Contents

Preface to the Third Edition v

1 Introduction 1
 1.1 The Problem Domain 2
  1.1.1 Industrial Strength Software 2
  1.1.2 Software is Expensive 3
  1.1.3 Late and Unreliable 5
  1.1.4 Maintenance and Rework 5
 1.2 The Software Engineering Challenges 7
  1.2.1 Scale 8
  1.2.2 Quality and Productivity 9
  1.2.3 Consistency and Repeatability 12
  1.2.4 Change 13
 1.3 The Software Engineering Approach 13
  1.3.1 Phased Development Process 14
  1.3.2 Managing the Process 17
 1.4 Summary 18
  Exercises 19

2 Software Processes 21
 2.1 Software Process 21
  2.1.1 Processes and Process Models 22
  2.1.2 Component Software Processes 23
  2.1.3 ETVX Approach for Process Specification 24
 2.2 Desired Characteristics of Software Process 25
  2.2.1 Predictability 26
  2.2.2 Support Testability and Maintainability 28
  2.2.3 Support Change 29
  2.2.4 Early Defect Removal 29
  2.2.5 Process Improvement and Feedback 31
 2.3 Software Development Process Models 31

2.3.1 Waterfall Model 32
2.3.2 Prototyping 34
2.3.3 Iterative Development 36
2.3.4 Timeboxing Model 39
2.3.5 Comparision of Models 42
2.4 Other Software Processes 44
2.4.1 Project Management Process 44
2.4.2 The Inspection Process 46
2.4.3 Software Configuration Management Process 52
2.4.4 Requirements Change Management Process 57
2.4.5 Process Management Process 58
2.5 Summary 62
Exercises 64

3 Software Requirements Analysis and Specification 67
3.1 Software Requirements 68
3.1.1 Need for SRS 68
3.1.2 Requirement Process 72
3.2 Problem Analysis 75
3.2.1 Informal Approach 76
3.2.2 Data Flow Modeling 77
3.2.3 Object-Oriented Modeling 88
3.2.4 Prototyping 95
3.3 Requirements Specification 99
3.3.1 Characteristics of an SRS 101
3.3.2 Components of an SRS 103
3.3.3 Specification Language 105
3.3.4 Structure of a Requirements Document 106
3.4 Functional Specification with Use Cases 109
3.4.1 Basics 110
3.4.2 Examples 112
3.4.3 Extensions 115
3.4.4 Developing Use Cases 116
3.5 Validation 118
3.6 Metrics 121
3.6.1 Size—Function Points 121
3.6.2 Quality Metrics 125
3.7 Summary 127
Exercises 128
Case Studies 130

4 Software Architecture 137
- 4.1 Role of Software Architecture 138
- 4.2 Architecture Views 141
- 4.3 Component and Connector View 144
  - 4.3.1 Components 144
  - 4.3.2 Connectors 145
  - 4.3.3 An Example 148
- 4.4 Architecture Styles for C&C View 151
  - 4.4.1 Pipe and Filter 152
  - 4.4.2 Shared-Data Style 154
  - 4.4.3 Client-Server Style 156
  - 4.4.4 Some Other Styles 157
- 4.5 Discussion 158
  - 4.5.1 Architecture and Design 158
  - 4.5.2 Preserving the Integrity of an Architecture 159
  - 4.5.3 Deployment View and Performance Analysis 162
  - 4.5.4 Documenting Architecture Design 164
- 4.6 Evaluating Architectures 167
  - 4.6.1 The ATAM Analysis Method 168
  - 4.6.2 An Example 169
- 4.7 Summary 172
  - Exercises 173
  - Case Studies 175

5 Planning a Software Project 179
- 5.1 Process Planning 180
- 5.2 Effort Estimation 180
  - 5.2.1 Uncertainties in Effort Estimation 181
  - 5.2.2 Building Effort Estimation Models 182
  - 5.2.3 A Bottom-Up Estimation Approach 184
  - 5.2.4 COCOMO Model 185
- 5.3 Project Scheduling and Staffing 189
  - 5.3.1 Overall Scheduling 190
  - 5.3.2 Detailed Scheduling 192
  - 5.3.3 An Example 193
  - 5.3.4 Team Structure 194
- 5.4 Software Configuration Management Plan 195
- 5.5 Quality Plan 196
  - 5.5.1 Defect Injection and Removal Cycle 197
  - 5.5.2 Approaches to Quality Management 198
  - 5.5.3 Quality Plan 199

5.6 Risk Management . . . 199
5.6.1 Risk Management Concepts . . . 200
5.6.2 Risk Assessment . . . 200
5.6.3 Risk Control . . . 204
5.6.4 A Practical Risk Management Approach . . . 205
5.7 Project Monitoring Plan . . . 207
5.7.1 Measurements . . . 207
5.7.2 Project Monitoring and Tracking . . . 208
5.8 Summary . . . 210
Exercises . . . 211
Case Studies . . . 213

