

Roland Petrasch · Oliver Meimberg

Model Driven Architecture

Eine praxisorientierte Einführung in die MDA

Mit Gastbeiträgen von Florian Fieber und Karsten Thoms



dpunkt.verlag

Roland Petrasch
petrasch@SoftwareQuality.de

Oliver Meimberg
oliver.meimberg@form4.de

Lektorat: René Schönfeldt
Copy-Editing: Sandra Gottmann, Münster
Satz & Herstellung: Birgit Bäuerlein
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: Koninklijke Wöhrmann B.V., Zutphen, Niederlande

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

ISBN 3-89864-343-3

1. Auflage 2006
Copyright © 2006 dpunkt.verlag GmbH
Ringstraße 19
69115 Heidelberg

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen. Insbesondere sind Model Driven Architecture, MDA, Object Management Group, Unified Modeling Language, UML, MOF, XMI Warenzeichen oder eingetragene Warenzeichen der Object Management Group (OMG), Inc., USA

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

Vorwort

Die modellgetriebene Software-Entwicklung hat sich in den letzten Jahren zu einem praxistauglichen Ansatz entwickelt, verschiedene Techniken, Methoden und Standards sorgen für ein breites Anwendungsspektrum, und die Werkzeuge werden entsprechend weiterentwickelt. Da erscheint es angebracht, die *Model Driven Architecture* (MDA) der *Object Management Group* (OMG) in einem einführenden Lehrbuch einem breiten Leserkreis näher zu bringen. Denn es zeichnet sich ab, dass die MDA zwar keine Revolution innerhalb weniger Jahre bewirken kann, die Software-Entwicklung jedoch mittelfristig, nachhaltig und signifikant voranbringen wird: Mit exakteren und formaleren Modellen als bisher, besonders in den frühen Phasen eines Projektes, steigen zwar die Anforderungen an die Qualifikation der Beteiligten, z.B. Systemanalytiker, Software-Architekten etc. Dies geschieht aber zugunsten der Qualität und Produktivität. Die MDA stellt damit einen weiteren Schritt in Richtung Professionalisierung der Informatik dar, der dringend notwendig erscheint: Nach wie vor sind Qualität, Kosten und Termintreue bei (zu) vielen Projekten unbefriedigend.

Die MDA verbessert die Software-Entwicklung nachhaltig.

Wie soll man dieses Buch lesen?

Das Buch ist kein Nachschlagewerk, sondern will als Lehrbuch von Anfang bis Ende möglichst sequenziell durchgearbeitet werden. Zum Durchblättern eignet es sich nicht, besonders da Rechnerübungen integraler Bestandteil des Buchkonzeptes sind. Wichtig ist auch die Bearbeitung der Aufgaben, damit eine Lernerfolgskontrolle stattfinden kann. Obwohl es prinzipiell möglich ist, sich den Stoff allein »im stillen Kämmerlein« anzueignen, ist eine Arbeit in der Gruppe, z.B. im Rahmen einer Lehrveranstaltung, sinnvoller, denn viele Fragen lassen sich bei einem Gedankenaustausch rasch klären, und weiterführende Themen können erörtert werden. Empfehlenswert ist ein Wochenzyklus, d.h. das Bearbeiten einiger Kapitel mit Beantwortung der Aufga-

Das Lehrbuch ist als Einstieg gedacht und keine vollständige MDA-Referenz.

ben an einem bestimmten Wochentag, so dass sich der Lernstoff zwi- schendurch »setzen« kann. In maximal drei Monaten müsste das Lehrbuch dann durchgearbeitet sein.

