

Object-Oriented Software Engineering

Stephen R. Schach

Vanderbilt University



Higher Education

Boston Burr Ridge, IL Dubuque, IA New York San Francisco St. Louis
Bangkok, Bogota Caracas Kuala Lumpur Lisbon London Madrid Mexico City
Milan Montreal New Delhi Santiago Seoul Singapore Sydney Taipei Toronto

Contents

PART ONE

INTRODUCTION TO OBJECT-ORIENTED SOFTWARE ENGINEERING 1

Chapter 1

The Scope of Object-Oriented Software Engineering 3

- Learning Objectives 3
- 1.1 Historical Aspects 4
- 1.2 Economic Aspects 7
- 1.3 Maintenance Aspects 8
 - 1.3.1 *The Modern View of Maintenance* 9
 - 1.3.2 *The Importance of Post-delivery Maintenance* 11
- 1.4 Requirements, Analysis, and Design Aspects 13
- 1.5 Team Development Aspects 15
- 1.6 Why There Is No Planning Phase 16
- 1.7 Why There Is No Testing Phase 17
- 1.8 Why There Is No Documentation Phase 18
- 1.9 The Object-Oriented Paradigm 18
- 1.10 Terminology 20
- 1.11 Ethical Issues 24
- Chapter Review 25
- For Further Reading 25
- Key Terms 26
- Problems 27
- References 28

Chapter 2

Software Life-Cycle Models 32

- Learning Objectives 32
- 2.1 Software Development in Theory 32
- 2.2 Winburg Mini Case Study 33
- 2.3 Lessons of the Winburg Mini Case Study 37
- 2.4 Teal Tractors Mini Case Study 37
- 2.5 Iteration and Incrementation 38
- 2.6 Winburg Mini Case Study Revisited 42
- 2.7 Risks and Other Aspects of Iteration and Incrementation 43

- 2.8 Managing Iteration and Incrementation 46
- 2.9 Other Life-Cycle Models 47
 - 2.9.1 *Code-and-Fix Life-Cycle Model* 47
 - 2.9.2 *Waterfall Life-Cycle Model* 48
 - 2.9.3 *Rapid-Prototyping Life-Cycle Model* 50
 - 2.9.4 *Open-Source Life-Cycle Model* 51
 - 2.9.5 *Agile Processes* 54
 - 2.9.6 *Synchronize-and-Stabilize Life-Cycle Model* 57
 - 2.9.7 *Spiral Life-Cycle Model* 57
- 2.10 Comparison of Life-Cycle Models 61
- Chapter Review 62
- For Further Reading 63
- Key Terms 64
- Problems 64
- References 65

Chapter 3

The Software Process 68

- Learning Objectives 68
- 3.1 The Unified Process 70
- 3.2 Iteration and Incrementation 72
- 3.3 The Requirements Workflow 73
- 3.4 The Analysis Workflow 74
- 3.5 The Design Workflow 76
- 3.6 The Implementation Workflow 77
- 3.7 The Test Workflow 78
 - 3.7.1 *Requirements Artifacts* 78
 - 3.7.2 *Analysis Artifacts* 79
 - 3.7.3 *Design Artifacts* 79
 - 3.7.4 *Implementation Artifacts* 79
- 3.8 Postdelivery Maintenance 81
- 3.9 Retirement 82
- 3.10 The Phases of the Unified Process 82
 - 3.10.1 *The Inception Phase* 83
 - 3.10.2 *The Elaboration Phase* 85
 - 3.10.3 *The Construction Phase* 86
 - 3.10.4 *The Transition Phase* 86
- 3.11 One- versus Two-Dimensional Life-Cycle Models 87
- 3.12 Improving the Software Process 89
- 3.13 Capability Maturity Models 89

- 3.14 Other Software Process Improvement Initiatives 92
- 3.15 Costs and Benefits of Software Process Improvement 93
 - Chapter Review 95
 - For Further Reading 95
 - Key Terms 96
 - Problems 97
 - References 97

Chapter 4 Teams 101

- Learning Objectives 101
- 4.1 Team Organization 101
- 4.2 Democratic Team Approach 103
 - 4.2.1 *Analysis of the Democratic Team Approach* 104
- 4.3 Chief Programmer Team Approach 104
 - 4.3.1 *The New York Times Project* 106
 - 4.3.2 *Impracticality of the Chief Programmer Team Approach* 107
- 4.4 Beyond Chief Programmer and Democratic Teams 107
- 4.5 Synchronize-and-Stabilize Teams 111
- 4.6 Teams for Agile Processes 112
- 4.7 Open-Source Programming Teams 112
- 4.8 People Capability Maturity Model 113
- 4.9 Choosing an Appropriate Team Organization 114
 - Chapter Review 115
 - For Further Reading 115
 - Key Terms 115
 - Problems 116
 - References 116

Chapter 5 The Tools of The Trade 118

- Learning Objectives 118
- 5.1 Stepwise Refinement 118
 - 5.1.1 *Stepwise Refinement Mini Case Study* 119
- 5.2 Cost-Benefit Analysis 124
- 5.3 Software Metrics 126
- 5.4 CASE 127

