Chapter 1 : Introduction to Software Engineering **1-1 to 1-10**

Syllabus : Nature of Software, Software Engineering, Software Process, Capability Maturity Model (CMM).

1.1 Nature of Software 1-1

1.1.1 Absence of Fundamental Theory 1-1

1.1.2 Ease of Change 1-1

1.1.3 Rapid Evolution of Technologies 1-2

1.1.4 Low Manufacturing Cost 1-2

1.2 Software Definition 1-2

1.3 Software Engineering : A Layered Technology 1-2

1.3.1 Quality Focus 1-3

1.3.2 Process 1-3

1.3.3 Methods 1-3

1.3.4 Tools 1-4

1.4 The Characteristics of Software 1-4

1.5 The Software Engineering Process Framework 1-4

1.5.1 Umbrella Activities 1-5

1.6 Capability Maturity Model (CMM) 1-6

Chapter 2 : Process Models **2-1 to 2-16**

Syllabus : Generic Process Model, Prescriptive Process Models: The Waterfall Model, V-model, Incremental Process Models, Evolutionary Process Models, Concurrent Models.

2.1 A Generic Process Model 2-1

2.1.1 Communication 2-1

2.1.2 Planning 2-1

2.1.3 Modelling 2-1

2.1.4 Construction 2-1

2.1.5 Deployment 2-2

2.2 Prescriptive Process Models 2-3

2.2.1 The Waterfall Model 2-3

2.2.1(A) V-Model (Software Development) 2-6

2.2.2 Incremental Process Models....2-6
2.2.2(A) The Incremental Model....2-6
2.2.2(B) The RAD Model....2-8
2.3 Evolutionary Process Models....2-10
2.3.1 The Prototyping Paradigm....2-10
2.3.2 The Spiral Model....2-12
2.3.3 The Concurrent Development Model....2-13
2.3.4 Differentiation between Prescriptive and Evolutionary Process Models....2-15

Chapter 3 : Agile Process 3-1 to 3-20

Syllabus : Agile process, Agility Principles, Extreme Programming (XP), Scrum, Kanban model. Self-learning Topics: Personal and Team Process Models.

3.1 Agile Process....3-1
3.1.1 Comparison between the Agile and Evolutionary Process Models....3-2
3.2 Agile Software Development....3-2
3.2.1 Agile methods....3-3
3.2.2 Agile Manifesto....3-3
3.2.3 Agility Principles....3-4
3.3 Extreme Programming Practices....3-5
3.3.1 XP Values....3-5
3.3.2 The XP Process....3-6
3.4 Scrum....3-6
3.4.1 Process Flow....3-7
3.4.2 Scrum Roles....3-8
3.4.3 Scrum Cycle Description....3-9
3.4.4 Product Backlog....3-10
3.4.5 Sprint Planning Meeting....3-11
3.4.6 Sprint Backlog....3-12
3.4.7 Sprint Execution....3-13
3.4.8 Daily Scrum Meeting....3-13
3.4.9 Maintaining Sprint Backlog and Burn-Down Chart....3-14
3.4.10 Sprint Review and Retrospective....3-15

3.5 Introduction to Agile Tools : Kanban Model 3-15

3.5.1 Kanban Boards 3-16

3.5.2 Kanban Cards 3-17

3.5.3 The Benefits of Kanban 3-17

3.5.4 Comparison between Kanban and Scrum 3-19

Chapter 4 : Requirement Engineering 4-1 to 4-30

Syllabus : Software Requirements : Functional & non-functional, user-system requirement engineering process, feasibility studies, elicitation, validation & management, software prototyping, S/W documentation, Analysis and modelling, Requirement Elicitation, Software requirement specification (SRS), Self-learning Topics : prioritizing requirements (Kano diagram) - real life application case study.

