

Contents

Preface xvi

Part I

Introduction and System Engineering 1

Chapter 1

Introduction 2

- 1.1 What Is Software Engineering? 2
- 1.2 Why Software Engineering? 3
- 1.3 Software Life-Cycle Activities 4
 - 1.3.1 Software Development Process 5
 - 1.3.2 Software Quality Assurance 9
 - 1.3.3 Software Project Management 10
- 1.4 Object-Oriented Software Engineering 11
 - 1.4.1 Object-Oriented Modeling and Design Languages 12
 - 1.4.2 Object-Oriented Development Processes 12
 - 1.4.3 Object-Oriented Development Methodologies 12
 - 1.4.4 Will OO Replace the Conventional Approaches? 13
- 1.5 Software Engineering and Computer Science 13
 - Summary 14
 - Further Reading 15
 - Chapter Review Questions 15
 - Exercises 15

Chapter 2

Software Process and Methodology 16

- 2.1 Challenges of System Development 17
- 2.2 Software Process 18

2.3 Merits and Problems of the Waterfall Process 19

2.4 Software Development Is a Wicked Problem 19

2.5 Software Process Models 21

2.5.1 Prototyping Process 21

2.5.2 Evolutionary Process 22

2.5.3 Spiral Process 22

2.5.4 The Unified Process 23

2.5.5 Personal Software Process 25

2.5.6 Team Software Process 28

2.5.7 Agile Processes 30

2.6 Software Development Methodology 37

2.6.1 Difference between Process and Methodology 37

2.6.2 Benefits of a Methodology 38

2.6.3 Structured Methodologies 39

2.6.4 Classical OO Methodologies 39

2.7 Agile Methods 40

2.7.1 Dynamic Systems Development Method 40

2.7.2 Scrum 42

2.7.3 Feature Driven Development 43

2.7.4 Extreme Programming 44

2.7.5 Agile or Plan-Driven 44

2.8 Overview of Process and Methodology of the Book 45

Summary 50

Further Reading 51

Chapter Review Questions 51

Exercises 51

Chapter 3

System Engineering 53

3.1 What Is a System? 54

3.2 What Is System Engineering? 55

- 3.3 System Requirements Definition 58**
 - 3.3.1 Identifying Business Needs 58**
 - 3.3.2 Defining System Requirements 60**
- 3.4 System Architectural Design 60**
 - 3.4.1 System Decomposition 61**
 - 3.4.2 Requirements Allocation 64**
 - 3.4.3 Architectural Design Diagrams 66**
 - 3.4.4 Specification of Subsystem Functions and Interfaces 70**
- 3.5 Subsystems Development 71**
 - 3.5.1 Object-Oriented Context Diagram 71**
 - 3.5.2 Usefulness of an Object-Oriented Context Diagram 72**
 - 3.5.3 Collaboration of Engineering Teams 73**
- 3.6 System Integration, Testing, and Deployment 73**
- 3.7 System Configuration Management 74**
 - Summary 76
 - Further Reading 76
 - Chapter Review Questions 76
 - Exercises 76

 Part II

Analysis and Architectural Design 79

Chapter 4

- Software Requirements Elicitation 80**
 - 4.1 What Is Requirements Elicitation? 81**
 - 4.2 Importance of Requirements Elicitation 82**
 - 4.3 Challenges of Requirements Elicitation 83**
 - 4.4 Types of Requirement 85**
 - 4.5 Steps for Requirements Elicitation 86**
 - 4.5.1 Collecting Information 87**
 - 4.5.2 Constructing Analysis Models 91**
 - 4.5.3 Deriving Requirements and Constraints 92**
 - 4.5.4 Requirements Specification Standards 97**
 - 4.5.5 Conducting Feasibility Study 97**
 - 4.5.6 Reviewing Requirements Specification 99**
 - 4.6 Applying Agile Principles 100**
 - 4.7 Requirements Management and Tools 101**

- Summary 102
- Further Reading 103
- Chapter Review Questions 103
- Exercises 103

