

Teil I

Einführung

1 Einleitung

1.1 Das Thema des Buches

Dieses Buch handelt von *modellgetriebener Softwareentwicklung* oder *Model Driven Software Development*, kurz *MDSD*. Eine etwas ungenauere, aber gängige Bezeichnung für diese Disziplin lautet *Model Driven Development (MDD)*.

Vielleicht werden Sie sich als Leser fragen, warum wir ein solches Buch schreiben. Wir denken, dass modellgetriebene Softwareentwicklung eine große Bedeutung hat und in Zukunft eine noch viel größere haben wird. Es handelt sich im Grunde um die natürliche Weiterentwicklung der Programmierung, wie wir sie heute kennen.

Die Verwendung von Modellen in der Softwareentwicklung hat zwar eine lange Tradition und spätestens seit der Definition der Unified Modeling Language (UML) auch eine zunehmende Verbreitung. Doch handelt es sich dabei in aller Regel »nur« um eine *Dokumentation* von Softwaresystemen, weil es lediglich eine gedankliche Verbindung zwischen Modell und Softwareimplementierung gibt. Diese Form der Modellnutzung in einem Entwicklungsprozess nennen wir *modellbasiert*. Sie birgt zwei gravierende Nachteile in sich: Zum einen sind Softwaresysteme nicht statisch, sondern insbesondere während der ersten Phasen ihres Lebenszyklus zum Teil starken Änderungen unterworfen. Die Dokumentation muss daher sorgfältig angepasst werden – dies kann je nach Detaillierungsgrad sehr aufwändig sein – oder sie wird inkonsistent. Zum anderen tragen solche Modelle nur mittelbar etwas zum Fortschritt bei, da ja erst durch die Interpretation der Softwareentwickler daraus Code entsteht. Diese Gründe veranlassen nicht ganz zu Unrecht viele Programmierer, Modelle als Overhead zu betrachten und sie allenfalls als Zwischenergebnisse anzusehen.

Die *modellgetriebene* Softwareentwicklung geht einen anderen Weg: Modelle sind hier nicht nur Dokumentation, sondern sie sind gleichzusetzen mit Code – die Umsetzung ist automatisiert. Ein Vergleich mit ausgereiften Ingenieursdisziplinen wie dem Maschinenbau lässt die Idee plastisch werden: Stellen Sie sich zum Beispiel eine CNC-Fräse vor, die mit CAD¹-Daten gespeist wird und dieses Modell vollautomatisch in ein entsprechendes Werkstück umsetzt. Oder nehmen

wir eine Produktionsstraße im Automobilbau: Sie geben Ihre Bestellung inklusive Ausstattungsmerkmalen (das »Modell« Ihres Autos) auf, die eigentliche Herstellung ist zu einem Großteil automatisiert.

Diese Beispiele zeigen bereits, dass der fachliche Bereich sowohl für die Modelle als auch für die Herstellungsautomation essenziell ist: Weder die kundenorientierte »Modellierungssprache« (Bestellformulare) eines Automobilherstellers noch seine Produktionsstraße sind z.B. dazu geeignet, Fertighäuser herzustellen. Diesen fachlichen Geltungsbereich bezeichnet man als *Domäne*.

Das Ziel von MDSD ist es daher, domänenspezifische Abstraktionen zu finden und diese einer formalen Modellierung zugänglich zu machen. Dadurch entsteht ein hohes Automationspotenzial für die Fertigung von Software, und dies wiederum lässt sich in den allermeisten Fällen in eine deutliche Produktivitätssteigerung umsetzen.

Auch verbessern sich dadurch Qualität und Wartbarkeit von Softwaresystemen. Zudem können die Modelle von Domänenexperten verstanden werden. Es handelt sich um einen ähnlichen Evolutionsschritt wie bei der Einführung der ersten Hochsprachen in der Ära der Assemblerprogrammierung. Das Adjektiv »getrieben« (»driven«) in MDSD soll im Gegensatz zu »basiert« deutlich machen, dass Modelle bei diesem Paradigma eine zentrale, treibende Position einnehmen und mindestens den gleichen Stellenwert wie Quellcode besitzen.

Um das Konzept der »domänenspezifischen Modelle« erfolgreich umsetzen zu können, werden vor allem zwei Dinge benötigt:

- Domänenspezifische Sprachen, um die Modelle überhaupt formulieren zu können.
- Generatoren, Transformatoren oder Interpreter, welche in der Lage sind, aus domänenspezifischen Modellen Programme zu erzeugen, die auf vorhandenen Plattformen ausgeführt werden können.

