

Contents

Part I What is Software Engineering?

1	Introduction to Software Engineering	3
1.1	Why Do We Need Software Engineering?	3
1.1.1	Brief Historical Summary	3
1.1.2	Success Rate of Software.	4
1.1.3	The Complexity of Software	4
1.2	The Software Engineering Project.	5
1.2.1	Defining the Software Engineering Project	5
1.2.2	Who is Involved in a Software Engineering Project?	5
1.3	Software Engineering Activities	6
1.3.1	Planning	6
1.3.2	Problem Solving.	7
1.3.3	Modeling.	8
1.3.4	Communication	8
1.3.5	Using and Managing Resources	9
1.3.6	Reaching Milestones and Producing Deliverables	9
1.3.7	Maintaining a Product.	9
1.4	Software Engineering is Not Limited to Programming	10
1.4.1	Computer Science Related	10
1.4.2	Business Related.	10
1.4.3	Psychology Related.	11
1.4.4	Engineering Related	11
1.5	Software Life-Cycles.	11
1.5.1	Understanding the Software Development Process	11
1.5.2	Evaluation of the Waterfall Model	12
1.5.3	Evaluation of the Spiral Model.	14
1.6	Chapter Conclusion and Summary	15
1.7	Exercises	16
	References	16

2	Object-Oriented Concepts	17
2.1	What is an Object?	18
2.2	Classes	20
2.2.1	Classes Versus Objects	20
2.2.2	The Class-Object Hierarchy	21
2.2.3	Why Use Objects and Classes?	21
2.3	Modularity	21
2.3.1	Reuse	22
2.3.2	Encapsulation and Information Hiding	23
2.3.3	Access Levels	24
2.3.4	Delegation	25
2.4	Inheritance	25
2.4.1	Overloading	27
2.4.2	Overriding	30
2.4.3	Polymorphism	30
2.5	Abstraction	32
2.5.1	Abstract Classes	32
2.5.2	Template	34
2.5.3	Generic Components	35
2.5.4	Interfaces	36
2.6	Chapter Conclusion and Summary	37
2.7	Exercises	37
	References	38
3	Modeling with UML	39
3.1	Introduction to the Unified Modeling Language	39
3.1.1	What is Modeling?	40
3.1.2	What is the Unified Modeling Language?	40
3.2	Object, Classes and Actors	41
3.2.1	Objects	41
3.2.2	Classes	42
3.2.3	Actors	42
3.3	Associations	44
3.3.1	Aggregation	44
3.3.2	Generalization	45
3.3.3	Dependency	45
3.3.4	Uses and Extends	45
3.3.5	Multiplicity	46
3.4	Models	47
3.4.1	Class Diagram	47
3.4.2	Use Case Diagram	48
3.4.3	Sequence Diagram	49
3.4.4	State Diagrams	50
3.4.5	Activity Diagram	51

3.4.6	Collaboration Diagram	53
3.4.7	Component Diagram	54
3.4.8	Deployment Diagram	54
3.5	Interfaces, Notes, and Packages	54
3.5.1	Interfaces	54
3.5.2	Packages	55
3.5.3	Notes	56
3.6	Chapter Conclusion and Summary	56
3.7	Exercises	57
	References	58

Part II The Software Engineering Project

4	Starting the Project	61
4.1	Overview of Project and Planning	61
4.1.1	Identifying the Purpose of the Project	62
4.1.2	Identifying the Deliverables of the Project	63
4.1.3	Cost Estimation	63
4.2	Staying on Task and on Schedule	64
4.2.1	Deadlines	65
4.2.2	Work Breakdown Structures	65
4.2.3	Gantt Chart	67
4.3	Communicating Ideas	67
4.3.1	Assessing Employee Skill Levels	69
4.3.2	Team Organization and Role Assignment	70
4.3.3	Project Communication	71
4.4	Handling Problems and Change	72
4.5	Document Format Specification	73
4.6	Software Project Management Plan	76
4.7	Chapter Summary and Conclusions	78
4.8	Exercises	78
	References	80
5	Requirements Elicitation	81
5.1	What are Requirements?	81
5.1.1	Functional Requirements	81
5.1.2	Non-Functional Requirements or Constraints	82
5.1.3	Domain Requirements	83
5.2	Requirement Elicitation	83
5.2.1	Communicating with the Client	83
5.2.2	Gathering Information	84
5.2.3	Issues in Requirement Elicitation	85

