Contents

PREFACE

I INTRODUCTION AND SYSTEM ENGINEERING

1 Introduction

1.1 What Is Software Engineering? . . . .

1.2 Why Software Engineering? . . . .

1.3 Software Life Cycle Activities . . . .

1.3.1 Software Development Process . . . .

1.3.2 Software Quality Assurance . . . .

1.3.3 Software Project Management . . . .

1.4 Object-Oriented Software Engineering . . . .

1.4.1 Object-Oriented Modeling and Design Languages . . . .

1.4.2 Object-Oriented Development Processes . . . .

1.4.3 Object-Oriented Development Methodologies . . . .

1.5 Software Engineering and Computer Science . . . .

1.6 Summary . . . .

1.7 Further Reading . . . .

1.8 Chapter Review Questions . . . .
1.9 Exercises . . . .

2 Software Process and Methodology

2.1 Challenges of System Development . . . .
2.2 Software Process . . . .
2.3 Merits and Problems of Waterfall Process . . . .
2.4 Software Development Is a Wicked Problem . . . .
2.5 Software Process Models . . . .
2.5.1 Prototyping Process . . . .
2.5.2 Evolutionary Process . . . .
2.5.3 Spiral Process . . . .
2.5.4 The Unified Process . . . .
2.5.5 Personal Software Process . . . .
2.5.6 Team Software Process . . . .
2.5.7 Agile Processes . . . .
2.6 Software Development Methodology . . . .
2.6.1 Difference between Process and Methodology . . . .
2.6.2 Benefits of a Methodology . . . .
2.6.3 Structured Methodologies . . . .
2.6.4 Classical OO Methodologies . . . .
2.7 Agile Methods . . . .
2.7.1 Dynamic Systems Development Method . . . .
2.7.2 Scrum . . . .
2.7.3 Feature Driven Development . . . .

2.7.4 Extreme Programming . . . .

2.7.5 Agile or Plan-Driven . . . .

2.8 Overview of Process and Methodology of the Book . . . .

2.9 Summary . . . .

2.10 Further Reading . . . .

2.11 Chapter Review Questions . . . .

2.12 Exercises . . . .

3 System Engineering

3.1 What Is a System? . . . .

3.2 What Is System Engineering? . . . .

3.3 System Requirements Definition . . . .

3.3.1 Identifying Business Needs . . . .

3.3.2 Defining System Requirements . . . .

3.4 System Architectural Design . . . .

3.4.1 System Decomposition . . . .

3.4.2 Requirements Allocation . . . .

3.4.3 Architectural Design Diagrams . . . .

3.4.4 Specification of Subsystem Functions and Interfaces . . . .

3.5 Subsystems Development . . . .

3.5.1 Object-Oriented Context Diagram . . . .

3.5.2 Usefulness of an Object-Oriented Context Diagram . . . .

3.5.3 Collaboration of Engineering Teams . . . .

3.6 System Integration, Testing and Deployment . . . .

3.7 System Configuration Management . . . .

3.8 Summary
3.9 Further Reading
3.10 Chapter Review Questions
3.11 Exercises

II ANALYSIS AND ARCHITECTURAL DESIGN

4 Software Requirements Elicitation
4.1 What Is Requirements Elicitation?
4.2 Importance of Requirements Elicitation
4.3 Challenges of Requirements Elicitation
4.4 Types of Requirement
4.5 Steps for Requirements Elicitation
4.5.1 Collecting Information
4.5.2 Constructing Analysis Models
4.5.3 Deriving Requirements and Constraints
4.5.4 Requirements Specification Standards
4.5.5 Conducting Feasibility Study
4.5.6 Reviewing Requirements Specification
4.6 Applying Agile Principles
4.7 Requirements Management and Tools
4.8 Summary
4.9 Further Reading
4.10 Chapter Review Questions
4.11 Exercises