Chapter 5

Domain Modeling 105

- 5.1 What Is Domain Modeling? 105**
- 5.2 Why Domain Modeling? 106**
- 5.3 Object-Orientation and Class Diagram 107**
 - 5.3.1 Extensional and Intentional Definitions 107**
 - 5.3.2 Class and Object 108**
 - 5.3.3 Object and Attribute 110**
 - 5.3.4 Association 110**
 - 5.3.5 Multiplicity and Role 111**
 - 5.3.6 Aggregation 113**
 - 5.3.7 Inheritance 114**
 - 5.3.8 Inheritance and Polymorphism 114**
 - 5.3.9 Association Class 115**
- 5.4 Steps for Domain Modeling 117**
 - 5.4.1 Collecting Application Domain Information 118**
 - 5.4.2 Brainstorming 119**
 - 5.4.3 Classifying Brainstorming Results 120**
 - 5.4.4 Visualizing the Domain Model 124**
 - 5.4.5 Domain Model Review Checklist 129**
- 5.5 Putting It Together 130**
- 5.6 Guidelines for Domain Modeling 133**
- 5.7 Applying Agile Principles 134**
- 5.8 Tool Support for Domain Modeling 135**
 - Summary 136
 - Further Reading 136
 - Chapter Review Questions 138
 - Exercises 138

Chapter 6

Architectural Design 139

- 6.1 What Is Architectural Design? 140**
- 6.2 The Importance of Architectural Design 140**

- 6.3 Architectural Design Process 141**
 - 6.3.1 Determine Architectural Design Objectives 142**
 - 6.3.2 Determine System Type 143**
 - 6.3.3 Applying Architectural Styles 147**
 - 6.3.4 Perform Custom Architectural Design 157**
 - 6.3.5 Specify Subsystem Functions and Interfaces 157**
 - 6.3.6 Review the Architectural Design 158**
- 6.4 Architectural Style and Package Diagram 158**
- 6.5 Applying Software Design Principles 160**
 - 6.5.1 What Are Software Design Principles? 161**
 - 6.5.2 Design for Change 161**
 - 6.5.3 Separation of Concerns 162**
 - 6.5.4 Information Hiding 163**
 - 6.5.5 High Cohesion 164**
 - 6.5.6 Low Coupling 165**
 - 6.5.7 Keep It Simple and Stupid 166**
- 6.6 Guidelines for Architectural Design 166**
- 6.7 Architectural Design and Design Patterns 167**
- 6.8 Applying Agile Principles 167**
 - Summary 168
 - Further Reading 168
 - Chapter Review Questions 169
 - Exercises 169

 Part **III**

Modeling and Design of Interactive Systems 171

Chapter **7**

- Deriving Use Cases from Requirements 172**
 - 7.1 What Is An Actor? 173**
 - 7.2 What Is a Use Case? 173**
 - 7.3 Business Process, Operation, and Action 174**
 - 7.4 Steps for Deriving Use Cases from Requirements 176**
 - 7.4.1 Identifying Use Cases 177**
 - 7.4.2 Specifying Use Case Scopes 184**
 - 7.4.3 Visualizing Use Case Contexts 186**
 - 7.4.4 Reviewing Use Case Specifications 190**
 - 7.4.5 Allocating the Use Cases to Iterations 191**

- 7.5 Guidelines for Use Case Derivation 192**
- 7.6 Applying Agile Principles 195**
- 7.7 Tool Support for Use Case Modeling 196**
- Summary 198
- Further Reading 198
- Chapter Review Questions 199
- Exercises 199

Chapter **8**

Actor-System Interaction Modeling 200

- 8.1 What Is Actor-System Interaction Modeling? 201**
- 8.2 Importance of Actor-System Interaction Modeling 202**
- 8.3 Steps for Actor-System Interaction Modeling 202**
 - 8.3.1 Initializing a Two-Column Table 202**
 - 8.3.2 Specifying Actor-System Interaction Steps 203**
 - 8.3.3 Reviewing Actor-System Interaction Specifications 204**
- 8.4 Specifying Alternative Flows 204**
- 8.5 Using User Interface Prototypes 204**
- 8.6 Do Not Show Exception Handling 208**
- 8.7 Use Case Precondition and Postcondition 209**
- 8.8 Including Other Use Cases 210**
- 8.9 Continuing with Other Use Cases 210**
- 8.10 Commonly Seen Problems 211**
- 8.11 Applying Agile Principles 213**
 - Summary 214
 - Further Reading 214
 - Chapter Review Questions 215
 - Exercises 215