5.3	Identifying the Requirements	86
5.3.1	Identifying the Functional Requirements	86
5.3.2	Identifying the Non-Functional Requirements.	87
5.3.3	Identifying the Application Domain	89
5.4	Common Problems Concerning Requirements	90
5.4.1	Problems of Scope	90
5.4.2	Problems of Understanding	91
5.4.3	Problems of Volatility	93
5.5	Validating the Requirements	93
5.6	Producing the Use Case Model.	94
5.6.1	Structuring Use Cases	95
5.6.2	Use Case Guidelines	95
5.7	The Software Requirements Specification Document	96
5.8	Chapter Summary and Conclusions.	98
5.9	Exercises	99
	References	101
6	Object-Oriented Analysis	103
6.1	Introduction to Analysis	103
6.1.1	What is Object-Oriented Analysis?	104
6.2	Requirements Specification and the Specification Document	105
6.2.1	Evaluating Requirements Specification	106
6.2.2	Refining Requirements Specification Through Prototyping	107
6.2.3	Verifying Requirements Specification	108
6.3	Analysis Modeling Concepts	109
6.3.1	Analysis Object Models.	110
6.3.2	Entity, Boundary, and Control Objects	111
6.4	Scenario-Based Modeling	112
6.5	Class-Based Modeling.	113
6.6	Analysis Process.	113
6.6.1	Identifying Entity, Boundary, and Control Objects	115
6.6.2	Identifying Use Cases	116
6.6.3	Scenario Development.	118
6.6.4	Modeling the System	119
6.6.5	Class Diagrams.	119
6.6.6	Use Case Diagrams.	120
6.7	Issues in Object-Oriented Analysis	121
6.8	Chapter Summary and Conclusions.	122
6.9	Exercises	123
	References	124

7	System Design	125
7.1	Categories of System Design	126
7.1.1	Global-Based Systems	126
7.1.2	Group-Based Systems	126
7.1.3	Local-Based Systems	127
7.2	Function-Oriented Approach	127
7.3	Structure-Oriented Approach	128
7.3.1	Process-Oriented Approach	128
7.3.2	Data-Oriented Approach	128
7.3.3	Creating the Data Dictionary	129
7.4	Object-Oriented Approach	129
7.4.1	Component-Based Design	131
7.4.2	Components and Objects	131
7.4.3	Component Models	131
7.4.4	The Component Concept	132
7.5	Structured Versus Object-Oriented System Design	134
7.5.1	Modularity	134
7.5.2	Top-Down Versus Bottom-Up Design	135
7.5.3	Reusability	135
7.6	Object-Oriented Design Concepts	136
7.6.1	Abstractions	136
7.6.2	Architecture	137
7.6.3	Design Patterns	137
7.6.4	Modularity	137
7.6.5	Encapsulation	138
7.6.6	Refinement	138
7.6.7	Class Design	139
7.6.8	Subsystems and Classes	139
7.6.9	Services and Subsystem Interfaces	140
7.6.10	Coupling and Cohesion	141
7.7	Architectural System Design	141
7.7.1	Data Design	142
7.7.2	Organization and Refinement	143
7.8	Issues in Object-Oriented Analysis	143
7.9	Chapter Summary and Conclusions	144
7.10	Exercises	145
	References	146
8	Object-Oriented Design	147
8.1	Overview of Object-Oriented Design	147
8.1.1	First Steps to OOD	149
8.1.2	Activities in OOD	150