Das Buch kann und will weder MDA erschöpfend behandeln, noch bietet es eine Lösung für die vielen unbeantworteten Fragen in Zusam- menhang mit der MDA. So ist nach wie vor unklar, ob und wie sich die verfügbaren Sprachen und Modellarten im Rahmen der MDA verwen- den lassen – zurzeit beschränken sich zahlreiche Werkzeuge auf die UML (*Unified Modeling Language*) und deren Klassenmodelle, obwohl andere Diagrammtypen wie Use-Case-Diagramme, State- Charts, Kompositionsstrukturdiagramme eine interessante Rolle spie- len könnten. Insofern muss an zwei Stellen »Bescheidenheit« herrschen:

1. Umfang: Ein kompaktes Lehrbuch lässt sich durch die MDA pro- blemlos sprengen, so dass hier viele Themen sehr stark vereinfacht oder weggelassen wurden. Entsprechende Literaturangaben sollen dabei helfen, diese Lücke zu schließen.
2. Werkzeug: Die MDA »lebt« mit den eingesetzten Tools, denn der Automatisierungsgrad stellt neben der Qualität schließlich eine der argumentativen Säulen dar. Leider stecken viele Werkzeuge noch in einer Art Orientierungsphase und sind daher nur bedingt MDA-tauglich.

Dennoch war und ist es der Anspruch dieses Buches, weder ein Trivial- beispiel zu präsentieren, noch einen naiven MDA-Ansatz im Sinne einer reinen Code-Generierung aus einem einzigen Modell vorzustel- len. Damit wäre weder dem MDA-Ansatz noch den Lesern gedient.

Für wen ist dieses Buch?

Sowohl Studierende im Hauptstudium (Diplom), im Masterstudium oder in den letzten Semestern des Bachelorstudiums als auch Software- Entwickler in der Praxis sollen mit diesem Buch angesprochen werden. Studierende können sich gleich während der Ausbildung für MDA »fit« machen. Der Praktiker (Software-Entwickler, IT-Architekt, Sys- temanalytiker, technischer Projektleiter) hingegen, für den die Soft- ware-Entwicklung das Tagesgeschäft ist, erhält einen fundierten Ein- stieg in die MDA und kann somit ein eigenes Urteil darüber fällen.

Der erste Teil des Buches (Kapitel 1 bis 4) hat zwar eher Lehrbuch- charakter, ist jedoch auch für den Praktiker interessant, um die Basis- information gerade im terminologischen, aber auch im konzeptionel- len Bereich, z.B. Model-to-Model-Transformation, zu erhalten. Der zweite Teil (Kapitel 5 bis 8) ist eine wirklich praxisnahe und detail- lierte Beschreibung, wie ein Model-to-Code-Ansatz mit der MDA aus-

sieht. Schritt für Schritt wird der Ablauf in Form von drei Iterationen einschließlich der Entwicklung einer kleinen Cartridge demonstriert. Das Beispiel umfasst die wichtigsten Komponenten einer Software-Anwendung: das User-Interface, die Business-Logik und die Persistenzschicht.

Wie ist das Buch aufgebaut?

Das Buch umfasst sieben Kapitel, die aufeinander aufbauen, d.h., es sollte »in einem Stück« durchgelesen bzw. durchgearbeitet werden.

Kapitel 1 stellt mit einer Einleitung zentrale Begriffe wie das Artefakt vor und erläutert die Bedeutung der Formalisierung.

Kapitel 2 erörtert die Grundlagen der MDA. Es bietet zunächst Definitionen für die wichtigsten Begriffe an und geht auf Modelle und SW-Architekturen ein, um mit dem Leser bzw. der Leserin eine gemeinsame terminologische Basis zu haben. Weiterhin geht es um die MDA-Spezifikationen der OMG. Wichtig ist: Die UML sollte schon bekannt sein, denn es geht hier um weiterführende Konzepte für MDA. Ein weiterer Kernbereich sind die verschiedenen Modelle der MDA (PIM, PSM, PM) und die Transformationen.

Kapitel 3 bringt schließlich die MDA-Grundlagen in einen Zusammenhang mit Software-Qualität und stellt einen methodischen Baukasten zur Verfügung. So werden beispielsweise die verschiedenen Transformationsarten erklärt.

Kapitel 4 stellt einen prozessorientierten Rahmen vor, damit die methodischen Bausteine der MDA in einer Vorgehensweise anwendbar werden.

Kapitel 5 befasst sich mit dem technischen Umfeld und gibt einen Überblick über die MDA-Tools. Etwas ausführlicher wird auf zwei Werkzeuge eingegangen. Dies ist einerseits AndroMDA und andererseits der openArchitectureWare/OpenGenerator.

