

CONTENTS IN DETAIL

ACKNOWLEDGMENTS

xvii

INTRODUCTION

xix

Assumptions and Prerequisites	xxi
What Is Great Code?	xxi
Programmer Classifications	xxii
Amateurs	xxii
Programmers	xxii
Software Engineers	xxiv
Great Programmers	xxiv
So You Want to Be a Great Programmer	xxv
A Final Note on Ethics and Character.	xxv
For More Information	xxvi

PART I: PERSONAL SOFTWARE ENGINEERING

1

1

SOFTWARE DEVELOPMENT METAPHORS

3

1.1 What Is Software?	3
1.1.1 Software Is Not Manufactured	4
1.1.2 Software Doesn't Wear Out	4
1.1.3 Most Software Is Custom.	4
1.1.4 Software Can Be Easily Upgraded.	5
1.1.5 Software Is Not an Independent Entity	5
1.2 Parallels to Other Fields	5
1.2.1 Programmer as Artist	5
1.2.2 Programmer as Architect	6
1.2.3 Programmer as Engineer.	7
1.2.4 Programmer as Craftsman.	7
1.2.5 Artist, Architect, Engineer, or Craftsman?	8
1.3 Software Engineering	8
1.3.1 A Formal Definition	9
1.3.2 Project Size.	10
1.3.3 Where Software Engineering Fails.	12
1.4 Software Craftsmanship	13
1.4.1 Education	13
1.4.2 Apprenticeship	13
1.4.3 The Software Journeyman	14
1.4.4 The Master Craftsman	15
1.4.5 Where Software Craftsmanship Fails	15

1.5	The Path to Writing Great Code	15
1.6	For More Information	16

2

PRODUCTIVITY 17

2.1	What Is Productivity?	17
2.2	Programmer Productivity vs. Team Productivity	18
2.3	Man-Hours and Real Time	19
2.4	Conceptual and Scope Complexity	20
2.5	Predicting Productivity	21
2.6	Metrics and Why We Need Them	22
2.6.1	Executable Size Metric	22
2.6.2	Machine Instructions Metric	23
2.6.3	Lines of Code Metric	23
2.6.4	Statement Count Metric	24
2.6.5	Function Point Analysis	24
2.6.6	McCabe’s Cyclomatic Complexity Metric	24
2.6.7	Other Metrics	25
2.6.8	The Problem with Metrics	25
2.7	How Do We Beat 10 Lines per Day?	26
2.8	Estimating Development Time	27
2.8.1	Estimating Small Project Development Time	27
2.8.2	Estimating Medium and Large Project Development Time	28
2.8.3	Problems with Estimating Development Time	29
2.9	Crisis Mode Project Management	30
2.10	How to Be More Productive	31
2.10.1	Choose Software Development Tools Wisely	31
2.10.2	Manage Overhead	33
2.10.3	Set Clear Goals and Milestones	33
2.10.4	Practice Self-Motivation	34
2.10.5	Focus and Eliminate Distractions	34
2.10.6	If You’re Bored, Work on Something Else	35
2.10.7	Be as Self-Sufficient as Possible	35
2.10.8	Recognize When You Need Help	36
2.10.9	Overcome Poor Morale	36
2.11	For More Information	37

3

SOFTWARE DEVELOPMENT MODELS 39

3.1	The Software Development Life Cycle	39
3.2	The Software Development Model	42
3.2.1	The Informal Model	43
3.2.2	The Waterfall Model	44
3.2.3	The V Model	45
3.2.4	The Iterative Model	46
3.2.5	The Spiral Model	48
3.2.6	The Rapid Application Development Model	49
3.2.7	The Incremental Model	51

3.3	Software Development Methodologies	52
3.3.1	Traditional (Predictive) Methodologies	52
3.3.2	Adaptive Methodologies	52
3.3.3	Agile	52
3.3.4	Extreme Programming	55
3.3.5	Scrum	65
3.3.6	Feature-Driven Development	66
3.4	Models and Methodologies for the Great Programmer	68
3.5	For More Information	69

