# CONTENTS

<table>
<thead>
<tr>
<th>Preface vii</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PART 1</strong> Introduction to the Software Life Cycle 1</td>
</tr>
<tr>
<td><strong>CHAPTER 1</strong> Scope of Software Engineering 3</td>
</tr>
<tr>
<td>1.1 Historical Aspects 5</td>
</tr>
<tr>
<td>1.2 Economic Aspects 7</td>
</tr>
<tr>
<td>1.3 Maintenance Aspects 8</td>
</tr>
<tr>
<td>1.4 Specification and Design Aspects 12</td>
</tr>
<tr>
<td>1.5 Team Programming Aspects 15</td>
</tr>
<tr>
<td>1.6 The Object-Oriented Paradigm 16</td>
</tr>
<tr>
<td>1.7 Terminology 21</td>
</tr>
<tr>
<td>Chapter Review 23</td>
</tr>
<tr>
<td>For Further Reading 24</td>
</tr>
<tr>
<td>Problems 25</td>
</tr>
<tr>
<td>References 26</td>
</tr>
<tr>
<td><strong>CHAPTER 2</strong> The Software Process 30</td>
</tr>
<tr>
<td>2.1 Client, Developer, and User 32</td>
</tr>
<tr>
<td>2.2 Requirements Phase 33</td>
</tr>
<tr>
<td>2.2.1 Requirements Phase Testing 34</td>
</tr>
<tr>
<td>2.3 Specification Phase 35</td>
</tr>
<tr>
<td>2.3.1 Specification Phase Testing 37</td>
</tr>
<tr>
<td>2.4 Design Phase 37</td>
</tr>
<tr>
<td>2.4.1 Design Phase Testing 39</td>
</tr>
<tr>
<td>2.5 Implementation Phase 39</td>
</tr>
<tr>
<td>2.5.1 Implementation Phase Testing 39</td>
</tr>
<tr>
<td>2.6 Integration Phase 40</td>
</tr>
<tr>
<td>2.6.1 Integration Phase Testing 40</td>
</tr>
<tr>
<td>2.7 Maintenance Phase 41</td>
</tr>
<tr>
<td>2.7.1 Maintenance Phase Testing 41</td>
</tr>
<tr>
<td>2.8 Retirement 42</td>
</tr>
<tr>
<td>2.9 Problems with Software Production: Essence and Accidents 43</td>
</tr>
<tr>
<td>2.9.1 Complexity 44</td>
</tr>
<tr>
<td>2.9.2 Conformity 46</td>
</tr>
<tr>
<td>2.9.3 Changeability 47</td>
</tr>
<tr>
<td>2.9.4 Invisibility 48</td>
</tr>
<tr>
<td>2.9.5 No Silver Bullet? 49</td>
</tr>
<tr>
<td>2.10 Improving the Software Process 50</td>
</tr>
<tr>
<td>2.11 Capability Maturity Models 50</td>
</tr>
<tr>
<td>2.12 ISO 9000 54</td>
</tr>
<tr>
<td>2.13 SPICE 55</td>
</tr>
<tr>
<td>2.14 Costs and Benefits of Software Process Improvement 56</td>
</tr>
<tr>
<td>Chapter Review 57</td>
</tr>
<tr>
<td>For Further Reading 58</td>
</tr>
<tr>
<td>Problems 59</td>
</tr>
<tr>
<td>References 60</td>
</tr>
<tr>
<td><strong>CHAPTER 3</strong> Software Life-Cycle Models 64</td>
</tr>
<tr>
<td>3.1 Build-and-Fix Model 64</td>
</tr>
<tr>
<td>3.2 Waterfall Model 65</td>
</tr>
<tr>
<td>3.2.1 Analysis of the Waterfall Model 68</td>
</tr>
<tr>
<td>3.3 Rapid Prototyping Model 70</td>
</tr>
<tr>
<td>3.3.1 Integrating the Waterfall and Rapid Prototyping Models 72</td>
</tr>
<tr>
<td>3.4 Incremental Model 72</td>
</tr>
<tr>
<td>3.4.1 Analysis of the Incremental Model 74</td>
</tr>
<tr>
<td>3.5 Synchronize-and-Stabilize Model 77</td>
</tr>
<tr>
<td>3.6 Spiral Model 77</td>
</tr>
<tr>
<td>3.6.1 Analysis of the Spiral Model 82</td>
</tr>
<tr>
<td>3.7 Object-Oriented Life-Cycle Models 83</td>
</tr>
<tr>
<td>3.8 Comparison of Life-Cycle Models 84</td>
</tr>
<tr>
<td>Chapter Review 86</td>
</tr>
<tr>
<td>For Further Reading 86</td>
</tr>
<tr>
<td>Problems 87</td>
</tr>
<tr>
<td>References 88</td>
</tr>
</tbody>
</table>
CHAPTER 4
Teams and the Tools of Their Trade  90

