

---

# Contents

<b>1</b>	<b>Introduction</b> .....	1
1.1	The Goals and Content of Volume 1 .....	2
1.2	Additional Goals of This Book .....	4
1.3	Overview .....	6
1.4	Notational Conventions .....	7
<b>2</b>	<b>Agile and UML-Based Methodology</b> .....	9
2.1	The Software Engineering Portfolio .....	11
2.2	Extreme Programming (XP) .....	13
2.3	Selected Development Practices .....	19
2.3.1	Pair Programming .....	19
2.3.2	Test-First Approach .....	20
2.3.3	Refactoring .....	23
2.4	Agile UML-Based Approach .....	24
2.5	Summary .....	30
<b>3</b>	<b>Compact Overview of UML/P</b> .....	33
3.1	Class Diagrams .....	34
3.1.1	Classes and Inheritance .....	34
3.1.2	Associations .....	35
3.1.3	Representation and Stereotypes .....	37
3.2	Object Constraint Language .....	39
3.2.1	OCL/P Overview .....	39
3.2.2	OCL Logic .....	42
3.2.3	Container Data Structures .....	42
3.2.4	Functions in OCL .....	49
3.3	Object Diagrams .....	51
3.3.1	Introduction to Object Diagrams .....	51
3.3.2	Compositions .....	54
3.3.3	The Meaning of an Object Diagram .....	54
3.3.4	The Logic of Object Diagrams .....	55

3.4	Statecharts .....	56
3.4.1	Properties of Statecharts .....	56
3.4.2	Representation of Statecharts .....	60
3.5	Sequence Diagrams .....	65
<b>4</b>	<b>Principles of Code Generation .....</b>	<b>71</b>
4.1	Concepts of Code Generation .....	74
4.1.1	Constructive Interpretation of Models .....	76
4.1.2	Tests versus Implementation .....	78
4.1.3	Tests and Implementation from the Same Model .....	81
4.2	Code Generation Techniques .....	82
4.2.1	Platform-Dependent Code Generation .....	82
4.2.2	Functionality and Flexibility .....	85
4.2.3	Controlling the Code Generation .....	88
4.3	Semantics of Code Generation .....	89
4.4	Flexible Parameterization of a Code Generator .....	91
4.4.1	Implementing Tools .....	92
4.4.2	Representation of Script Transformations .....	94
<b>5</b>	<b>Transformations for Code Generation .....</b>	<b>99</b>
5.1	Transformations for Class Diagrams .....	100
5.1.1	Attributes .....	100
5.1.2	Methods .....	103
5.1.3	Associations .....	106
5.1.4	Qualified Associations .....	110
5.1.5	Compositions .....	114
5.1.6	Classes .....	116
5.1.7	Object Instantiation .....	119
5.2	Transformations for Object Diagrams .....	123
5.2.1	Object Diagrams Used For Constructive Code .....	123
5.2.2	Example of a Constructive Code Generation .....	125
5.2.3	Object Diagram Used as Predicate .....	125
5.2.4	An Object Diagram Describes a Structure Modification .....	129
5.2.5	Object Diagrams and OCL .....	131
5.3	Code Generation from OCL .....	132
5.3.1	An OCL Expression as a Predicate .....	133
5.3.2	OCL Logic .....	135
5.3.3	OCL Types .....	137
5.3.4	A Type as an Extension .....	139
5.3.5	Navigation and Flattening .....	140
5.3.6	Quantifiers and Special Operators .....	141
5.3.7	Method Specifications .....	141
5.3.8	Inheritance of Method Specifications .....	145
5.4	Executing Statecharts .....	146
5.4.1	Method Statecharts .....	146

5.4.2	The Transformation of States .....	147
5.4.3	The Transformation of Transitions .....	152
5.5	Transformations for Sequence Diagrams .....	155
5.5.1	A Sequence Diagram as a Test Driver .....	155
5.5.2	A Sequence Diagram as a Predicate .....	157
5.6	Summary of Code Generation .....	158
<b>6</b>	<b>Principles of Testing with Models</b> .....	<b>161</b>
6.1	An Introduction to the Challenges of Testing .....	162
6.1.1	Terminology for Testing .....	163
6.1.2	The Goals of Testing Activities .....	165
6.1.3	Error Categories .....	167
6.1.4	Terminology Definitions for Test Procedures .....	168
6.1.5	Finding Suitable Test Data .....	169
6.1.6	Language-Specific Sources for Errors .....	169
6.1.7	UML/P as the Test and Implementation Language ....	171
6.1.8	A Notation for Defining Test Cases .....	174
6.2	Defining Test Cases .....	177
6.2.1	Implementing a Test Case Operatively .....	177
6.2.2	Comparing Test Results .....	178
6.2.3	The JUnit tool .....	181
<b>7</b>	<b>Model-Based Tests</b> .....	<b>185</b>
7.1	Test Data and Expected Results using Object Diagrams .....	186
7.2	Invariants as Code Instrumentations .....	189
7.3	Method Specifications .....	191
7.3.1	Method Specification for Code Instrumentation .....	191
7.3.2	Method Specifications for Determining Test Cases ....	191
7.3.3	Defining Test Cases using Method Specifications .....	194
7.4	Sequence Diagrams .....	195
7.4.1	Triggers .....	196
7.4.2	Completeness and Matching .....	198
7.4.3	Noncausal Sequence Diagrams .....	199
7.4.4	Multiple Sequence Diagrams in a Single Test .....	199
7.4.5	Multiple Triggers in a Sequence Diagram .....	200
7.4.6	Interaction Patterns .....	200
7.5	Statecharts .....	202
7.5.1	Executable Statecharts .....	202
7.5.2	Using Statecharts to Describe Sequences .....	205
7.5.3	Statecharts used in Testing .....	206
7.5.4	Coverage Metrics .....	208
7.5.5	Transition Tests instead of Test Sequences .....	211
7.5.6	Further Approaches .....	212
7.6	Summary and Open Issues Regarding Testing .....	212

