Table of Contents

Foreword xiii
Brian R. LARSON

Foreword xv
Dominique POTIER

Introduction xix
Fabrice KORDON, Jérôme HUGUES, Agusti CANALS and Alain DOHET

PART 1. General Concepts 1

Chapter 1. Elements for the Design of Embedded Computer Systems . . . 3
Fabrice KORDON, Jérôme HUGUES, Agusti CANALS and Alain DOHET

1.1. Introduction 3
1.2. System modeling 5
1.3. A brief presentation of UML 6
1.3.1. The UML static diagrams 7
1.3.2. The UML dynamic diagrams 9
1.4. Model-driven development approaches 10
1.4.1. The concepts 10
1.4.2. The technologies 11
1.4.3. The context of the wider field 12
1.5. System analysis 14
1.5.1. Formal verification via proving 15
1.5.2. Formal verification by model-checking 15
1.5.3. The languages to express specifications 16
1.5.4. The actual limits of formal approaches 19
1.6. Methodological aspects of the development
of embedded computer systems 20

1.6.1. The main technical processes 22
1.6.2. The importance of the models 23
1.7. Conclusion 24
1.8. Bibliography 25

Chapter 2. Case Study: Pacemaker 29
Fabrice KORDON, Jérôme HUGUES, Agusti CANALS and Alain DOHET

2.1. Introduction 29
2.2. The heart and the pacemaker 30
2.2.1. The heart 30
2.2.2. Presentation of a pacemaker 32
2.3. Case study specification 33
2.3.1. System definition 34
2.3.2. System lifecycle 35
2.3.3. System requirements 36
2.3.4. Pacemaker behavior 39
2.4. Conclusion 42
2.5. Bibliography 43

PART 2. SysML 45

Chapter 3. Presentation of SysML Concepts 47
Jean-Michel BRUEL and Pascal ROQUES

3.1. Introduction 47
3.2. The origins of SysML 48
3.3. General overview: the nine types of diagrams 49
3.4. Modeling the requirements 50
3.4.1. Use case diagram 50
3.4.2. Requirement diagram 51
3.5. Structural modeling 53
3.5.1. Block definition diagram 54
3.5.2. Internal block diagram 56
3.5.3. Package diagram 58
3.6. Dynamic modeling 59
3.6.1. Sequence diagram 59
3.6.2. State machine diagram 61
3.6.3. Activity diagram 63
3.7. Transverse modeling 65
3.7.1. Parametric diagram 65
3.7.2. Allocation and traceability 67
3.8. Environment and tools 68
3.9. Conclusion 68
3.10. Bibliography 68

Chapter 4. Modeling of the Case Study Using SysML 71
Loïc FEJOZ, Philippe LEBLANC and Agusti CANALS

4.1. Introduction 71
4.2. System specification 73
4.2.1. Context 73
4.2.2. Requirements model and operational scenarios 75
4.2.3. Requirements model 78
4.3. System design 80
4.3.1. Functional model 81
4.3.2. Domain-specific data 83
4.3.3. Logical architectural model 86
4.3.4. Physical architectural model 90
4.4. Traceability and allocations 90
4.4.1. "Technical needs: divers" traceability diagram 90
4.4.2. Traceability diagram "technical needs: behavior of the pacemaker" 91
4.4.3. Allocation diagram 92
4.5. Test model 93
4.5.1. Traceability diagram "system test: requirements verification" . . . 93
4.5.2. Sequence diagram for the test game TC-PM-07 94
4.5.3. Diagrams presenting a general view of the requirements 94
4.6. Conclusion 95
4.7. Bibliography 97

Chapter 5. Requirements Analysis 99
Ludovic APVRILLE and Pierre DE SAQUI-SANNES

5.1. Introduction 99
5.2. The AVATAR language and the TTool tool 100
5.2.1. Method 101
5.2.2. AVATAR language and SysML standard 101
5.2.3. The TEPE language for expressing properties 102
5.2.4. TTool 103
5.3. An AVATAR expression of the SysML model
of the enhanced pacemaker 103
5.3.1. Functioning of the pacemaker and modeling hypotheses 103
5.3.2. Requirements diagram 104
5.4. Architecture 105
5.5. Behavior 106
5.6. Formal verification of the VVI mode 107
5.6.1. General properties 108
5.6.2. Expressing properties using TEPE 108
5.6.3. The use of temporal logic 109
5.6.4. Observer-guided verification 111

5.6.5. Coming back to the model 112
5.7. Related work 113
5.7.1. Languages 113
5.7.2. Tools 114
5.8. Conclusion 115
5.9. Appendix: TTool 116
5.10. Bibliography 116

PART 3. MARTE 119

Chapter 6. An Introduction to MARTE Concepts 121
Sébastien GÉRARD and François TERRIER

6.1. Introduction 121
6.2. General remarks 121
6.2.1. Possible uses of MARTE 122
6.2.2. How should we read the norm? 123
6.2.3. The MARTE architecture 124
6.2.4. MARTE and SysML 127
6.2.5. An open source support 128
6.3. Several MARTE details 128
6.3.1. Modeling non-functional properties 128
6.3.2. A components model for the real-time embedded system 133
6.4. Conclusion 137
6.5. Bibliography 137

