Table of contents

- Preface
- The Issue of Scale
- This Edition Compared with the First
- How Instructors Can Use This Book
- Acknowledgments
- Part I Introduction To Software Engineering
- Chapter 1 The Goals and Terminology of Software Engineering
- What is Software Engineering
- Why Software Engineering Is Critical: Software Disasters
- Why Software Fails and Succeeds
- Software Engineering Activities
- Software Engineering Principles
- Ethics in Software Engineering
- Case Studies
- Summary
- Exercises
- Bibliography
- Chapter 2 Introduction to Quality and Metrics in Software Engineering
- 2.1 The Meaning of Software Quality
- 2.2 Defects in Software
- 2.3 Verification and Validation
- 2.4 Planning for Quality
- 2.5 Metrics
- 2.6 Summary
- 2.7 Exercises
- Bibliography
- Part II Software Process
- Chapter 3 Software Process
- 3.1 The Activities of Software Process
- 3.2 Software Process Models
- 3.3 Case Study: Student Team Guidance
- 3.4 Summary
- 3.5 Exercises
- Bibliography
- Chapter 4 Agile Software Processes
- 4.1 Agile History and Agile Manifesto
- 4.2 Agile Principles
- 4.3 Agile Methods
- 4.4 Agile Processes
- 4.5 Integrating Agile with Non-Agile Processes
- 4.6 Summary
- 4.7 Exercises
- Bibliography
- Chapter 5 Quality in the Software Process

- **5.1 Principles of Managing Quality**
- **5.2 Managing Quality in Agile Processes**
- **5.3 Quality Planning**
- **5.4 Inspections**
- **5.5 QA Reviews and Audits**
- **5.6 Defect Management**
- **5.7 Process Improvement and Process Metrics**
- **5.8 Organization-Level Quality and the CMMI**
- **5.9 Case Study**
- **5.10 Summary**
- **5.11 Exercises**
- **Bibliography**
- **Chapter 6 Software Configuration Management**
- **6.1 Software Configuration Management Goals**
- **6.2 SCM Activities**
- **6.3 Configuration Management Plans**
- **6.4 Configuration Management Systems**
- **6.5 Case Study: Encounter Video Game**
- **6.6 Case Study: Eclipse**
- **6.7 Student Team Guidance: Configuration Management**
- **6.8 Summary**
- **6.9 Exercises**
- **Bibliography**
- **Part III Project Management**
- **Chapter 7 Principles of Software Project Management I: Organization, Tools, and Risk Management**
- **7.1 Software Project Organization**
- **7.2 Team Size**
- **7.3 Geographically Distributed Development**
- **7.4 The Team Software Process**
- **7.5 Software Project Tools and Techniques**
- **7.6 Risk Management**
- **7.7 Student Team Guidance: Organizing the Software Projec's Management**
- **7.8 Summary**
- **7.9 Exercises**
- **Bibliography**
- **Chapter 8 Principles of Software Project Management II: Estimation, Scheduling, and Planning**
- **8.1 Cost Estimation**
- **8.2 Scheduling**
- **8.3 The Software Project Management Plan**
- **8.4 Case Study: Encounter Project Management Plan**
- **8.5 Case Study: Project Management in Eclipse**
- **8.6 Case Study: Project Management for OpenOffice**
- **8.7 Case Study: Student Team Guidance**
- **8.8 Summary**

- 8.9 Exercises
- Bibliography
- Chapter 9 Quality and Metrics in Project Management
- 9.1 Cultivating and Planning Internal Quality
- 9.2 Project Metrics
- 9.3 Using Metrics for Improvement
- 9.4 Software Verification and Validation Plan
- 9.5 Case Study: Software Verification and Validation Plan for Encounter
- 9.6 Summary
- 9.7 Exercises
- Bibliography
- Part IV Requirements Analysis
- Chapter 10 Principles of Requirements Analysis
- 10.1 The Value of Requirements Analysis
- 10.2 Sources of Requirements
- 10.3 High-level vs. Detailed
- Requirements
- 10.4 Types of Requirements
- 10.5 Nonfunctional Requirements
- 10.6 Documenting Requirements
- 10.7 Traceability
- 10.8 Agile Methods and Requirements
- 10.9 Updating the Project to Reflect Requirements Analysis
- 10.10 Summary
- 10.11 Exercises
- Bibliography
- Chapter 11 Analyzing High-Level Requirements
- 11.1 Examples of Customer Wants
- 11.2 Stakeholder Vision
- 11.3 The Interview and Documentation Process
- 11.4 Writing an Overview
- 11.5 Describing Main Functions and Used Cases
- 11.6 Agile Methods for High-Level Requirements
- 11.7 Specifying User Interfaces: High Level
- 11.8 Security Requirements
- 11.9 Using Diagrams for High-Level Requirements
- 11.10 Case Study: High-Level Software Requirements Specifications (SRS) for the Encounter Video Game
- 11.11 Case Study: High-Level Requirements for Eclipse
- 11.12 Case Study: High-Level Requirements for OpenOffice
- 11.13 Summary
- 11.14 Exercises
- Bibliography
- Chapter 12 Analyzing Detailed Requirements
- 12.1 The Meaning of Detailed Requirements
- 12.2 Organizing Detailed Requirements