Im Rahmen der modellgetriebenen Entwicklung werden zwar oft grafische Modelle verwendet, dies ist jedoch keineswegs zwingend und auch nicht immer opportun – textuelle Modelle sind genauso gut möglich. Typischerweise werden Modelle bei einem modellgetriebenen Vorgehen in programmiersprachlichen Quellcode übersetzt, um anschließend kompiliert und ausgeführt zu werden. Es gibt allerdings auch die Möglichkeit, Modelle zu interpretieren. Beide Ansätze geben einer domänenspezifischen Sprache eine formale Bedeutung.

MDA

Wenn Sie bereits mit der Model Driven Architecture (MDA) der Object Management Group (OMG) vertraut sind, werden Sie nun vielleicht sagen: »Das klingt wie MDA.« Zum Teil ist das auch richtig. MDA hat prinzipiell ähnliche Ansätze,

1. CAD = Computer Aided Design

unterscheidet sich aber in verschiedenen Details – zum Teil auch in der Motivation – und nimmt tendenziell Einschränkungen vor, wie z.B. die Fokussierung auf UML-basierte Modellierungssprachen. Das Ziel von MDA ist in erster Linie Interoperabilität zwischen Werkzeugen und auf längere Sicht die Standardisierung von Modellen für populäre Anwendungsbereiche. MDSD hingegen zielt auf die Bereitstellung von praktisch einsetzbaren Bausteinen für Softwareentwicklungsprozesse ab, welche heute im Zusammenhang mit modellgetriebenen Ansätzen im Allgemeinen einsetzbar sind, und zwar unabhängig von der Werkzeugwahl oder dem Reifegrad der OMG-MDA-Standards.

Die MDA-Standards der OMG können grundsätzlich auch für die MDSD eingesetzt werden. So wird die UML 2 mit ihrem Profile-Mechanismus zur Definition einfacher DSLs verwendet, auch die Transformationssprache QVT (Queries, Views, Transformations), die wir im Anhang B ausführlich beschreiben, kann hilfreich sein.

Die Relation zwischen MDA und MDSD werden wir im Rahmen dieses Buches noch genauer betrachten, im Prinzip kann man aber sagen, dass die MDA eine Standardisierungsinitiative der OMG zum Thema MDSD ist.

1.2 Zielgruppen

Bestimmte Konzepte, Begrifflichkeiten und Grundlagen müssen von allen Projektbeteiligten verstanden werden, sonst lässt sich ein MDSD-Projekt nicht erfolgreich abschließen. Das Buch beschreibt diese Dinge vor allem in den einführenden Kapiteln.

Darüber hinaus richtet sich das Buch besonders an einige spezielle Personengruppen, auf die wir in den folgenden Abschnitten gesondert eingehen.

Architekten

Modellgetriebene Entwicklung betrifft in erster Linie Softwarearchitekten, was sich in dreierlei Hinsicht zeigt. Zunächst erfordert der Ansatz eine klare und formale Definition der architektonischen Konzepte einer Anwendung. Des Weiteren findet MDSD teilweise nicht nur im Rahmen der Entwicklung einer einzelnen Anwendung statt, sondern im Kontext von Produktlinien/Software-Systemfamilien. Diese besitzen ihre eigenen architektonischen Anforderungen, die von den Architekten adressiert werden müssen. Und schließlich kommt eine ganz neue Sichtweise und Art der Entwicklung ins Projekt, weil man eigene Programmiersprachen, Generatoren und Werkzeuge baut, anstatt nur vorhandene Sprachen zu verwenden. Das schlägt sich auch im Entwicklungsprozess nieder, und alle diese Themen betreffen die Arbeit von Architekten.

Entwickler

Das vorliegende Buch beschreibt ein Softwareentwicklungs-Paradigma, und deshalb ist es selbstverständlich, dass die Rolle des Softwareentwicklers eine zentrale Bedeutung hat. MDSD impliziert zum Teil schärfere und klarere Sichtweisen auf Dinge wie die Bedeutung von Modellen, die Trennung von fachlichem und technischem Code, das Verhältnis zwischen Design und Implementierung, Roundtrip-Problematiken, Architektur und Generierung, Framework-Entwicklung, Versionierung und Tests. Richtig eingesetzte MDSD erleichtert die Arbeit des Softwareentwicklers erheblich, vermeidet redundanten Code und steigert die Softwarequalität durch formalisierte Strukturen.