Kapitel 6, 7 und 8 stellen in Form von drei Iterationen ein ausführliches Anwendungsbeispiel mit Hilfe der Vorgehensweise aus Kapitel 4 und des Werkzeuges AndroMDA vor.

Kapitel 9 gibt eine Zusammenfassung, diskutiert einige Vor- und Nachteile der MDA und blickt in die Zukunft.

Zu beachten sind die aktuellen Informationen über dieses Buch im Web. Die Domain ist www.MDABuch.de. Hier finden sich auch die Lösungen der Übungsaufgaben und die Code-Beispiele. Auch das Beispiel mit dem openArchitectureWare/OpenGenerator gibt es als Bonuskapitel auf der Webseite.

*Informationen zum Buch
finden sich im Web unter
www.MDABuch.de.*

Welche Vorkenntnisse werden vorausgesetzt?

Vorkenntnisse für dieses
Buch sind notwendig,
um den Rahmen
nicht zu sprengen:
UML, Java, Eclipse

Es ist unabdingbar, dass bereits UML-Kenntnisse vorhanden sind, bevor man dieses Buch zur Hand nimmt. Die Grundlagen der Objektorientierung müssen auf der Konzeptebene genauso beherrscht werden wie die Sprache Java einschließlich einer Entwicklungsumgebung, z. B. Eclipse.

Ebenfalls nicht näher erklärt sind die Technologien für verteilte, unternehmensweite Java-Anwendungen, auch wenn versucht wurde, diesen Bereich so einfach wie möglich zu halten. Es empfiehlt sich jedoch, vorab das Open-Source Framework »Apache struts« sowie die J2EE (Java Enterprise Edition) wenigstens in einem Beispiel kennen zu lernen, damit die Konzepte für JSPs (Java Server Pages) und EJBs (Enterprise Java Beans) bekannt sind. Damit verbunden ist auch die Verwendung von Servern (Web-, Applikations-, Datenbankschicht) wie z. B. Apache Tomcat, JBoss oder MySQL. Wenn also auch hier ein (kleines) Fundament gegeben ist, kann es eigentlich losgehen.

Ganz so einfach ist es leider nicht, denn bevor es tatsächlich an die Tastatur geht, sind die ersten vier Kapitel zu »überstehen«, sprich: durcharbeiten. MDA ist eben mehr als nur flexible Code-Generierung oder »besseres« *executable UML*, aber MDA kann eben auch nicht »zaubern«, denn eine Software-Architektur ist nun einmal nur so gut wie der Architekt, der sie erschuf – daran wird sich auf absehbare Zeit nicht sehr viel ändern, aber ein Architekt kann (oder besser: muss) sich bei seiner Arbeit durch zahlreiche Verfahren, Methoden, Techniken und Werkzeuge unterstützen lassen: Genau hier setzt MDA an.

Was ist MDA?

MDA: Anspruch und
Wirklichkeit liegen
zuweilen auseinander.

Es stellt sich die berechtigte Frage, was genau das »MDA-Ding« ist, denn es gibt weder eine einzelne MDA-Spezifikation, noch ist es ein Produkt, sondern mehr eine Idee und ein Satz von bestehenden OMG-Spezifikationen, z. B. die UML. Da es noch nicht möglich ist, so etwas wie die »MDA-Konformität« zu prüfen, so wie dies z. B. bei den *UML Compliance Levels* ([UML2Super], S. 2) der Fall ist, lassen wir es beim Begriff des *Konzeptes* i. S. des lateinischen Wortes *capere* (begreifen) und sprechen vom *MDA-Konzept*. Ob es in Zukunft eine MDA-Norm, einen MDA-Standard, eine MDA-Konformität oder zertifizierte MDA-Produkte geben wird, bleibt damit offen.

Das Buch als
Wegbegleiter:
Die Reise zur MDA muss
jeder selbst antreten.