4.1 Team Organization 90
4.2 Democratic Team Approach 92
  4.2.1 Analysis of the Democratic Team Approach 93
4.3 Classical Chief Programmer Team Approach 94
  4.3.1 The New York Times Project 96
  4.3.2 Impracticality of the Classical Chief Programmer Team Approach 96
4.4 Beyond Chief Programmer and Democratic Teams 97
4.5 Synchronize-and-Stabilize Teams 101
4.6 Stepwise Refinement 102
  4.6.1 Stepwise Refinement Example 103
4.7 Cost–Benefit Analysis 109
4.8 Software Metrics 110
4.9 CASE 112
4.10 Taxonomy of CASE 113
4.11 Scope of CASE 114
4.12 Software Versions 119
  4.12.1 Revisions 119
  4.12.2 Variations 120
4.13 Configuration Control 120
  4.13.1 Configuration Control during Product Maintenance 123
  4.13.2 Baselines 124
  4.13.3 Configuration Control during Product Development 124
4.14 Build Tools 125
4.15 Productivity Gains with CASE Technology 126
Chapter Review 128
For Further Reading 128
Problems 129
References 131

CHAPTER 5
Testing  134

5.1 Quality Issues 135
  5.1.1 Software Quality Assurance 135
  5.1.2 Managerial Independence 136
5.2 Nonexecution-Based Testing 137
  5.2.1 Walkthroughs 137
  5.2.2 Managing Walkthroughs 138
  5.2.3 Inspections 139
  5.2.4 Comparison of Inspections and Walkthroughs 141
  5.2.5 Strengths and Weaknesses of Reviews 142
  5.2.6 Metrics for Inspections 142
5.3 Execution-Based Testing 143
5.4 What Should Be Tested? 143
  5.4.1 Utility 144
  5.4.2 Reliability 145
  5.4.3 Robustness 145
  5.4.4 Performance 146
  5.4.5 Correctness 146
5.5 Testing versus Correctness Proofs 148
  5.5.1 Example of a Correctness Proof 148
  5.5.2 Correctness Proof Case Study 152
  5.5.3 Correctness Proofs and Software Engineering 154
5.6 Who Should Perform Execution-Based Testing? 157
5.7 Testing Distributed Software 158
5.8 Testing Real-Time Software 160
5.9 When Testing Stops 162
Chapter Review 163
For Further Reading 163
Problems 165
References 166

CHAPTER 6
Introduction to Objects  171

6.1 What Is a Module? 171
6.2 Cohesion 175
  6.2.1 Coincidental Cohesion 175
  6.2.2 Logical Cohesion 176
  6.2.3 Temporal Cohesion 178
  6.2.4 Procedural Cohesion 178
  6.2.5 Communicational Cohesion 178
  6.2.6 Informational Cohesion 179
  6.2.7 Functional Cohesion 180
  6.2.8 Cohesion Example 180
6.3 Coupling 181
  6.3.1 Content Coupling 182
  6.3.2 Common Coupling 182
CHAPTER 7
Reusability, Portability, and Interoperability 217

