

Contents

Part I	Introduction	1
1	Introduction	3
1.1	The Subject of the Book	3
1.1.1	MDA	4
1.2	Target Audience	4
1.2.1	Software Architects	5
1.2.2	Software Developers	5
1.2.3	Managers and Project Leaders	5
1.3	The Goals of the Book	5
1.4	The Scope of the Book	6
1.5	The Structure of the Book and Reader Guidelines	6
1.6	The Accompanying Web site	8
1.7	About the Authors	8
1.8	About the Cover	8
1.9	Acknowledgments	9
2	MDSO – Basic Ideas and Terminology	11
2.1	The Challenge	11
2.1.1	Historical View	12
2.1.2	The Status Quo	12
2.2	The Goals of MDSO	13
2.3	The MDSO Approach	14
2.4	Basic Terminology	16
2.4.1	An Overview of MDA Concepts	18
2.5	Architecture-Centric MDSO	21
2.5.1	Motivation	21
2.5.2	Generative Software Architectures	22
2.5.3	Architecture-Centric Design	24
2.5.4	Development Process	26
2.5.5	The Properties of Architecture-Centric MDSO	27

3	Case Study: A Typical Web Application	29
3.1	Application Development	29
3.1.1	The Application Example	30
3.1.2	MDSO Tools	32
3.1.3	Example 1: Simple Changes to Models	33
3.1.4	Example 2: Model Changes and Protected Regions	35
3.1.5	Example 3: Working with Dynamic Models	37
3.1.6	Interaction Between Development and Architecture	39
3.1.7	Intermediate Result	39
3.2	Architecture Development	40
3.2.1	The UML Profile	40
3.2.2	Transformations	42
3.2.3	The Mode of Operation of the MDSO Generator	47
3.2.4	Bootstrapping	49
3.2.5	Adaptations of the Generative Software Architecture	49
3.2.6	The Boundary of Infrastructure Code	53
3.2.7	Structuring Metaprograms	53
3.3	Conclusion and Outlook	54
4	Concept Formation	55
4.1	Common MDSO Concepts and Terminology	55
4.1.1	Modeling	56
4.1.2	Platforms	59
4.1.3	Transformations	60
4.1.4	Software System Families	61
4.2	Model-Driven Architecture	63
4.3	Architecture-Centric MDSO	64
4.4	Generative Programming	65
4.5	Software Factories	68
4.5.1	The Software Factory Schema	68
4.5.2	The Software Factory Template	69
4.5.3	The Role of DSLs and their Relationship to MDSO	69
4.6	Model-Integrated Computing	70
4.7	Language-Oriented Programming	70
4.8	Domain-Specific Modeling	71
5	Classification	73
5.1	MDSO vs. CASE, 4GL and Wizards	73
5.2	MDSO vs. Roundtrip Engineering	74
5.3	MDSO and Patterns	75
5.3.1	Patterns and Transformations	75
5.3.2	Patterns and Profiles	76
5.3.3	Patterns Languages as a Source of DSLs	77

5.4	MDSO and Domain-Driven Design	77
5.5	MDSO, Data-Driven Development and Interpreters	78
5.6	MDSO and Agile Software Development	78
5.6.1	The Agile Manifesto and MDSO	79
5.6.2	Agile Techniques	80
Part II Domain Architectures		83
6	Metamodeling	85
6.1	What Is Metamodeling?	85
6.2	Metalevels vs. Level of Abstraction	88
6.3	MOF and UML	88
6.4	Extending UML	89
6.4.1	Extension Based on the Metamodel	89
6.4.2	Extension With Stereotypes in UML 1.x	92
6.4.3	Extension With Profiles in UML 2	92
6.5	UML Profiles	93
6.6	Metamodeling and OCL	96
6.7	Metamodeling: Example 1	98
6.8	Metamodeling: Example 2	99
6.9	Tool-supported Model Validation	102
6.10	Metamodeling and Behavior	106
6.11	A More Complex Example	107
6.11.1	The Basics	108
6.11.2	Value Types	109
6.11.3	Physical Quantities	110
6.12	Pitfalls in Metamodeling	113
6.12.1	Interfaces	113
6.12.2	Dependencies	114
6.12.3	IDs	115
6.12.4	Primary Keys	116
6.12.5	Metalevels and Instanceof	116
7	MDSO-Capable Target Architectures	119
7.1	Software Architecture in the Context of MDSO	119
7.2	What Is a Sound Architecture?	120
7.3	How Do You Arrive at a Sound Architecture?	121
7.3.1	Architectural Patterns and Styles	121
7.4	Building Blocks for Software Architecture	122
7.4.1	Frameworks	122
7.4.2	Middleware	123
7.4.3	Components	123
7.5	Architecture Reference Model	124