4.1 Introduction to Software Requirements 4-1

4.2 Requirements elicitation – validation & management 4-1

4.2.1 Inception 4-2

4.2.2 Elicitation 4-2

4.2.3 Elaboration 4-3

4.2.4 Negotiation 4-3

4.2.5 Specification 4-3

4.2.6 Validation 4-3

4.2.7 Requirement Management 4-4

4.2.8 Initiating the Requirement Engineering Process 4-4

4.3 Feasibility studies 4-6

4.4 Requirements Validation 4-10

4.5 Requirements Management 4-11

4.6 Functional and Non- Functional Requirements 4-12

4.6.1 Functional Requirements 4-12

4.6.2 Non-functional Requirements 4-12

4.7 User-system requirement engineering process 4-13

4.8 Software Prototyping 4-14

4.9 S/W documentation 4-15

4.9.1 Requirement Documentation 4-16

4.10 Requirements Analysis and Modelling 4-17

4.10.1 Analysis Rules of Thumb 4-18

4.10.2 Domain Analysis 4-18
4.10.3 Requirements Modelling Approaches 4-19
4.11 Requirements Elicitation 4-19
4.11.1 Collaborative Requirements Gathering 4-19
4.11.2 Quality Function Deployment 4-19
4.11.3 Usage Scenarios 4-20
4.11.4 Elicitation Workproduct 4-21
4.11.5 Elicitation Techniques 4-21
4.11.6 Developing Use Cases 4-21
4.12 Software Requirements Specification (SRS) 4-23
4.12.1 Writing Software Requirements Specifications 4-25
4.12.2 What is a Software Requirements Specification? 4-25
4.12.3 What Kind of Information Should an SRS Include? 4-26
4.12.4 SRS Template 4-26
4.12.5 Characteristics of an SRS 4-27
4.12.6 Structured Specifications for an Insulin Pump Case Study 4-27
4.12.7 Tabular Specifications for an Insulin Pump Case Study 4-28

Chapter 5 : Software Estimation Metrics **5-1 to 5-24**

Syllabus : Management Spectrum, 3Ps (people, product and process) Process and Project metrics, Software Project Estimation: LOC, FP, Empirical Estimation Models - COCOMO II Model, Specialized Estimation Techniques, Object based estimation, use-case based estimation.

5.1 Management Spectrum : 3P's (people, product and process) 5-1
5.1.1 The People 5-1
5.1.1(A) Stake Holders 5-2
5.1.1(B) Team Leaders 5-2
5.1.1(C) Software Team 5-3
5.1.1(D) Agile Teams 5-4
5.1.1(E) Co-ordination and Communication Issues 5-4
5.1.2 The Product 5-5
5.1.3 The Process 5-5
5.1.4 The Project 5-5

5.2 Software Metrics 5-6

5.2.1 Process Metrics 5-6

5.2.2 Project Metrics 5-7

5.3 Software Project Estimation 5-7

5.3.1 Observations on Estimation 5-8

5.3.2 Software Sizing 5-8

5.3.3 Problem-Based Estimation 5-9

5.3.4 An Example of LOC-Based Estimation 5-9

5.3.5 An Example of FP-Based Estimation 5-10

5.3.6 Process-Based Estimation 5-12

5.3.6(A) An Example of Process-Based Estimation 5-13

5.3.7 Estimation with Use-Cases 5-13

5.3.7(A) An Example of Use-Case Based Estimation 5-14

5.3.8 Reconciling Estimates 5-14

5.3.9 Software Scope and Feasibility 5-15

5.3.9(A) Obtaining Information Necessary for Scope 5-15

5.3.9(B) Feasibility 5-16

5.3.9(C) A Scoping Example 5-16

5.4 Empirical Estimation Models 5-16

5.4.1 The Structure of Estimation Models 5-17

5.4.2 The COCOMO II Model 5-17

5.4.3 The Software Equation 5-19

5.5 Specialized Estimation Techniques 5-20

5.5.1 Estimation for Agile Development 5-20

5.5.2 Estimation for Web Engineering Projects 5-20

5.6 Object based Estimation 5-21

5.7 Use-case based Estimation 5-22

5.7.1 An Example of Use-Case Based Estimation 5-22

Chapter 6 : Project Scheduling and Tracking **6-1 to 6-8**

Syllabus : Project scheduling: Defining a Task Set for the Software Project, Timeline charts, Tracking the Schedule, Earned Value Analysis . Self-learning Topics: Cost Estimation Tools and Techniques, Typical Problems with IT Cost Estimates.

6.1 Project Scheduling **6-1**

6.1.1 Defining a Task Set for the Software Project 6-2

6.1.2 Scheduling 6-3

6.1.2(A) Time-line Charts 6-3

6.2 Tracking Software Project Estimation **6-3**

6.3 Tracking the Schedule **6-4**

6.3.1 Microsoft Project 6-5

6.3.2 Daily Activity Reporting and Tracking (DART) 6-6

6.4 Earned Value Analysis **6-7**

Chapter 7 : Design Engineering **7-1 to 7-16**

Syllabus : Design Process & quality, Design Concepts , The design Model, Pattern-based Software Desig