- 12.3 User Interfaces: Detailed Requirements
- 12.4 Detailed Security Requirements
- 12.5 Error Conditions
- 12.6 Traceability of Detailed Requirements
- 12.7 Using Detailed Requirements to Manage Projects
- 12.8 Prioritizing Requirements
- 12.9 Associating Requirements with Tests
- 12.10 Agile Methods for Detailed Requirements
- 12.11 Using Tools and the Web for Requirements Analysis
- 12.12 The Effects on Projects of the Detailed Requirements Process
- 12.13 Student Project Guide: Requirements for the Encounter Case Study
- 12.14 Case Study: Detailed Requirements for the Encounter Video Game
- 12.15 Summary
- 12.16 Exercises
- Bibliography
- Chapter 13 Quality and Metrics in Requirements Analysis
- 13.1 Quality of Requirements for Agile Projects
- 13.2 Accessibility of Requirements
- 13.3 Comprehensiveness of Requirements
- 13.4 Understandability of Requirements
- 13.5 Un-ambiguity of Requirements
- 13.6 Consistency of Requirements
- 13.7 Prioritization of Requirements
- 13.8 Security and High-Level Requirements
- 13.9 Self-Completeness of Requirements
- 13.10 Testability of Requirements
- 13.11 Traceability of Requirements
- 13.12 Metrics for Requirements Analysis
- 13.13 Inspecting Detailed Requirements
- 13.14 Summary
- 13.15 Exercises
- Chapter 14 Online Chapter - Formal and Emerging Methods in Requirements Analysis as follows: An Introduction
- 14.1 Provable Requirements Method
- 14.2 Introduction to Formal Methods
- 14.3 Mathematical Preliminaries
- 14.4 The Z-Specification Language
- 14.5 The B Language System
- 14.6 Trade-offs for Using a B-like system
- 14.7 Summary
- 14.8 Exercises
- Bibliography
- Part V Software Design
- Chapter 15 Principles of Software Design
- 15.1 The Goals of Software Design
- 15.2 Integrating Design Models

- 15.3 Frameworks
- 15.4 IEEE Standards for Expressing Designs
- 15.6 Summary
- 15.7 Exercises
- Chapter 16 The Unified Modeling Language
- 16.1 Classes in UML
- 16.2 Class Relationships in UML
- 16.3 Multiplicity
- 16.4 Inheritance
- 16.5 Sequence Diagrams
- 16.6 State Diagrams
- 16.7 Activity Diagrams
- 16.8 Data Flow Models
- 16.9 A Design Example with UML
- 16.10 Summary
- 16.11 Exercises
- Bibliography
- Chapter 17 Software Design Patterns
- 17.1 Examples of a Recurring Design Purpose
- 17.2 An Introduction to Design Patterns
- 17.3 Summary of Design Patterns by Type: Creational, Structural, and Behavioral
- 17.4 Characteristics of Design Patterns: Viewpoints, Roles, and Levels
- 17.5 Selected Creational Design Patterns
- 17.6 Selected Structural Design Patterns
- 17.7 Selected Behavioral Design Patterns
- 17.8 Design Pattern Forms: Delegation and Recursion
- 17.9 Summary
- 17.10 Exercises
- Bibliography
- Chapter 18 Software Architecture
- 18.1 A Categorization of Architectures
- 18.2 Software Architecture Alternatives and Their Class Models
- 18.3 Trading Off Architecture Alternatives
- 18.4 Tools for Architectures
- 18.5 IEEE Standards for Expressing Designs
- 18.6 Effects of Architecture Selection on the Project Plan
- 18.7 Case Study: Preparing to Design Encounter (Student Project Guide continued)
- 18.8 Case Study: Software Design Document for the Role-Playing Video Game Framework
- 18.9 Case Study: Software Design Document for Encounter (Uses the Framework)
- 18.10 Case Study: Architecture of Eclipse
- 18.11 Case Study: OpenOffice Architecture
- 18.12 Summary