Chapter **9**

Object Interaction Modeling 216

- 9.1 What Is Object Interaction Modeling? 216**
- 9.2 UML Sequence Diagram 218**
 - 9.2.1 Notions and Notations 218**
 - 9.2.2 Representing Instances of a Class 218**
 - 9.2.3 Sequence Diagrams Illustrated 220**

- 9.2.4 Sequence Diagram for Analysis and Design 222
- 9.2.5 Using the Notations Correctly 224
- 9.3 Steps for Object Interaction Modeling 225
 - 9.3.1 Collecting Information About Business Processes 226
 - 9.3.2 Identifying Nontrivial Steps 227
 - 9.3.3 Writing Scenarios for Nontrivial Steps 228
 - 9.3.4 Constructing Scenario Tables 230
 - 9.3.5 Scenarios: How to Write Them 232
 - 9.3.6 Deriving Sequence Diagrams from Scenario Tables 236
 - 9.3.7 Object Interaction Modeling Review Checklist 245
- 9.4 Applying Agile Principles 246
- 9.5 Tool Support for Object Interaction Modeling 248
 - Summary 249
 - Further Reading 249
 - Chapter Review Questions 249
 - Exercises 249

Chapter 10

Applying Responsibility-Assignment Patterns 251

- 10.1 What Are Design Patterns? 252
- 10.2 Why Design Patterns? 253
- 10.3 Situation-Specific and Responsibility-Assignment Patterns 253
- 10.4 Pattern Specification 254
- 10.5 The Controller Pattern 254
 - 10.5.1 A Motivating Example 255
 - 10.5.2 What Is a Controller? 258
 - 10.5.3 Applying the Controller Pattern 258
 - 10.5.4 Types of Controller 261
 - 10.5.5 Keeping Track of Use Case State 261
 - 10.5.6 Bloated Controller 263
 - 10.5.7 Comparing Different Designs 264
 - 10.5.8 When Does One Apply the Controller Pattern? 265
 - 10.5.9 Guidelines for Using Controller 265
- 10.6 The Expert Pattern 267
 - 10.6.1 The Information Expert 267

- 10.6.2 Applying the Expert Pattern 267
- 10.6.3 Expert Pattern Involving More Than One Object 269
- 10.6.4 When Does One Apply the Expert Pattern? 269
- 10.6.5 Guidelines for Using Expert 270
- 10.7 The Creator Pattern 270
 - 10.7.1 What Is a Creator? 270
 - 10.7.2 Applying the Creator Pattern 271
 - 10.7.3 Benefits of the Creator Pattern 272
 - 10.7.4 When Does One Apply the Creator Pattern? 273
- Summary 273
- Further Reading 274
- Chapter Review Questions 274
- Exercises 275

Chapter 11

Deriving a Design Class Diagram 276

- 11.1 What Is a Design Class Diagram? 278
- 11.2 Usefulness of a Design Class Diagram 278
- 11.3 Steps for Deriving a Design Class Diagram 279
 - 11.3.1 Identifying Classes 279
 - 11.3.2 Identifying Methods 281
 - 11.3.3 Identifying Attributes 281
 - 11.3.4 Relationships between Classes 285
 - 11.3.5 Identifying Relationships 285
 - 11.3.6 Design Class Diagram Review Checklist 288
- 11.4 Organize Classes with Package Diagram 288
- 11.5 Applying Agile Principles 291
- 11.6 Tool Support for Design Class Diagram 292
 - Summary 292
 - Further Reading 292
 - Chapter Review Questions 292
 - Exercises 292

Chapter 12

User Interface Design 293

- 12.1 What Is User Interface Design? 294
- 12.2 Why Is User Interface Design Important? 295