7.6	Balancing the MDS Platform	125
7.6.1	Examples	126
7.6.2	Integration of Frameworks	127
7.7	Architecture Conformance	127
7.8	MDS and CBD	129
7.8.1	Three viewpoints	129
7.8.2	Viewpoint Dependencies	132
7.8.3	Aspect Models	132
7.8.4	Variations	134
7.8.5	Component Implementation	136
7.9	SOA, BPM and MDS	137
7.9.1	SOA	137
7.9.2	BPM	139
7.9.3	SOA and BPM	140
8	Building Domain Architectures	143
8.1	DSL construction	143
8.1.1	Choose a suitable DSL	143
8.1.2	Configuration and Construction – Variants	144
8.1.3	Modeling Behavior	146
8.1.4	Concrete Syntax Matters!	148
8.1.5	Continuous Validation of the Metamodel	149
8.2	General Transformation Architecture	150
8.2.1	Which Parts of the Target Architecture Should Be Generated?	150
8.2.2	Believe in Reincarnation	150
8.2.3	Exploit the Model	150
8.2.4	Generate Good-looking Code – Whenever Possible	152
8.2.5	Model-driven Integration	153
8.2.6	Separation of Generated and Non-generated Code	154
8.2.7	Modular Transformations	155
8.2.8	Cascaded Model-Driven Development	158
8.3	Technical Aspects of Building Transformations	159
8.3.1	Explicit Integration of Generated Code and Manual Parts	159
8.3.2	Dummy Code	164
8.3.3	Technical Subdomains	166
8.3.4	Proxy Elements	167
8.3.5	External Model Markings	168
8.3.6	Aspect Orientation and MDS	169
8.3.7	Descriptive Meta Objects	170
8.3.8	Generated Reflection Layers	172
8.4	The Use of Interpreters	173
8.4.1	Interpreters	174

8.4.2	MDS D Terminology Revisited	175
8.4.3	Non-functional Properties of Interpreters	176
8.4.4	Integrating Interpreters into a System	177
8.4.5	Interpreters and Testing	179
9	Code Generation Techniques	181
9.1	Code Generation – Why?	181
9.1.1	Performance	181
9.1.2	Code Volume	181
9.1.3	Analyzability	182
9.1.4	Early Error Detection	182
9.1.5	Platform Compatibility	182
9.1.6	Restrictions of the (Programming) Language	182
9.1.7	Aspects	182
9.1.8	Introspection	182
9.2	Categorization	183
9.2.1	Metaprogramming	183
9.2.2	Separation/Mixing of Program and Metaprogram	183
9.2.3	Implicit or Explicit Integration of Generated With Non-generated Code	184
9.2.4	Relationships	184
9.2.5	Examples of the Blending of Program and Metaprogram	185
9.3	Generation Techniques	186
9.3.1	Templates and Filtering	187
9.3.2	Templates and Metamodel	188
9.3.3	Frame Processors	189
9.3.4	API-based Generators	192
9.3.5	In-line Generation	194
9.3.6	Code Attributes	196
9.3.7	Code Weaving	197
9.3.8	Combining Different Techniques	198
9.3.9	Commonalities and Differences between the Different Approaches	199
9.3.10	Other Systems	201
10	Model Transformations with QVT	203
10.1	History	203
10.2	M2M language requirements	204
10.3	Overall Architecture	207
10.4	An Example Transformation	209
10.4.1	The Example in the QVT Relations language	212
10.4.2	The Example in the QVT Operational Mappings language	217