- 18.13 Exercises
- Bibliography
- Chapter 19 Detailed Design
- 19.1 Relating Use Cases, Architecture, and Detailed Design
- 19.2 A Typical Road Map for the Detailed Design Process
- 19.3 Object-Oriented Design Principles
- 19.4 Designing against Interfaces
- 19.5 Specifying Classes, Functions, and Algorithms
- 19.6 Reusing Components
- 19.7 Sequence and Data Flow Diagrams for Detailed Design
- 19.8 Detailed Design and Agile Processes
- 19.9 Design in the Unified Development Process
- 19.10 IEEE Standard 890 for Detailed Design
- 19.11 Updating a Project with Detailed Design
- 19.12 Case Study: Detailed Design of Encounter
- 19.13 Case Study: Detailed Design of Eclipse
- 19.14 Summary
- 19.15 Exercises
- Bibliography
- Chapter 20 Design Quality and Metrics
- 20.1 Degree of Understandability, Cohesion, and Coupling
- 20.2 Degree of Sufficiency as a Quality Goal
- 20.3 Degree of Robustness as a Quality Goal
- 20.4 Degree of Flexibility as a Design Quality Goal
- 20.5 Degree of Reusability as a Design Quality Goal
- 20.6 Degree of Time Efficiency as a Design Quality Measure
- 20.7 Degree of Space Efficiency as a Design Quality Measure
- 20.8 Degree of Reliability as a Design Quality Measure
- 20.9 Degree of Security as a Design Quality Measure
- 20.10 Assessing Quality in Architecture Selection
- 20.11 Assessing the Quality of Detailed Designs
- 20.12 Summary
- 20.13 Exercises
- Bibliography
- Chapter 21 Online Chapter - Advanced and Emerging Methods in Software Design
- 21.1 Designing in a Distributed Environment
- 21.2 Introduction to Aspect-Oriented Programming
- 21.3 Designing for Security with UMLsec
- 21.4 Model-Driven Architectures
- 21.5 The Formal Design Process in B
- 21.6 Summary
- 21.7 Exercises
- Bibliography
- Part VI Implementation
- Chapter 22 Principles of Implementation

- **22.1 Agile and Non-Agile Approaches to Implementation**
- **22.2 Choosing a Programming Language**
- **22.3 Identifying Classes**
- **22.4 Defining Methods**
- **22.5 Implementation Practices**
- **22.6 Defensive Programming**
- **22.7 Coding Standards**
- **22.8 Comments**
- **22.9 Tools and Environments for Programming**
- **22.10 Case Study: Encounter Implementation**
- **22.11 Case Study: Eclipse**
- **22.12 Case Study: OpenOffice**
- **22.13 Student Team Guidance for Implementation**
- **22.14 Summary**
- **22.15 Code Listings Referred to in This Chapter**
- **22.16 Exercises**
- **Bibliography**
- **Chapter 23 Quality and Metrics in Implementation**
- **23.1 Quality of Implementation**
- **23.2 Code Inspections**
- **23.3 Summary**
- **23.4 Exercises**
- **Chapter 24 Refactoring**
- **24.1 Big Refactorings**
- **24.2 Composing Methods**
- **24.3 Moving Features between Objects**
- **24.4 Organizing Data**
- **24.5 Generalization**
- **24.6 Introducing Modules**
- **24.7 Refactoring in Projects**
- **24.8 Summary**
- **24.9 Exercises**
- **Bibliography**
- **Part VII Testing And Maintenance**
- **Chapter 25 Introduction to Software Testing**
- **25.1 Testing Early and Often and the Agile Connection**
- **25.2 Retesting: Regression Testing**
- **25.3 Black Box and White Box Testing**
- **25.4 Unit Testing vs. Post-Unit Testing**
- **25.5 Testing Object-Oriented Implementations**
- **25.6 Documenting Tests**
- **25.7 Test Planning**
- **25.8 Testing Test Suites by Fault Injection**
- **25.9 Summary**
- **25.10 Exercises**
- **Chapter 26 Unit Testing**

- 26.1 The Sources of Units for Unit Testing
- 26.2 Unit Test Methods
- 26.3 Testing Methods
- 26.4 Test-Driven Development
- 26.5 Case Study: Encounter Video Game
- 26.6 Summary
- 26.7 Exercises
- Bibliography
- Chapter 27 Module and Integration Testing
- 27.1 Stubs and Drivers
- 27.2 Testing a Class
- 27.3 Integration
- 27.4 Daily Builds
- 27.5 Interface Testing
- 27.6 Module Integration
- 27.7 Case Study: Class Test for Encounter
- 27.8 Case Study: Encounter Integration Plan
- 27.9 Summary
- 27.10 Exercises
- Bibliography
- Chapter 28 Testing at the System Level
- 28.1 Functional Testing
- 28.2 Nonfunctional Testing
- 28.3 Testing with Lightweight Requirements
- 28.4 Testing Shortly Before Release
- 28.5 Case Study: Encounter Software Test Documentation
- 28.6 Case Study: Eclipse
- 28.7 Case Study: OpenOffice
- 28.8 Summary
- 28.9 Exercises
- Bibliography
- Chapter 29 Software Maintenance
- 29.1 Types of Software Maintenance
- 29.2 Issues of Software Maintenance
- 29.3 Maintenance Process
- 29.4 IEEE Maintenance Standards
- 29.5 Software Evolution
- 29.6 Maintenance Metrics
- 29.7 Case Studies
- 29.8 Summary
- 29.9 Exercises
- Bibliography
- Glossary
- Index