SOFTWARE ENGINEERING

ENCT 352

Lecture	**: 3**	**Year : III**
Tutorial	**: 1**	**Part : II**
Practical	**: 3/2**	

Course Objectives:

The objective of this course is to provide students with a foundation in software engineering, covering software characteristics, principles, process models, requirements engineering, system and object-oriented modeling, architectural and design principles, coding standards, testing and quality assurance. Students will gain both theoretical understanding and practical experience in applying software engineering concepts to a project development, including modern practices such as CI/CD, containerization and AI-assisted development.

1 Introduction (4 hours)

- 1.1 Software and software engineering
- 1.2 Nature and characteristics of software
- 1.3 Software application domains
- 1.4 Legacy software
- 1.5 Software crisis
- 1.6 Software myths
- 1.7 Software engineering practice: Essence of practice, general principles

2 The Software Process (8 hours)

- 2.1 Process framework and umbrella activities
- 2.2 Traditional (Plan-driven) process models
  - 2.2.1 Waterfall model and its extensions
  - 2.2.2 Incremental process model
  - 2.2.3 Evolutionary process models (Prototyping, spiral)
- 2.3 Agile and adaptive process models
  - 2.3.1 Agile manifesto and the 12 principles
  - 2.3.2 Agile versus plan-driven development
  - 2.3.3 Scrum framework (Roles, artifacts, ceremonies)
  - 2.3.4 Extreme programming (XP) practices
  - 2.3.5 Lean software development
- 2.4 Model selection considerations

3 Software Requirements Engineering (6 hours)

- 3.1 Requirement engineering process

- 3.2 SRS (Structure, characteristics, users)
- 3.3 Functional and non-functional requirements
- 3.4 Gathering requirements using use cases modeling and scenarios
- 3.5 Agile requirements engineering
  - 3.5.1 User stories and acceptance criteria
  - 3.5.2 Product backlog creation and prioritization
  - 3.5.3 Story mapping basics
  - 3.5.4 Continuous requirements refinement (Backlog grooming)

4 Architectural Design (3 hours)

- 4.1 Introductions and importance
- 4.2 Architectural design principles
- 4.3 Taxonomy of architectural styles
- 4.4 Modular design, cohesion and coupling

5 System Modeling (9 hours)

- 5.1 System modeling
  - 5.1.1 Need for system modeling
  - 5.1.2 Role of abstractions in managing complexity
- 5.2 Process modeling using DFD
- 5.3 Scenario-based analysis
  - 5.3.1 Concept of scenarios
  - 5.3.2 Use-case descriptions and diagrams
- 5.4 Behavioral and structural modeling
  - 5.4.1 Activity diagrams
  - 5.4.2 Class-based modeling

6 Coding and Testing (5 hours)

- 6.1 Coding standards and guidelines
- 6.2 Code review
  - 6.2.1 Code walkthrough
  - 6.2.2 Code inspection
  - 6.2.3 Cleanroom technique
- 6.3 Software testing fundamentals
  - 6.3.1 Verification and validation
  - 6.3.2 Unit testing
  - 6.3.3 Integration testing
  - 6.3.4 System testing
  - 6.3.5 Acceptance testing
- 6.4 Black-box and white-box testing approach
- 6.5 Agile testing practices
  - 6.5.1 Test-driven development (TDD)
  - 6.5.2 Continuous testing

6.5.3 Automated unit testing basics
6.5.4 Refactoring for code quality

7 Software Quality, Assurance, Maintenance (4 hours)

7.1 Quality concepts
7.2 Quality attributes
7.3 Reviews, inspections and QA concepts
7.4 ISO standards, CMMI levels
7.5 Software maintenance and its types
7.6 Maintenance effort and lifecycle considerations

8 Software Configuration Management (3 hours)

8.1 Software configuration management
8.2 Version control and continuous integration
8.3 Change management process
8.4 Branching strategies (Git-flow basics)

9 Recent Trends (3 hours)

9.1 DevOps and continuous deployment pipelines
9.2 Continuous integration/continuous delivery (CI/CD) in modern projects
9.3 Cloud-native, microservices architectures and infrastructure as code (IaC)
9.4 AI-assisted software development
9.5 Low-code/no-code, green software engineering

Tutorial (15 hours)

1. Analyze the project domain, identify challenges, and discuss software myths
2. Select and justify an appropriate software process model
3. Prepare preliminary SRS, including functional and non-functional requirements
4. Elicit and model requirements using use cases (Actors, scenarios, system boundary, diagram, and descriptions)
5. Develop behavioral models (Scenario identification and activity diagrams)
6. Develop a structural model (Class diagram)
7. Propose an architectural style and apply modular design principles
8. Prepare user stories and product backlog; Conduct a mock sprint planning session
9. Define coding standards and perform basic code review/refactoring
10. Design black-box and white-box test cases
11. Apply version control using git (Commit, branch, merge, conflict resolution)
12. Plan a basic CI/CD workflow and compile the final documentation of all artifacts

Practical **(22.5 hours)**

1. Preparation and organization of software requirements specification (SRS) using standard templates
2. Construction of UML diagrams (Use case, activity, class, and sequence) using modeling tools
3. Design of system architecture and component structure using CASE tools
4. Implementation of key system modules following coding standards and best practices
5. Execution of testing procedures (Unit and integration) and documentation of results
6. Application of version control (Git), CI/CD workflow setup, and containerization using docker
7. Term project

Final Exam

The questions will cover all the chapters in the syllabus. The evaluation scheme will be as indicated in the table below:

Chapter	Hours	Marks distribution*
1	4	5
2	8	11
3	6	8
4	3	4
5	9	12
6	5	7
7	4	5
8	3	4
9	3	4
Total	**45**	**60**

* There may be minor deviation in marks distribution.

References

1. Pressman, R. S., Maxim, B. R. (2019). Software engineering: A practitioner's approach. McGraw-Hill Education.
2. Mall, R. (2018). Fundamentals of software engineering. PHI Learning.
3. Sommerville, I. (2016). Software engineering. Pearson.
4. Martin, R. C. (2009). Clean code: A handbook of agile software craftsmanship. Pearson.
5. Schwaber, K., Beedle, M. (2002). Agile software development with Scrum. Prentice Hall.