Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Siegfried Nolte

QVT - Relations Language

Modellierung mit der Query Views Transformation

Springer

Siegfried Nolte
Beethovenstr. 57
22941 Bargteheide
Deutschland
siegfried.nolte@alice-dsl.net

Xpert.press ISSN 1439-5428
ISBN 978-3-540-92170-7 e-ISBN 978-3-540-92171-4
DOI 10.1007/978-3-540-92171-4
Springer Dordrecht Heidelberg London New York

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

© Springer-Verlag Berlin Heidelberg 2009
Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.
Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Einbandentwurf: KuenkelLopka GmbH, Heidelberg

Gedruckt auf säurefreiem Papier

9 8 7 6 5 4 3 2 1

Springer ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.com)

Vorwort

Warum ein Buch über QVT Relations Language?

Dieses Buch beschäftigt sich mit der Transformation von Modellen. Dazu gibt es zurzeit bereits mehrere Mittel und Techniken, die jedoch vorwiegend die Transformation nach Text unterstützen, also die Generierung von Code-Artefakten aus formalen Modellen. Das ist schon etwas, schon ein Stück modellgetriebene Entwicklung. Die Transformation eines Modells, welches in einer grafischen Modellierungssprache erstellt worden ist, in ein anderes, ebenfalls grafisch repräsentiertes, ist bis heute noch die Ausnahme. Im Sinne des von der *Object Management Group* (OMG) veröffentlichten MDA-Konzeptes ist jedoch die Modelltransformation neben der Modellierung von Sachverhalten eine klassische Aktivität der modellgetriebenen Anwendungsentwicklung.

Modellierung erfolgt mit formalen Modellen, mit Modellen, die auf der Basis von Metamodellen formal spezifiziert worden sind – auch dies ein standardisiertes Konzept der OMG. Und die Modellierungssprache UML ist ein Klassiker, der heutzutage aus der Entwicklung von DV-gestützten Anwendungssystemen nicht mehr wegzudenken ist. Anmerken möchte ich an dieser Stelle, dass ich, wenn ich hier und im Weiteren von Standard rede, nicht Standard im Sinne der Normierung und Standardisierung durch ein nationales (DIN, ANSI) oder internationales Institut (ISO) meine. Standard heißt die Spezifikation und Veröffentlichung durch die *Object Management Group*, ein internationales Konsortium, das sich unter Beteiligung verschiedener IT-Unternehmen zusammengetan hat, um objektorientierte Technologien zu standardisieren. Das soll uns soweit reichen. Die Konzepte und Spezifikationen der OMG im MDA-Kontext stehen im Mittelpunkt dieses Buches.

Aber warum nun ein Buch über QVT? Mit Ausnahme der publizierten Spezifikation vom April 2008 gibt es bisher keins. Und die Spezifikation ist als Lehrmaterial meiner Meinung nach nicht brauchbar. Auch in den mir bekannten MDA-Büchern wird kaum oder gar nicht auf QVT eingegangen. Doch QVT ist ja genau das Mittel zur Transformation von Modellen, welches im Rahmen des MDA-

Konzeptes mit der MOF/QVT-Spezifikation Anfang 2008 als Standard herausgegeben worden ist. Diese Lücke will das vorliegende Fachbuch füllen.

Für wen ist das Buch gedacht?

Modellierung von Sachverhalten der realen Welt ist ein wesentlicher Bestandteil des ingenieursmäßigen Software-Entwicklungsprozesses. In der Szene der Entwicklungswerkzeuge ist die modellgetriebene Software-Entwicklung, die neben der Modellierung auch einen generativen Aspekt verfolgt, schon seit langem ein beliebtes Feld, das bereits mit Kompetenz und Erfahrung bedient wird. Mit Together 2006 zum Beispiel hat Borland ein Werkzeug auf den Markt gebracht, welches neben der Modellierung mit unterschiedlichen Modellierungssprachen – UML2, BPMN, SysML etc. – auch Modelltransformation mit QVT *Operational Mappings* unterstützt. Mit der Veröffentlichung des QVT-Standards, sogar schon mit der Vorankündigung, der *Final Adopted Specification*, Juli 2007, gibt es auch andere Hersteller, die QVT-Produkte entwickeln und anbieten. Eins davon, mediniQVT der ikv++ technologies AG, werde ich hier vorstellen und anwenden. Mit mediniQVT, ein Eclipse-basiertes Werkzeug zur Modelltransformation mit der *Relations Language*, sind nahezu alle Beispiele dieses Buches entwickelt worden.

