

Contents

1	Introduction	1
1.1	Why This Book?	1
1.2	A Bit of History	1
1.3	What Is Agile Software Development?	2
1.4	Why Be Agile?	3
1.5	What This Book Is About?	3
1.6	Implementation Languages	3
1.7	The Structure of the Book	4
1.8	Where to Get More Information?	6
1.9	Where to Go Online?	6
2	Agile Methods and the Agile Manifesto	9
2.1	Introduction	9
2.2	What Is Agile?	9
2.3	The Agile Manifesto	10
2.4	What Are Agile Methods?	12
2.5	Agile Modelling	14
2.6	XP: eXtreme Programming	16
2.6.1	The XP Project Lifecycle	17
2.6.2	User Stories	17
2.6.3	Architectural Spike	18
2.6.4	Release Planning	18
2.6.5	Iterations	18
2.6.6	Acceptance Testing	19
2.6.7	Release	19
2.6.8	Why Is XP Controversial?	19
2.7	DSDM	21
2.8	SCRUM	25
2.8.1	Feature-Driven Development	26
2.9	Summary	30
3	Agile Modelling	31
3.1	Introduction	31
3.2	Modelling Misconceptions	31

- 3.3 Agile Modelling 35
 - 3.3.1 Agile Models Add Value 36
 - 3.3.2 Agile Models Fulfil Their Purpose 37
 - 3.3.3 Agile Models Are Understandable 37
 - 3.3.4 Accuracy and Consistency 37
 - 3.3.5 Agile Models Are Sufficiently Detailed 39
 - 3.3.6 Agile Models Are as Simple as Possible 39
- 3.4 What Sort of Models? 40
- 3.5 Tool Misconceptions 41
- 3.6 Updating Agile Models 42
- 3.7 Summary 43

- 4 **How to Become an Agile Modeller** 45
 - 4.1 Introduction 45
 - 4.2 Agile Modelling Practices 45
 - 4.2.1 The Core Practices 45
 - 4.2.2 The Supplementary Practices 46
 - 4.2.3 Interactions Between Practices 48
 - 4.3 Adopt the Core Agile Modelling Practices 49
 - 4.3.1 Iterative and Incremental Modelling 49
 - 4.3.2 Working as a Team 50
 - 4.3.3 Promoting Simplicity 52
 - 4.3.4 Validating the Models 55
 - 4.4 Consider the Supplementary Practices 56
 - 4.4.1 Improving Productivity 56
 - 4.4.2 Design Patterns 57
 - 4.4.3 Controlling Documentation 59
 - 4.4.4 Motivations for Modelling 61
 - 4.5 Maximise Your Modelling Potential 61
 - 4.5.1 Know Your Tools 61
 - 4.5.2 Refactoring 62
 - 4.5.3 Test-First Design 62
 - 4.5.4 Model in Increments 62
 - 4.5.5 Think Small 63
 - 4.5.6 Agile Models Are Good Enough 63
 - 4.6 Agile Modelling Sessions 63
 - 4.7 Agile Models 65
 - 4.8 Agile Documentation 65
 - 4.9 Summary 67

- 5 **Extreme Programming (XP)** 69
 - 5.1 Introduction 69
 - 5.2 Core XP Values 70
 - 5.2.1 Communication 70
 - 5.2.2 Simplicity 71
 - 5.2.3 Feedback 72
 - 5.2.4 Courage 73

5.3	User Stories	73
5.4	The Twelve XP Practises	73
5.4.1	The Planning Game	74
5.4.2	Small Releases	79
5.4.3	Simple Design	79
5.4.4	Testing	80
5.4.5	Refactoring	81
5.4.6	Pair Programming	82
5.4.7	Collective Ownership	83
5.4.8	Continuous Integration	83
5.4.9	On-Site Customer	84
5.4.10	Coding Standards	84
5.4.11	40-Hour Week	85
5.4.12	System Metaphor	85
5.5	What Is So Extreme About Extreme Programming?	86
5.6	Review	86
6	Putting XP into Practise	89
6.1	Introduction	89
6.2	Planning XP Projects	90
6.2.1	Playing the Planning Game	91
6.2.2	The Goal of the Game	91
6.2.3	The Strategy	92
6.2.4	The Game Pieces	92
6.2.5	The Players	92
6.2.6	The Moves/Playing the Game	93
6.2.7	Planning Your XP Project	97
6.3	Test First Coding	100
6.3.1	How to Write Tests First?	101
6.3.2	What to Test?	108
6.3.3	Confidence in the Test Suite	108
6.4	Making Pair Programming Work	109
6.5	Refactoring	113
6.5.1	The Very Idea	113
6.5.2	When to Refactor?	114
6.5.3	How to Refactor?	115
6.5.4	When Not to Refactor?	115
6.6	Keeping on Track	116
6.6.1	Small Releases	116
6.6.2	Simple Design	116
6.6.3	Continuous Integration	119
6.6.4	Making Collective Ownership Happen	121
6.6.5	Getting an On-Site Customer	122
6.6.6	Stand-Up Meetings	122
6.7	Summary	123

