitioner's Approach, McGraw-Hill
2. Len Bass, Paul Clements, Rick Kazman, Software Architecture in Practice, Pearson,
3. Jez Humble, David Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison-Wesley
4. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley
5. Robert C. Martin, Agile Software Development, Principles, Patterns, and Practices, Pearson
6. Ken Schawber, Mike Beedle, Agile Software Development with Scrum, Pearson
7. Lisa Crispin, Janet Gregory, Agile Testing: A Practical Guide for Testers and Agile Teams, Addison Wesley
8. Robert C. Martin, Agile Software Development, Principles, Patterns and Practices, Prentice Hall
9. Mike Cohn, User Stories Applied: For Agile Software, Addison Wesley
10. Laurie Williams, Thomas Zimmermann, AI for Software Engineering: An Open-Source Guide, O'Reilly Media
11. Andreas Zeller, Rahul Gopinath, Learning Software Testing with Machine Learning, Cambridge University Press
12. Mark Dowd, John McDonald, Justin Schuh, The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities, Addison-Wesley
13. Gene Kim, Patrick Debois, John Willis, Jez Humble, The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations, IT Revolution Press.

Suggested List of Experiments:

Sr. No.	Name of Experiments/Exercises	Hours
1	Comparative Study of Waterfall, Spiral, V-Model, and RAD using a Case Study	02
2	Software Cost Estimation using COCOMO (Size-Oriented & Function-Oriented Estimation)	02
3	Risk Identification, Analysis, and Mitigation using a Risk Management Framework	02
4	Implementing Scrum: Creating a Product Backlog, Sprint Planning, and Burndown Chart	04
5	Test-Driven Development (TDD) using xUnit Framework (JUnit, PyTest, or similar tool)	04
6	Behavior-Driven Development (BDD) using Cucumber or Similar Tool	04
7	A. Threat Modeling and Identifying OWASP Top 10 Security Risks for a Web Application B. Conducting Static and Dynamic Code Analysis using SonarQube or a Similar Tool	04
8	Reliability Testing: Implementing Chaos Engineering for Fault Tolerance	02
9	Functional Specification and Cleanroom Testing using Formal Specification Language (Z or OCL)	02
10	Web Application Design using Model-View-Controller (MVC) Pattern. Sample 2 to 3 modules can be designed.	04

[illegible]