<b>8</b>	<b>Design Patterns for Testing</b> .....	217
8.1	Dummies .....	219
8.1.1	Dummies for Layers of the Architecture .....	221
8.1.2	Dummies with a Memory .....	222
8.1.3	Using a Sequence Diagram instead of Memory .....	223
8.1.4	Catching Side Effects .....	224
8.2	Designing Testable Programs .....	224
8.2.1	Static Variables and Methods .....	225
8.2.2	Side Effects in Constructors .....	228
8.2.3	Object Instantiation .....	228
8.2.4	Predefined Frameworks and Components .....	230
8.3	Handling of Time .....	232
8.3.1	Simulating Time in a Dummy .....	233
8.3.2	A Variable Time Setting in a Sequence Diagram .....	234
8.3.3	Patterns for Simulating Time .....	236
8.3.4	Timers .....	237
8.4	Concurrency with Threads .....	237
8.4.1	Separate Scheduling .....	238
8.4.2	Sequence Diagrams as Scheduling Models .....	240
8.4.3	Handling Threads .....	241
8.4.4	A Pattern for Handling Threads .....	241
8.4.5	The Problems of Forcing Sequential Tests .....	243
8.5	Distribution and Communication .....	245
8.5.1	Simulating the Distribution .....	245
8.5.2	Simulating a Singleton .....	247
8.5.3	OCL Constraints across Several Locations .....	248
8.5.4	Communication Simulates Distributed Processes .....	249
8.5.5	Pattern for Distribution and Communication .....	251
8.6	Summary .....	253
<b>9</b>	<b>Refactoring as a Model Transformation</b> .....	255
9.1	Introductory Examples for Transformations .....	256
9.2	The Methodology of Refactoring .....	261
9.2.1	Technical and Methodological Prerequisites for Refactoring .....	261
9.2.2	The Quality of the Design .....	263
9.2.3	Refactoring, Evolution, and Reuse .....	264
9.3	Model Transformations .....	265
9.3.1	Forms of Model Transformations .....	265
9.3.2	The Semantics of a Model Transformation .....	266
9.3.3	The Concept of Observation .....	272
9.3.4	Transformation Rules .....	277
9.3.5	The Correctness of Transformation Rules .....	278
9.3.6	Transformational Software Development Approaches .....	280
9.3.7	Transformation Languages .....	282

<b>10 Refactoring of Models</b> .....	285
10.1 Sources for UML/P Refactoring Rules .....	286
10.1.1 Defining and Representing Refactoring Rules .....	288
10.1.2 Refactoring in Java/P .....	289
10.1.3 Refactoring Class Diagrams .....	295
10.1.4 Refactoring in OCL .....	301
10.1.5 Introducing Test Patterns as Refactoring .....	302
10.2 A Superimposition Method for Changing Data Structures ...	306
10.2.1 Approach for Changing the Data Structure .....	306
10.2.2 Example: Representing a Bag of Money .....	308
10.2.3 Example: Introducing the Chair in the Auction System .....	312
10.3 Summary of Refactoring Techniques .....	320
<b>11 Summary, Further Reading and Outlook</b> .....	323
11.1 Summary .....	324
11.2 Outlook .....	325
11.3 Agile Model Based Software Engineering .....	328
11.4 Generative Software Engineering .....	331
11.5 Unified Modeling Language (UML) .....	332
11.6 Domain Specific Languages (DSLs) .....	332
11.7 Software Language Engineering (SLE) .....	335
11.8 Modeling Software Architecture and the MontiArc Tool ...	339
11.9 Variability and Software Product Lines (SPL) .....	342
11.10 Semantics of Modeling Languages .....	344
11.11 Compositionality and Modularity of Models and Languages .....	348
11.12 Evolution and Transformation of Models .....	349
11.13 State Based Modeling (Automata) .....	351
11.14 Modelling Cyber-Physical Systems (CPS) .....	354
11.15 Applications in Cloud Computing and Data-Intensive Systems .....	355
11.16 Modelling for Energy Management .....	356
11.17 Modelling Robotics .....	358
11.18 Automotive Software .....	359
11.19 Autonomic Driving and Driver Intelligence .....	360
<b>References</b> .....	363
<b>Index</b> .....	385



<http://www.springer.com/978-3-319-58861-2>

Agile Modeling with UML  
Code Generation, Testing, Refactoring  
Rumpe, B.  
2017, XIII, 388 p. 176 illus., 101 illus. in color.,  
Hardcover  
ISBN: 978-3-319-58861-2