PART II: UML

71

4

AN INTRODUCTION TO UML AND USE CASES

73

4.1	The UML Standard	73
4.2	The UML Use Case Model	74
4.2.1	Use Case Diagram Elements	74
4.2.2	Use Case Packages	76
4.2.3	Use Case Inclusion	76
4.2.4	Use Case Generalization	77
4.2.5	Use Case Extension	79
4.2.6	Use Case Narratives	80
4.2.7	Use Case Scenarios	86
4.3	The UML System Boundary Diagrams	87
4.4	Beyond Use Cases	88
4.5	For More Information	88

5

UML ACTIVITY DIAGRAMS

89

5.1	UML Activity State Symbols	89
5.1.1	Start and Stop States	90
5.1.2	Activities	90
5.1.3	States	91
5.1.4	Transitions	91
5.1.5	Conditionals	91
5.1.6	Merge Points	93
5.1.7	Events and Triggers	94
5.1.8	Forks and Joins (Synchronization)	96
5.1.9	Call Symbols	96
5.1.10	Partitions	97
5.1.11	Comments and Annotations	98
5.1.12	Connectors	98
5.1.13	Additional Activity Diagram Symbols	98
5.2	Extending UML Activity Diagrams	99
5.3	For More Information	100

6

UML CLASS DIAGRAMS **103**

6.1	Object-Oriented Analysis and Design in UML	103
6.2	Visibility in a Class Diagram	105
6.2.1	Public Class Visibility	105
6.2.2	Private Class Visibility	106
6.2.3	Protected Class Visibility	107
6.2.4	Package Class Visibility	107
6.2.5	Unsupported Visibility Types	108
6.3	Class Attributes	108
6.3.1	Attribute Visibility	109
6.3.2	Attribute Derived Values	109
6.3.3	Attribute Names	109
6.3.4	Attribute Data Types	110
6.3.5	Operation Data Types (Return Values)	110
6.3.6	Attribute Multiplicity	111
6.3.7	Default Attribute Values	111
6.3.8	Property Strings	112
6.3.9	Attribute Syntax	112
6.4	Class Operations	112
6.5	UML Class Relationships	114
6.5.1	Class Dependency Relationships	114
6.5.2	Class Association Relationships	115
6.5.3	Class Aggregation Relationships	116
6.5.4	Class Composition Relationships	117
6.5.5	Relationship Features	117
6.5.6	Class Inheritance Relationships	125
6.6	Objects	125
6.7	For More Information	126

7

UML INTERACTION DIAGRAMS **127**

7.1	Sequence Diagrams	128
7.1.1	Lifelines	128
7.1.2	Message Types	129
7.1.3	Message Labels	130
7.1.4	Message Numbers	130
7.1.5	Guard Conditions	131
7.1.6	Iterations	132
7.1.7	Long Delays and Time Constraints	132
7.1.8	External Objects	133
7.1.9	Activation Bars	133
7.1.10	Branching	134
7.1.11	Alternative Flows	135
7.1.12	Object Creation and Destruction	136
7.1.13	Sequence Fragments	137
7.2	Collaboration Diagrams	152
7.3	For More Information	153

8		
	MISCELLANEOUS UML DIAGRAMS	155
8.1	Component Diagrams	155
8.2	Package Diagrams	158
8.3	Deployment Diagrams	159
8.4	Composite Structure Diagrams	160
8.5	Statechart Diagrams	163
8.6	More UML	165
8.7	For More Information	165

PART III: DOCUMENTATION 167

9		
	SYSTEM DOCUMENTATION	169
9.1	System Documentation Types	170
9.2	Traceability	171
	9.2.1 Ways to Build Traceability into Your Documentation	172
	9.2.2 Tag Formats	172
	9.2.3 The Requirements/Reverse Traceability Matrix	178
9.3	Validation, Verification, and Reviews	181
9.4	Reducing Development Costs Using Documentation	182
	9.4.1 Reducing Costs via Validation	182
	9.4.2 Reducing Costs via Verification	183
9.5	For More Information	184