Chapter 7. Case Study Modeling Using MARTE 139
Jérôme DELATOUR and Joël CHAMPEAU

7.1. Introduction 139
7.1.1. Hypotheses used in modeling 139
7.1.2. The modeling methodology used 140
7.1.3. Chapter layout 141
7.2. Software analysis 141
7.2.1. Use case and interface characterization 141
7.2.2. The sphere of application 144
7.3. Preliminary software design – the architectural component 145
7.3.1. The candidate architecture 146
7.3.2. Identifying the components 146
7.3.3. Presentation of the candidate architecture 148
7.3.4. A presentation of the detailed interfaces 150
7.4. Software preliminary design – behavioral component 151
7.4.1. The controller 151
7.4.2. The cardiologist 153

7.4.3. The operating modes of the cardiologist 153
7.5. Conclusion 155
7.6. Bibliography 156

Chapter 8. Model-Based Analysis 157
Frederic BONIOL, Philippe DHAUSSY, Luka LE ROUX and Jean-Charles ROGER

8.1. Introduction 157
8.2. Model and requirements to be verified 161
8.2.1. The UML-MARTE model that needs to be translated in Fiacre . . 161
8.2.2. Fiacre language 162
8.2.3. The translation principles of the UML model in Fiacre 163
8.2.4. Requirements 165
8.3. Model-checking of the requirements 166
8.3.1. Use case 166
8.3.2. Properties 167
8.3.3. Property check 170
8.3.4. First assessment 172
8.4. Context exploitation 172
8.4.1. Identifying the context scenarios 173
8.4.2. Automatic partitioning of the context graphs 174
8.4.3. CDL language 175
8.4.4. CDL model exploitation in a model-checker 177
8.4.5. Description of a CDL context 178
8.4.6. Results 179
8.5. Assessment 180
8.6. Conclusion 181
8.7. Bibliography 182

Chapter 9. Model-Based Deployment and Code Generation 185
Chokri MRAIDHA, Ansgar RADERMACHER and Sébastien GÉRARD

9.1. Introduction 185
9.2. Input models 187
9.2.1. Description of the executable component-based model 187
9.2.2. Description of the platform model 188
9.2.3. Description of the deployment model 189
9.3. Generation of the implementation model 190
9.3.1. Main concepts 191
9.3.2. Connector pattern 191
9.3.3. Container pattern 193
9.3.4. Implementation of the components 195
9.3.5. Resulting implementation components 197
9.4. Code generation 197
9.4.1. Deployment of the components 198

9.4.2. Transformation into an object-oriented model 199
9.4.3. Generating code 200
9.5. Support tools 201
9.6. Conclusion 202
9.7. Bibliography 202

PART 4. AADL 205

Chapter 10. Presentation of the AADL Concepts 207
Jérôme HUGUES and Xavier RENAULT

10.1. Introduction 207
10.2. General ADL concepts 207
10.3. AADLv2, an ADL for design and analysis 208
10.3.1. A history of the AADL 208
10.3.2. A brief introduction to AADL 209
10.3.3. Tools 211
10.4. Taxonomy of the AADL entities 211
10.4.1. Language elements: the components 212
10.4.2. Connections between the components 214
10.4.3. Language elements: attributes 215
10.4.4. Language elements: extensions and refinements 219
10.5. AADL annexes 220
10.5.1. Data modeling annex 220
10.6. Analysis of AADL models 221
10.6.1. Structural properties 222
10.6.2. Qualitative properties 222
10.6.3. Quantitative properties 223
10.7. Conclusion 224
10.8. Bibliography 225

Chapter 11. Case Study Modeling Using AADL 227
Etienne BORDE

11.1. Introduction 227
11.2. Review of the structure of a pacemaker 229
11.3. AADL modeling of the structure of the pacemaker 230
11.3.1. Decomposition of the system into several subsystems 230
11.3.2. Execution and communication infrastructure 233
11.4. Overview of the functioning of the pacemaker 235
11.4.1. The operational modes of the pacemaker 235
11.4.2. The operational sub-modes of the pacemaker 235
11.4.3. Some functionalities of the pacemaker 237
11.5. AADL modeling of the software architecture
of the pulse generator 240

11.5.1. AADL modeling of the operational modes of the pulse generator . . . 240
11.5.2. AADL modeling of the features of the pulse generator in the permanent mode . . . 242
11.6. Modeling of the deployment of the pacemaker . . . 247
11.7. Conclusion . . . 249
11.8. Bibliography . . . 250

Chapter 12. Model-Based Analysis . . . 251
Thomas ROBERT and Jérôme HUGUES

12.1. Introduction . . . 251
12.2. Behavioral validation, per mode and global . . . 252
12.2.1. Validation context and fine tuning of the requirements . . . 253
12.2.2. Translation of the behavioral automata into UPPAAL . . . 253
12.2.3. Refining requirements 22-23/P . . . 258
12.2.4. Study of the permanent/VVT mode . . . 260
12.2.5. Study of the changing of the permanent/VVT→Magnet/VOO mode . . . 261
12.3. Conclusion . . . 262
12.4. Bibliography . . . 263

Chapter 13. Model-Based Code Generation . . . 265
Laurent PAUTET and Béchir ZALILA

13.1. Introduction . . . 265
13.2. Software component generation . . . 268
13.2.1. Data conversion . . . 269
13.2.2. Conversion of subprograms . . . 272
13.2.3. Conversion of execution threads . . . 275
13.2.4. Conversion of the instances of shared data . . . 283
13.3. Middleware components generation . . . 283
13.4. Configuration and deployment of middleware components . . . 284
13.4.1. Deployment . . . 284
13.5. Integration of the compilation chain . . . 285
13.6. Conclusion . . . 287
13.7. Bibliography . . . 287

List of Authors . . . 289

Index . . . 291