- 7 Agile Modelling and XP 125
 - 7.1 Introduction 125
 - 7.2 The Fit 125
 - 7.3 Common Practises 126
 - 7.4 Modelling Specific Practises 127
 - 7.4.1 Model with a Purpose 127
 - 7.4.2 Multiple Models 129
 - 7.4.3 Know Your Models 131
 - 7.5 XP Objections to Agile Modelling 131
 - 7.6 Agile Modelling and Planning XP Projects 132
 - 7.6.1 Initial Project Planning 132
 - 7.6.2 Iteration/Release Planning 133
 - 7.7 XP Implementation Phase 134
 - 7.7.1 Refactoring 135
 - 7.7.2 Test-First Coding 137
 - 7.7.3 Simple Design 138
 - 7.7.4 Pair Programming 140
 - 7.8 Focus on XP 141

- 8 Agile Modelling and XP Reviewed 143
 - 8.1 Introduction 143
 - 8.2 Review of XP/AM Practices 143
 - 8.2.1 The Planning Game 143
 - 8.2.2 Small Releases 143
 - 8.2.3 Simple Design 144
 - 8.2.4 Testing 144
 - 8.2.5 Refactoring 145
 - 8.2.6 Pair Programming 146
 - 8.2.7 Collective Ownership 148
 - 8.2.8 Continuous Integration 148
 - 8.2.9 On-Site Customer 148
 - 8.2.10 Coding Standards 149
 - 8.2.11 40-hour Week 150
 - 8.2.12 System Metaphor 151
 - 8.3 Other Factors 151
 - 8.3.1 Scalability 151
 - 8.3.2 Post Project Review 151
 - 8.3.3 Environment 152
 - 8.3.4 Daily Meeting 152
 - 8.4 Architecture 152
 - 8.4.1 Why Have an Architecture 153
 - 8.4.2 Characteristics of a Good Architecture 156
 - 8.4.3 So What is an Architecture? 156
 - 8.4.4 Architecture Can Make XP Work 157
 - 8.5 XP on Large Projects 157
 - 8.6 Where XP Works Best 159
 - 8.7 Summary 159

9	Feature-Driven Development	161
9.1	Introduction	161
9.2	Incremental Software Development	163
9.3	Regaining Control: The Motivation behind FDD	164
9.3.1	Feature-Centric Development	165
9.3.2	Timeboxing Iterations	166
9.3.3	Being Adaptive but Managed	167
9.4	Planning an Iterative Project	168
9.4.1	Iterations, Timeboxes and Releases	168
9.4.2	Planning an Iteration	171
9.4.3	An Aside on Planning within an FDD Project	173
9.4.4	Estimating the Cost of a Feature	173
9.5	Architecture Centric	175
9.5.1	Why Architecture Centric?	175
9.5.2	Architecture Defined	175
9.5.3	Why Have an Architecture?	176
9.5.4	Architecture Myths	177
9.5.5	Plan Incremental Build of Software	178
9.6	FDD and XP	178
9.7	Summary	180
10	Planning a Sample FDD Project	183
10.1	Introduction	183
10.2	Initiating the Project	183
10.3	The Overall Project Plan	184
10.4	Planning the First Iteration	186
10.4.1	Selecting Features for Iteration 1	186
10.4.2	Feature to Task Mapping	187
10.4.3	Ordering Tasks for Iteration 1	190
10.4.4	The Gantt Chart for Iteration 1	191
10.5	Post Delivery	192
10.6	Summary	192
11	Agile Methods with RUP and PRINCE2	193
11.1	Introduction	193
11.2	Agile Modelling and RUP	194
11.2.1	Overview of the Unified Process	195
11.2.2	Lifecycle Phases	197
11.2.3	Phases, Iterations and Disciplines	198
11.2.4	Modelling and the Unified Process	201
11.2.5	Agile Modelling and Documentation	204
11.3	FDD and RUP	204
11.4	Agile Methods and Prince2	205
11.5	Summary	209
12	Introducing Agile Methods into Your Organisation	211
12.1	Introduction	211
12.2	Selling Agile Methods	211

- 12.3 Identifying a Suitable First Project 212
- 12.4 Promoting an Agile Culture 213
- 12.5 Building an Agile Team 214
- 12.6 Adopting Agile Processes One at a Time 214
- 12.7 Managing Existing Processes 215
- 12.8 Working with Distributed Teams 216
- 12.9 Get Some Experience 216

- 13 Tools to Help with Agile Development 215**
 - 13.1 Introduction 215
 - 13.2 What Tools Do You Need? 215
 - 13.3 Eclipse: An Agile IDE 216
 - 13.4 Lightweight Modelling within Eclipse 221
 - 13.5 Building Applications with ANT 223
 - 13.6 Version Control with CVS 226
 - 13.6.1 So What Is It All About? 226
 - 13.6.2 Code Central Station 226
 - 13.7 Testing with JUnit 227
 - 13.7.1 JUnit within Eclipse 228
 - 13.7.2 Adding JUnit to an Eclipse Project 229
 - 13.7.3 Using the Eclipse JUnit Wizard 230
 - 13.8 Online References 237

- 14 Obstacles to Agile Software Development 239**
 - 14.1 Introduction 239
 - 14.2 Management Intransigence 239
 - 14.3 The Failed Project Syndrome 240
 - 14.4 Developer Resistance 241
 - 14.5 Customer Opposition 242
 - 14.6 Contractual Difficulties 243
 - 14.7 Familiarity with Agility 245

- 15 References 247**

- Index 251**