Wie wir sehen, konzentrieren sich die Hersteller von Modellierungsprodukten zunehmend auf die QVT, was sicher auch darin begründet ist, dass sie zumeist ebenfalls in den Gremien der OMG vertreten sind. Ein weiterer interessanter Trend ist, dass das QVT-Konzept auch von der Eclipse *Modeling Tool* Initiative aufgegriffen worden ist und nun auch aus der Szene interessante Lösungen zu erwarten sind, zum Beispiel SmartQVT und QVTO, Transformationswerkzeuge, welche *Operational Mappings* unterstützen. Mit *Operational Mappings* und SmartQVT werde ich mich in einem weiteren Buch befassen.

Aber hier erst einmal die *Relations Language*. Dieses Fachbuch ist als Lehrbuch gedacht, mit dem die Sprache *Relations Language* erlernt und geübt werden kann. Ich habe versucht, das Thema anhand umfassender und ausführlicher Beispiele anschaulich und nachvollziehbar aufzuarbeiten. Grundkenntnisse der Modellierung mit UML, insbesondere im Zusammenhang mit Modellen und Metamodellen, setze ich dabei voraus, Erfahrungen mit höheren Programmiersprachen sind sicher hilfreich.

MDA beschreibt ein Konzept, in dem sich vorwiegend Analytiker und Anwendungsentwickler wiederfinden wie auch, gerade in dem Zweig der Architekturentwicklung, natürlich IT-Architekten. Diese Analytiker- und Entwicklerrollen möchte ich primär ansprechen. Gerade IT-Architekten sollen in die Lage versetzt werden, Metamodelle zu entwickeln und darauf aufbauend Modelltransformationen vorzunehmen. Aber es würde mich freuen, wenn auch Manager von IT-Organisationen wie auch Leiter von IT-Entwicklungsprojekten einige interessante Eindrücke und Erkenntnisse aus dem Buch entnehmen könnten, insbesondere was

die Einordnung der Modelltransformation in einen modellgetriebenen Entwicklungsprozess bedeutet.

Wie sollte man es lesen?

Das Hauptthema ist natürlich die *Relations Language*. Zuvor gebe ich in einem einführenden ersten Kapitel einen entwicklungsgeschichtlichen Einblick in das Thema der ingenieursmäßigen Software-Entwicklung, um darauf aufbauend eine Einordnung des modellgetriebenen Vorgehens in den systematischen Entwicklungsprozess zu begründen. Daran schließt eine Erläuterung der wesentlichen Begriffe und Akronyme des MDA-Konzeptes an. Im Weiteren erfolgt eine Erläuterung der Architektur des QVT-Ansatzes. Dieser beschreibt drei Transformationssprachen, *Core Language*, *Relations Language* und *Operational Mappings*, die ich kurz vorstellen und im Zusammenhang einordnen werde.

Das zweite Kapitel beschäftigt sich mit Metamodellierung. Wie die Modellierung mit formalen Modellierungssprachen, so ist auch der QVT-Ansatz zur Transformation grundsätzlich metamodellbasiert. Den Sachverhalt möchte ich verdeutlichen; an verhältnismäßig einfachen Beispielen, bei denen ich mich auf die Vorgabe der QVT-Spezifikation stütze, soll die Entwicklung von Metamodellen erläutert werden.

Im dritten Kapitel geht es um die *Relations Language*. Auch diese soll ausgehend von einem sehr einfachen Beispiel, bei dem ich das berühmte `HelloWorld` bemühen werde, erläutert werden. Danach stelle ich die Entwicklungsumgebung mediniQVT vor, mit der alle in diesem Buch vorgestellten Beispiele erarbeitet worden sind. mediniQVT ist ein freies Produkt, das die spezifizierte QVT *Relations Language* sehr konsequent unterstützt und damit für ein Üben mit der *Relations Language* recht hilfreich ist. In dem folgenden Kapitel gehe ich nun formal auf sämtliche Sprachkonzepte ein, um so die Sprache in ihrer Gänze vorzustellen.

