Foundations of Software Engineering
Foundations of Software Engineering

Ashfaque Ahmed
SCM Consulting, Bhilai, Chattisgarh, India

Bhanu Prasad
Florida A&M University, Tallahassee, USA
## Contents

**Preface**  
xxvii  

**Acknowledgments**  
xxi  

**Authors**  
xxiii  

### Chapter 1  Introduction to Software Engineering  
1  

1.1 Introduction  
1  

1.2 Components of a Computer  
1  

1.3 Building the Software Products  
2  

1.4 What Is Software Engineering?  
4  

1.5 Why Software Engineering?  
4  

1.5.1 Reduction of Development Costs  
5  

1.5.2 Reduction of Development Time  
5  

1.5.3 Increasing the Quality  
6  

1.6 Challenges in Software Engineering  
6  

1.7 Project Management and Software Engineering  
7  

1.8 Costs Involved in Software Development  
7  

1.9 Methodologies Used for Software Development  
8  

1.10 Some Careers in Software Engineering  
8  

1.11 Software Industry Size  
9  

1.12 Code of Ethics  
9  

1.13 Book Organization  
9  

1.14 Chapter Summary  
11  

Recommended Reading  
11  

### Chapter 2  Software Engineering Methodologies  
13  

2.1 Introduction  
13  

2.2 Why a Methodology?  
14  

2.3 Agile Methodologies  
14  

2.4 Waterfall Model  
15  

2.4.1 Details of the Waterfall Model  
15  

2.4.2 Salient Features of the Waterfall Model  
16
## CONTENTS

2.4.3 Drawbacks of the Waterfall Model 18
2.4.4 When to Use the Waterfall Model 19

2.5 Rational Unified Process 19
2.5.1 Salient Features of RUP 21
2.5.2 Drawbacks of RUP 21
2.5.3 When to Use RUP 21

2.6 Spiral Model 22
2.6.1 Comparison with Other Models 24
2.6.2 What Are Prototypes? 25
2.6.3 Salient Features of the Spiral Model 25
2.6.4 Drawbacks of the Spiral Model 25
2.6.5 When to Use the Spiral Model 26

2.7 Incremental Iteration Model 26

2.8 Extreme Programming 28
2.8.1 XP Activities 29
  2.8.1.1 Coding 29
  2.8.1.2 Testing 29
  2.8.1.3 Listening 29
  2.8.1.4 Designing 29
2.8.2 XP Concepts 29
  2.8.2.1 Customer 30
  2.8.2.2 Pair Programming 30
  2.8.2.3 Planning Game 30
  2.8.2.4 Release Planning 30
  2.8.2.5 Iteration Planning 30
  2.8.2.6 Whole Team 31
  2.8.2.7 Continuous Integration 31
  2.8.2.8 Refactoring 31
  2.8.2.9 Smaller Releases 31
2.8.3 Salient Features of XP 31
2.8.4 Drawbacks of XP 31
2.8.5 When to Use XP 32

2.9 Scrum 32
2.9.1 Scrum Roles 33
2.9.2 Scrum Events 33
  2.9.2.1 Sprint 33
  2.9.2.2 Meetings 34
  2.9.2.3 Backlog Refinement 34
  2.9.2.4 Scrum of Scrums 34
2.9.3 Scrum Artifacts 35
  2.9.3.1 Product Backlog 35
  2.9.3.2 Sprint Backlog 35
2.9.4 Salient Features of Scrum 35
2.9.5 Drawbacks of Scrum 35
2.9.6 Benefits with Scrum 36

2.10 Methodology for Implementing SaaS Products 36
2.10.1 What Is a Customer-Specific Version? 39

2.11 Methodology for Implementing COTS Products 40

2.12 Usage of Software Development Models 41

2.13 Popular Process Standards 42

2.14 Process Standards and Software Engineering Methodologies 43
CONTENTS

2.15 Software Process Activities 43
2.15.1 Feasibility Study Phase 44
2.15.2 Requirement Phase 45
2.15.3 Design Phase 45
2.15.4 Construction Phase 46
2.15.5 Testing Phase 46
2.15.6 Release Phase 47
2.15.7 Software Maintenance 47
2.15.8 Software Configuration Management 47
2.15.9 Project Management Processes 48
2.16 Chapter Summary 48
Recommended Reading 49

