

Inhaltsverzeichnis

Teil I Einführung

1	Einleitung	3
1.1	Das Thema des Buches	3
1.2	Zielgruppen	5
1.3	Ziele des Buches	6
1.4	Abgrenzung	7
1.5	Struktur des Buches und Leitfaden für den Leser	7
1.6	Die zweite Auflage	9
1.7	Webseite zum Buch	10
1.8	Danksagungen	10
2	Einführung in MDSD	11
2.1	Was ist MDSD	11
2.1.1	Formale Modelle	11
2.1.2	Lauffähige Software erzeugen	12
2.1.3	Automatisch	13
2.2	Gründe für MDSD	13
2.2.1	Abstraktion	13
2.2.2	Einheitliche Architektur	14
2.2.3	Entwicklungsgeschwindigkeit	14
2.2.4	Wiederverwendung	15
2.2.5	Interoperabilität und Plattformunabhängigkeit	15
2.2.6	Softwarequalität	16
2.3	Erstes Fallbeispiel: Generator für einfache Webanwendungen	16
2.3.1	Die technische Basis	16
2.3.2	Entitäten, Komponenten und Webseiten	17
2.3.3	Anwendungscode	19

2.3.4	Generator	20
2.3.5	Gesamtsicht.....	23
2.3.6	Fazit und Ausblick	25
3	Begriffe und Konzepte	27
3.1	Definitionen	27
3.1.1	Modellierung.....	28
3.1.2	Software-Systemfamilien	34
3.2	Verwandte Ansätze.....	35
3.2.1	Model Driven Architecture – MDA.....	36
3.2.2	Generative Programming.....	37
3.2.3	Software Factories.....	40
3.2.4	Model-integrated Computing	41
3.2.5	Sprachorientierte Programmierung.....	42
3.2.6	Domänenspezifische Modellierung.....	43
3.3	Vergleich mit anderen Techniken.....	43
3.3.1	CASE, 4GL und Wizards.....	43
3.3.2	Roundtrip Engineering	44
3.3.3	Patterns	46
3.3.4	Domain Driven Design.....	48
3.3.5	MDSD, datengetriebene Entwicklung und Interpreter	48
3.3.6	MDSD und Agile Softwareentwicklung.....	49
Teil II	Domänenarchitekturen	
4	Das Beispiel – eine Versicherungsanwendung	55
4.1	Das Projekt.....	55
4.2	Fachlichkeit	56
4.2.1	Funktionalität	56
4.2.2	Entitäten	57
4.3	Architektur	57
4.3.1	Client-Server-Architektur	57
4.3.2	Plattform.....	58
4.3.3	Serverseitiges Komponentenmodell	58
5	Metamodellierung	59
5.1	Einführung	59
5.1.1	Was ist Metamodellierung?	59
5.1.2	Constraints und Modellvalidierung	60
5.1.3	Metametamodelle.....	61

5.2	Konkrete Technologien	64
5.2.1	MOF/UML.....	64
5.2.2	EMF und Ecore.....	71
5.2.3	Klassen als Metamodell.....	73
5.2.4	Abstrakter Syntaxbaum.....	74
5.2.5	XML und XSD.....	74
5.2.6	Vergleich der Technologien.....	74
5.3	Best Practices.....	75
5.3.1	Lebendiges Metamodell.....	75
5.3.2	Metamodell an erster Stelle	76
5.3.3	Modulares Metamodell	76
5.3.4	Extension Points.....	77
5.3.5	Metamodell als Projektsprache.....	77
5.4	Typische Metamodellbestandteile.....	78
5.4.1	Komponenteninfrastruktur.....	78
5.4.2	Implementierung von Komponenten	86
5.4.3	Ausdrücke.....	87
5.5	Validierung von Metamodellen	89
5.6	Beispiel.....	89
5.6.1	Typen und Komponenten.....	89
5.6.2	Persistenz	91
5.6.3	Formbriefe	93
6	Domänenspezifische Sprachen	97
6.1	Kategorisierung	98
6.1.1	Interne vs. externe DSLs.....	98
6.1.2	Wiederverwendung vs. Neuimplementierung	100
6.1.3	Grafische vs. textuelle DSLs	101
6.2	Techniken der DSL-Entwicklung.....	103
6.2.1	Textuelle Syntax mit Xtext.....	104
6.2.2	Das Graphical Modeling Framework (GMF)	105
6.2.3	Parsergeneratoren.....	107
6.2.4	Andere Arten von DSL-Editoren	109
6.2.5	Integrierte Metamodellierungs-IDEs.....	110
6.3	Best Practices.....	113
6.3.1	Evolution von DSLs (Don't break the public API).....	113
6.3.2	Modellierung von Verhalten.....	114
6.3.3	Konkrete Syntax ist wichtig!	115
6.3.4	Don't Repeat Yourself (DRY)	115