- 5.5 Taxonomy of CASE 128
- 5.6 Scope of CASE 130
- 5.7 Software Versions 133
 - 5.7.1 *Revisions* 134
 - 5.7.2 *Variations* 134
- 5.8 Configuration Control 135
 - 5.8.1 *Configuration Control during Postdelivery Maintenance* 137
 - 5.8.2 *Baselines* 138
 - 5.8.3 *Configuration Control during Development* 138
- 5.9 Build Tools 138
- 5.10 Productivity Gains with CASE Technology 139
 - Chapter Review 141
 - For Further Reading 141
 - Key Terms 141
 - Problems 142
 - References 143

Chapter 6 Testing 145

- Learning Objectives 145
- 6.1 Quality Issues 146
 - 6.1.1 *Software Quality Assurance* 147
 - 6.1.2 *Managerial Independence* 147
- 6.2 Non-Execution-Based Testing 148
 - 6.2.1 *Walkthroughs* 149
 - 6.2.2 *Managing Walkthroughs* 149
 - 6.2.3 *Inspections* 150
 - 6.2.4 *Comparison of Inspections and Walkthroughs* 152
 - 6.2.5 *Strengths and Weaknesses of Reviews* 153
 - 6.2.6 *Metrics for Inspections* 153
- 6.3 Execution-Based Testing 153
- 6.4 What Should Be Tested? 154
 - 6.4.1 *Utility* 155
 - 6.4.2 *Reliability* 155
 - 6.4.3 *Robustness* 156
 - 6.4.4 *Performance* 156
 - 6.4.5 *Correctness* 157
- 6.5 Testing versus Correctness Proofs 158
 - 6.5.1 *Example of a Correctness Proof* 158
 - 6.5.2 *Correctness Proof Mini Case Study* 162

6.5.3	<i>Correctness Proofs and Software Engineering</i>	163
6.6	Who Should Perform Execution-Based Testing?	166
6.7	When Testing Stops	167
	Chapter Review	167
	For Further Reading	168
	Key Terms	168
	Problems	169
	References	170
 Chapter 7		
From Modules to Objects 173		
	Learning Objectives	173
7.1	What Is a Module?	174
7.2	Cohesion	176
	7.2.1 Coincidental Cohesion	177
	7.2.2 Logical Cohesion	178
	7.2.3 Temporal Cohesion	178
	7.2.4 Procedural Cohesion	179
	7.2.5 Communicational Cohesion	179
*	7.2.6 Functional Cohesion	180
	7.2.7 Informational Cohesion	180
	7.2.8 Cohesion Example	181
7.3	Coupling	181
	7.3.1 Content Coupling	182
	7.3.2 Common Coupling	183
	7.3.3 Control Coupling	185
	7.3.4 Stamp Coupling	185
	7.3.5 Data Coupling	186
	7.3.6 Coupling Example	187
	7.3.7 The Importance of Coupling	188
7.4	Data Encapsulation	189
	7.4.1 Data Encapsulation and Development	191
	7.4.2 Data Encapsulation and Maintenance	192
7.5	Abstract Data Types	197
7.6	Information Hiding	199
7.7	Objects	201
7.8	Inheritance, Polymorphism, and Dynamic Binding	205
7.9	The Object-Oriented Paradigm	207
	Chapter Review	210
	For Further Reading	211
	Key Terms	211

	Problems	212
	References	212
 Chapter 8		
Reusability and Portability 215		
	Learning Objectives	215
8.1	Reuse Concepts	216
8.2	Impediments to Reuse	218
8.3	Reuse Case Studies	219
	8.3.1 Raytheon Missile Systems Division	220
	8.3.2 European Space Agency	221
8.4	Objects and Reuse	222
8.5	Reuse during Design and Implementation	222
	8.5.1 Design Reuse	222
	8.5.2 Application Frameworks	224
	8.5.3 Design Patterns	224
	8.5.4 Software Architecture	224
	8.5.5 Component-Based Software Engineering	221
8.6	More on Design Patterns	227
	8.6.1 FLIC Mini Case Study	228
	8.6.2 Adapter Design Pattern	229
	8.6.3 Bridge Design Pattern	230
	8.6.4 Iterator Design Pattern	233
	8.6.5 Abstract Factory Design Pattern	233
8.7	Categories of Design Patterns	235
8.8	Strengths and Weaknesses of Design Patterns	237
8.9	Reuse and Postdelivery Maintenance	238
8.10	Portability	239
	8.10.1 Hardware Incompatibilities	239
	8.10.2 Operating System X Incompatibilities	240
	8.10.3 Numerical Software Incompatibilities	241
	8.10.4 Compiler Incompatibilities	241
8.11	Why Portability?	244
8.12	Techniques for Achieving Portability	245
	8.12.1 Portable System Software	246
	8.12.2 Portable Application Software	246
	8.12.3 Portable Data	241
	8.12.4 Web-Based Applications	248
	Chapter Review	249
	For Further Reading	249