CHAPTER 3 FEASIBILITY STUDY

3.1 Introduction 51
3.2 Feasibility Study for Software Projects 52
3.3 Finding the Economic Feasibility of Requirements 53
3.3.1 Build/Buy Decision 54
3.3.2 Budget Instead of Cost Analysis 55
3.4 Finding the Technical Feasibility of Requirements 55
3.4.1 Difficult-to-Figure Scenarios 56
3.4.2 When Is Feasibility Study Not Required? 56
3.5 Prototyping 57
3.5.1 Throwaway Prototyping 58
3.5.2 Evolutionary Prototyping 59
3.5.3 Incorporating Prototyping in Software Development 60
3.6 Pilot Projects 61
3.7 Chapter Summary 62
Recommended Reading 63

CHAPTER 4 SOFTWARE REQUIREMENTS SPECIFICATIONS

4.1 Introduction 65
4.2 Software Engineering Methodologies and Requirements Management 66
4.2.1 Requirements Management in eXtreme Programming 66
4.2.2 Requirements Management in Scrum 67
4.2.3 Requirements Management in Waterfall 67
4.3 Implementation of the Requirements 68
4.4 Requirement Types 69
4.4.1 Functional Requirements 69
4.4.2 Nonfunctional Requirements 70
4.5 Sources of Requirements 72
4.6 Categories of Users 73
4.7 Software Requirement Life Cycle 74
4.8 Requirements Gathering (Elicitation) 76
4.8.1 Requirements Meetings 77
4.8.2 E-mail 77
4.9 Requirements Analysis 77
4.10 Requirements Specification 78
4.10.1 Use Cases 79
4.10.2 Relevance of Use Cases in Software Design 82
4.10.3 Use Case Example 82
4.11 Requirements Management 84
5.13.3 Technology Availability
  5.13.3.1 Integrated Development Environment Infrastructure
  5.13.3.2 Platform Independence
  5.13.3.3 Library Infrastructure
5.13.4 Code Reuse
5.14 Types of Programming Languages: A Brief Review
  5.14.1 Special Languages for Web-Based Systems
5.15 Security
  5.15.1 Authorization
  5.15.2 Role-Based Security
  5.15.3 Security Based on the User Data
  5.15.4 Data Encryption
5.16 Performance
5.17 Case Study
  5.17.1 Three-Tier Architecture
  5.17.2 User Interface
  5.17.3 Middle Layer
  5.17.4 Database
  5.17.5 Component Diagram
  5.17.6 Data Flow Diagram
  5.17.7 Code Reuse
  5.17.8 Maintainability
  5.17.9 Security
  5.17.10 Performance
  5.17.11 Methodology
5.18 Chapter Summary
Recommended Reading

Chapter 6 Software User Interface Design and Construction
6.1 Introduction
6.2 Graphical User Interface
  6.2.1 Rich Windows-Based GUIs
  6.2.2 Web Browser
6.3 Graphic Control Elements
  6.3.1 Containers
    6.3.1.1 Window
    6.3.1.2 Modal Window
    6.3.1.3 Dialog Box
    6.3.1.4 Frame
    6.3.1.5 Canvas
    6.3.1.6 Form
  6.3.2 Selection and Display Elements
    6.3.2.1 Button
    6.3.2.2 Radio Button
    6.3.2.3 Checkbox
    6.3.2.4 Slider
    6.3.2.5 List Box
    6.3.2.6 Drop-Down List
    6.3.2.7 Menu (Menu Item)
    6.3.2.8 Context Menu
    6.3.2.9 Menu Bar
## CONTENTS

7.18.3 Detailed Design for OBAAS 1.2 238
   7.18.3.1 Class Diagram 238
   7.18.3.2 Object Diagram 238
7.18.4 Implementation Considerations 238
   7.18.4.1 User Validation 240