8.2	Object-Oriented Design Concepts	151
8.2.1	Functional and Non-Functional Requirements	151
8.2.2	Types, Signatures and Visibility	151
8.2.3	Object Contracts: Invariants, Preconditions and Postconditions	152
8.3	Reuse Concepts: Objects and Design Patterns	152
8.3.1	Objects	152
8.3.2	Inheritance	153
8.3.3	The Liskov Substitution Principle	153
8.4	Specifying Interfaces	153
8.5	Interface Specification Concepts	154
8.5.1	Class Implementer	154
8.5.2	Object Constraint Language	156
8.5.3	OCL Collections	160
8.5.4	OCL Quantifiers	161
8.6	Managing Object Design	161
8.6.1	Documenting Object Design	161
8.6.2	Roles in OOD	162
8.6.3	The Unified Process	163
8.7	Objects as Models	163
8.7.1	Forward Engineering	164
8.7.2	Reverse Engineering	164
8.8	Design Tips	164
8.9	Chapter Summary and Conclusions	165
8.10	Exercises	166
	References	168
9	Implementation	169
9.1	Introduction to the Implementation Phase	169
9.1.1	Implementation Standards	170
9.1.2	Library Utilization and Management	171
9.1.3	Version Controls	171
9.2	Tasks and Activities	172
9.2.1	Implementation Updates	172
9.2.2	New Model Training Plan	173
9.2.3	Data Entry	174
9.2.4	Post-Implementation Assessment	174
9.2.5	Documenting Updates	174
9.3	Roles and Responsibilities	174
9.4	Deliverables	175
9.4.1	Delivered System	175
9.4.2	Change Notice	175
9.4.3	Version Description	175
9.4.4	Post-Implementation Review	176

9.5	Post-Implementation Considerations	178
9.6	Language Choice	178
9.7	Development Paradigms	179
	9.7.1 Component-Based Development	179
	9.7.2 Extreme Programming.	179
9.8	Code Style Standards	180
9.9	Code Reuse	181
9.10	Integration	181
	9.10.1 Top-Down Integration	182
	9.10.2 Bottom-Up Integration.	183
	9.10.3 Sandwich Integration.	184
	9.10.4 Integration of Object-Oriented Products.	185
	9.10.5 Integration Management	185
9.11	Implementation Workflow	185
	9.11.1 Challenges of Implementation Workflow.	186
	9.11.2 Metrics of Implementation Workflow	186
9.12	Managing Implementation	187
	9.12.1 Documenting Transformations	187
	9.12.2 Assigning Responsibilities	188
9.13	Chapter Summary and Conclusions.	188
9.14	Exercises	189
	References	190
10	Testing.	191
10.1	Introduction to Testing	191
	10.1.1 Objective of Testing	192
	10.1.2 Testing Concepts and Theory.	192
	10.1.3 Test Planning	194
10.2	Quality and Internal Controls	194
	10.2.1 Assuring Quality Software.	194
	10.2.2 Managerial Independence.	195
10.3	Testing Management.	195
10.4	Non-Execution Based Testing	196
	10.4.1 Walkthrough	196
	10.4.2 Managing Walkthroughs	196
	10.4.3 Inspections.	197
10.5	Things to be Tested	198
	10.5.1 Utility	199
	10.5.2 Reliability	199
	10.5.3 Robustness.	199
	10.5.4 Performance.	200
	10.5.5 Correctness	200
	10.5.6 Usability Testing	200
	10.5.7 System Integration	201