Mit Sicherheit kann die MDA auch nach den fünf Jahren, in denen der Ansatz »auf dem Markt« ist, längst nicht als ausgereift bezeichnet werden. Es liegt noch viel Arbeit besonders bei der Standardisierung vor uns. Insofern wäre ein blindes Vertrauen in die MDA wenig hilf-

reich. Lassen Sie sich, liebe Leserin, lieber Leser, also nicht durch die zahlreichen Begriffe und Ansätze im Bereich *modellgetriebene Software-Entwicklung* verwirren und machen sich ein eigenes Bild über die MDA. Wir hoffen, dass Ihnen dieses Buch dabei hilft.

Welche Sprache und welche Werkzeuge kommen zum Einsatz?

Die Sprachen sind vornehmlich UML 1.4 bzw. 2.0 und Java (Standard Edition) sowie die Java 2 Enterprise Edition (J2EE 1.4). Als UML-Modellierungswerkzeug wird MagicDraw von No Magic, Inc. empfohlen. Als integrierte Entwicklungsumgebung wurde Eclipse ausgewählt, weil sie wie das UML-Tool und Java auf den meisten Plattformen verfügbar, weit verbreitet und kostenlos erhältlich ist.

Auch bei den MDA-Werkzeugen wurde darauf geachtet, dass sie frei verfügbar bzw. Open-Source sind und eine gewisse Verbreitung haben. Dies gilt für die beiden eingesetzten Tools AndroMDA und openArchitectureWare. Beide existieren bereits mehrere Jahre, sind kostenlos einsetzbar und haben eine recht aktive Community. Auch das Leistungsspektrum, mit dem die Werkzeuge mittlerweile aufwarten, kann sich sehen lassen, obwohl es bei solchen innovativen und Open-Source-Produkten natürlich Einschränkungen gegenüber den kommerziellen Lösungen, z.B. IBM Rational Software Architect, gibt. Aber genau die leichtgewichtigen und frei verfügbaren Produkte sind es, die aus unserer Sicht einen Einstieg in das Thema erleichtern und sich daher für dieses Buch besonders gut eignen.

Konventionen für die Notation

Ziel ist es, mit möglichst wenigen Sonderzeichen, Markierungen und Spezialregelungen im Text auszukommen, um ein »normales« Lesen des Buches zu ermöglichen. Um einige Festlegungen kommen wir jedoch nicht herum:

- Neue Begriffe und Abbildungsbezug: Wenn ein neuer, wichtiger Begriff eingeführt wird, so ist dieser *kursiv* gekennzeichnet. Auch kommt dieser Stil zum Einsatz, wenn mit einem Begriff nicht das reale Objekt, sondern seine Abbildung oder sein Konzept gemeint ist, z.B. *Abstraktion* ist das Weglassen von irrelevanten Details.
- Technische Konstrukte: Sie sind mit einer serifenlosen, nichtproportionalen Schrift gekennzeichnet, z.B. das Verzeichnis `/opt/eclipse`.
- Anführungszeichen: Bei wörtlichen Zitaten, bei Sprichwörtern und bei nicht ganz erst gemeinten oder metaphorischen Aussagen bieten sie sich an, z.B. auch für die MDA gilt: »Ex nihilo nihil fit.«

- Englisch-Deutsch und die »Bindestrichkrankheit«: Alle Begriffe einzudeutschen ist uns nicht gelungen (ehrlich gesagt, haben wir es auch gar nicht so richtig versucht), so dass zahlreiche Wortverbindungen beide Sprachen enthalten. In diesem Fall sind die Wörter mit einem Bindestrich getrennt, um so Lesehürden zu vermeiden. Das gilt auch für rein englische Begriffe, z.B. heißt es bei uns *Software-Engineering* statt *Softwareengineering* (das Doppel-»e« sieht doch zu »unglücklich« aus). Auch finden sich die englischen Begriffe in Großschreibweise, wenn es sich dabei um ein Substantiv handelt, z.B. *Use-Case* statt *use case*.

Was ist noch zu beachten?

Auch MDA bedeutet harte Arbeit – Software zum Nulltarif gibt es (noch) nicht.