7.19 Chapter Summary 241
Recommended Reading 243

### Chapter 8 Database Design and Construction 245

8.1 Introduction 245
8.2 Databases and Software Engineering Methodologies 247
8.3 Database Types 248
   8.3.1 NoSQL Databases 248
   8.3.2 Relational Databases 249
8.4 Database Languages 249
   8.4.1 Data Dictionary Language 250
   8.4.2 Data Manipulation Language 251
8.5 Database Entities 252
   8.5.1 Schema 253
   8.5.2 Tables 253
   8.5.3 Primary Keys 254
   8.5.4 Foreign Keys 256
   8.5.5 Indexes 258
   8.5.6 Sequences 259
   8.5.7 Stored Procedures 259
   8.5.8 Triggers 261
8.6 Database Design 261
   8.6.1 ER Diagram 261
   8.6.2 Normalization 264
   8.6.3 Relationship among Tables 267
      8.6.3.1 One-to-Many 267
      8.6.3.2 Many-to-Many 268
      8.6.3.3 One-to-One 268
   8.6.4 Integrity Rules 269
      8.6.4.1 Entity Integrity Rule 269
      8.6.4.2 Referential Integrity Rule 269
   8.6.5 Master and Transaction Data 270
8.7 Database Management Systems 272
8.8 Object Relational Databases 272
8.9 Case Study 273
   8.9.1 ER Diagram 273
   8.9.2 Schema 274
   8.9.3 Database Transactions 276
   8.9.4 Future Enhancements 276
8.10 Chapter Summary 276
    Recommended Reading 278

### Chapter 9 Software Testing (Verification and Validation) 279

9.1 Introduction 279
9.2 Software Testing and Software Engineering Methodologies 281
9.3 Introduction to Different Types of Software Testing 282
9.4 Introduction to Verification and Validation 282
CONTENTS

9.5 Introduction to Levels of Software Testing (Validation) 284
  9.5.1 V Model 284

9.6 Verification 286
  9.6.1 Requirement Reviews 286
  9.6.2 Design Reviews 287
  9.6.3 Code Walkthroughs 287
  9.6.4 Code Inspection/Code Review 288

9.7 Unit Testing 288
  9.7.1 Database Unit Testing 292

9.8 Integration Testing 294

9.9 System Testing 298
  9.9.1 Functional Testing 298
  9.9.2 Nonfunctional Tests 299
  9.9.3 Regression Tests 300

9.10 User Acceptance Testing 300

9.11 Other Important Tests 301
  9.11.1 Sanity Tests 301
  9.11.2 Smoke Testing 302
  9.11.3 Cyclomatic Test 303
  9.11.4 Incremental Testing 303
  9.11.5 Exploratory Testing 303
  9.11.6 Coverage-Based Testing 304
  9.11.7 Alpha Testing 304
  9.11.8 Beta Testing 305

9.12 Test Case Design 305
  9.12.1 White Box (Unit and Integration) Test Case Design 307
    9.12.1.1 Test Coverage 308
    9.12.1.2 Types of Coverage 308
    9.12.1.3 Statement Tests 309
    9.12.1.4 Decision Tree Tests 309
  9.12.2 Black Box (System and User Acceptance) Test Case Design 310
    9.12.2.1 Test Case Workflow 310
    9.12.2.2 Equivalence Partitioning 310
    9.12.2.3 Boundary Value Analysis 310
    9.12.2.4 Decision Table Testing 311
    9.12.2.5 State Transition Testing 312
    9.12.2.6 Use Case Testing 312
    9.12.2.7 Experience-Based Testing 313
    9.12.2.8 Choosing the Right Testing Techniques 313
  9.12.3 Test Case Design for New Technologies 313

9.13 Test Preparation 314
  9.13.1 Test Environment (Test Bed) Preparation 315
  9.13.2 Test Data Preparation 315
  9.13.3 Configuration Management Preparation 315