13.3	Coding Standards	422	
13.4	Code Reuse	423	
13.5	Integration	423	
	13.5.1 Top-down Integration	424	
	13.5.2 Bottom-up Integration	426	
	13.5.3 Sandwich Integration	426	
	13.5.4 Integration Techniques	428	
	13.5.5 Management of Integration	428	
13.6	The Implementation Workflow	429	
13.7	The Implementation Workflow: The MSG Foundation Case Study	429	
13.8	The Test Workflow: Implementation	429	
13.9	Test Case Selection	430	
	13.9.1 Testing to Specifications versus Testing to Code	430	
	13.9.2 Feasibility of Testing to Specifications	430	
	13.9.3 Feasibility of Testing to Code	431	
13.10	Black-Box Unit-Testing Techniques	433	
	13.10.1 Equivalence Testing and Boundary Value Analysis	434	
	13.10.2 Functional Testing	435	
13.11	Black-Box Test Cases: The MSG Foundation Case Study	436	
13.12	Glass-Box Unit-Testing Techniques	436	
	13.12.1 Structural Testing: Statement, Branch, and Path Coverage	438	
	13.12.2 Complexity Metrics	440	
13.13	Code Walkthroughs and Inspections	441	
13.14	Comparison of Unit-Testing Techniques	441	
13.15	Cleanroom	442	
13.16	Testing Issues	443	
13.17	Management Aspects of Unit Testing	445	
13.18	When to Rewrite Rather than Debug a Code Artifact	446	
13.19	Integration Testing	447	
13.20	Product Testing	448	
13.21	Acceptance Testing	449	
13.22	The Test Workflow: The MSG Foundation Case Study	450	
13.23	CASE Tools for Implementation	450	
	13.23.7 CASE Tools for the Complete Software Process	450	
	13.23.2 Integrated Development Environments	451	
	13.23.3 Environments for Business Applications	452	
	13.23.4 Public Tool Infrastructures	452	
	13.23.5 Potential Problems with Environments	452	
13.24	CASE Tools for the Test Workflow	453	
13.25	Metrics for the Implementation Workflow	453	
13.26	Challenges of the Implementation Workflow	454	
	Chapter Review	455	
	For Further Reading	455	
	Key Terms	456	
	Problems	457	J
	References	459	
Chapter 14			
Postdelivery Maintenance 462			
	Learning Objectives	462	
14.1	Development and Maintenance	462	
14.2	Why Postdelivery Maintenance Is Necessary	464	
14.3	What Is Required of Postdelivery Maintenance Programmers?	465	
14.4	Postdelivery Maintenance Mini Case Study	467	
14.5	Management of Postdelivery Maintenance	468	
	14.5.7 Defect Reports	468	
	14.5.2 Authorizing Changes to the Product	469	
	14.5.3 Ensuring Maintainability	470	
	14.5.4 Problem of Repeated Maintenance	471	
14.6	Maintenance Issues	471	
14.7	Postdelivery Maintenance Skills versus Development Skills	474	
14.8	Reverse Engineering	474	
14.9	Testing during Postdelivery Maintenance	475	
14.10	CASE Tools for Postdelivery Maintenance	476	
14.11	Metrics for Postdelivery Maintenance	477	
14.12	Postdelivery Maintenance: The MSG Foundation Case Study	477	
14.13	Challenges of Postdelivery Maintenance	477	

Chapter Review	478
For Further Reading	478
Key Terms	479
Problems	479
References	480

Chapter 15

More on UML 482

Learning Objectives	482
15.1 UML Is <i>Not</i> a Methodology	483
15.2 Class Diagrams	483
15.2.1 <i>Aggregation</i>	484
15.2.2 <i>Multiplicity</i>	485
15.2.3 <i>Composition</i>	486
15.2.4 <i>Generalization</i>	487
15.2.5 <i>Association</i>	487
15.3 Notes	488
15.4 Use-Case Diagrams	488
15.5 Stereotypes	488
15.6 Interaction Diagrams	489
15.7 Statecharts	491
15.8 Activity Diagrams	494
15.9 Packages	496
15.10 Component Diagrams	497
15.11 Deployment Diagrams	497
15.12 Review of UML Diagrams	498
15.13 UML and Iteration	498
Chapter Review	498
For Further Reading	499
Key Terms	499
Problems	499
References	500

Bibliography 501

Appendix A

Term Project: Osric's Office Appliances
and Decor 524

Appendix B

Software Engineering Resources 528

Appendix C

The Requirements Workflow: The MSG
Foundation Case Study 530

Appendix D

The Analysis Workflow: The MSG
Foundation Case Study 531

Appendix E

Software Project Management Plan:
The MSG Foundation Case Study 532

Appendix F

The Design Workflow: The MSG
Foundation Case Study 537

Appendix C

The Implementation Workflow: The MSG
Foundation Case Study (C++ Version) 542

Appendix H

The Implementation Workflow: The MSG
Foundation Case Study (Java Version) 543

Appendix I

The Test Workflow: The MSG Foundation
Case Study 544 !