10		
	REQUIREMENTS DOCUMENTATION	185
10.1	Requirement Origins and Traceability	185
	10.1.1 A Suggested Requirements Format	186
	10.1.2 Characteristics of Good Requirements	187
10.2	Design Goals	193
10.3	The System Requirements Specification Document	193
10.4	The Software Requirements Specification Document	194
	10.4.1 Introduction	195
	10.4.2 Overall Description	196
	10.4.3 Specific Requirements	199
	10.4.4 Supporting Information	203
	10.4.5 A Sample Software Requirements Specification	203
10.5	Creating Requirements	212
10.6	Use Cases	214
	10.6.1 Enable/Disable Debug Mode	215
	10.6.2 Enable/Disable Ethernet	216
	10.6.3 Enable/Disable RS-232	218
	10.6.4 Enable/Disable Test Mode	218
	10.6.5 Enable/Disable USB	218
	10.6.6 Read DIP Switches	218

10.7	Creating DAQ Software Requirements from the Use Cases	218
10.8	(Selected) DAQ Software Requirements (from SRS)	219
10.9	Updating the Traceability Matrix with Requirement Information	222
10.9.1	Requirements to Be Verified by Review	223
10.9.2	Requirements to Be Verified by Testing	225
10.10	For More Information	225

11 SOFTWARE DESIGN DESCRIPTION DOCUMENTATION 227

11.1	IEEE Std 1016-1998 vs. IEEE Std 1016-2009	228
11.2	IEEE 1016-2009 Conceptual Model.	228
11.2.1	Design Concerns and Design Stakeholders	228
11.2.2	Design Viewpoints and Design Elements	229
11.2.3	Design Views, Overlays, and Rationales	239
11.2.4	The IEEE Std 1016-2009 Conceptual Model	242
11.3	SDD Required Contents	244
11.3.1	SDD Identification	244
11.3.2	Design Stakeholders and Their Design Concerns	244
11.3.3	Design Views, Viewpoints, Overlays, and Rationales	245
11.4	SDD Traceability and Tags	245
11.5	A Suggested SDD Outline	245
11.6	A Sample SDD	247
11.7	Updating the Traceability Matrix with Design Information	259
11.8	Creating a Software Design	260
11.9	For More Information	260

12 SOFTWARE TEST DOCUMENTATION 261

12.1	The Software Test Documents in Std 829	262
12.1.1	Process Support	262
12.1.2	Integrity Levels and Risk Assessment	263
12.1.3	Software Development Testing Levels	265
12.2	Test Plans	266
12.2.1	Master Test Plan	266
12.2.2	Level Test Plan	267
12.2.3	Level Test Design Documentation	269
12.3	Software Review List Documentation.	270
12.3.1	Sample SRL Outline	270
12.3.2	Sample SRL	271
12.3.3	Adding SRL Items to the Traceability Matrix.	274
12.4	Software Test Case Documentation	274
12.4.1	Introduction in the STC Document.	277
12.4.2	Details	278
12.4.3	General	280
12.4.4	A Sample Software Test Case Document.	281
12.4.5	Updating the RTM with STC Information	288
12.5	Software Test Procedure Documentation	288
12.5.1	The IEEE Std 829-2009 Software Test Procedure	289
12.5.2	Extended Outline for Software Test Procedure	290
12.5.3	Introduction in the STP Document	292

12.5.4	Test Procedures	294
12.5.5	General	296
12.5.6	Index	297
12.5.7	A Sample STP	297
12.5.8	Updating the RTM with STP Information	302
12.6	<i>Level</i> Test Logs	303
12.6.1	Introduction in the <i>Level</i> Test Logs Document	304
12.6.2	Details	305
12.6.3	Glossary	305
12.6.4	A Few Comments on Test Logs	305
12.7	Anomaly Reports	308
12.7.1	Introduction in the Anomaly Reports Document	309
12.7.2	Details	310
12.7.3	A Few Comments on Anomaly Reports	311
12.8	Test Reports.	312
12.8.1	Brief Mention of the Master Test Report	313
12.8.2	<i>Level</i> Test Reports	313
12.9	Do You Really Need All of This?	315
12.10	For More Information	315

AFTERWORD: DESIGNING GREAT CODE 317

GLOSSARY 319

INDEX 327