9.14 Test Life Cycle 316
  9.14.1 Test Planning 316
  9.14.2 Test Case Design and Creation 317
  9.14.3 Test Case Execution 318
  9.14.4 Defect Life Cycle/Tracking 318
  9.14.5 Test Reporting 319
  9.14.6 Exit Criteria and Test Closure 319
9.15 Case Study
  9.15.1 System Parts
  9.15.2 Test Strategy
  9.15.3 User Interface Testing
  9.15.4 Client-Side User Input Validation
  9.15.5 Unit Testing
  9.15.6 Integration Testing
  9.15.7 System Testing
9.16 Chapter Summary
Recommended Reading

Chapter 10 Software Release
  10.1 Introduction
  10.2 Software Release and Software Engineering Methodology
  10.3 Integration
  10.4 Documentation
    10.4.1 User Manual
    10.4.2 Technical Manual
  10.5 Release Cycles
    10.5.1 Release Cycle for Waterfall Projects
    10.5.2 Release Cycles for Incremental Building of Products
    10.5.3 Difference between Alpha–Beta–Final Release and Incremental Release
  10.6 User Training
  10.7 Deployment
  10.8 Software Migration Strategies
  10.9 Software Product Release Checklist
  10.10 Chapter Summary
Recommended Reading

Chapter 11 Software Maintenance
  11.1 Introduction
  11.2 Software Maintenance and Software Engineering Methodology
  11.3 Production Environment Maintenance
  11.4 Production Environment Monitoring
  11.5 Maintenance Process
  11.6 Types of Software Maintenance
  11.7 Software Maintenance Strategies
  11.8 Reverse Engineering
  11.9 Chapter Summary
Recommended Reading

Chapter 12 Configuration and Version Management
  12.1 Introduction
  12.2 Configuration Management and Version Control Concepts
    12.2.1 Artifact Versions and Many Copies
    12.2.2 Limitations of the File System of an Operating System
    12.2.3 Software Build
    12.2.4 File Locking
  12.3 CVS Systems
    12.3.1 Software Build on CVS Systems
    12.3.2 Branching in CVS Systems
    12.3.3 Branches and Different Product Releases
Software engineering is a vast field. It has the following core areas: requirements management, software architecture, software design and construction, and software testing. All of these areas are vast and are mostly separated from each other. The core area software design and construction is further divided into user interface design and construction, business logic design and construction, and database design and construction.

The authors have several years of experience working with and teaching all areas of software engineering. The lead author has also been writing university-level textbooks in areas related to software engineering for the last 10 years.

This book provides in-depth coverage of the areas of software engineering that are essential to learning this field. Advanced areas such as commercial-grade security mechanisms, and performance management techniques for high-end software products, are not discussed in this book. Discussing these areas may distract the reader from the primary goal of learning software engineering.

This book was written after receiving feedback from several professors and software engineers from the United States and India. A book on software engineering was thought to be necessary, and it not only should cover the theory of software engineering but also should have content for practice. Without practical examples to work on, it is difficult to learn any subject. This is especially true for a subject such as software engineering, which is used to build large and complex software products. At the same time, it is also true that many universities have a curriculum in software engineering that includes building a software product by undertaking a software project. This book covers both theory and practical examples of building software products.
As a result of extensive research conducted by the authors on what content to include in the book and how to present that content, the material provided in this book is relevant for teaching software engineering to students. Rather than using UML or other formal notations, the content is explained in plain (and easy-to-understand) English. Basic programming knowledge using an object-oriented language is helpful to understand the contents of this book. We have provided a complete case study using Java. The knowledge gained from this book can be readily used in other relevant courses or in a typical software development environment of a company.

The best way to learn software engineering is by understanding its core and peripheral areas. In this book, a complete chapter is devoted to each of the core and peripheral areas. Requirements management is covered by explaining requirement specifications and use cases. Software high-level design aspects are covered including software architecture patterns, software architectures, component design, component diagrams, and data flow diagrams. Software detailed design and construction for business logic development covers classes, class diagrams, objects, object diagrams, statechart diagrams, sequence diagrams, database programming, web-based programming, refactoring, and so on. User interface design includes Model–View–Controller architecture, HTML, client-side scripts, and AJAX, among others. Database design covers schemas, relational database concepts, referential and entity integrity rules, primary keys, secondary keys, normalization, and so on. Software testing includes unit testing, integration testing, system testing, user acceptance testing, and testing strategy, to name a few.