Dem folgen mehrere vollständige Beispiele, die ich Schritt für Schritt entwickeln werde, um so die Erarbeitung von Transformationsscripten anhand eines exemplarischen Vorgang zu verdeutlichen, zunächst das Standardbeispiel der QVT-Spezifikation `UML2RDBM` mit den im zweiten Kapitel erarbeiteten einfachen Metamodellen `SimpleUML` und `SimpleRDBM` und anschließend ein umfassendes Projekt auf der Basis der UML2. Zuletzt werde ich einige fortgeschrittene Konzepte erläutern, zum Beispiel die Arbeit mit UML-Profilen, BlackBoxes etc.

Wem bin ich zu Dank verpflichtet?

Auf jeden Fall meiner Familie, zu allererst meinen Eltern Marie Louise und Gerhard, denen es gelungen ist, drei Kindern in einer früher schon wirtschaftlich schwierigen Zeit den Abschluss einer akademischen Ausbildung zu ermöglichen, womit letztendlich der Grundstein für dieses Buch gelegt worden ist. Dann den wichtigsten Frauen in meinem Leben, Konstanze und Christine, die während der Entstehung dieses Buches nicht nur auf einen Teil gemeinsamer Zeit verzichten,

sondern auch noch einen Teil ihrer Zeit aufbringen mussten, um sich durch frühe Fassungen des Buches hindurchzuarbeiten. Beide gehören nicht zu dem oben angeführten Leserkreis, doch ihre Kritik und Anregungen waren sehr hilfreich.

Als nächstes natürlich den Personen und Institutionen, die mich bei der Veröffentlichung des Buches unterstützten, zum einen dem Springer-Verlag und zum anderen Frau Monika Riepl von der Letex publishing services oHG. Der Springer-Verlag ist stets – so auch in diesem Fall – bereit, auch neue, manchmal unerfahrene Autoren bei der Veröffentlichung von Büchern zu unterstützen. Und Frau Riepl hat mir behutsam und mit Kompetenz geholfen, eine angemessene Form des Buches zu realisieren.

Und schließlich hat die ikv++ technologies AG mit der Herausgabe des Produktes mediniQVT und der Antwort auf viele Fragen im Forum dazu beigetragen, dass die Beschreibung der Sprachkonzepte mit vielen Beispielen unterlegt und untermauert werden konnte. Für die Beispiele im Kapitel Metamodelle ist das freie UML-Werkzeug TOPCASED eingesetzt worden. Fast alle Abbildungen sind UML-Diagramme, die mit dem Werkzeug MagicDraw der Firma nomagic erstellt worden sind.