5 Domain Modeling

5.1 What Is Domain Modeling? . . . .

5.2 Why Domain Modeling? . . . .

5.3 Object-Orientation and Class Diagram . . . .

5.3.1 Extensional and Intentional Definitions . . . .

5.3.2 Class and Object . . . .

5.3.3 Object and Attribute . . . .

5.3.4 Association . . . .

5.3.5 Multiplicity and Role . . . .

5.3.6 Aggregation . . . .

5.3.7 Inheritance . . . .

5.3.8 Inheritance and Polymorphism . . . .

5.3.9 Association Class . . . .

5.4 Steps for Domain Modeling . . . .

5.4.1 Collecting Application Domain Information . . . .

5.4.2 Brainstorming . . . .

5.4.3 Classifying Brainstorming Results . . . .

5.4.4 Visualizing the Domain Model . . . .

5.4.5 Domain Model Review Checklist . . . .

5.5 Putting It Together . . . .

5.6 Guidelines for Domain Modeling . . . .

5.7 Applying Agile Principles . . . .

5.8 Tool Support for Domain Modeling . . . .

5.9 Summary . . . .

5.10 Further Reading . . . .
5.11 Chapter Review Questions . . . .
5.12 Exercises . . . .

6 Architectural Design

6.1 What Is Architectural Design? . . . .
6.2 The Importance of Architectural Design . . . .
6.3 Architectural Design Process . . . .
6.3.1 Determine Architectural Design Objectives . . . .
6.3.2 Determine System Type . . . .
6.3.3 Applying Architectural Styles . . . .
6.3.4 Perform Custom Architectural Design . . . .
6.3.5 Specify Subsystem Functions and Interfaces . . . .
6.3.6 Review the Architectural Design . . . .
6.4 Architectural Style and Package Diagram . . . .
6.5 Applying Software Design Principles . . . .
6.5.1 What Are Software Design Principles? . . . .
6.5.2 Design for Change . . . .
6.5.3 Separation of Concerns . . . .
6.5.4 Information Hiding . . . .
6.5.5 High Cohesion . . . .
6.5.6 Low Coupling . . . .
6.5.7 Keep It Simple and Stupid . . . .
6.6 Guidelines for Architectural Design . . . .
6.7 Architectural Design and Design Patterns . . . .