6.3.5	Configuration by Exception (Defaults).....	116
6.3.6	Convention over Configuration.....	116
6.4	Anwendung im Versicherungsbeispiel.....	117
6.4.1	Typen und Komponenten.....	117
6.4.2	Persistenz	120
6.4.3	Formbriefe	121
7	Konstruktion MDSD-tauglicher Zielarchitekturen	123
7.1	Softwarearchitektur im Kontext von MDSD.....	123
7.2	Was ist eine gute Architektur?	125
7.3	Wie kommt man zu einer guten Architektur?	126
7.3.1	Architekturmuster und -stile.....	126
7.4	Bausteine für Softwarearchitektur.....	127
7.4.1	Frameworks	127
7.4.2	Middleware.....	127
7.4.3	Komponenten.....	128
7.5	Architektur-Referenzmodell.....	129
7.6	Ausbalancierung der Plattform	130
7.6.1	Beispiele	131
7.6.2	Integration von Frameworks	131
7.7	SOA, BPM und MDSD.....	132
7.7.1	SOA	132
7.7.2	BPM.....	135
7.7.3	SOA und BPM	136
8	Codegenerierung	139
8.1	Workflows und Cartridges	139
8.1.1	Die Workflow-Engine des MWE-Projekts	140
8.1.2	Cartridges	140
8.2	Codegenerierung – warum?	141
8.2.1	Performance	141
8.2.2	Codegröße.....	141
8.2.3	Analysierbarkeit	141
8.2.4	Fehlerfrüherkennung.....	142
8.2.5	Einschränkungen der (Programmier-)Sprache.....	142
8.2.6	Aspekte	142
8.2.7	Introspection	142

8.3	Kategorisierung	143
8.3.1	Metaprogramme	143
8.3.2	Präprozessoren (Makroprozessoren)	143
8.3.3	Integrierte Metaprogrammierfunktionalität.....	144
8.3.4	Interpreter	144
8.3.5	Codegeneratoren bei der MDSD	145
8.4	Konstruktion von Codegeneratoren.....	145
8.4.1	Templates.....	146
8.4.2	Generieren mit imperativen Programmiersprachen.....	149
8.4.3	Codegenerierung mit openArchitectureWare 4.x.....	152
8.5	Best Practices.....	157
8.5.1	Generiere gut aussehenden Code – wann immer möglich	158
8.5.2	Verquickung von generiertem und nicht-generiertem Code ..	159
8.6	Testen von Generatoren	166
8.6.1	Testen gegen die konkrete Syntax.....	166
8.6.2	Testen mit Referenzmodell	167
8.7	Versicherungsbeispiel	168
8.7.1	Verzeichnisstruktur	169
8.7.2	Entitäten	169
8.7.3	Komponenten.....	172
9	Interpreter	173
9.1	Interpreter und Generatoren	174
9.2	MDSD-Terminologie aus Sicht von Interpretern.....	175
9.2.1	Domäne	176
9.2.2	Metamodell.....	176
9.2.3	Metametamodell	176
9.2.4	Formales Modell	177
9.2.5	Plattform.....	177
9.3	Nichtfunktionale Eigenschaften von Interpretern	177
9.3.1	Performance	178
9.3.2	Umfang des Programmcodes	178
9.3.3	Zeitpunkt des Bindens.....	178
9.4	Integration eines Interpreters in ein System.....	179
9.4.1	Aufruf des Interpreters	179
9.4.2	Erweiterungspunkte	180
9.4.3	Bereitstellen der Modelle	180

