

Contents

	<i>Preface</i>	xv
1	Introduction	1
	1.1 What Is Software Engineering?	1
	1.1.1 Definition of Software Engineering	2
	1.1.2 A Tale of Two Companies	4
	1.2 The Requirements Challenge	5
	1.2.1 Identifying Users and Requirements	6
	1.2.2 Dealing with Requirements Changes	7
	1.3 Software Is Inherently Complex	8
	1.3.1 Sources of Complexity	8
	1.3.2 Architecture: Dealing with Program Complexity	9
	1.4 Defects Are Inevitable	11
	1.4.1 Fix Faults to Avoid Failures	11
	1.4.2 Introduction to Testing	12
	1.4.3 Black-Box and White-Box Testing	13
	1.5 Balancing Constraints: The Iron Triangle	13
	1.5.1 Scope. Cost. Time. Pick Any Two!	14
	1.6 Social Responsibility	15
	1.6.1 Case Study: The Volkswagen Emissions Scandal	15
	1.6.2 The ACM Code	15
	1.6.3 Case Study: The Therac-25 Accidents	16
	1.6.4 Lessons for Software Projects	17
	1.7 Conclusion	19
	Further Reading	20
	Exercises	20
2	Software Development Processes	24
	2.1 Processes and Values Guide Development	24
	2.1.1 What Is a Process?	25
	2.1.2 Two Development Cultures: Plan versus Grow	26
	2.1.3 Role Models: Unix Culture and Agile Values	28
	2.1.4 Selecting a Process Model	30

2.2	Structuring Teamwork: The Scrum Framework	32
2.2.1	Overview of Scrum	32
2.2.2	Scrum Roles	34
2.2.3	Scrum Events	34
2.2.4	Scrum Artifacts	36
2.2.5	Summary	37
2.3	Agile Development: Extreme Programming	37
2.3.1	Listening to Customers: User Stories	38
2.3.2	Testing: Make It Central to Development	39
2.3.3	When to Design?	41
2.3.4	A Scrum+XP Hybrid	43
2.3.5	Summary	44
2.4	Limitations of Waterfall Processes	44
2.4.1	The Perils of Big-Bang Integration and Testing	45
2.4.2	The Waterfall Cost-of-Change Curve	46
2.4.3	Managing the Risks of Waterfall Processes	47
2.4.4	Summary	49
2.5	Levels of Design and Testing: V Processes	49
2.5.1	Overview of V Processes	50
2.5.2	Levels of Testing, from Unit to Acceptance	51
2.5.3	Summary	51
2.6	Additional Project Risks	51
2.6.1	Rough Risk Assessment	53
2.6.2	Netscape 3.0: A Successful Iterative Project	53
2.6.3	Netscape 4.0: A Troubled Project	54
2.7	Risk Reduction: The Spiral Framework	55
2.7.1	Overview of the Spiral Framework	56
2.7.2	Summary	58
2.8	Conclusion	58
	Further Reading	60
	Exercises	60
3	User Requirements	64
3.1	What Is a Requirement?	64
3.1.1	The Basic Requirements Cycle	65
3.1.2	Case Study: Requirements Challenges	66
3.1.3	Kinds of Requirements	67
3.2	Developing Requirements and Software	69
3.2.1	Agile Methods Validate Working Software	69
3.2.2	Case Study: An Agile Emphasis on Requirements	70
3.2.3	Plan-Driven Methods Validate a Specification	72
3.3	Eliciting User Needs	73
3.3.1	A Classification of Needs	73
3.3.2	Accessing User Needs	75