The book also covers peripheral areas of feasibility study, software engineering methodologies, project management, configuration management, and software maintenance.

Some of the salient features of the book include the following:

- Complete coverage of all important areas of software engineering.
- Strong foundation for almost all areas of software engineering. Each software engineering concept is backed by solid examples.
- Complete alignment with university-level courses in software engineering offered at several universities in the United States and India.
- Complete case study with source code (source code is available at the course website).
- Object-oriented design and programming with complete concepts in modeling and design using classes and objects.
- Database design with entity-relationship concept.

The case studies that have been provided for all core areas of software engineering are a useful feature of this book. They provide an understanding of software engineering as well as how things work while building a software product. The following
artifacts are provided for a case study that deals with an online banking system called OBAAS:

- Complete requirements specifications along with the use cases
- Software high-level design including software architecture, software component diagrams, and data flow diagrams
- Detailed design for developing the business logic including class diagrams, object diagrams, statechart diagrams, and sequence diagrams
- Software construction including complete source code and technical documentation
- Database design (entity-relationship diagrams and table structure)
- User interface design (mockup screens)
- Testing strategy including testing samples

This case study is provided at the end of each of the core chapters under a separate section. While all the design documents and requirements specifications are provided in the book itself, the source code for this software product is provided at the course website: http://ahmedashfaque.wordpress.com. People can easily download the source code as well as the database scripts from this website. In case this website is down for any reason, the details of the new website will be posted at http://www.bhanuprasad.org/fse.html. There are two sets of source code: OBAAS Version 1.1 and OBAAS Version 1.2. We have used Java technology to help readers learning how to apply software engineering concepts develop an object-oriented project. The other case study deals with an order management system for a restaurant. It is provided within the chapters.

The case study on OBAAS can also be used for doing additional exercises. We have provided some ideas about enhancing OBAAS at many places in the case study. For example, we are currently using hard-coded values to populate the service types in a drop-down list in the source code. This functionality can be extended so that instead of hard-coded values these values can come from a database. You can find a list of future enhancements in Chapters 4 and 8, which can be taken as a practical exercise or assignment in the classroom. These exercises can be in any area of software engineering. For example, a feasibility study can be done to find out if any of the enhancements listed there are feasible.

All in all, this book has everything one needs to learn software engineering easily. There are also myriad code samples, comparisons, and examples provided throughout the book on all the topics covered. They will help the reader clearly understand the concepts.

This book has an appendix that contains answers to all the questions at the end of each chapter. Instructor material is available at the course website for instructors who adopt this book as a textbook.

The book will be very useful for students taking a course in software engineering or its equivalent at the undergraduate or graduate level. People working in the software
industry and those who want to bridge their knowledge gap in any area related to software engineering may also find this book useful.

We invite you to e-mail us any comments concerning this book at the addresses provided below. We will use them to improve future editions of this book. We also invite you to e-mail us any errors that you notice. The errata will be posted at the course website. Hope you enjoy reading this book!

Ashfaque Ahmed
ashfaque.a@gmail.com

Bhanu Prasad
bhanupvsr@gmail.com
Acknowledgments

This book was prepared after getting feedback from a large number of professors who teach courses in software engineering or its equivalent at several universities in the United States and India. We are thankful to all these people for their invaluable feedback.

I thank Dr. Nadimpalli Mahadev, professor, Department of Computer Science, Fitchburg State University, Massachusetts; Dr. Victoria Hilford, assistant professor, Department of Computer Science, University of Houston, Texas; Dr. Guowei Yang, assistant professor, Department of Computer Science, Texas State University, San Marcos; and Dr. Scott Stoller, professor, Computer Science Department, Stony Brook University, New York. We are also thankful to many other professors, whose invaluable advice and feedback helped in creating the structure and content of this book.