7.1 Introduction to Design Engineering **7-1**

7.2 Design Process **7-1**

7.3 Design Quality **7-1**

7.3.1 Quality of Design Guidelines 7-2

7.3.2 The Quality Attributes 7-2

7.4 Design Concepts **7-3**

7.4.1 Abstraction 7-3

7.4.2 Architecture 7-4

7.4.3 Patterns 7-4

7.4.4 Modularity 7-4

7.4.5 Information Hiding 7-6

7.4.6 Functional Independence 7-6

7.4.7 Refinement 7-8

7.4.8 Refactoring 7-9

7.4.8(A) Importance of refactoring 7-9

7.4.9 Design Classes 7-9

7.4.10 Differentiation between Abstraction and Refinement 7-9

7.5 The Design Model 7-10

7.5.1 Data Design Elements 7-11

7.5.2 Architectural Design Elements 7-11

7.5.3 Interface Design Elements 7-11

7.5.4 Component-Level Design Elements 7-12

7.5.5 Deployment-Level Design Elements 7-13

7.5.6 Translating Requirements Model to Design Model 7-13

7.5.7 Guidelines for the Data Design 7-14

7.6 Pattern-Based Software Design 7-14

7.6.1 Describing a Design Pattern 7-15

7.6.2 Using Patterns in Design 7-15

7.6.3 Frameworks 7-16

Chapter 8 : Architectural Design 8-1 to 8-14

Syllabus : Design Decisions, Views, Patterns, Application Architectures, Modeling , Component level Design : component, Designing class based components, conducting component-level design.

8.1 Introduction to Architectural Design 8-1

8.2 Architectural Design Decisions 8-3

8.3 Architectural Views 8-4

8.4 Architectural Patterns 8-6

8.4.1 Software Architecture 8-6

8.5 Application Architectures 8-7

8.5.1 Transaction Processing Systems 8-8

8.5.2 Language Processing Systems 8-9

8.6 Conducting Component level Design 8-11

8.7 Class-based Components 8-13

8.7.1 Basic Design Principles 8-14

8.7.2 Conducting Component-Level Design 8-14

Chapter 9 : User Interface Design **9-1 to 9-14**

Syllabus : The golden rules, Interface Design steps & Analysis, Design Evaluation, Case Study : Web App Interface Design. Self-learning Topics: Refinement, Aspects, Refactoring.

9.1 User Interface Design 9-1

9.1.1 Type of User Interface 9-2

9.1.2 Characteristics of Good User Interface 9-4

9.1.3 Benefits of Good Interface Design 9-4

9.2 The Golden Rules 9-4

9.2.1 Place the user in Control 9-5

9.2.2 Reduce the User's Memory Load 9-6

9.2.3 Make the Interface Consistent 9-6

9.2.4 Necessity of a Good User Interface 9-7

9.3 Shneiderman's 8 Golden Rules for UI Analysis 9-7

9.4 Interface Analysis and Design Models 9-9

9.4.1 Interface Analysis and Design Models 9-9

9.4.2 User Interface Design Process 9-9

9.5 Interface Design Steps and Analysis 9-10

9.5.1 Applying Interface Design Steps 9-11

9.5.2 User Interface Design Patterns 9-11

9.5.3 Interface Design Issues 9-11

9.5.4 Interface Design Evaluation 9-12

9.6 Design Evaluation 9-13

9.7 WebApp Interface Design 9-14

9.7.1 WebApp Design Principles 9-14

Chapter 10 : Risk Management **10-1 to 10-14**

Syllabus : Risk Identification, Risk Assessment, Risk Projection, RMMM.

10.1 Introduction to Risk Management 10-1

10.1.1 Reactive Versus Proactive Risk Strategies 10-2

10.1.2 Various steps in Risk Management 10-3

10.2 Risk Identification 10-3

10.2.1 Risk Components and Drivers 10-4

10.3 Risk Assessment 10-6

10.4 Risk Projection 10-6

10.4.1 Developing a Risk Table 10-7

10.4.2 Assessing Risk 10-8

10.5 RMMM (Risk Mitigation, Monitoring and Planning) 10-9

10.5.1 The RMMM Plan 10-9

Chapter 11 : Software Configuration Management 11-1 to 11-12

Syllabus : Software Configuration management, SCM repositories, SCM process.

11.1 Software Configuration Management (SCM) 11-1

11.1.1 SCM Basics (Configuration Management System Elements) 11-2

11.1.2 Baselines 11-2

11.1.3 Software Configuration Items 11-2

11.2 The SCM Repository 11-4

11.2.1 The Role of the Repository 11-4

11.2.2 General Features and Content 11-5

11.2.3 SCM Features 11-6

11.3 The SCM Process 11-7

11.3.1 Identification of Objects in the Software Configuration 11-8

11.3.2 Version Control 11-8

11.3.3 Change Control 11-9

11.3.4 Configuration Audit 11-10

11.3.5 Status Reporting 11-11

Chapter 12 : Software Quality Assurance (SQA) 12-1 to 12-16

Syllabus : Software Quality Assurance Task and Plan, Metrics, Software Reliability, Formal Technical Review (FTR), Walkthrough. Self-learning Topics:: Configuration management for WebApps.