		76
3.3.3	Case Study: Intuit’s Design for Delight	76
3.4	Writing Requirements: Stories and Features	77
3.4.1	Guidelines for Effective User Stories	77
3.4.2	Guidelines for System Features	80
3.4.3	Perspective on User Stories	82
3.5	Writing User-Experience Scenarios	83
3.5.1	Guidelines for User-Experience Scenarios	83
3.5.2	Case Study: A Medical Scenario	84
3.6	Clarifying User Goals	85
3.6.1	Properties of Goals	86
3.6.2	Asking Clarifying Questions	87
3.6.3	Organizing Goals into Hierarchies	89
3.6.4	Contributing and Conflicting Goals	91
3.7	Identifying Security Attacks	93
3.7.1	Attack Trees: Think Like an Attacker	93
3.7.2	Finding Possible Attacks	93
3.8	Conclusion	95
	Further Reading	97
	Exercises	97
4	Requirements Analysis	100
4.1	A Checklist Approach	100
4.2	Relative Estimation: Iteration Planning	103
4.2.1	Anchoring Can Bias Decisions	103
4.2.2	Agile Story Points	104
4.2.3	Velocity of Work	106
4.3	Structured Group Consensus Estimates	106
4.3.1	Wideband Delphi and Planning Poker	107
4.3.2	The Original Delphi Method	108
4.4	Balancing Priorities	110
4.4.1	Must-Should-Could-Won’t (MoSCoW) Prioritization	110
4.4.2	Balancing Value and Cost	111
4.4.3	Balancing Value, Cost, and Risk	112
4.5	Customer Satisfiers and Dissatisfiers	113
4.5.1	Kano Analysis	114
4.5.2	Classification of Features	115
4.5.3	Life Cycles of Attractiveness	117
4.5.4	Degrees of Sufficiency	118
4.6	Plan-Driven Estimation Models	118
4.6.1	How Are Size and Effort Related?	119
4.6.2	The Cocomo Family of Estimation Models	121
4.7	Conclusion	122
	Further Reading	122
	Exercises	123

5	Use Cases	125
5.1	Elements of a Use Case	125
5.1.1	Actors and Goals Outline a System	125
5.1.2	Flows and Basic Flows	127
5.2	Alternative Flows: Conditional Behaviors	127
5.2.1	Specific Alternative Flows	129
5.2.2	Extension Points	130
5.2.3	Bounded Alternative Flows	131
5.3	Writing Use Cases	132
5.3.1	A Template for Use Cases	133
5.3.2	From Actor Intentions to System Interactions	134
5.3.3	How to Build Use Cases	135
5.4	Use-Case Diagrams	136
5.4.1	Diagrams Highlight Goals and Actors	136
5.5	Relationships between Use Cases	137
5.5.1	Subflows	137
5.5.2	Inclusion of Use Cases	137
5.5.3	Extensions of Use Cases	138
5.6	Conclusion	139
	Further Reading	140
	Exercises	140
6	Design and Architecture	142
6.1	The Role of Architecture	142
6.1.1	What Is Architecture?	143
6.1.2	Design Includes Architecture	144
6.1.3	What Is a Good Software Architecture?	144
6.2	Designing Modular Systems	145
6.2.1	The Modularity Principle	145
6.2.2	Coupling and Cohesion	148
6.2.3	Design Guidelines for Modules	149
6.3	Class Diagrams	150
6.3.1	Representing a Class	150
6.3.2	Relationships between Classes	153
6.4	Architectural Views	156
6.4.1	The 4+1 Grouping of Views	156
6.4.2	Structures and Views	160
6.5	Describing System Architecture	162
6.5.1	Outline for an Architecture Description	162
6.5.2	System Overview of a Communications App	164
6.5.3	A Development View: Module Hierarchies	165
6.6	Conclusion	167
	Further Reading	168
	Exercises	168

7	Architectural Patterns	172
	7.1 Software Layering	172
	7.1.1 The Layered Pattern	174
	7.1.2 Design Trade-offs	176
	7.2 Three Building Blocks	178
	7.2.1 The Shared-Data Pattern	179
	7.2.2 Observers and Subscribers	180
	7.3 User Interfaces: Model-View-Controller	181
	7.3.1 Design Decisions	182
	7.3.2 The Basic Model-View-Controller Pattern	183
	7.3.3 Keep Views Simple	185
	7.4 Dataflow Architectures	186
	7.4.1 Dataflow Pipelines	186
	7.4.2 Dataflow Networks	189
	7.4.3 Unbounded Streams	190
	7.4.4 Big Dataflows	191
	7.5 Connecting Clients with Servers	192
	7.5.1 The Client-Server Pattern	193
	7.5.2 Deploying Test Servers	194
	7.5.3 The Broker Pattern	196
	7.6 Families and Product Lines	197
	7.6.1 Commonalities and Variabilities	197
	7.6.2 Software Architecture and Product Lines	198
	7.6.3 Economics of Product-Line Engineering	198
	7.7 Conclusion	199
	Further Reading	199
	Exercises	200
8	Static Checking	203
	8.1 Architecture Reviews	203
	8.1.1 Guiding Principles for Architecture Reviews	204
	8.1.2 Discovery, Deep-Dive, and Retrospective Reviews	206
	8.2 Conducting Software Inspections	207
	8.2.1 The Phases of a Traditional Inspection	207
	8.2.2 Case Study: Using Data to Ensure Effectiveness	209
	8.2.3 Organizing an Inspection	211
	8.3 Code Reviews: Check Intent and Trust	212
	8.3.1 Invested Expert Reviewers	212
	8.3.2 Reviewing Is Done within Hours	213
	8.4 Automated Static Analysis	214
	8.4.1 A Variety of Static Checkers	215
	8.4.2 False Positives and False Negatives	218
	8.5 Conclusion	218