6.8 Applying Agile Principles . . . .

6.9 Summary . . . .

6.10 Further Reading . . . .

6.11 Chapter Review Questions . . . .

6.12 Exercises . . . .

III MODELING AND DESIGN OF INTERACTIVE SYSTEMS

7 Deriving Use Cases from Requirements

7.1 What Is an Actor? . . . .

7.2 What Is a Use Case? . . . .

7.3 Business Process, Operation, and Action . . . .

7.4 Steps for Deriving Use Cases from Requirements . . . .

7.4.1 Identifying Use Cases . . . .

7.4.2 Specifying Use Case Scopes . . . .

7.4.3 Visualizing Use Case Contexts . . . .

7.4.4 Reviewing Use Case Specifications . . . .

7.4.5 Allocating the Use Cases to Iterations . . . .

7.5 Guidelines for Use Case Derivation . . . .

7.6 Applying Agile Principles . . . .

7.7 Tool Support for Use Case Modeling . . . .

7.8 Summary . . . .

7.9 Further Reading . . . .

7.10 Chapter Review Questions . . . .

7.11 Exercises . . . .

8 Actor-System Interaction Modeling

8.1 What Is Actor-System Interaction Modeling? . . . .

8.2 Importance of Actor-System Interaction Modeling . . . .

8.3 Steps for Actor-System Interaction Modeling . . . .

8.3.1 Initializing a Two Column Table . . . .

8.3.2 Specifying Actor-System Interaction Steps . . . .

8.3.3 Reviewing Actor-System Interaction Specifications . . . .

8.4 Specifying Alternative Flows . . . .

8.5 Using User Interface Prototypes . . . .

8.6 Do Not Show Exception Handling . . . .

8.7 Use Case Precondition and Postcondition . . . .

8.8 Including Other Use Cases . . . .

8.9 Continuing with Other Use Cases . . . .

8.10 Commonly Seen Problems . . . .

8.11 Applying Agile Principles . . . .

8.12 Summary . . . .

8.13 Further Reading . . . .

8.14 Chapter Review Questions . . . .

8.15 Exercises . . . .

9 Object Interaction Modeling

9.1 What Is Object Interaction Modeling? . . . .

9.2 UML Sequence Diagram . . . .

9.2.1 Notions and Notations . . . .

9.2.2 Representing Instances of a Class . . . .

9.2.3 Sequence Diagrams Illustrated . . . .

9.2.4 Sequence Diagram for Analysis and Design . . . .

9.2.5 Using the Notations Correctly . . . .

9.3 Steps for Object Interaction Modeling . . . .

9.3.1 Collecting Information about Business Processes . . . .

9.3.2 Identifying Nontrivial Steps . . . .

9.3.3 Writing Scenarios for Nontrivial Steps . . . .

9.3.4 Constructing Scenario Tables . . . .

9.3.5 Scenarios: How to Write Them? . . . .

9.3.6 Deriving Sequence Diagrams from Scenario Tables . . . .

9.3.7 Object Interaction Modeling Review Checklist . . . .

9.4 Applying Agile Principles . . . .

9.5 Tool Support for Object Interaction Modeling . . . .

9.6 Summary . . . .

9.7 Further Reading . . . .

9.8 Chapter Review Questions . . . .

9.9 Exercises . . . .

10 Applying Responsibility-Assignment Patterns

10.1 What Are Design Patterns? . . . .

10.2 Why Design Patterns? . . . .

10.3 Situation-Specific and Responsibility-Assignment Patterns . . . .

10.4 Pattern Specification . . . .

10.5 The Controller Pattern . . . .

10.5.1 A Motivating Example . . . .

10.5.2 What Is a Controller? . . . .

10.5.3 Applying the Controller Pattern . . . .

10.5.4 Types of Controller . . . .

10.5.5 Keeping Track of Use Case State . . . .

10.5.6 Overloaded Controller . . . .

10.5.7 Comparing Different Designs . . . .

10.5.8 When to Apply the Controller Pattern? . . . .

10.5.9 Guidelines for Using Controller . . . .

10.6 The Expert Pattern . . . .

10.6.1 The Information Expert . . . .

10.6.2 Applying the Expert Pattern . . . .

10.6.3 Expert Pattern Involving More Than One Object . . . .

10.6.4 When to Apply the Expert Pattern? . . . .

10.6.5 Guidelines for Using Expert . . . .

10.7 The Creator Pattern . . . .

10.7.1 What Is a Creator? . . . .

10.7.2 Applying the Creator Pattern . . . .

10.7.3 Benefits of the Creator Pattern . . . .

10.7.4 When to Apply the Creator Pattern? . . . .

10.8 Summary . . . .

10.9 Further Reading . . . .

10.10 Chapter Review Questions . . . .

10.11 Exercises . . . .

11 Deriving Design Class Diagram

11.1 What Is a Design Class Diagram? . . . .

11.2 Usefulness of a Design Class Diagram . . . .