9.5	Interpretation von Ausdrücken.....	181
9.5.1	Implementierung eines Interpreters.....	181
9.5.2	Immutable Execution Context.....	186
9.6	Interpreter und Tests	188
9.6.1	Testen des Interpreters	188
9.6.2	Testen der Modelle.....	190
9.7	Interpreter im Versicherungsbeispiel.....	190
9.7.1	Auswertung des Parse-Baums	191
9.7.2	Eingabetexte speziell für einen Brief	193
10	Modell-zu-Modell-Transformationen	195
10.1	Wozu Modelltransformationen?.....	195
10.1.1	Cartridges	197
10.2	Kategorisierung	199
10.2.1	Modellmodifikation	199
10.2.2	Modelltransformation	199
10.2.3	Modell-Weaving (Linking)	200
10.3	Herausforderungen.....	201
10.3.1	Umgang mit Mengen.....	201
10.3.2	Zyklen.....	201
10.3.3	Debug-Fähigkeiten	202
10.3.4	Inkrementelle Transformationen	203
10.3.5	Protected Regions bei Modelltransformationen	203
10.3.6	Bidirektionale Transformationen.....	203
10.4	Konkrete Lösungen	204
10.4.1	QVT.....	204
10.4.2	Atlas Transformation Language (ATL)	205
10.4.3	Xtend	206
10.4.4	Vergleich	207
10.5	Testen von Modelltransformationen.....	207
10.6	Modelltransformationen im Versicherungsbeispiel	208
10.6.1	Modelltransformation im Kontext	209
10.6.2	Konkrete Transformation.....	209

Teil III Prozesse und Engineering

11	MDSD-Prozessbausteine und Best Practices	215
11.1	Einleitung	215
11.2	Trennung von Anwendungs- und Domänenarchitekturentwicklung..	215
11.2.1	Grundprinzip	215
11.2.2	Architektur-Entwicklungsstrang	217
11.2.3	Anwendungs-Entwicklungsstrang.....	224
11.2.4	Organisatorische Aspekte	225
11.3	Zweigleisig iterative Entwicklung	225
11.4	Entwicklungsprozess für Zielarchitekturen.....	227
11.4.1	Drei Phasen.....	229
11.4.2	Phase 1: Elaboration	229
11.4.3	Phase 2: Iteration	234
11.4.4	Phase 3: Automation	235
11.5	Grundlagen des Product Line Engineering	237
11.5.1	Software-Systemfamilien und Produktlinien	238
11.5.2	Einordnung in den MDSD-Prozess	238
11.5.3	Methodik	239
11.5.4	Domänenmodellierung	244
11.5.5	Weiterführende Literatur.....	245
12	Testen	247
12.1	Testen von Softwaresystemen	248
12.1.1	Unit-Tests und interne DSLs.....	248
12.1.2	Lasttests	251
12.1.3	Nicht-funktionale Tests.....	252
12.1.4	Oberflächentests.....	252
12.2	Testen mit MDSD	253
12.2.1	Generieren von automatischen Testfällen	253
12.2.2	Unit-Tests und DSLs	253
12.3	Modellgetriebenes Testen	256
12.3.1	Abgrenzung.....	257
12.3.2	Konzeptionelle Grundlagen	257
12.3.3	Fallstudie: Oberflächentests mit oAW-Test und JMeter.....	260

13	Versionierung	267
13.1	Was wird versioniert?	267
13.2	Projekte und Abhängigkeiten.....	268
13.3	Struktur von Anwendungsprojekten	269
13.4	Versionsmanagement und Build-Prozess bei gemischten Dateien	270
13.5	Modellierung im Team und Versionierung von Teilmodellen	272
13.5.1	Partitionierung vs. Subdomänen.....	272
13.5.2	Verschiedene generative Softwarearchitekturen.....	273
13.5.3	Weiterentwicklung der DSL	273
13.5.4	Partitionierung und Integration	276
13.5.5	»Echte« dateiübergreifende Referenzierung	276
14	Fallstudie: Eingebettete Komponenteninfrastrukturen	279
14.1	Überblick.....	279
14.1.1	Einführung und Motivation	279
14.1.2	Komponenteninfrastrukturen	280
14.1.3	Anforderungen an Komponenteninfrastrukturen bei eingebetteten Systemen.....	281
14.1.4	Grundsätzlicher Ansatz	281
14.2	Product Line Engineering	282
14.2.1	Domain Scoping.....	282
14.2.2	Variabilitätsanalyse und Domänenstrukturierung	283
14.2.3	Domänenendesign.....	287
14.2.4	Domänenimplementierung	290
14.3	Modellierung.....	290
14.3.1	Definition von Interfaces	290
14.3.2	Definition von Komponenten und Ports	292
14.3.3	Definition eines Systems	293
14.3.4	Gesamtmodell	296
14.3.5	Verarbeitung	296
14.4	Implementierung von Komponenten.....	297
14.5	Generatorentwicklung	299
14.5.1	Definition der textuellen Syntax	299
14.5.2	Parse und Zusammenführen des Gesamtmodells	300
14.5.3	Deklarative Modellvalidierung	302