	Further Reading	220
	Exercises	220
9	Testing	222
	9.1 Overview of Testing	222
	9.1.1 Issues during Testing	223
	9.1.2 Test Selection	225
	9.1.3 Test Adequacy: Deciding When to Stop	225
	9.1.4 Test Oracles: Evaluating the Response to a Test	226
	9.2 Levels of Testing	227
	9.2.1 Unit Testing	228
	9.2.2 Integration Testing	229
	9.2.3 Functional, System, and Acceptance Testing	231
	9.2.4 Case Study: Test Early and Often	232
	9.3 Code Coverage I: White-Box Testing	233
	9.3.1 Control-Flow Graphs	233
	9.3.2 Control-Flow Coverage Criteria	235
	9.4 Input Coverage I: Black-Box Testing	238
	9.4.1 Equivalence-Class Coverage	239
	9.4.2 Boundary-Value Coverage	240
	9.5 Code Coverage II: MC/DC	241
	9.5.1 Condition and Decision Coverage Are Independent	242
	9.5.2 MC/DC Pairs of Tests	242
	9.6 Input Coverage II: Combinatorial Testing	245
	9.7 Conclusion	249
	Further Reading	250
	Exercises	250
10	Quality Metrics	253
	10.1 Meaningful Metrics	253
	10.1.1 Metrics Quantify Attributes	254
	10.1.2 Selecting Useful Metrics	255
	10.1.3 Goal-Directed Measurement	257
	10.2 Software Quality	257
	10.2.1 The Many Forms of Software Quality	258
	10.2.2 Measuring Customer Support	260
	10.3 Graphical Displays of Data Sets	261
	10.3.1 Data Sets	261
	10.3.2 Scales of Measurement	262
	10.3.3 Bar Charts Display Data by Category	264
	10.3.4 Gantt Charts Display Schedules	266
	10.4 Product Quality: Measuring Defects	267
	10.4.1 Severity of Defects	268
	10.4.2 Defect-Removal Efficiency	268

	10.4.3 Customer-Found Defects (CFDs)	270
	10.4.4 CFDs Measure Installs, Not Quality	270
10.5	Ops Quality Improvement: A Case Study	272
	10.5.1 How to Improve Software Quality	272
	10.5.2 The Customer Quality Metric	274
	10.5.3 Subgoals: Product and Process Improvement	275
	10.5.4 Measuring Process Improvements	276
10.6	Data Dispersion: Boxplots and Histograms	277
	10.6.1 Medians and Quartiles	277
	10.6.2 Box Plots Summarize Data by Quartile	279
	10.6.3 Histograms of Data Spread	280
10.7	Data Dispersion: Statistics	282
	10.7.1 Variance from the Mean	282
	10.7.2 Discrete Probability Distribution	284
	10.7.3 Continuous Distributions	286
	10.7.4 Introduction to Normal Distributions	286
	10.7.5 Introduction to Student's t-Distributions	287
10.8	Confidence Intervals	287
	10.8.1 Definition of Confidence Interval	288
	10.8.2 If the Population Standard Deviation Is Known	289
	10.8.3 If the Population Standard Deviation Is Unknown	290
10.9	Simple Linear Regression	292
	10.9.1 The Simpler Case: Line through the Origin	293
	10.9.2 Ordinary Least-Squares Fit	294
10.10	Conclusion	296
	Further Reading	298
	Exercises	298
Appendix	A Team Project	300
	A.1 Overview	300
	A.1.1 Goals for a Project	301
	A.1.2 The Team Experience	302
	A.1.3 Coupling the Classroom and Project Experiences	302
	A.2 Project Proposal	304
	A.3 Skeletal System: Status Report 1	306
	A.4 Viable System: Status Report 2	308
	A.5 Comprehensive Final Report	310
	<i>Notes</i>	314
	<i>References</i>	325
	<i>Index</i>	337