10.6	Mathematically Proving Correctness	201
10.7	Execution-Based Testing	202
10.7.1	Who Should Perform Execution-Based Testing? . . .	203
10.8	Levels of Testing	203
10.8.1	Systems Testing	204
10.8.2	Web Application Testing	204
10.9	Unit Testing	205
10.10	Acceptance Testing	206
10.10.1	Alpha Testing	206
10.10.2	Beta Testing	208
10.11	UML Model Testing	209
10.12	Testing for Object-Oriented Systems	211
10.13	Testing in a Box	211
10.13.1	Black-Box Testing	212
10.13.2	White-Box Testing	212
10.14	Testing Alternatives	213
10.15	When to Release the Software	213
10.16	Chapter Summary and Conclusions	214
10.17	Exercises	215
	References	215
11	Project Wrap-Up, Delivery, and Maintenance	217
11.1	Project Management and Success Criteria	217
11.1.1	Define Project Success Criteria	217
11.1.2	Define Business Objectives	218
11.1.3	Identify Project Constraints	222
11.1.4	Derive Project Success Criteria	223
11.2	Project Termination and Release	225
11.3	Project Wrap-Up and Result Presentation	225
11.3.1	Postmortem Review	226
11.3.2	Release Management	227
11.4	Development and Maintenance	229
11.5	Why Post-Delivery Maintenance is Necessary	229
11.6	What is Required of Post-Delivery Maintenance Programmers?	229
11.7	Managing Post-Delivery Maintenance	230
11.7.1	Defect Reports	230
11.7.2	Authorizing Changes to the Product	231
11.7.3	Ensuring Maintainability	231
11.7.4	Problem of Repeated Maintenance	231
11.8	Maintenance of Object-Oriented Software	232
11.9	Post-Delivery Maintenance Skill Versus Development Skills	232
11.10	Reverse Engineering	233

11.11	Agile Modeling and Extreme Programming	234
11.12	Testing During Post-Delivery Maintenance	235
11.13	Metrics and Challenges of Post-Delivery Maintenance	235
11.14	Chapter Summary and Conclusions.	236
11.15	Exercises	237
	References	237
12	Software Metrics and Measurements.	239
12.1	Theory and Practice	239
12.1.1	Challenges Using and Understanding Metrics.	240
12.1.2	Properties of a Good Measurement or Metric.	241
12.1.3	Etiquette	241
12.1.4	Private Versus Public Metrics	242
12.1.5	Baseline Measurements	242
12.1.6	Attributes of a Good Metric.	243
12.1.7	Establishing Uniform Measures	244
12.2	Quality Metrics	244
12.2.1	Garvin's Quality Dimensions	245
12.2.2	McCall's Quality Factors.	246
12.2.3	ISO 9126 Quality Factors	247
12.2.4	The Quantitative View	247
12.3	Design Metrics	247
12.3.1	Interface	248
12.3.2	Web Design Evaluation.	248
12.3.3	Object-Oriented Design Metrics	249
12.3.4	Architectural Design Evaluation	250
12.4	Object-Oriented Metrics	250
12.5	Project Metrics	251
12.5.1	Use-Case Metrics	252
12.5.2	Size Metrics.	252
12.6	Process Metrics	252
12.7	Post Release Metrics.	253
12.8	Chapter Summary and Conclusions.	253
12.9	Exercises	254
	References	254
13	Hands-On Software Engineering Project	255
13.1	Phase I: Team Composition and Problem Definition.	255
13.1.1	Team Composition	256
13.1.2	Writing for Software Engineering.	257
13.1.3	Time and Resource Management	258
13.2	Phase II: Object-Oriented Software Requirements Specification	261
13.2.1	Specification Documentation	264
13.2.2	Change Management.	265

13.3	Phase III: Object Oriented System Design	266
13.3.1	Stake Holders and Interests List	266
13.3.2	Use Case	266
13.3.3	Use Case Diagrams	268
13.3.4	Class and Object Diagrams	268
13.4	Phase IV: Implementation	271
13.4.1	Commenting Code	272
13.5	Phase V: System Integration and Testing	273
13.5.1	Testing	273
13.5.2	Quality Testing	274
13.5.3	Performance Testing	275
13.5.4	Reporting Test Results	275
13.5.5	When and How to Comprise Time and Functionality	276
13.6	Phase VI: System Demonstration	277
13.6.1	What to Bring	277
13.6.2	The Demonstration	278
13.6.3	Presenting	278
13.7	Phase VII: Product Delivery With Documentation	279
13.7.1	Peer Reviews	279
13.8	Chapter Summary and Conclusions	280
13.9	Exercises	280
	References	281
	About the Author	283
	Index	285



<http://www.springer.com/978-94-6239-005-8>

Software Engineering: A Hands-On Approach

Lee, R.Y.

2013, XXIV, 288 p. 70 illus., 3 illus. in color., Hardcover

ISBN: 978-94-6239-005-8

A product of Atlantis Press