Writing a large textbook is very demanding and labor-intensive. While I was working on writing the book, I was fully supported by my family. Without their support, I could not have done it. Credit goes to my father, mother, wife, and brother Javed, who have always stood by me and my commitment to writing this book. Javed also helped in writing the source code for the case studies. During breaks in writing, my family’s kids, Arisha, Shija, Aashi, Jasim, and Ismail, enlivened me. Playing with kids is so much fun and relieves my stress. After playing with kids, I get recharged for the next load of hard work.

Sadly, during the time that I was (too engrossed) writing this book, my youngest brother, Aslam, passed away because of heart failure. It jolted me so much that for quite some time I was not able to concentrate on anything. Somehow, I mustered enough courage and strength to finish writing this book.

Ashfaque Ahmed
I am thankful to many people, including my colleagues, students, relatives, and friends, who helped me in writing this book and for providing valuable input from time to time. I am thankful to Florida A&M University for providing an environment where I was able to work on this book. I am thankful to Dr. Bobby Granville, Dr. Zoran Majkic, Samuel, John, Sreedevi, Anjali, Ram, and Brown for their valuable feedback and help.

I am thankful to India, the country where I was born and brought up and from which I received my academic education. I am also thankful to all my teachers and professors. Finally and most importantly, I am thankful to my parents for all their support and encouragement in pursuing my academic education.

Bhanu Prasad
Authors

Ashfaque Ahmed is a seasoned software engineering professional with more than 27 years of experience. He has worked as a programmer analyst, software engineer, project manager, test engineer, test manager, and business analyst in his career. He has work experience in all areas of software engineering including requirements management, software design, software construction, software testing, and database design. He has worked for leading multinational companies in his career, in countries including the United States, Canada, United Arab Emirates, India, Libya, and Nigeria.

Ahmed is also a well-known author. He has written *Software Testing as a Service* (2009, CRC Press, USA); *Software Project Management: A Process-Driven Approach* (2011, CRC Press, USA); *The SAP Materials Management Handbook* (2014, CRC Press, USA); and two smaller books. His books are recommended for reading at large universities in countries including the United States, India, Canada, South Korea, Thailand, and Indonesia.

Ahmed is passionate about helping mankind by using technology and spreading awareness to combat climate change and global warming. Using renewable energy and finding better ways to reduce our carbon footprint is one area that is close to his heart.

Ashfaque Ahmed earned his bachelor of engineering degree from the National Institute of Technology, Raipur, India, in 1988 and master of business administration degree from Indira Gandhi National Open University, India, in 1997.
Bhanu Prasad earned his master of technology and PhD degrees, both in computer science, from Andhra University and the Indian Institute of Technology Madras, respectively. He is currently serving as an associate professor in the Department of Computer and Information Sciences at Florida A&M University, Tallahassee, where he was also the chairman of the department. He served as an assistant professor of computer science at Georgia Southwestern State University, Americus. He worked with Hyperion Solutions Corporation (now part of Oracle) in the United States and Infosys Technologies and Future Software (now part of Aricent) in India.

Dr. Prasad’s current research interests are in the areas of artificial intelligence and software engineering. He has authored or co-authored more than 70 research papers in various international conference proceedings, journals, or books in computer science. Some of the courses he has taught at the graduate level include software engineering, software development and maintenance, advanced system design principles, human–computer interaction and interface design, internetworking architecture and protocols, programming languages, topics in advanced artificial intelligence, operating systems, and theory of computation. He has also taught software engineering and approximately 20 other courses at the undergraduate level. Dr. Prasad received several certificates including the Rational Rose certificate and Rational Unified Process certificate from the Rational Software Corporation (now part of IBM). He has supervised graduate students to successfully complete their thesis work and served as an external reviewer or committee member for graduate theses or dissertations from several universities around the world. He is or was a reviewer, program committee member, organizing committee member, co-chair, chair, keynote speaker, editorial board member, guest editor, co-editor, editor, and editor in chief for more than a hundred international conferences, journals, or books in computer science.