Inhaltsverzeichnis

1 Einführung ... 1

1.1 Modellgetriebene Software-Entwicklung – die Geschichte ... 1

1.2 MDA – Model Driven Architecture ... 7

1.2.1 Definition der Model Driven Architecture ... 8

System ... 11

Plattform ... 11

Modell ... 11

Modellierung ... 12

Architektur ... 12

Transformation ... 14

1.2.2 MOF – Modelle und Metamodelle ... 17

1.2.3 QVT – Query Views Transformation ... 17

Deskriptive Sprachen ... 19

Imperative Sprachen ... 20

1.3 Zusammenfassung und Ausblick ... 20

1.3.1 Hinweise zur Notation ... 21

1.3.2 Werkzeuge ... 23

2 Metamodelle und ihre Darstellung ... 25

2.1 Das Metamodell SimpleUML ... 26

2.2 Das Metamodell SimpleRDBM ... 29

2.3 Serialisierung der Metamodelle ... 31

2.3.1 Die Deklaration der Metamodelle als QVT-Datenstruktur ... 32

2.3.2 EMOF – Datenstrukturen im XMI-Format ... 34

2.3.3 Die Verwendung der Metamodelle ... 38

2.4 Werkzeugunterstützung ... 39

2.4.1 Erstellung von Metamodellen mit Topcased ... 40

2.5 Zusammenfassung und weiterführende Literatur ... 46

3 Relations Language 47
3.1 mediniQVT 47
3.1.1 Aufbau der mediniQVT-Entwicklungsplattform 48
3.1.2 Bearbeitung von QVT-Projekten 49
3.2 Das berühmteste Programm der Welt 51
3.3 Der generelle Aufbau von *Relations Language*-Programmen 54
3.4 Formale Darstellung der Konzepte 57
3.4.1 Transformationen 57
3.4.2 Relationen 59
Charakterisierung von Relationen 61
3.4.3 Domänen 62
3.4.4 when- und where-Klauseln 67
3.4.5 Relation Calls und Function Calls 69
3.4.6 Primitive Domänen 72
3.4.7 Hilfsfunktionen – Queries 75
3.4.8 Variablen und Objekte 77
3.4.9 Object Template Expressions und Inline-Objekterzeugung . 80
3.4.10 Rekursionen 84
Das Prinzip des rekursiven Abstiegs 86
3.4.11 Keys 89
3.4.12 Kommentare 90
3.4.13 Relations Language und OCL 90
3.5 Exemplarische Entwicklung von Transformationen 93
3.5.1 Signatur: Vorbereitung der Transformation 95
3.5.2 Festlegen und Klassifizieren der Relationen 96
1. Lösung der Klassifizierung von Relationen 97
2. Lösung der Klassifizierung von Relationen 98
3.5.3 Festlegen der Domänen 99
3.5.4 Beschreibung der Domänenmuster 101
PackageToSchema 101
ClassToTable 102
PrimaryKeys mittels *Inline* Objekterzeugung 104
3.5.5 Auflösung der Assoziationen – AssocToTable 106
Anlegen der Tabellen 106
Auflösen der Beziehungen 108
3.5.6 Behandlung von Attributen – AttributeToColumn 113
Attribute mit einfachen Datentypen 113
Attribute mit komplexen Datentypen 115
Attribute aus Vererbungshierarchien 118
3.5.7 Relation Calls und primitive Domänen 121
3.6 Weiterführende Konzepte 124
3.6.1 Behandlung komplexer Codes 125
3.6.2 Redefinition und Mehrfachverwendung 129
3.6.3 Implementierung von Domänen durch BlackBoxes 131

3.6.4 Bidirektionale Transformation 133
3.7 UML2RDBM im rekursiven Abstieg 134
3.7.1 Aufbau des Grundgerüsts der Transformation 135
3.7.2 Spezifikation der Domänenmuster 136
3.7.3 Behandlung von Beziehungstypen 138
3.7.4 Attribute mit ihren primitiven Datentypen 138
3.7.5 Attribute mit ihren komplexen Datentypen 140
3.7.6 Attribute aus Vererbungsbeziehungen 140
3.8 UmlToEjb – Ein Beispiel mit dem UML2-Metamodell 141
3.8.1 Aufgabe 144
3.8.2 Identifizieren der relevanten UML-Elemente 145
3.8.3 Domänen und Domänenmuster 148
3.8.4 Business-Klassen mit ihren Attributen und Operationen 152
3.8.5 Business-Klassen mit ihren Assoziationen 154
3.8.6 Generieren der SessionBean-Methoden und Interfaces 155
3.9 QVT und UML-Profile 162
3.10 Schlussbemerkungen und Ausblick 168

A Die Syntax der Relational Language 171
Reservierte Wörter 171
Ableitungsregeln 171

B SimpleUML und SimpleRDBM 173
Deklaration der Metamodelle als QVT-Datentypen 173
Ecore-Repräsentation 175
<!-- SimpleUML --> 175
<!-- SimpleRDBM --> 177
Benutzung der Ecore-Metamodelle 180

C Relations Language-Beispiele 181
UmlToRdbm – Das vollständige *Relations Language*-Beispiel 181
UML2EJB – Das vollständige *Relations Language*-Beispiel 188