11.3 Steps for Deriving a Design Class Diagram . . . .

11.3.1 Identifying Classes . . . .

11.3.2 Identifying Methods . . . .

11.3.3 Identifying Attributes . . . .

11.3.4 Relationships between Classes . . . .

11.3.5 Identifying Relationships . . . .

11.3.6 Design Class Diagram Review Checklist . . . .

11.4 Organize Classes with Package Diagram . . . .

11.5 Applying Agile Principles . . . .

11.6 Tool Support for Design Class Diagram . . . .

11.7 Summary . . . .

11.8 Further Reading . . . .

11.9 Chapter Review Questions . . . .

11.10 Exercises . . . .

12 User Interface Design

12.1 What Is User Interface Design? . . . .

12.2 Why Is User Interface Design Important? . . . .

12.3 Graphical User Interface Widgets . . . .

12.3.1 Container Widgets . . . .

12.3.2 Input, Output and Information Presentation Widgets . . . .

12.3.3 Guidelines for Using GUI Widgets . . . .

12.4 User Interface Design Process . . . .

12.4.1 Case Study: User Interface Design for a Diagram Editor . . . .

12.4.2 Identifying Major System Displays . . . .

12.4.3 Producing a Draft Layout Design . . . .

12.4.4 Specifying Interaction Behavior . . . .

12.4.5 Constructing a Prototype . . . .

12.4.6 Evaluating the User Interface Design with Users . . . .

12.4.7 User Interface Design Review Checklist . . . .

12.5 Designing User Support Capabilities . . . .

12.6 Guidelines for User Interface Design . . . .

12.7 Applying Agile Principles . . . .

12.8 Tool Support for User Interface Design . . . .

12.9 Summary . . . .

12.10 Further Reading . . . .

12.11 Chapter Review Questions . . . .

12.12 Exercises . . . .

IV MODELING AND DESIGN OF OTHER TYPES OF SYSTEMS

13 Object State Modeling for Event-Driven Systems

13.1 What Is Object State Modeling? . . . .

13.2 Why Object State Modeling? . . . .

13.3 Basic Definitions . . . .

13.4 Steps for Object State Modeling . . . .

13.4.1 Collecting and Classifying State Behavior Information . . . .

13.4.2 Constructing a Domain Model to Show the Context . . . .

13.4.3 Constructing State Transition Tables . . . .

13.4.4 Usefulness of the State Transition Table . . . .

13.4.5 Converting State Transition Table to Analysis State Diagram . . . .

13.4.6 Converting Analysis State Diagram to Design State Diagram . . . .

13.4.7 State Modeling Review Checklists . . . .

13.5 The State Pattern . . . .

13.5.1 Conventional Approaches . . . .

13.5.2 What Is State Pattern? . . . .

13.5.3 Applying State Pattern . . . .

13.6 Real-Time Systems Modeling and Design . . . .

13.6.1 The Transformational Schema . . . .

13.6.2 Timed State Machine . . . .

13.6.3 Interrupt Handling . . . .

13.7 Applying Agile Principles . . . .

13.8 Tool Support for Object State Modeling . . . .

13.9 Summary . . . .

13.10 Further Reading . . . .

13.11 Chapter Review Questions . . . .

13.12 Exercises . . . .

14 Activity Modeling for Transformational Systems

14.1 What Is Activity Modeling? . . . .

14.2 Why Activity Modeling? . . . .

14.3 Activity Modeling: Technical Background . . . .

14.3.1 Flowchart . . . .

14.3.2 Petri Net . . . .

14.3.3 Data Flow Diagram . . . .

14.4 UML Activity Diagram . . . .

14.5 Steps for Activity Modeling . . . .

14.5.1 Identifying Activities and Work Flows . . . .

14.5.2 Producing a Preliminary Activity Diagram . . . .

14.5.3 Introducing Branching, Forking and Joining . . . .

14.5.4 Refining Complex Activities . . . .

14.5.5 Activity Modeling Review Checklist . . . .

14.6 Relationships to Other Diagrams . . . .

14.7 Applying Agile Principles . . . .

14.8 Tool Support for Activity Modeling . . . .

14.9 Summary . . . .

14.10 Further Reading . . . .

14.11 Chapter Review Questions . . . .

14.12 Exercises . . . .

15 Modeling and Design of Rule-Based Systems

15.1 What Is a Decision Table? . . . .

15.2 Usefulness of Decision Table . . . .

15.3 Systematic Decision Table Construction . . . .

15.4 Progressive Decision Table Construction . . . .

15.5 Checking for Desired Properties . . . .

15.6 Decision Table Consolidation . . . .

15.7 Generating Code from a Decision Table . . . .

15.8 Applying the Interpreter Pattern . . . .

