

CONTENTS

Preface vii

PART 1

Introduction to the Software Life Cycle 1

CHAPTER 1 Scope of Software Engineering 3

- 1.1 Historical Aspects 5
- 1.2 Economic Aspects 7
- 1.3 Maintenance Aspects 8
- 1.4 Specification and Design Aspects 12
- 1.5 Team Programming Aspects 15
- 1.6 The Object-Oriented Paradigm 16
- 1.7 Terminology 21
- Chapter Review 23
- For Further Reading 24
- Problems 25
- References 26

CHAPTER 2 The Software Process 30

- 2.1 Client, Developer, and User 32
- 2.2 Requirements Phase 33
 - 2.2.1 Requirements Phase Testing 34
- 2.3 Specification Phase 35
 - 2.3.1 Specification Phase Testing 37
- 2.4 Design Phase 37
 - 2.4.1 Design Phase Testing 39
- 2.5 Implementation Phase 39
 - 2.5.1 Implementation Phase Testing 39
- 2.6 Integration Phase 40
 - 2.6.1 Integration Phase Testing 40
- 2.7 Maintenance Phase 41
 - 2.7.1 Maintenance Phase Testing 41
- 2.8 Retirement 42

2.9 Problems with Software Production:
Essence and Accidents 43

- 2.9.1 Complexity 44
- 2.9.2 Conformity 46
- 2.9.3 Changeability 47
- 2.9.4 Invisibility 48
- 2.9.5 No Silver Bullet? 49
- 2.10 Improving the Software Process 50
- 2.11 Capability Maturity Models 50
- 2.12 ISO 9000 54
- 2.13 SPICE 55
- 2.14 Costs and Benefits of Software Process Improvement 56
- Chapter Review 57
- For Further Reading 58
- Problems 59
- References 60

CHAPTER 3 Software Life-Cycle Models 64

- 3.1 Build-and-Fix Model 64
- 3.2 Waterfall Model 65
 - 3.2.1 Analysis of the Waterfall Model 68
- 3.3 Rapid Prototyping Model 70
 - 3.3.1 Integrating the Waterfall and Rapid Prototyping Models 72
- 3.4 Incremental Model 72
 - 3.4.1 Analysis of the Incremental Model 74
- 3.5 Synchronize-and-Stabilize Model 77
- 3.6 Spiral Model 77
 - 3.6.1 Analysis of the Spiral Model 82
- 3.7 Object-Oriented Life-Cycle Models 83
- 3.8 Comparison of Life-Cycle Models 84
- Chapter Review 86
- For Further Reading 86
- Problems 87
- References 88

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

6.3.3	Control Coupling	184
6.3.4	Stamp Coupling	185
6.3.5	Data Coupling	186
6.3.6	Coupling Example	186
6.3.7	The Importance of Coupling	188
6.4	Data Encapsulation	189
6.4.1	Data Encapsulation and Product Development	192
6.4.2	Data Encapsulation and Product Maintenance	194
6.5	Abstract Data Types	198
6.6	Information Hiding	201
6.7	Objects	203
6.8	Inheritance, Polymorphism, and Dynamic Binding	207
6.9	Cohesion and Coupling of Objects	209
	Chapter Review	210
	For Further Reading	210
	Problems	211
	References	213

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

- 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

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

Problems	432
References	433

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

CHAPTER 14**Implementation and
Integration Phase 479**

- 14.1 Implementation and Integration 479
 - 14.1.1 Top-Down Implementation and Integration 480
 - 14.1.2 Bottom-Up Implementation and Integration 482
 - 14.1.3 Sandwich Implementation and Integration 483
 - 14.1.4 Implementation and Integration of Object-Oriented Products 485
 - 14.1.5 Management Issues during the Implementation and Integration Phase 485
- 14.2 Testing during the Implementation and Integration Phase 486
- 14.3 Integration Testing of Graphical User Interfaces 486
- 14.4 Product Testing 487
- 14.5 Acceptance Testing 488
- 14.6 CASE Tools for the Implementation and Integration Phase 489
- 14.7 CASE Tools for the Complete Software Process 490
- 14.8 Integrated Environments 490
 - 14.8.1 Process Integration 490
 - 14.8.2 Tool Integration 491
 - 14.8.3 Other Forms of Integration 494
- 14.9 Environments for Business Applications 494
- 14.10 Public Tool Infrastructures 495
- 14.11 Potential Problems with Environments 495
- 14.12 Metrics for the Implementation and Integration Phase 496
- 14.13 Osbert Oglesby Case Study: Implementation and Integration Phase 497
- Chapter Review 498
- For Further Reading 498
- Problems 499
- References 500

CHAPTER 15**Maintenance Phase 502**

- 15.1 Why Maintenance Is Necessary 502
- 15.2 What Is Required of Maintenance Programmers 503
- 15.3 Maintenance Case Study 505
- 15.4 Management of Maintenance 507
 - 15.4.1 Fault Reports 507
 - 15.4.2 Authorizing Changes to the Product 508
 - 15.4.3 Ensuring Maintainability 509
 - 15.4.4 Problem of Repeated Maintenance 509
- 15.5 Maintenance of Object-Oriented Software 510
- 15.6 Maintenance Skills versus Development Skills 514
- 15.7 Reverse Engineering 514
- 15.8 Testing during the Maintenance Phase 515
- 15.9 CASE Tools for the Maintenance Phase 516
- 15.10 Metrics for the Maintenance Phase 517
- 15.11 Osbert Oglesby Case Study: Maintenance 517
- Chapter Review 518
- For Further Reading 519
- Problems 519
- References 520

APPENDIX A**Air Gourmet 523****APPENDIX B****Software Engineering
Resources 526****APPENDIX C****Osbert Oglesby Case Study:
Rapid Prototype 529**

APPENDIX D Osbert Oglesby Case Study: Structured Systems Analysis	530	APPENDIX H Osbert Oglesby Case Study: Black-Box Test Cases	559
APPENDIX E Osbert Oglesby Case Study: Object-Oriented Analysis	534	APPENDIX I Osbert Oglesby Case Study: Complete Source Code	563
APPENDIX F Osbert Oglesby Case Study: Software Project Management Plan	535	Bibliography	564
APPENDIX G Osbert Oglesby Case Study: Design	540	Author Index	597
		Subject Index	603