D Die wichtigsten OCL-Standardfunktionen 200
OCL-Standardfunktionen auf Sammlungen 200
OCL-Iterator-Funktionen 202

Glossar 205

Abkürzungsverzeichnis 213

Quellenverzeichnis **215**
Literatur 215
Referenzen im Internet 218

Index **221**

Abbildungsverzeichnis

Abb.1.1 SWE – die konservative Methode 2
Abb.1.2: SWE – erste Verbesserung durch Beschreibung 4
Abb.1.3: SWE – Analyse, Spezifikation und Implementierung 5
Abb.1.4: Der klassische Anwendungsentwicklungsprozess 6
Abb.1.5: Ein modellgetriebener Anwendungsentwicklungsprozess 9
Abb.1.6: Der MDA-Entwicklungsprozess 13
Abb.1.7: Das MDA-Transformationspattern 15
Abb.1.8: Ein exemplarisches MDA-Transformationspattern 16
Abb.1.9: Architektur der QVT-Sprachen 18
Abb.1.10: Model-To-Model/Model-To-Text-Abgrenzung 22
Abb.1.11: Die Architektur der QVT-Entwicklungsumgebung 24
Abb.2.1: Das Metamodell `SimpleUML` als Klassendiagramm 26
Abb.2.2: Beispiel eines `SimpleUML`-Modells 28
Abb.2.3: Das Metamodell `SimpleRDBM` 30
Abb.2.4: Beispiel eines SimpleRDBM-Modells 31
Abb.2.5: `SimpleUML` im Topcased Ecore/UML-Editor 40
Abb.2.6: Generierung der Metamodelle als Eclipse-Plugins 42
Abb.2.7: Deployment der Metamodelle 43
Abb.2.8: Als Eclipse Plugins veröffentlichte Metamodelle 44
Abb.2.9: Das Wohnungsbaukreditgeschäft im `SimpleUML` 45
Abb.3.1: Die *Relations Language*-Plattform mediniQVT 48
Abb.3.2: Konfiguration der Metamodelle 49
Abb.3.3: Ausführen einer Modelltransformation mit mediniQVT 50
Abb.3.4: `HelloWorld` im *Sample Reflective Ecore Model Editor* 54
Abb.3.5: Der Aufbau von *Relations Language*-Scripten 55
Abb.3.6: Eine rekursive Struktur im Metamodell `SimpleUML` 87
Abb.3.7: UML2RDBM im MDA-Pattern 94
Abb.3.8: Das Package `Darlehen` – ein simples UML-Diagramm 95
Abb.3.9: Das Schema Darlehen im SimpleRDBM 104

Abb.3.10: `Darlehen` – nach Transformation der Klassen 106
Abb.3.11: `Darlehen` – nach Auflösen der Assoziationen 112
Abb.3.12: `Darlehen` – mit Columns aus einfachen Attributen 115
Abb.3.13: Package `Darlehen` mit komplexen Attributen 116
Abb.3.14: Schema `Darlehen` mit komplexen Columns 118
Abb.3.15: Das erweiterte `SimpleUML`-Modell `Darlehen` 119
Abb.3.16: `SimpleRDBM`-Schema nach Generalisierungen 121
Abb.3.17: Das Metamodell UML2 im Kontext *Element* 142
Abb.3.18: Das Metamodell UML2 im Kontext *Classifier* 143
Abb.3.19: Die Transformation UML2EJB im MDA-Pattern 144
Abb.3.20: Das Wohnungsbaukreditgeschäft als UML-Diagramm 146
Abb.3.21: Das Wohnungsbaukreditgeschäft als EJB-Diagramm 161
Abb.3.22: Die SessionBean SB_Immobilie 161
Abb.3.23: UML-Profil mit Stereotyp <<persistent>> 163
Abb:3.24: Anwendung des Stereotyps im UML-Modell 164
Abb.C.1: Überblick über das Transformationsscript `UmlToRdbms` 179
Abb.C.2: Das Transformationsscript UML2EJB im Überblick 186
Abb.C.3: Detailstruktur der Relation `Classes` 187