15.8.1 Defining a Business Rule Grammar . . . .

15.8.2 Representing Rules in a Class Diagram . . . .

15.8.3 Constructing a Parser and a Variable Look Up Context . . . .

15.8.4 Interpreting Business Rules . . . .

15.8.5 Updating Rules Dynamically . . . .

15.8.6 Merits of the Interpretation Approach . . . .

15.9 Using Decision Table in Test Driven Development . . . .

15.10 Decision Tree . . . .

15.11 Applying Agile Principles . . . .

15.12 Summary . . . .

15.13 Further Reading . . . .

15.14 Chapter Review Questions . . . .

15.15 Exercises . . . .

V APPLYING SITUATION-SPECIFIC PATTERNS

16 Applying Patterns to Design a State Diagram Editor

16.1 Process for Applying Patterns . . . .

16.2 Case Study: State Diagram Editor . . . .

16.3 Working with Complex Structures . . . .

16.3.1 Representing Recursive Whole-Part Structures . . . .

16.3.2 Providing Layout Choices with Strategy . . . .

16.3.3 Accessing Complex Structures with Iterator . . . .

16.3.4 Analyzing Complex Structures with Visitor . . . .
16.3.5 Storing and Restoring Object State with Memento . . . .
16.4 Creating and Constructing Complex Objects . . . .
16.4.1 Creating Families of Products . . . .
16.4.2 Building Large Complex Objects . . . .
16.4.3 Reusing Objects with Flyweight . . . .
16.5 Designing Graphical User Interface and Display . . . .
16.5.1 Keeping Track of Editing States . . . .
16.5.2 Responding to Editing Events . . . .
16.5.3 Converting One Interface to Another . . . .
16.5.4 Providing Context Dependent Help . . . .
16.5.5 Enhancing Display Capability with Decorator . . . .
16.6 Applying Agile Principles . . . .
16.7 Summary . . . .
16.8 Further Reading . . . .
16.9 Chapter Review Questions . . . .
16.10 Exercises . . . .

17 Applying Patterns to Design a Persistence Framework

17.1 Problems with Direct Database Access . . . .
17.2 Hiding Persistence Storage with Bridge . . . .
17.3 Encapsulating Database Requests as Commands . . . .
17.4 Hiding Network Access with Remote Proxy . . . .
17.5 Sharing Common Code with Template Method . . . .
17.6 Retrieving Different Objects with Factory Method . . . .

17.7 Reducing Number of Classes with Prototype . . . .

17.8 Applying Agile Principles . . . .

17.9 Summary . . . .

17.10 Further Reading . . . .

17.11 Chapter Review Questions . . . .

17.12 Exercises . . . .

VI IMPLEMENTATION AND QUALITY ASSURANCE

18 Implementation Considerations

18.1 Coding Standards . . . .

18.1.1 What Are Coding Standards? . . . .

18.1.2 Why Coding Standards? . . . .

18.1.3 Code Review Checklist . . . .

18.1.4 Guidelines for Practicing Coding Standards . . . .

18.2 Organizing the Implementation Artifacts . . . .

18.3 Generating Code from Design . . . .

18.3.1 Implementing Classes and Interfaces . . . .

18.3.2 From Sequence Diagram to Method Code Skeleton . . . .

18.3.3 Implementing Association Relationships . . . .

18.4 Assigning Implementation Work to Team Members . . . .

18.5 Pair Programming . . . .

18.6 Test Driven Development . . . .

18.6.1 Test Driven Development Work Flow . . . .

18.6.2 Merits of Test Driven Development . . . .

18.6.3 Potential Problems . . . .
18.7 Applying Agile Principles . . . .
18.8 Tool Support for Implementation . . . .
18.9 Summary . . . .
18.10 Further Reading . . . .
18.11 Chapter Review Questions . . . .
18.12 Exercises . . . .

19 Software Quality Assurance

19.1 Benefits of Software Quality Assurance . . . .
19.2 Software Quality Attributes . . . .
19.3 Quality Measurements and Metrics . . . .
19.3.1 Usefulness of Quality Measurements and Metrics . . . .
19.3.2 Conventional Quality Metrics . . . .
19.3.3 Reusing Conventional Metrics for Object-Oriented Software . . . .
19.3.4 Object-Oriented Quality Metrics . . . .
19.4 Software Verification and Validation Techniques . . . .
19.4.1 Inspection . . . .
19.4.2 Walkthrough . . . .
19.4.3 Peer Review . . . .
19.5 Verification and Validation in the Life Cycle . . . .
19.6 Software Quality Assurance Functions . . . .
19.6.1 Definition of Processes and Standards . . . .
19.6.2 Quality Management . . . .
19.6.3 Process Improvement . . . .