Auf einen letzten Aspekt sei an dieser Stelle hingewiesen, um Enttäuschungen durch das Buch zu vermeiden: Frustration und Kritik könnte schnell dadurch entstehen, dass die »MDA-Versprechungen« hier anscheinend nicht eingehalten bzw. nicht vermittelt werden. Es ist jedoch nicht das Ziel von MDA (und diesem Buch), Software-Entwicklung für komplexe Systeme auf einen einfachen Knopfdruck zu reduzieren – im Gegenteil: Die (Meta-)Modellierung ist eine anspruchsvolle, komplexe und zuweilen auch recht abstrakte Angelegenheit – durch das MDA-Konzept werden komplexe Sachverhalte nicht automatisch zu einfachen »Peanuts«, aber die MDA hilft u.a. dabei, die Komplexität beherrschbarer, die Essenz des Systems sichtbarer bzw. wiederverwendbarer und das Projektrisiko berechenbarer zu machen. Die Anforderungen lassen sich bis zum Code verfolgen, und die Fehleranfälligkeit wird vermindert. Das Ergebnis (oder die Hoffnung) ist eine verbesserte Produkt- und Prozessqualität.

Fazit: Statt der Quadratur des Kreises zum Nulltarif ist die MDA ein weiterer Schritt in Richtung Professionalisierung der Informatik, der mit viel (Lern-)Aufwand verbunden ist.

Roland Petrasch, Oliver Meimberg
Berlin im April 2006

Danksagung von Roland Petrasch

Dem fertigen Werke merkt man es nicht wirklich an: das Verständnis und die liebevolle Hilfe der Familie und Freunde, die fachliche Unterstützung der Kollegen und Mitautoren sowie die Bereitstellung der Technik und Infrastruktur durch alle Beteiligten.

Bei meiner lieben Frau Leticia und meinen Eltern will ich mich herzlich bedanken und besonders auch bei meinem Sohn Richard, der mich inspiriert hat und mich stets daran erinnerte, dass es tatsächlich ein Leben außerhalb der modellgetriebenen Software-Architekturen gibt. Ebenfalls Dank gebührt dem Mitautor Oliver Meimberg für die sehr gute Zusammenarbeit – die vielen fachlichen Diskussionen waren eine Bereicherung und trugen entscheidend zum Gelingen bei. Auch den Autoren der Gastbeiträge, Florian Fieber und Karsten Thoms, sei an dieser Stelle gedankt.

Meinen Studenten möchte ich ebenfalls ein Dankeschön zurufen. Oftmals ohne es zu merken, haben sie durch Gespräche oder Fragen wertvolle Beiträge geleistet und mir »zu denken« gegeben. Besonders fruchtbar waren die Unterredungen in meinem berühmten »Büro« auf dem Flur am Kaffeeautomaten.

Und last, but not least ist die Begleitung durch den Verlag eine essenzielle Angelegenheit. Insofern danke ich allen Mitarbeitern und besonders Herrn Schönfeldt für die Betreuung und die Geduld mit uns. Gleiches gilt auch den Gutachtern für ihre wertvollen Hinweise.

Danksagung von Oliver Meimberg

Mein besonderer Dank gilt zunächst meiner Freundin und Partnerin Petra, die mich bei meiner Arbeit an diesem Buch stets unterstützt und inspiriert hat, sowie meinem Hund Max, der mich häufig davor bewahrt hat, vollständig im Stoff der Metamodellierung zu versinken.

Weiterhin möchte ich mich bei meinem Freund und ehemaligen Geschäftspartner Oliver Charlet bedanken. Die lange Zusammenarbeit mit ihm als technischer Pionier haben mir viele Aspekte der Software-Entwicklung eröffnet, die sich auch in diesem Werk wiederfinden. Auch meinem treuen Mitarbeiter Jan-Henrik Hempel möchte ich für die Unterstützung bei der Überarbeitung des Praxisteils danken. Vielen Dank auch an den Mitautor Roland Petrasch für die hervorragende Zusammenarbeit und die vielen fruchtbaren Diskussionen quer durch alle Metaebenen.

Auch von mir ein herzliches Dankeschön an den dpunkt.verlag und insbesondere an René Schönfeldt für seine Unterstützung sowie an die Gastautoren Florian Fieber und Karsten Thoms für ihre Mitarbeit.

Zuletzt möchte ich noch meinen Eltern danken, die mich bei meinen ersten Schritten zum Software-Entwickler begleitet und unterstützt haben.