10.5	The OMG Standardization Process and Tool Availability	220
10.6	Assessment	221

11 MDSO Tools: Roles, Architecture, Selection Criteria, and Pointers **223**

11.1	The Role of Tools in the Development Process	223
11.1.1	Modeling	223
11.1.2	Model Validation and Code Generation	224
11.1.3	Build Tool	225
11.1.4	Recipe Frameworks	226
11.1.5	IDE Toolkit	227
11.2	Tool Architecture and Selection Criteria	227
11.2.1	Implement the Metamodel	227
11.2.2	Ignore the Concrete Syntax	227
11.2.3	Modular Transformations	229
11.2.4	Model Transformations are ‘First-Class Citizens’	229
11.3	Pointers	230
11.3.1	The Eclipse World	230
11.3.2	Trends in UML tools	233
11.3.3	UML 2 Composite Structure Diagrams	233
11.3.4	Other kinds of Editors	235
11.3.5	Integrated Metamodeling IDEs	236

12 The MDA Standard **239**

12.1	Goals	239
12.2	Core Concepts	239
12.2.1	UML 2.0	240
12.2.2	MOF – The Meta Object Facility	241
12.2.3	XMI	242
12.2.4	PIM/PSM/PDM	243
12.2.5	Multi-stage Transformations	244
12.2.6	Action Languages	244
12.2.7	Core Models	247
12.2.8	Controlling the PIM to PSM Transformation	248
12.2.9	Executable UML	250

Part III Processes and Engineering **251**

13 MDSO Process Building Blocks and Best Practices **253**

13.1	Introduction	253
13.2	Separation between Application and Domain Architecture Development	253
13.2.1	The Basic Principle	253

13.2.2	Domain Architecture Development Thread	255
13.2.3	Application Development Thread	260
13.3	Two-Track Iterative Development	262
13.4	Target Architecture Development Process	263
13.4.1	Three Phases	264
13.4.2	Phase 1: Elaborate	265
13.4.3	Phase 2: Iterate	269
13.4.4	Phase 3: Automate	269
13.5	Product-Line Engineering	271
13.5.1	Software System Families and Product Lines	272
13.5.2	Integration into the MDSO Process	272
13.5.3	Methodology	272
13.5.4	Domain Modeling	277
13.5.5	Further Reading	277
14	Testing	279
14.1	Test Types	279
14.2	Tests in Model-Driven Application Development	280
14.2.1	Unit Tests	281
14.2.2	Acceptance Tests	286
14.2.3	Load Tests	287
14.2.4	Non-functional Tests	288
14.2.5	Model Validation	288
14.3	Testing the Domain Architecture	290
14.3.1	Testing the Reference Implementation and the MDSO Platform	290
14.3.2	Acceptance Test of the DSL	290
14.3.3	Test of the MDSO Transformations	290
15	Versioning	293
15.1	What Is Versioned?	293
15.2	Projects and Dependencies	293
15.3	The Structure of Application Projects	294
15.4	Version Management and Build Process for Mixed Files	295
15.5	Modeling in a Team and Versioning of Partial Models	297
15.5.1	Partitioning vs. Subdomains	297
15.5.2	Various Generative Software Architectures	298
15.5.3	Evolution of the DSL	298
15.5.4	Partitioning and Integration	300
16	Case Study: Embedded Component Infrastructures	305
16.1	Overview	305
16.1.1	Introduction and Motivation	306
16.1.2	Component Infrastructures	306