- 12.3 Graphical User Interface Widgets 296**
 - 12.3.1** Container Widgets 297
 - 12.3.2** Input, Output, and Information Presentation Widgets 298
 - 12.3.3** Guidelines for Using GUI Widgets 298
- 12.4 User Interface Design Process 300**
 - 12.4.1** Case Study: User Interface Design for a Diagram Editor 301
 - 12.4.2** Identifying Major System Displays 302
 - 12.4.3** Producing a Draft Layout Design 304
 - 12.4.4** Specifying Interaction Behavior 306
 - 12.4.5** Constructing a Prototype 307
 - 12.4.6** Evaluating the User Interface Design with Users 308
 - 12.4.7** User Interface Design Review Checklist 310
- 12.5 Designing User Support Capabilities 310**
- 12.6 Guidelines for User Interface Design 311**
- 12.7 Applying Agile Principles 313**
- 12.8 Tool Support for User Interface Design 314**
 - Summary 315
 - Further Reading 315
 - Chapter Review Questions 315
 - Exercises 315

- 13.4.3** Constructing State Transition Tables 327
- 13.4.4** Usefulness of the State Transition Table 329
- 13.4.5** Converting State Transition Table to Analysis State Diagram 330
- 13.4.6** Converting Analysis State Diagram to Design State Diagram 333
- 13.4.7** State Modeling Review Checklists 334
- 13.5 The State Pattern 334**
 - 13.5.1** Conventional Approaches 334
 - 13.5.2** What Is State Pattern? 335
 - 13.5.3** Applying State Pattern 337
- 13.6 Real-Time Systems Modeling and Design 339**
 - 13.6.1** The Transformational Schema 339
 - 13.6.2** Timed State Machine 342
 - 13.6.3** Interrupt Handling 343
- 13.7 Applying Agile Principles 344**
- 13.8 Tool Support for Object State Modeling 345**
 - Summary 345
 - Further Reading 346
 - Chapter Review Questions 346
 - Exercises 346

Part **IV**

Modeling and Design of Other Types of Systems 317

Chapter 13

Object State Modeling for Event-Driven Systems 318

- 13.1** What Is Object State Modeling? 319
- 13.2** Why Object State Modeling? 319
- 13.3** Basic Definitions 320
- 13.4** Steps for Object State Modeling 321
 - 13.4.1** Collecting and Classifying State Behavior Information 322
 - 13.4.2** Constructing a Domain Model to Show the Context 325

Chapter 14

Activity Modeling for Transformational Systems 349

- 14.1** What Is Activity Modeling? 350
- 14.2** Why Activity Modeling? 351
- 14.3** Activity Modeling: Technical Background 351
 - 14.3.1** Flowchart 352
 - 14.3.2** Petri Net 352
 - 14.3.3** Data Flow Diagram 353
- 14.4** UML Activity Diagram 355
- 14.5** Steps for Activity Modeling 356
 - 14.5.1** Identifying Activities and Workflows 357
 - 14.5.2** Producing a Preliminary Activity Diagram 360
 - 14.5.3** Introducing Branching, Forking, and Joining 362

- 14.5.4 Refining Complex Activities 362
- 14.5.5 Activity Modeling Review Checklist 363
- 14.6 Relationships to Other Diagrams 363
- 14.7 Applying Agile Principles 364
- 14.8 Tool Support for Activity Modeling 365
 - Summary 365
 - Further Reading 365
 - Chapter Review Questions 366
 - Exercises 366

Chapter 15

Modeling and Design of Rule-Based Systems 367

- 15.1 What Is a Decision Table? 368
- 15.2 Usefulness of Decision Table 369
- 15.3 Systematic Decision Table Construction 370
- 15.4 Progressive Decision Table Construction 371
- 15.5 Checking for Desired Properties 373
- 15.6 Decision Table Consolidation 374
- 15.7 Generating Code from a Decision Table 375
- 15.8 Applying the Interpreter Pattern 375
 - 15.8.1 Defining a Business Rule Grammar 376
 - 15.8.2 Representing Rules in a Class Diagram 376
 - 15.8.3 Constructing a Parser and a Variable Look Up Context 377
 - 15.8.4 Interpreting Business Rules 378
 - 15.8.5 Updating Rules Dynamically 378
 - 15.8.6 Merits of the Interpretation Approach 379
- 15.9 Using a Decision Table in Test-Driven Development 379
- 15.10 Decision Trees 380
- 15.11 Applying Agile Principles 380
 - Summary 381
 - Further Reading 382
 - Chapter Review Questions 382
 - Exercises 382