19.7 Applying Agile Principles . . . .

19.8 Tool Support for SQA . . . .

19.9 Summary . . . .

19.10 Further Reading . . . .

19.11 Chapter Review Questions . . . .

19.12 Exercises . . . .

20 Software Testing

20.1 What Is Software Testing? . . . .

20.2 Why Software Testing? . . . .

20.3 Conventional Black-Box Testing . . . .

20.3.1 Functional Testing: An Example . . . .

20.3.2 Equivalence Partitioning . . . .

20.3.3 Boundary Value Analysis . . . .

20.3.4 Cause-Effect Analysis . . . .

20.4 Conventional White-Box Testing . . . .

20.4.1 Basis Path Testing . . . .

20.4.2 Cyclomatic Complexity . . . .

20.4.3 Flow Graph Test Coverage Criteria . . . .

20.4.4 Testing Loops . . . .

20.4.5 Data Flow Testing . . . .

20.4.6 Coverage Criteria for Data Flow Testing . . . .

20.4.7 Interprocedural Data Flow Testing . . . .

20.5 Test Coverage . . . .

20.6 A Generic Software Testing Process . . . .

20.7 Object-Oriented Software Testing . . . .

20.7.1 Use Case Based Testing . . . .

20.7.2 Object State Testing with ClassBench . . . .

20.7.3 Testing Class Hierarchy . . . .

20.7.4 Testing Exception Handling Capabilities . . . .

20.8 Testing Web Applications . . . .

20.8.1 Object-Oriented Model for Web Application Testing . . . .

20.8.2 Static Analysis Using the Object-Oriented Model . . . .

20.8.3 Test Case Generation Using the Object-Oriented Model . . . .

20.8.4 Web Application Testing with HttpUnit . . . .

20.9 Testing for Nonfunctional Requirements . . . .

20.9.1 Performance and Stress Testings . . . .

20.9.2 Testing for Security . . . .

20.9.3 Testing User Interface . . . .

20.10 Software Testing in the Life Cycle . . . .

20.11 Regression Testing . . . .

20.12 When to Stop Testing? . . . .

20.13 Applying Agile Principles . . . .

20.14 Tool Support for Testing . . . .

20.15 Summary . . . .

20.16 Further Reading . . . .

20.17 Chapter Review Questions . . . .

20.18 Exercises . . . .

VII MAINTENANCE AND CONFIGURATION MANAGEMENT

21 Software Maintenance

21.1 What Is Software Maintenance? . . . .

21.2 Factors That Mandate Change . . . .

21.3 Lehman's Laws of System Evolution . . . .

21.4 Types of Software Maintenance . . . .

21.5 Software Maintenance Process and Activities . . . .

21.5.1 Maintenance Process Models . . . .

21.5.2 Program Understanding . . . .

21.5.3 Change Identification and Analysis . . . .

21.5.4 Configuration Change Control . . . .

21.5.5 Change Implementation, Testing and Delivery . . . .

21.6 Reverse Engineering . . . .

21.6.1 Reverse Engineering Work Flow . . . .

21.6.2 Usefulness of Reverse Engineering . . . .

21.6.3 Reverse Engineering: a Case Study . . . .

21.7 Software Re-Engineering . . . .

21.7.1 Objectives of Re-Engineering . . . .

21.7.2 Software Re-Engineering Process . . . .

21.7.3 Software Re-Engineering: a Case Study . . . .

21.8 Patterns for Software Maintenance . . . .

21.8.1 Simplifying Client Interface with Facade . . . .

21.8.2 Simplifying Component Interaction with Mediator . . . .

21.8.3 Summary of Patterns for Software Maintenance . . . .

21.9 Applying Agile Principles . . . .

21.10 Tool Support for Software Maintenance . . . .

21.11 Summary . . . .

21.12 Further Reading . . . .

21.13 Chapter Review Questions . . . .

21.14 Exercises . . . .

22 Software Configuration Management

22.1 The Baselines of a Software Life Cycle . . . .

22.2 What Is Software Configuration Management? . . . .

22.3 Why Software Configuration Management? . . . .