7.1 Reuse Concepts 217
7.2 Impediments to Reuse 219
7.3 Reuse Case Studies 220
  7.3.1 Raytheon Missile Systems Division 220
  7.3.2 Toshiba Software Factory 222
  7.3.3 NASA Software 223
  7.3.4 GTE Data Services 224
  7.3.5 Hewlett-Packard 224
  7.3.6 European Space Agency 225
7.4 Objects and Productivity 226
7.5 Reuse during the Design and Implementation Phases 228
  7.5.1 Module Reuse 228
  7.5.2 Application Frameworks 229
  7.5.3 Design Patterns 230
  7.5.4 Software Architecture 235
7.6 Reuse and Maintenance 235
7.7 Portability 236
  7.7.1 Hardware Incompatibilities 237
  7.7.2 Operating System Incompatibilities 238
7.7.3 Numerical Software Incompatibilities 239
7.7.4 Compiler Incompatibilities 239
7.8 Why Portability? 245
7.9 Techniques for Achieving Portability 246
  7.9.1 Portable System Software 246
  7.9.2 Portable Application Software 247
  7.9.3 Portable Data 248
7.10 Interoperability 249
  7.10.1 OLE, COM, and ActiveX 250
  7.10.2 CORBA 251
  7.10.3 Comparing OLE/COM and CORBA 251
7.11 Future Trends in Interoperability 252
Chapter Review 252
For Further Reading 253
Problems 254
References 256

CHAPTER 8
Planning and Estimating 262

8.1 Planning and the Software Process 262
8.2 Estimating Duration and Cost 264
  8.2.1 Metrics for the Size of a Product 265
  8.2.2 Techniques of Cost Estimation 270
  8.2.3 Intermediate COCOMO 273
  8.2.4 COCOMO II 276
  8.2.5 Tracking Duration and Cost Estimates 277
8.3 Components of a Software Project Management Plan 278
8.4 Software Project Management Plan Framework 280
8.5 IEEE Software Project Management Plan 281
8.6 Planning of Testing 284
8.7 Planning of Object-Oriented Projects 285
8.8 Training Requirements 286
8.9 Documentation Standards 287
8.10 CASE Tools for Planning and Estimating 288
CONTENTS

8.11 Testing the Software Project Management Plan 291
Chapter Review 291
For Further Reading 291
Problems 293
References 294

PART 2
The Phases of the Software Life Cycle 299

CHAPTER 9
Requirements Phase 301

9.1 Requirements Analysis Techniques 302
9.2 Rapid Prototyping 303
9.3 Human Factors 305
9.4 Rapid Prototyping as a Specification Technique 307
9.5 Reusing the Rapid Prototype 309
9.6 Other Uses of Rapid Prototyping 311
9.7 Management Implications of the Rapid Prototyping Model 312
9.8 Experiences with Rapid Prototyping 313
9.9 Joint Application Design (JAD) 315
9.10 Comparison of Requirements Analysis Techniques 315
9.11 Testing during the Requirements Phase 316
9.12 CASE Tools for the Requirements Phase 316
9.13 Metrics for the Requirements Phase 318
9.14 Osbert Oglesby Case Study: Requirements Phase 318
9.15 Osbert Oglesby Case Study: Rapid Prototype 321
9.16 Object-Oriented Requirements? 324
Chapter Review 324
For Further Reading 325
Problems 326
References 327

CHAPTER 10
Specification Phase 329

10.1 The Specification Document 329
10.2 Informal Specifications 331
10.2.1 Case Study: Text Processing 332
10.3 Structured Systems Analysis 333
10.3.1 Sally's Software Shop 333
10.4 Other Semiformal Techniques 341
10.5 Entity-Relationship Modeling 342
10.6 Finite State Machines 344
10.6.1 Elevator Problem: Finite State Machines 346
10.7 Petri Nets 351
10.7.1 Elevator Problem: Petri Nets 355
10.8 Z 357
10.8.1 Elevator Problem: Z 358
10.8.2 Analysis of Z 360
10.9 Other Formal Techniques 362
10.10 Comparison of Specification Techniques 363
10.11 Testing during the Specification Phase 364
10.12 CASE Tools for the Specification Phase 365
10.13 Metrics for the Specification Phase 366
10.14 Osbert Oglesby Case Study: Structured Systems Analysis 366
10.15 Osbert Oglesby Case Study: Software Project Management Plan 367
Chapter Review 367
For Further Reading 368
Problems 369
References 372