14.6	Codegenerierung	305
14.6.1	Referenzen	305
14.6.2	Polymorphismus.....	307
14.6.3	Trennung von Verantwortlichkeiten im Metamodell.....	309
14.6.4	Generierung der Build-Files	310
14.6.5	Verwendung von AspectJ	310
14.7	Kaskadierte Domänenarchitekturen	312
14.7.1	Modellierung.....	313
14.7.2	Generierung	314

Teil VI Management

15	Entscheidungshilfen	319
15.1	Betriebswirtschaftliches Potenzial	319
15.2	Automation und Wiederverwendung.....	321
15.3	Qualität.....	326
15.3.1	Wohldefinierte Architektur	326
15.3.2	Konserviertes Expertenwissen	326
15.3.3	Stringentes Programmiermodell.....	327
15.3.4	Aktuelle und nutzbare Dokumentation	327
15.3.5	Qualität von generiertem Code	328
15.3.6	Testaufwand und mögliche Fehlerquellen.....	328
15.4	Wiederverwendung.....	329
15.5	Portabilität, Änderbarkeit.....	330
15.6	Investitionen und erzielbare Gewinne	331
15.6.1	Architekturzentrierte MDSD	331
15.6.2	Fachlich ausgerichtete MDSD-Domänen	336
15.7	Kritische Fragen	337
15.8	Zusammenfassung.....	342
15.9	Weiterführende Literatur.....	342
16	Organisatorische Aspekte	343
16.1	Rollenverteilung	343
16.1.1	Domänenarchitektur-Entwicklung	343
16.1.2	Anwendungsentwicklung	347

16.2	Teamstruktur	347
16.2.1	Ausgestaltung der Rollen und Personalbedarf	349
16.2.2	Querschnitts-Teams	350
16.2.3	Aufgaben der Architekturgruppe	350
16.3	Software-Produktentwicklungsmodelle	352
16.3.1	Terminologie	352
16.3.2	In-House-Entwicklung	353
16.3.3	Klassisches Outsourcing	354
16.3.4	Offshoring.....	355
16.3.5	Radikales Offshoring	356
16.3.6	Kontrolliertes Offshoring	357
16.3.7	Komponentenweise Entscheidung	359
17	Adaptionstrategien für MDSD	361
17.1	Voraussetzungen	361
17.2	Getting Started – MDSD-Pilotierung	362
17.2.1	Risikoanalyse	363
17.2.2	Projektinitialisierung	363
17.3	MDSD-Adaptierung bestehender Systeme.....	364
17.4	Klassifikation des Software-Inventars	366
17.5	Bauen, Kaufen oder Open Source	368
17.6	Entwurf einer Zulieferkette	369
17.7	Inkrementelle Evolution von Domänenarchitekturen.....	370
17.8	Risikomanagement	371
17.8.1	Risiko: Toolzentriertheit	371
17.8.2	Risiko: Für MDSD kontraproduktive Entwicklungs-Toolkette	371
17.8.3	Risiko: Überlastetes Domänenarchitekturteam.....	372
17.8.4	Risiko: Wasserfall-Vorgehensmodell, datenbankzentrierte Entwicklung	372
17.8.5	Risiko: Elfenbeinturm	373
17.8.6	Risiko: Keine Trennung von Anwendung und Domänenarchitektur	373

Anhang

A	MDA-Standard	377
A.1	Ziele	377
A.2	Kernkonzepte	378
A.3	Herausforderungen für die MDA	390
B	Queries/Views/Transformations (QVT)	393
B.1	Historie	393
B.2	Architektur	394
B.3	Eine Beispiel-Transformation	396
B.4	Bewertung	409
C	Quelltext der Modelltransformationen	413
C.1	Complete QVT Relations alma2db Example	413
C.2	Complete QVT Operational Mappings alma2db Example	419
D	Literatur	423
E	Gastautoren	431
	Stichwortverzeichnis	433