16.1.3	Component Infrastructure Requirements for Embedded Systems	307
16.1.4	The Basic Approach	307
16.2	Product-Line Engineering	307
16.2.1	Domain Scoping	308
16.2.2	Variability Analysis and Domain Structuring	309
16.2.3	Domain Design	312
16.2.4	Domain Implementation	315
16.3	Modeling	315
16.3.1	Definition of Interfaces	315
16.3.2	Definition of Components and Ports	316
16.3.3	Definition of a System	318
16.3.4	The Complete Model	320
16.3.5	Processing	320
16.4	Implementation of Components	321
16.5	Generator Adaptation	323
16.5.1	Parsing of the textual syntax	323
16.5.2	Parsing the System Definition XML	325
16.5.3	Parsing and Merging the Complete Model	326
16.5.4	Pseudo-declarative Metamodel Implementation	328
16.6	Code Generation	330
16.6.1	References	330
16.6.2	Polymorphism	333
16.6.3	Separation of Concerns in the Metamodel	335
16.6.4	Generation of Build Files	337
16.6.5	Use of AspectJ	338
17	Case Study: An Enterprise System	341
17.1	Overview	341
17.2	Phase 1: Elaboration	341
17.2.1	Technology-Independent Architecture	341
17.2.2	Programming Model	342
17.2.3	Technology Mapping	343
17.2.4	Mock Platform	344
17.2.5	Vertical Prototype	344
17.3	Phase 2: Iterate	344
17.4	Phase 3: Automate	345
17.4.1	Architecture Metamodel	345
17.4.2	Glue Code Generation	346
17.4.3	DSL-based Programming Model	346
17.4.4	Model-Based Architecture Validation	354
17.5	Discussion	355

Part IV Management	357
18 Decision Support	359
18.1 Business Potential	359
18.2 Automation and Reuse	361
18.3 Quality	365
18.3.1 Well-defined Architecture	365
18.3.2 Preserved Expert Knowledge	365
18.3.3 A Stringent Programming Model	365
18.3.4 Up-to-date and Usable Documentation	366
18.3.5 The Quality of Generated Code	366
18.3.6 Test Effort and Possible Sources of Errors	367
18.4 Reuse	367
18.5 Portability, Changeability	368
18.6 Investment and Possible Benefits	369
18.6.1 Architecture-centric MDS	369
18.6.2 Functional/Professional MDS Domains	373
18.7 Critical Questions	374
18.8 Conclusion	378
18.9 Recommended Reading	378
19 Organizational Aspects	379
19.1 Assignment of Roles	379
19.1.1 Domain Architecture Development	379
19.1.2 Application Development	382
19.2 Team Structure	383
19.2.1 Definition of Roles and Staffing Requirements	384
19.2.2 Cross-Cutting Teams	385
19.2.3 Tasks of the Architecture Group	385
19.3 Software Product Development Models	386
19.3.1 Terminology	387
19.3.2 In-house Development	387
19.3.3 Classical Outsourcing	388
19.3.4 Offshoring	389
19.3.5 Radical Offshoring	389
19.3.6 Controlled Offshoring	390
19.3.7 Component-wise Decision	391
20 Adoption Strategies for MDS	393
20.1 Prerequisites	393
20.2 Getting Started – MDS Piloting	393
20.2.1 Risk Analysis	394
20.2.2 Project Initialization	395

20.3	MDSB Adaptation of Existing Systems	396
20.4	Classification of the Software Inventory	397
20.5	Build, Buy, or Open Source	398
20.6	The Design of a Supply Chain	399
20.7	Incremental Evolution of Domain Architectures	400
20.8	Risk Management	400
20.8.1	Risk: Tool-centeredness	400
20.8.2	Risk: A Development Tool Chain Counterproductive to MDSB	401
20.8.3	Risk: An Overburdened Domain Architecture Team	401
20.8.4	Risk: Waterfall Process Model, Database-centered Development	402
20.8.5	Risk: The Ivory Tower	402
20.8.6	Risk: No Separation of Application and Domain Architecture	403
A	Model Transformation Code	405
A.1	Complete QVT Relations alma2db Example	405
A.2	Complete QVT Operational Mappings alma2db Example	411
	References	415
	Index	421