CHAPTER 11
Object-Oriented Analysis Phase 376

11.1 Object-Oriented versus Structured Paradigm 376
11.2 Object-Oriented Analysis 378
CONTENTS

11.3  Elevator Problem: Object-Oriented Analysis 380
11.4  Use-Case Modeling 381
11.5  Class Modeling 382
   11.5.1  Noun Extraction 382
   11.5.2  CRC Cards 385
11.6  Dynamic Modeling 387
11.7  Testing during the Object-Oriented Analysis Phase 388
11.8  CASE Tools for the Object-Oriented Analysis Phase 391
11.9  Osbert Oglesby Case Study: Object-Oriented Analysis 393
11.10 Osbert Oglesby Case Study: Software Project Management Plan 398

Chapter Review 399
For Further Reading 399
Problems 400
References 401

CHAPTER 12
Design Phase 403

12.1  Design and Abstraction 403
12.2  Action-Oriented Design 404
12.3  Data Flow Analysis 405
   12.3.1  Data Flow Analysis Example 406
   12.3.2  Extensions 411
12.4  Transaction Analysis 411
12.5  Data-Oriented Design 413
12.6  Object-Oriented Design 414
12.7  Elevator Problem: Object-Oriented Design 414
12.8  Formal Techniques for Detailed Design 420
12.9  Real-Time Design Techniques 422
12.10 Testing during the Design Phase 423
12.11 CASE Tools for the Design Phase 424
12.12 Metrics for the Design Phase 425
12.13 Osbert Oglesby Case Study: Object-Oriented Design 426

Chapter Review 431
For Further Reading 431

CHAPTER 13
Implementation Phase 437

13.1  Choice of Programming Language 437
13.2  Fourth Generation Languages 440
13.3  Good Programming Practice 443
13.4  Coding Standards 449
13.5  Module Reuse 450
13.6  Module Test Case Selection 451
   13.6.1  Testing to Specifications versus Testing to Code 451
   13.6.2  Feasibility of Testing to Specifications 451
   13.6.3  Feasibility of Testing to Code 452
13.7  Black-Box Module-Testing Techniques 455
   13.7.1  Equivalence Testing and Boundary Value Analysis 455
   13.7.2  Functional Testing 456
13.8  Glass-Box Module-Testing Techniques 457
   13.8.1  Structural Testing: Statement, Branch, and Path Coverage 458
   13.8.2  Complexity Metrics 459
13.9  Code Walkthroughs and Inspections 462
13.10 Comparison of Module-Testing Techniques 462
13.11 Cleanroom 463
13.12 Potential Problems When Testing Objects 464
13.13 Management Aspects of Module Testing 467
13.14 When to Rewrite Rather than Debug a Module 467
13.15 CASE Tools for the Implementation Phase 469
13.16 Osbert Oglesby Case Study: Black-Box Test Cases 469

Chapter Review 471
For Further Reading 471
Problems 472
References 474
APPENDIX D
Osbert Oglesby Case Study:
Structured Systems Analysis 530

APPENDIX E
Osbert Oglesby Case Study:
Object-Oriented Analysis 534

APPENDIX F
Osbert Oglesby Case Study:
Software Project
Management Plan 535

APPENDIX G
Osbert Oglesby Case Study:
Design 540

APPENDIX H
Osbert Oglesby Case Study:
Black-Box Test Cases 559

APPENDIX I
Osbert Oglesby Case Study:
Complete Source Code 563

Bibliography 564

Author Index 597

Subject Index 603