Part V

Applying Situation-Specific Patterns 385

Chapter 16

Applying Patterns to Design a State Diagram Editor 386

- 16.1 Process for Applying Patterns 387
- 16.2 Case Study: State Diagram Editor 390
- 16.3 Working with Complex Structures 391
 - 16.3.1 Representing Recursive Whole-Part Structures 391
 - 16.3.2 Providing Layout Choices with Strategy 395
 - 16.3.3 Accessing Complex Structures with Iterator 395
 - 16.3.4 Analyzing Complex Structures with Visitor 398
 - 16.3.5 Storing and Restoring Object State with Memento 402
- 16.4 Creating and Constructing Complex Objects 404
 - 16.4.1 Creating Families of Products 404
 - 16.4.2 Building Large Complex Objects 407
 - 16.4.3 Reusing Objects with Flyweight 410
- 16.5 Designing Graphical User Interface and Display 411
 - 16.5.1 Keeping Track of Editing States 411
 - 16.5.2 Responding to Editing Events 412
 - 16.5.3 Converting One Interface to Another 414
 - 16.5.4 Providing Context-Dependent Help 418
 - 16.5.5 Enhancing Display Capability with a Decorator 420
- 16.6 Applying Agile Principles 423
 - Summary 424
 - Further Reading 424
 - Chapter Review Questions 425
 - Exercises 425

Chapter 17

Applying Patterns to Design a Persistence Framework 426

- 17.1 Problems with Direct Database Access 427
- 17.2 Hiding Persistence Storage with Bridge 428
- 17.3 Encapsulating Database Requests as Commands 431
- 17.4 Hiding Network Access with Remote Proxy 435
- 17.5 Sharing Common Code with Template Method 439
- 17.6 Retrieving Different Objects with Factory Method 442
- 17.7 Reducing Number of Classes with Prototype 444
- 17.8 Applying Agile Principles 447
 - Summary 447
 - Further Reading 448
 - Chapter Review Questions 448
 - Exercises 448

Part VI

Implementation and Quality Assurance 449

Chapter 18

Implementation Considerations 450

- 18.1 Coding Standards 450
 - 18.1.1 What Are Coding Standards? 451
 - 18.1.2 Why Coding Standards? 455
 - 18.1.3 Code Review Checklist 455
 - 18.1.4 Guidelines for Practicing Coding Standards 456
- 18.2 Organizing the Implementation Artifacts 457
- 18.3 Generating Code from Design 459
 - 18.3.1 Implementing Classes and Interfaces 459
 - 18.3.2 From Sequence Diagram to Method Code Skeleton 460

- 18.3.3 Implementing Association Relationships 460

- 18.4 Assigning Implementation Work to Team Members 461
- 18.5 Pair Programming 462
- 18.6 Test-Driven Development 463
 - 18.6.1 Test-Driven Development Workflow 463
 - 18.6.2 Merits of Test-Driven Development 465
 - 18.6.3 Potential Problems 466
- 18.7 Applying Agile Principles 466
- 18.8 Tool Support for Implementation 467
 - Summary 467
 - Further Reading 467
 - Chapter Review Questions 468
 - Exercises 468

Chapter 19

Software Quality Assurance 469

- 19.1 Benefits of Software Quality Assurance 469
- 19.2 Software Quality Attributes 470
- 19.3 Quality Measurements and Metrics 472
 - 19.3.1 Usefulness of Quality Measurements and Metrics 473
 - 19.3.2 Conventional Quality Metrics 474
 - 19.3.3 Reusing Conventional Metrics for Object-Oriented Software 480
 - 19.3.4 Object-Oriented Quality Metrics 480
- 19.4 Software Verification and Validation Techniques 483
 - 19.4.1 Inspection 484
 - 19.4.2 Walkthrough 485
 - 19.4.3 Peer Review 486
- 19.5 Verification and Validation in the Life Cycle 487
- 19.6 Software Quality Assurance Functions 490
 - 19.6.1 Definition of Processes and Standards 490
 - 19.6.2 Quality Management 494
 - 19.6.3 Process Improvement 495

- 19.7 Applying Agile Principles** 497
- 19.8 Tool Support for SQA** 498
 - Summary 498
 - Further Reading 499
 - Chapter Review Questions 499
 - Exercises 499

