Detailed Contents

Software Engineering

Text Book 5th Edition

Unit 1

1. Introduction 1–33

- 1.1 Evolution—From an Art Form to an Engineering Discipline *4*
- 1.1.1 Evolution of an Art into an Engineering Discipline *4*
- 1.1.2 Evolution Pattern for Engineering Disciplines *4*
- 1.1.3 A Solution to the Software Crisis *6*
- 1.2 Software Development Projects *7*
- 1.2.1 Programs *versus* Products *7*
- 1.2.2 Types of Software Development Projects *8*
- 1.2.3 Software Projects Being Undertaken by Indian Companies *10*
- 1.3 Exploratory Style of Software Development *10*
- 1.3.1 Perceived Problem Complexity: An Interpretation Based on Human Cognition Mechanism *12*
- 1.3.2 Principles Deployed by Software Engineering to Overcome Human cognitive Limitations *15*
- 1.4 Emergence of Software Engineering *18*
- 1.4.1 Early Computer Programming *19*
- 1.4.2 High-level Language Programming *19*
- 1.4.3 Control Flow-based Design *19*
- 1.4.4 Data Structure-oriented Design *23*
- 1.4.5 Data Flow-oriented Design *23*
- 1.4.6 Object-oriented Design *24*
- 1.4.7 What Next? *25*
- 1.4.8 Other Developments *26*
- 1.5 Notable Changes in Software Development Practices *26*
- 1.6 Computer Systems Engineering *28*

2. Software Life Cycle Models 34–92

Unit 2

- 2.1 A Few Basic Concepts *36*
- 2.2 Waterfall Model and Its Extensions *40*
- 2.2.1 Classical Waterfall Model *40*
- 2.2.2 Iterative Waterfall Model *48*
- 2.2.3 V-Model *52*
- 2.2.4 Prototyping Model *53*
- 2.2.5 Incremental Development Model *56*
- 2.2.6 Evolutionary Model *58*
- 2.3 Rapid Application Development (RAD) *60*
- 2.3.1 Working of RAD *61*
- 2.3.2 Applicability of RAD Model *62*
- 2.3.3 Comparison of RAD with Other Models *63*
- 2.4 Agile Development Models *64*
- 2.4.1 Essential Idea behind Agile Models *65*
- 2.4.2 Agile ***versus*** Other Models *66*
- 2.4.3 Extreme Programming Model *67*
- 2.4.4 Scrum *70*
- 2.4.5 Lean Software Development *73*
- 2.5 Spiral Model *74*
- 2.5.1 Phases of the Spiral Model *76*
- 2.6 A Comparison of Different Life Cycle Models *77*
- 2.6.1 Selecting an Appropriate Life Cycle Model for a Project *78*

4. Requirements Analysis and Specification

Unit 2

- 4.1 Requirements Gathering and Analysis *173*
- 4.1.1 Requirements Gathering *174*
- 4.1.2 Requirements Analysis *177*
- 4.2 Software Requirements Specification (SRS) *179*
- 4.2.1 Users of SRS Document *179*
- 4.2.2 Why Spend Time and Resource to Develop an SRS Document? *180*
- 4.2.3 Characteristics of a Good SRS Document *181*
- 4.2.4 Attributes of Bad SRS Documents *182*
- 4.2.5 Important Categories of Customer Requirements *183*
- 4.2.6 Functional Requirements *185*
- 4.2.7 How to Identify the Functional Requirements? *188*
- 4.2.8 How to Document the Functional Requirements? *188*
- 4.2.9 Traceability *190*
- 4.2.10 Organization of the SRS Document *191*
- 4.2.11 Techniques for Representing Complex Logic *197*

5. Software Design 219–239

Unit 2

- 5.1 Overview of the Design Process *220*
- 5.1.1 Outcome of the Design Process *220*
- 5.1.2 Classification of Design Activities *221*
- 5.1.3 Classification of Design Methodologies *222*
- 5.2 How to Characterize a Good Software Design? *223*
- 5.2.1 Understandability of a Design: A Major Concern *223*
- 5.3 Cohesion and Coupling *226*
- 5.3.1 Classification of Cohesiveness *227*
- 5.3.2 Classification of Coupling *229*
- 5.4 Layered Arrangement of Modules *230*
- 5.5 Approaches to Software Design *232*
- 5.5.1 Function-oriented Design *232*
- 5.5.2 Object-oriented Design *233*

6. Function-Oriented Software Design 240–294

Unit 2

- 6.1 Overview of SA/SD Methodology *241*
- 6.2 Structured Analysis *243*
- 6.2.1 Data Flow Diagrams (DFDs) *243*
- 6.3 Developing the DFD Model of a System *247*
- 6.3.1 Context Diagram *247*
- 6.3.2 Level 1 DFD *249*
- 6.3.3 Extending DFD Technique to make it Applicable to Real-time Systems *264*
- 6.4 Structured Design *265*
- 6.4.1 Transformation of a DFD Model into Structure Chart *266*
- 6.5 Detailed Design *270*
- 6.6 Design Review *271*

Unit 2

7. Object Modelling Using UML 295–358

- 7.1 Basic Object-Orientation Concepts *297*
- 7.1.1 Basic Concepts *297*
- 7.1.2 Class Relationships *301*
- 7.1.3 How to Identify Class Relationships? *308*
- 7.1.4 Other Key Concepts *308*
- 7.1.5 Related Technical Terms *314*
- 7.1.6 Advantages and Disadvantages of OOD *314*
- 7.2 Unified Modelling Language (UML) *316*
- 7.2.1 Origin of UML *316*
- 7.2.2 Evolution of UML *318*
- 7.3 UML Diagrams *320*
- 7.4 Use Case Model *322*
- 7.4.1 Representation of Use Cases *323*
- 7.4.2 Why Develop the Use Case Diagram? *327*
- 7.4.3 How to Identify the Use Cases of a System? *327*
- 7.4.4 Essential Use Case ***versus*** Real Use Case *327*
- 7.4.5 Factoring of Commonality among Use Cases *328*
- 7.4.6 Use Case Packaging *330*
- 7.5 Class Diagrams *331*
- 7.6 Sequence Diagram *338*
- 7.7 Activity Diagram *340*
- 7.8 State Chart Diagram *342*
- 7.9 Postscript *343*
- 7.9.1 Package, Component, and Deployment Diagrams *344*
- 7.9.2 UML 2.0 *345*

9. User Interface Design 398–422

Unit 2

- 9.1 Characteristics of a Good User Interface *400*
- 9.2 Basic Concepts *402*
- 9.2.1 User Guidance and On-line Help *402*
- 9.2.2 Mode-based ***versus*** Modeless Interface *403*
- 9.2.3 Graphical User Interface (GUI) ***versus*** Text-based User Interface *403*
- 9.3 Types of User Interfaces *404*
- 9.3.1 Command Language-based Interface *404*
- 9.3.2 Menu-based Interface *405*
- 9.3.3 Direct Manipulation Interfaces *407*

Unit 3

- Verification and Validation
- Software Quality Frameworks
- ISO 9000 Models