12.1 Software Quality Assurance (SQA) 12-1

12.1.1 Types of Standards 12-2

12.1.2 Software Quality Assurance Activities 12-2

12.1.3 SQA Relationships to Other Assurance Activities 12-3

12.2 SQA Plans 12-7

12.3 SQA Task 12-8

12.3.1 McCall's Quality Factors 12-9

12.3.2 ISO 9126 Quality Factors 12-10

12.4 SQA Metrics 12-11

12.4.1 Measuring Quality 12-11

12.4.2 Defect Removal Efficiency 12-12

12.5 Formal Technical Reviews (FTR) 12-12

12.5.1 Review Meetings 12-13

12.5.2 Review Guidelines 12-13

12.6 Software Reliability 12-14

12.6.1 Measures of Reliability and Availability 12-14

12.6.2 Software Safety 12-14

12.7 Walkthrough 12-15

12.7.1 Differentiate between Walkthrough and FTR 12-15

Chapter 13 : Testing Strategies and Tactics 13-1 to 13-24

Syllabus : Software Quality, Strategic Approach, Strategic Issues, Strategies for Conventiona, Software, Object oriented software, WebApps - Validating Testing- System Testing- Art of Debugging.

13.1 Software Quality 13-1

13.2 A Strategic Approach to Software Testing 13-1

13.2.1 Verification and Validation 13-2

13.2.1(A) Difference between Verification and Validation 13-3

13.2.2 Organizing for Software Testing 13-3

13.2.3 A Software Testing Strategy for Conventional Software Architectures 13-4

13.2.4 Software Testing 13-6

13.3 Strategic Issues 13-6

13.4 Test Strategies for Conventional Architectures 13-7

13.4.1 Unit Testing 13-7

13.4.2 Integration Testing 13-9

13.4.3 Regression Testing 13-11

13.4.4 Smoke Testing 13-11
13.4.5 Comments on Integration Testing 13-11
13.4.6 Integration Test Documentation 13-11
13.5 Test Strategies for Object-Oriented Architectures 13-12
13.5.1 Unit Testing in the OO Context 13-12
13.5.2 Integration Testing in the OO Context 13-12
13.6 Test Strategies for WebApps 13-12
13.7 Validation Testing 13-13
13.7.1 Validation Test Criteria 13-13
13.7.2 Configuration Review 13-13
13.7.3 Acceptance Testing 13-13
13.7.4 Alpha and Beta Testing 13-14
13.7.4(A) Difference between Alpha Testing and Beta Testing 13-15
13.8 System Testing 13-15
13.8.1 Recovery Testing 13-15
13.8.2 Security Testing 13-16
13.8.3 Stress Testing 13-16
13.8.4 Performance Testing 13-17
13.8.5 Verification and Validation 13-17
13.9 Object-Oriented Testing Strategies 13-17
13.9.1 Unit Testing in the OO Context 13-17
13.9.2 Integration Testing in the OO Context 13-17
13.10 Object-Oriented Testing Methods 13-18
13.10.1 The Test Cases and the Class Hierarchy 13-18
13.10.2 Applicability of Conventional Test Case Design Methods 13-19
13.10.3 Fault-Based Testing 13-19
13.10.4 Scenario-Based Testing 13-19
13.10.5 Testing Surface Structure and Deep Structure 13-19
13.11 Bug Reporting (The art of debugging) 13-20
13.11.1 The Debugging Process 13-20
13.11.2 Psychological Considerations 13-21
13.11.3 Debugging Approaches 13-22

Chapter 14 : Software Maintenance **14-1 to 14-12**

Syllabus : Software Maintenance, Software Supportability, Reengineering, Business Process Reengineering, Software Reengineering, Reverse Engineering, Restructuring, Forward Engineering. Self-learning Topics: Test Strategies for WebApps.

14.1 Software Maintenance 14-1

14.1.1 Modifiability 14-1

14.1.2 Types of Maintenance 14-2

14.1.2(A) Corrective Maintenance 14-2

14.1.2(B) Adaptive Maintenance 14-2

14.1.2(C) Perfective Maintenance 14-3

14.1.2(D) Preventive Maintenance 14-3

14.1.3 Need of Maintenance 14-3

14.2 Software Supportability 14-3

14.3 Reengineering 14-4

14.3.1 Re-Engineering Process Model 14-4

14.4 Business Process Reengineering 14-5

14.5 Software Reengineering 14-7

14.6 Reverse Engineering 14-8

14.6.1 Abstraction Level 14-9

14.6.2 Completeness 14-9

14.6.3 Directionality 14-9

14.7 Restructuring 14-10

14.8 Forward Engineering 14-11

❑❑❑