6 Function-Oriented Design 215
6.1 Design Principles . . . 216
6.1.1 Problem Partitioning and Hierarchy . . . 218
6.1.2 Abstraction . . . 219
6.1.3 Modularity . . . 220
6.1.4 Top-Down and Bottom-Up Strategies . . . 220
6.2 Module-Level Concepts . . . 222
6.2.1 Coupling . . . 222
6.2.2 Cohesion . . . 224
6.3 Design Notation and Specification . . . 226
6.3.1 Structure Charts . . . 227
6.3.2 Specification . . . 230
6.4 Structured Design Methodology . . . 231
6.4.1 Restate the Problem as a Data Flow Diagram . . . 232
6.4.2 Identify the Most Abstract Input and Output Data Elements . . 234
6.4.3 First-Level Factoring . . . 236
6.4.4 Factoring the Input, Output, and Transform Branches . . . 237
6.4.5 Design Heuristics . . . 239
6.4.6 Transaction Analysis . . . 241
6.4.7 Discussion . . . 242
6.5 Verification . . . 244
6.6 Metrics . . . 246
6.6.1 Network Metrics . . . 246
6.6.2 Stability Metrics . . . 247
6.6.3 Information Flow Metrics . . . 250
6.7 Summary . . . 252
Exercises . . . 253
Case Studies . . . 255

7 Object-Oriented Design 263
7.1 OO Analysis and OO Design 264
7.2 OO Concepts 266
7.2.1 Classes and Objects 266
7.2.2 Relationships Among Objects 271
7.2.3 Inheritance and Polymorphism 273
7.3 Design Concepts 280
7.3.1 Coupling 281
7.3.2 Cohesion 282
7.3.3 The Open-Closed Principle 284
7.3.4 Some Design Guidelines 286
7.4 Unified Modeling Language (UML) 287
7.4.1 Class Diagram 288
7.4.2 Sequence and Collaboration Diagrams 291
7.4.3 Other Diagrams and Capabilities 294
7.5 A Design Methodology 296
7.5.1 Dynamic Modeling 298
7.5.2 Functional Modeling 300
7.5.3 Defining Internal Classes and Operations 300
7.5.4 Optimize and Package 301
7.5.5 Examples 302
7.6 Metrics 309
7.7 Summary 313
Exercises 314
Case Studies 316

8 Detailed Design 323
8.1 Detailed Design and PDL 323
8.1.1 PDL 323
8.1.2 Logic/Algorithm Design 326
8.1.3 State Modeling of Classes 330
8.2 Verification 331
8.2.1 Design Walkthroughs ✗ 331
8.2.2 Critical Design Review 332
8.2.3 Consistency Checkers 333
8.3 Metrics 333
8.3.1 Cyclomatic Complexity 334
8.3.2 Data Bindings 336
8.3.3 Cohesion Metric 337
8.4 Summary 338
Exercises 339

9 Coding 341
9.1 Programming Principles and Guidelines 342
9.1.1 Common Coding Errors 342
9.1.2 Structured Programming 347
9.1.3 Information Hiding 349
9.1.4 Some Programming Practices 350
9.1.5 Coding Standards 354
9.2 Coding Process 357
9.2.1 An Incremental Coding Process 357
9.2.2 Test Driven Development 358
9.2.3 Pair Programming 360
9.2.4 Source Code Control and Build 361
9.3 Refactoring 363
9.3.1 Basic Concepts 364
9.3.2 An example 366
9.3.3 Bad Smells 369
9.3.4 Common Refactorings 370
9.4 Verification 374
9.4.1 Code Inspections 374
9.4.2 Static Analysis 376
9.4.3 Proving Correctness 381
9.4.4 Unit Testing 387
9.4.5 Combining Different Techniques 390
9.5 Metrics 394
9.5.1 Size Measures 394
9.5.2 Complexity Metrics 395
9.6 Summary 398
Exercises 400
Case Studies 403

10 Testing 405
10.1 Testing Fundamentals 405
10.1.1 Error, Fault, and Failure 406
10.1.2 Test Oracles 407
10.1.3 Test Cases and Test Criteria 408
10.1.4 Psychology of Testing 410
10.2 Black-Box Testing 411
10.2.1 Equivalence Class Partitioning 412
10.2.2 Boundary Value Analysis 414
10.2.3 Cause-Effect Graphing 415
10.2.4 Pair-wise Testing 418

10.2.5 Special Cases . . . 421
10.2.6 State-Based Testing . . . 421
10.3 White-Box Testing . . . 424
10.3.1 Control Flow-Based Criteria . . . 425
10.3.2 Data Flow-Based Testing . . . 428
10.3.3 An Example . . . 431
10.3.4 Mutation Testing . . . 433
10.3.5 Test Case Generation and Tool Support . . . 437
10.4 Testing Process . . . 439
10.4.1 Levels of Testing . . . 439
10.4.2 Test Plan . . . 441
10.4.3 Test Case Specifications . . . 443
10.4.4 Test Case Execution and Analysis . . . 445
10.4.5 Defect Logging and Tracking . . . 446
10.5 Defect Analysis and Prevention . . . 449
10.5.1 Pareto Analysis . . . 449
10.5.2 Perform Causal Analysis . . . 451
10.5.3 Develop and Implement Solutions . . . 453
10.6 Metrics—Reliability Estimation . . . 454
10.6.1 Basic Concepts and Definitions . . . 455
10.6.2 A Reliability Model . . . 456
10.6.3 Failure Data and Parameter Estimation . . . 460
10.6.4 Translating to Calendar Time . . . 462
10.6.5 An Example . . . 463
10.7 Summary . . . 465
Exercises . . . 466
Case Studies . . . 469

Bibliography **473**

Index **483**