Chapter **20**

Software Testing 501

- 20.1 What Is Software Testing?** 502
- 20.2 Why Software Testing?** 503
- 20.3 Conventional Black-Box Testing** 504
 - 20.3.1 Functional Testing: An Example** 504
 - 20.3.2 Equivalence Partitioning** 505
 - 20.3.3 Boundary Value Analysis** 507
 - 20.3.4 Cause-Effect Analysis** 509
- 20.4 Conventional White-Box Testing** 510
 - 20.4.1 Basis Path Testing** 510
 - 20.4.2 Cyclomatic Complexity** 511
 - 20.4.3 Flow Graph Test Coverage Criteria** 512
 - 20.4.4 Testing Loops** 512
 - 20.4.5 Data Flow Testing** 514
 - 20.4.6 Coverage Criteria for Data Flow Testing** 515
 - 20.4.7 Interprocedural Data Flow Testing** 515
- 20.5 Test Coverage** 516
- 20.6 A Generic Software Testing Process** 517
- 20.7 Object-Oriented Software Testing** 518
 - 20.7.1 Use Case-Based Testing** 518
 - 20.7.2 Object State Testing with ClassBench** 520
 - 20.7.3 Testing Class Hierarchy** 523
 - 20.7.4 Testing Exception-Handling Capabilities** 524
- 20.8 Testing Web Applications** 525
 - 20.8.1 Object-Oriented Model for Web Application Testing** 525
 - 20.8.2 Static Analysis Using the Object-Oriented Model** 526
 - 20.8.3 Test Case Generation Using the Object-Oriented Model** 527
- 20.8.4 Web Application Testing with HttpUnit** 527
- 20.9 Testing for Nonfunctional Requirements** 527
 - 20.9.1 Performance and Stress Testings** 527
 - 20.9.2 Testing for Security** 528
 - 20.9.3 Testing User Interface** 529
- 20.10 Software Testing in the Life Cycle** 529
- 20.11 Regression Testing** 532
- 20.12 When to Stop Testing?** 533
- 20.13 Applying Agile Principles** 534
- 20.14 Tool Support for Testing** 534
 - Summary 535
 - Further Reading 535
 - Chapter Review Questions 535
 - Exercises 535

Part **VII**

Maintenance and Configuration Management 537

Chapter **21**

Software Maintenance 538

- 21.1 What Is Software Maintenance?** 539
- 21.2 Factors That Mandate Change** 539
- 21.3 Lehman's Laws of System Evolution** 540
- 21.4 Types of Software Maintenance** 541
- 21.5 Software Maintenance Process and Activities** 542
 - 21.5.1 Maintenance Process Models** 542
 - 21.5.2 Program Understanding** 543
 - 21.5.3 Change Identification and Analysis** 544
 - 21.5.4 Configuration Change Control** 547
 - 21.5.5 Change Implementation, Testing, and Delivery** 547
- 21.6 Reverse-Engineering** 547
 - 21.6.1 Reverse-Engineering Workflow** 548
 - 21.6.2 Usefulness of Reverse-Engineering** 548
 - 21.6.3 Reverse-Engineering: A Case Study** 549

- 21.7 Software Reengineering 549**
 - 21.7.1** Objectives of Reengineering 550
 - 21.7.2** Software Reengineering Process 551
 - 21.7.3** Software Reengineering: A Case Study 551
- 21.8 Patterns for Software Maintenance 553**
 - 21.8.1** Simplifying Client Interface with Facade 553
 - 21.8.2** Simplifying Component Interaction with Mediator 553
 - 21.8.3** Other Patterns for Software Maintenance 555
- 21.9 Applying Agile Principles 555**
- 21.10 Tool Support for Software Maintenance 557**
 - Summary 559
 - Further Reading 560
 - Chapter Review Questions 560
 - Exercises 560

Chapter **22**

Software Configuration Management 562

- 22.1** The Baselines of a Software Life Cycle 563
- 22.2** What Is Software Configuration Management? 564
- 22.3** Why Software Configuration Management? 565
- 22.4** Software Configuration Management Functions 565
 - 22.4.1** Software Configuration Identification 566
 - 22.4.2** Software Configuration Change Control 568
 - 22.4.3** Software Configuration Auditing 569
 - 22.4.4** Software Configuration Status Accounting 570
- 22.5** Configuration Management in an Agile Project 570
- 22.6** Software Configuration Management Tools 570
 - Summary 572
 - Further Reading 573
 - Chapter Review Questions 573
 - Exercises 573