Entscheider/Projektleiter

Die Entscheidung, modellgetrieben zu arbeiten, fußt nicht zuletzt auf betriebswirtschaftlichen Fragestellungen wie dem Kosten-Nutzen-Verhältnis oder der Gewinnschwelle. There is no free lunch; auch modellgetriebene Entwicklung hat ihre Kosten. Unter vielen Umständen rechnet sich ein modellgetriebener Ansatz, unter einigen Umständen ist eher davon abzuraten. Auch wenn dieses Buch einen technischen Schwerpunkt hat, so betrachten wir durchaus auch organisatorische und ökonomische Aspekte, die aus Sicht eines Projektes oder des ganzen Unternehmens relevant sind.

Des Weiteren hat ein modellgetriebenes Vorgehen Auswirkungen auf Projektorganisation, Teamstruktur und den Softwareentwicklungsprozess. Auch darauf gehen wir in diesem Buch ausführlich ein.

1.3 Ziele des Buches

Dieses Buch gibt eine praktische Anleitung für Sie, den Leser, um MDSD in Projekten erfolgreich einzusetzen. Modellgetriebenes Arbeiten ist inzwischen schon eine ganze Weile im praktischen Projekteinsatz, und die damit verbundenen Werkzeuge und Techniken haben einen recht guten Reifegrad erreicht. MDSD ist ein absolut praktikabler Ansatz – und in der Tat häufig der traditionellen Entwicklung überlegen. Wir wollen Sie motivieren, diesen Ansatz alsbald einzusetzen, denn es handelt sich weder um bloße Visionen noch um graue Theorie. Dazu möchten wir Ihnen alles an die Hand geben, was Sie benötigen. Falls Sie MDSD bereits praktizieren, soll dieses Buch ggf. eine Hilfestellung oder Vertiefung für spezifische Fragen oder Bereiche darstellen.

Im Detail verfolgen wir dementsprechend eine ganze Reihe von »Unterzielen« – unabhängig vom Aufbau des Buches –, die wir hier kurz und knapp erläutern möchten:

Zunächst führen wir den theoretischen Rahmen, die Grundkonzepte und die Grundbegriffe ein. Dabei werden wir auch auf andere, verwandte Themen und Ansätze eingehen, wie z.B. die Model Driven Architecture (MDA) der OMG.

Des Weiteren geben wir konkrete Hilfestellung für spezifische, MDSD-relevante Themen. Dazu gehören Metamodellierung, domänenspezifische Sprachen, Codegenerierung, Interpreter und Modelltransformationen sowie die Unterstützung bei der Toolauswahl auf technischer Seite. Sehr wichtig sind uns aber auch organisatorische und prozessbezogene Hilfestellungen. Zusätzlich liefern wir Argumentationen für MDSD aus ökonomischer Sicht.

Auch wenn man ohne einige theoretische Grundlagen nicht auskommen kann, so soll das Buch vor allem praktische Hilfestellungen geben bzw. praxisrelevante Teilmengen der angesprochenen Themen näher beleuchten. Best Practices sowie die Weitergabe von konkreten Erfahrungen (und teils auch persönlichen Meinungen) sind uns wichtig. Im Zentrum der praktischen Sicht stehen die Fallstudien im ersten und zweiten Teil des Buches. Im dritten Teil findet sich eine weitere Fallstudie, die die dort behandelten Themengebiete illustriert.

Wir geben Antworten auf aktuelle Fragestellungen und greifen aktuelle Diskussionen auf. Ein Ausblick auf Trends und Visionen im Kontext von MDSD rundet das Buch ab.

1.4 Abgrenzung

Dies ist kein MDA-Buch. Wir beschreiben zwar die Grundlagen und die Terminologie dieses OMG-Standards sowie die zugrundeliegende Vision, auch geben wir in Anhang A einen Abriss über den Stand der Standardisierung, aber betrachten modellgetriebenes Arbeiten in einem größeren Rahmen. Sekundärliteratur (neben der Spezifikation) zum Thema MDA bietet z.B. [Fra02].

Dieses Buch hat nicht zum Ziel, einen geschlossenen, allgemeingültigen MDSD-Entwicklungsprozess zu definieren. Stattdessen beschränken wir uns auf die Vermittlung von Best Practices, welche sich im Zusammenspiel mit agilen Ansätzen wie Crystal [Coc01] und Methodenbausteinen der Produktlinienentwicklung (siehe Abschnitt 11.5) zur Konstruktion eines auf den jeweiligen Kontext maßgeschneiderten Entwicklungsprozesses anbieten.