22.4 Software Configuration Management Functions . . . .

22.4.1 Software Configuration Identification . . . .

22.4.2 Software Configuration Change Control . . . .

22.4.3 Software Configuration Auditing . . . .

22.4.4 Software Configuration Status Accounting . . . .

22.5 Configuration Management in an Agile Project . . . .

22.6 Software Configuration Management Tools . . . .

22.7 Summary . . . .

22.8 Further Reading . . . .

22.9 Chapter Review Questions . . . .

22.10 Exercises . . . .

VIII PROJECT MANAGEMENT AND SOFTWARE SECURITY

23 Software Project Management

23.1 Project Organization . . . .

23.1.1 Project Format . . . .

23.1.2 Team Structure . . . .

23.2 Effort Estimation Methods . . . .

23.2.1 The Function Point Method . . . .

23.2.2 The COCOMO II Model . . . .

23.2.3 The Delphi Estimation Method . . . .

23.2.4 Agile Estimation . . . .

23.3 Project Planning and Scheduling . . . .

23.3.1 PERT Chart . . . .

23.3.2 Gantt Chart and Staff Allocation . . . .

23.3.3 Agile Planning . . . .

23.4 Risk Management . . . .

23.4.1 Risk identification . . . .

23.4.2 Risk Analysis and Prioritizing . . . .

23.4.3 Risk Management Planning . . . .

23.4.4 Risk Resolution and Monitoring . . . .

23.5 Process Improvement . . . .

23.6 Applying Agile Principles . . . .

23.7 Tool Support for Project Management . . . .

23.8 Summary . . . .

23.9 Further Reading . . . .

23.10 Chapter Review Questions . . . .

23.11 Exercises . . . .

24 Software Security

24.1 What Is Software Security? . . . .

24.2 Security Requirements . . . .

24.3 Secure Software Design Principles . . . .

24.4 Secure Software Design Patterns . . . .

24.5 Seven Best Practices of Software Security . . . .

24.6 Risk Analysis with an Attack Tree . . . .

24.7 Software Security in the Life Cycle . . . .

24.7.1 Security in the Planning Phase . . . .

24.7.2 Security in the Incremental Phase . . . .

24.8 Applying Agile Principles . . . .

24.9 Tool Support for Software Security . . . .

24.10 Summary . . . .

24.11 Further Reading . . . .

24.12 Chapter Review Questions . . . .

24.13 Exercises . . . .

A Personal Software Process — Estimation, Planning and Quality Assurance

B Java Technologies

B.1 Getting Started with Database Connectivity . . . .

B.1.1 What Is Database Connectivity? . . . .

B.1.2 Setting Up Data Sources . . . .

B.1.3 Accessing Databases from a Program . . . .

B.2 Getting Started with Swing . . . .

B.2.1 Creating Main Window with JFrame . . . .

B.2.2 Using Layout Managers to Arrange Components . . . .

B.2.3 Processing Button Events with Action Listener . . . .

B.2.4 Implementing Drawing Capabilities . . . .

B.3 Getting Started with Java Server Pages . . . .

B.3.1 What Are Java Server Pages? . . . .

B.3.2 JSP Workflow . . . .

B.3.3 Installing a Web Server with a JSP Container . . . .

B.3.4 Using Java Server Pages . . . .

C Software Tools

C.1 NetBeans . . . .

C.2 Using JUnit . . . .

C.3 Running JUnit in NetBeans . . . .

C.4 The Emma Coverage Tool . . . .

C.5 The Cobertura Coverage Tool . . . .

C.6 Web Application Testing with HttpUnit . . . .

C.7 Using CVS and Subversion in NetBeans . . . .

D Project Descriptions

D.1 Car Rental System . . . .

D.2 National Trade Show Service System . . . .

D.3 Study Abroad Management System (SAMS) . . . .

D.4 UML Class Diagram Editor . . . .

D.5 Radio Communication Simulator . . . .

D.6 Object State Testing Environment . . . .

E References

E.1 References . . . .