Part **VIII**

Project Management and Software Security 575

Chapter **23**

Software Project Management 576

- 23.1 Project Organization 577**
 - 23.1.1** Project Format 578
 - 23.1.2** Team Structure 579
- 23.2 Effort Estimation Methods 580**
 - 23.2.1** The Function Point Method 581
 - 23.2.2** The COCOMO II Model 583
 - 23.2.3** The Delphi Estimation Method 588
 - 23.2.4** Agile Estimation 589
- 23.3 Project Planning and Scheduling 591**
 - 23.3.1** PERT Chart 591
 - 23.3.2** Gantt Chart and Staff Allocation 593
 - 23.3.3** Agile Planning 594
- 23.4 Risk Management 595**
 - 23.4.1** Risk Identification 596
 - 23.4.2** Risk Analysis and Prioritizing 597
 - 23.4.3** Risk Management Planning 599
 - 23.4.4** Risk Resolution and Monitoring 599
- 23.5 Process Improvement 599**
- 23.6 Applying Agile Principles 601**
- 23.7 Tool Support for Project Management 602**
 - Summary 603
 - Further Reading 603
 - Chapter Review Questions 604
 - Exercises 604

Chapter **24**

Software Security 606

- 24.1** What Is Software Security? 607
- 24.2** Security Requirements 608
- 24.3** Secure Software Design Principles 609
- 24.4** Secure Software Design Patterns 610
- 24.5** Seven Best Practices of Software Security 612
- 24.6** Risk Analysis with an Attack Tree 613

24.7	Software Security in the Life Cycle	614
24.7.1	Security in the Planning Phase	615
24.7.2	Security in the Iterative Phase	623
24.8	Applying Agile Principles	627
24.9	Tool Support for Software Security	628
	Summary	629
	Further Reading	629
	Chapter Review Questions	630
	Exercises	630

Appendices

A Personal Software Process: Estimation, Planning, and Quality Assurance 631

A.1	Effort Estimation in PSP	631
A.2	Software Quality Assurance in PSP	632
A.3	Design and Quality	633

B Java Technologies 634

B.1	Getting Started with Database Connectivity	634
B.1.1	What Is Database Connectivity?	634
B.1.2	Setting Up Data Sources	634
B.1.3	Accessing Databases from a Program	635
B.2	Getting Started with Swing	636
B.2.1	Creating Main Window with JFrame	637
B.2.2	Using Layout Managers to Arrange Components	638
B.2.3	Processing Button Events with ActionListener	640
B.2.4	Implementing Drawing Capabilities	640
B.3	Getting Started with Java Server Pages	642
B.3.1	What Are Java Server Pages?	642
B.3.2	JSP Workflow	642
B.3.3	Installing a Web Server with a JSP Container	643
B.3.4	Using Java Server Pages	643

C Software Tools 647

C.1	NetBeans	647
C.2	Using JUnit	648
C.3	Running JUnit in NetBeans	652
C.4	The Emma Coverage Tool	652
C.5	The Cobertura Coverage Tool	653
C.6	Web Application Testing with HttpUnit	655
C.6.1	Configure an IDE to Use HttpUnit	655
C.6.2	Implementing Test Cases in HttpUnit	655
C.7	Using CVS and Subversion in NetBeans	656
C.7.1	Creating a CVS Remote Repository	656
C.7.2	Setting Up Subversion in NetBeans	658
C.7.3	Checking Out Files from a Repository	659
C.7.4	Editing Sources and Viewing Changes	661
C.7.5	Viewing File Status	662
C.7.6	Comparing File Revisions	662
C.7.7	Merging Changes from Repository	662
C.7.8	Resolving Conflicts	663
C.7.9	Updating Local Copies	663
C.7.10	Committing Local Files to a Repository	663
C.7.11	Importing Files into a Repository	664

D Project Descriptions 665

D.1	Car Rental System	665
D.2	National Trade Show Service System	666
D.3	Study Abroad Management System	667
D.4	UML Class Diagram Editor	669
D.5	Radio Communication Simulator	670
D.6	Object State Testing Environment	672

References 675

Index 682