1.5 Struktur des Buches und Leitfaden für den Leser

Das Buch beschreibt modellgetriebene Ansätze, wie sie u.a. von den Autoren in der Praxis seit vielen Jahren erfolgreich eingesetzt wurden und werden. Wir beleuchten das Thema im Wesentlichen aus den Perspektiven *Technik*, *Engineering* und *Management*, welche sich in der Aufteilung des Buches wiederfinden:

- **Teil 1 – Einführung:** Dieser Teil enthält die Einleitung, die Sie gerade lesen, sowie eine Erläuterung der wichtigsten MDSD-Grundideen und der zugehörigen Basisterminologie. Dann zeigen wir die konkreten Techniken anhand einer Fallstudie aus dem Bereich E-Business/Web-Applikationen, um einen ersten Vorgeschmack auf die Arbeit in einem MDSD-Projekt zu vermitteln. Anschließend führen wir auf der Grundlage der erworbenen Kenntnisse eine umfassendere MDSD-Begriffsbildung ein. Dieses Kapitel ist wichtig, weil der Rest des Buches auf der dort definierten Terminologie aufbaut. Hier findet sich die konzeptionelle, artefaktbezogene Definition von MDSD. Abgerundet wird der erste Teil durch eine Einordnung respektive Abgrenzung verwandter Themen wie der MDA und Software Factories, aber auch agiler Softwareentwicklung.
- **Teil 2 – Domänenarchitekturen:** Die Domänenarchitektur ist das MDSD-Schlüsselkonzept. Sie beinhaltet unter anderem die Modellierungssprachen für die verschiedenen Domänen und die dazugehörigen Editoren, aber auch den Generator, welcher die Modelle auf eine konkrete Plattform abbildet. Dieser Teil beginnt mit der Einführung eines weiteren Fallbeispiels aus der Versicherungsbranche, bevor in den folgenden Kapiteln dann die wichtigsten Arbeitsschritte zur Konstruktion von Domänenarchitekturen besprochen werden. Dabei beginnt jedes Kapitel mit einem konzeptionellen Abschnitt. Hier werden sowohl Anwendungsfälle als auch bekannte Technologien und Best Practices diskutiert. Im letzten Abschnitt jedes Kapitels wird das Besprochene anhand des Versicherungsbeispiels praktisch umgesetzt. Die Kapitel behandeln die Themen Metamodellierung, domänenspezifische Sprachen, Zielarchitekturen, Codegenerierung, Interpreter und Modelltransformationen.
- **Teil 3 – Prozesse und Engineering:** In diesem Teil behandeln wir die prozessualen MDSD-Aspekte und Engineering-Themen, die durch MDSD eine spezifische Ausprägung erhalten. Spätestens hier wird deutlich, dass MDSD nicht einfach nur eine Technik ist. Wir stellen eine Reihe von Best Practices vor, die zu einem praktischen und pragmatischen Entwicklungsprozess kombiniert werden können. Anschließend gehen wir auf die Themen Testen und Versionierung ein. Es folgt eine Behandlung von Product Line Engineering, das durch ein reales Projektbeispiel begleitet und durch eine ebenso reale Fallstudie im Anschluss konkretisiert wird.
- **Teil 4 – Management:** Dieser Teil des Buches wendet sich primär an IT-Manager und Projektleiter und kann weitgehend unabhängig vom Rest des Buches gelesen werden. Wir gehen dabei auf betriebswirtschaftliche und organisatorische Aspekte ein und behandeln Adaptionstrategien. Das erste Kapitel dieses Teils enthält einen FAQ²-Abschnitt zum Thema MDSD.

2. FAQ = Frequently Asked Questions

- **Anhang:** Der Anhang enthält ausführlichere Darstellungen des MDA- und des QVT-Standards der OMG sowie einen Index und das Literaturverzeichnis.

Grundsätzlich haben wir uns beim Aufbau des Buches große Mühe gegeben, dass es beim sequenziellen Lesen – trotz der inhaltlich unvermeidlichen zyklischen Abhängigkeiten – didaktisch sinnvoll aufgebaut ist. Da wir jedoch zum Teil stark unterschiedliche Zielgruppen ansprechen, wird sicherlich bei dem einen oder anderen Leser der Wunsch bestehen, (zunächst) selektiv zu lesen. Dazu ein paar Hinweise:

Der primär technisch interessierte Leser, der eventuell bereits über einige Vorkenntnisse verfügt, kann ggf. direkt mit Kapitel 3 beginnen, um dort die im Buch verwendete Begrifflichkeit kennenzulernen. Die technischen Fragestellungen werden im Detail im gesamten zweiten Teil des Buches diskutiert. Im Anschluss können die verschiedenen Techniken durch die Lektüre des dritten Teils vertieft werden.

Falls Sie sich fragen, welche ökonomischen Vorteile MDSD denn konkret bietet, bevor Sie sich im Detail mit der Frage auseinandersetzen wollen, was MDSD ist, können Sie das Kapitel 15 vorziehen.

1.6 Die zweite Auflage

Für die zweite Auflage wurden große Teile des Buches überarbeitet, um auf aktuelle Entwicklungen unter besonderer Berücksichtigung der Eclipse-Plattform einzugehen. Das betrifft vor allem den zweiten Teil, genauer die dortige Behandlung der einzelnen Bestandteile einer Werkzeugkette für modellgetriebenes Arbeiten.

Zum einen gibt es seit der ersten Auflage neue Erfahrungen und bessere Werkzeuge, was einige neue Möglichkeiten eröffnet. Das sind vor allem Modelltransformations-Engines, mit denen man inzwischen recht leicht das Überwinden der Abstraktionslücke zwischen Plattform und Modell auf mehrere Schritte aufteilen kann. Das ermöglicht unter anderem elegantere, leistungsfähigere und besser wiederverwendbare Cartridges. Außerdem ist die IDE-Integration der MDSD-Werkzeuge zunehmend von Bedeutung, was sich in den Beschreibungen der einzelnen Techniken niederschlägt.

Zum anderen ist das Generator-Framework `openArchitectureWare` inzwischen ein Bestandteil der Eclipse-Plattform, wo eine zunehmend leistungsfähige Sammlung von MDSD-Werkzeugen entsteht, die z.B. auch EMF (ein sehr nützliches Modellierungs-Framework) oder ATL (eine Modelltransformations-Engine) umfasst. Das haben wir zum Anlass genommen, ausführlicher als in der ersten Auflage auf konkrete Werkzeuge einzugehen und dabei den Schwerpunkt auf die Werkzeuge der Eclipse-Plattform zu legen. Dabei sei darauf hingewiesen, dass die Werkzeuge zwar in Java implementiert sind, dass man aber mit ihnen MDSD für jede beliebige Zielsprache betreiben kann.

1.7 Webseite zum Buch

Einige Themen, die wir im Buch behandeln, sind derzeit einer starken Evolution unterworfen. Andere Themen konnten wir aus Platzgründen nur streifen. Auf der Webseite zum Buch www.mdsd-buch.de stellen wir daher aktuelle Informationen und interessante Links für Sie zusammen.

1.8 Danksagungen

Ein Buch, wie Sie es hier vor sich haben, ist mit viel mehr Aufwand verbunden, als man gemeinhin denkt. Ohne die Unterstützung einer ganzen Reihe von Leuten wäre die Fertigstellung noch deutlich schwieriger gewesen. Deshalb ist es an der Zeit, dass wir unseren Dank aussprechen.

Da wären zunächst unsere Gastautoren Jorn Bettin, Simon Helsen und Michael Kunz. Jorn hat vor allem zum Management-Teil des Buches sehr viel beigesteuert, uns bei der Begriffsbildung unterstützt und wichtige Hinweise zur Struktur des Buches gegeben. Simon war uns dank seinem einzigartigen Einblick in die QVT-Sprachfamilie eine große Hilfe beim Verfassen des Kapitels über Modelltransformationen. Er hat weiterhin den Anhang über die QVT geschrieben. Michael hat einen großen Teil des Kapitels 12 zum Testen in der Anwendungsentwicklung verfasst und wurde dabei von Carsten und Stefan Sensler unterstützt.

Last – but not least – möchten wir auch dem dpunkt.verlag danken, insbesondere René Schönfeldt, unserem Lektor. Er hat nicht nur Verständnis für unsere mehrfachen Terminverschiebungen aufgebracht, sondern ist uns vor allem gegen Ende mit der einen oder anderen »unüblichen« Vorgehensweise entgegengekommen.

Mit der zweiten Auflage dieses Buches haben wir zwei neue Hauptautoren »an Bord« – Sven und Arno. Ihnen möchten Thomas und Markus ihren ganz besonderen Dank aussprechen, denn ohne ihr Engagement wäre eine so umfassende Überarbeitung und Aktualisierung des Buches nicht möglich gewesen.

Thomas bedankt sich außerdem bei seiner Frau Anja und seinen Kindern, die ihm den notwendigen Rückhalt gegeben und ihn durch große Rücksichtnahme unterstützt haben.

Sven möchte sich besonders bei seiner Frau Kristina und seinem Sohn Cedric bedanken, die ihn stets mit viel Liebe und Geduld bei der Arbeit unterstützen.

Arnos besonderer Dank geht an seine Freunde – speziell den Hauskreis –, die mit aufbauenden Worten und Verständnis für gelegentliches Gestresstsein besonders gegen Ende des Buchprojektes eine echte Unterstützung waren.