

Model Driven Architecture Mit MDA aus der „Softwarekrise“?

Die Komplexität eingebetteter Systeme wächst beständig. Das gilt in besonderem Maße für Anwendungen aus dem Automotive-Bereich. Der technologische Fortschritt bei der Hardware ermöglicht auch immer ausgefeiltere Softwareanwendungen. Die Zeiten, in denen eine Handvoll Programmierer in der Garage ihre Programme entwickelten, sind längst passé. Hunderte von Entwicklern, mitunter über die ganze Welt verstreut, sind heute mit der Entwicklung eines einzigen Systems beschäftigt. In mancherlei Hinsicht ist die Komplexität der Softwaresysteme kaum noch zu bewältigen. Die Folge: Schlechte Softwarequalität und verärgerte Kunden. Manch Marktteilnehmer spricht sogar bereits wieder von einer „Softwarekrise“.

Automobilzulieferer haben neben einem äußerst heterogenen Hardwaremarkt die Schwierigkeit, daß sie eine Vielzahl von Varianten ihrer Produkte für die Hersteller anbieten müssen. Jedes Antiblockiersystem, zum Beispiel, besitzt eine maßgeschneiderte Software. Eine große Herausforderung für jedes Konfigurationsmanagement. Hinzu kommen all die anderen Anwendungen: ESP, ASR, ACC und immer wieder neue Entwicklungen.

Evolution statt Revolution

Hinter der *Model Driven Architecture* (MDA) steckt der Grundgedanke, aus generischen Modellen, die die technischen Anforderungen einer Zielplattform noch nicht berücksichtigen, speziellere Modelle abzuleiten, aus denen der Code für die Zielplattform automatisch generiert werden kann. Für ein System im Automotive-Bereich könnte das bedeuten, daß, unabhängig von den technischen Anforderungen eines bestimmten Fahrzeugs, ein Basissystem mit der Grundfunktionalität modelliert wird, welches unter Berücksichtigung der technischen Gegebenheiten zu einem Modell für ein bestimmtes Fahrzeug verfeinert wird. Der Vorteil, der daraus entsteht, liegt darin, daß ein plattformunabhängiges Basismodell beliebig oft, für verschiedene Fahrzeugtypen und -hersteller wiederverwendet werden kann. Das spart Zeit und Kosten, und die Qualität steigt.

Die MDA ist der nächste Schritt der *Object Management Group* (OMG) zu mehr Plattformunab-

hängigkeit durch einen höheren Grad an Abstraktion. Aus Modellen soll soviel Code wie möglich für die Zielplattform automatisch generiert werden.

MDA ist keine Revolution, sondern eine Evolution. Nach Assembler, Hochsprachen wie C, OOP mit C++ und Java und UML bedeutet MDA der nächste logische Schritt auf dem Weg des Abstraktionsgedankens. Nach der Assembler-Programmierung kamen die Compiler, die C in Assembler übersetzten; es folgten die objektorientierten Programmiersprachen und schließlich die UML-basierten CASE-Tools, die Modelle in Hochsprachen übersetzten. In Zukunft werden Modelle in Modelle übersetzt. Grundlage bilden die UML 2 und andere domänenspezifische Modellierungssprachen. Die folgende Abbildung zeigt die Vorgehensweise:

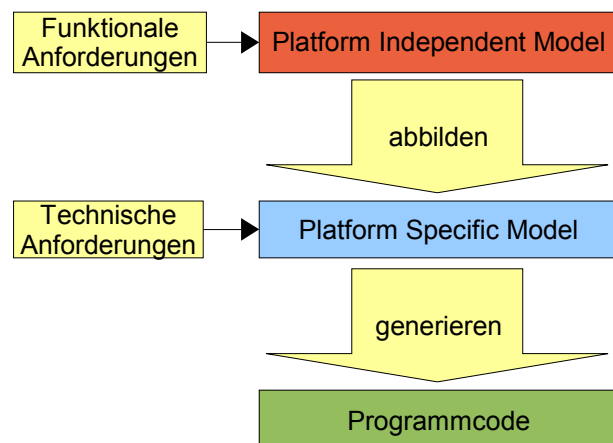


Abb. 1: Vom Modell zum Programmcode

Aus plattformunabhängigen Modellen werden vom Entwickler plattformspezifische Modelle abgeleitet, aus denen der Code für die Zielplattform automatisch generiert werden kann.

In einem *Plattform Independent Model* (PIM) werden die reinen funktionalen Anforderungen zusammengefaßt. Das PIM ist von den Hardware- und Softwareanforderungen der Zielplattform (z.B. Betriebssystem und Programmiersprache) unabhängig. Erst in einem weiteren Schritt fließen die technischen Anforderungen des Zielsystems in das Modell mit ein und münden in einem *Plattform Specific Model* (PSM). Aus dem PSM wird schließlich der Code für die Zielplattform automatisch generiert.

Beispielsweise könnte bei einem ABS das PIM sämtliche funktionalen Anforderungen eines Basis-Antiblockiersystems abdecken; also alle Funktionen in Form von (UML-)Klassen und Methoden, die in jedem System des Zulieferers enthalten sein müssen. Durch Vererbung und Ein-

bindung der technischen Anforderungen wird das plattformunabhängige Modell zu einem plattform-spezifischen Modell verfeinert. Das PSM enthält alle Klassen und Methoden, die das ABS für ein bestimmtes Fahrzeug eines bestimmten Herstellers ausmachen. Auf diese Weise kann dasselbe Modell „Basis-ABS“ für verschiedene Hersteller, Fahrzeuge und Fahrzeugtypen wiederverwendet werden. Durch den Einsatz von Plattformprofilen kann das Mapping eines plattformunabhängigen Modells auch weitestgehend automatisiert erfolgen.

Die Vorteile, die durch den Einsatz der MDA entstehen liegen auf der Hand:

- Beherrschung der immer größer werdenden Systemanforderungen durch Trennung von plattform-spezifischen und funktionalen Anforderungen.
- Vereinfachter Umgang mit Varianten durch Vererbung und Modellverfeinerung
- Wiederverwendbarkeit von Design und Implementierung und damit schnelle Reaktionsmöglichkeit auf veränderte Anforderungen des Kunden
- Bessere Qualität des Entwicklungsprozesses durch eine verbesserte Kommunikation aller Projektbeteiligten
- Geringere Entwicklungszeiten und -kosten
- Höhere Softwarequalität und damit größere Kundenzufriedenheit

Die MDA ist nicht auf eine bestimmte Modellierungssprache beschränkt, sondern sieht auch anwendungsspezifische Sprachen vor. Die UML in den Versionen 1.x war für viele Anwender nicht immer ganz frei von Widersprüchen; mit UML 2 sollten diese Mängel endgültig der Vergangenheit angehören. Mit ihren Erweiterungen bietet sie sich auch zur Modellierung von Echtzeitsystemen an.

Werkzeuge

Die meisten verfügbaren Werkzeuge sind für *Enterprise Applications* vorgesehen, aber auch die Anzahl für den Embedded- und Echtzeit-Bereich nimmt zu. Zu erwähnen wären hier z.B. die Entwicklungswerkzeuge Ameos von der Firma Aonix, das Real-time Studio von Artisan, Rhapsody von I-Logix oder TAU von Telelogic. Die Werkzeuge sind in ihrem Funktionsumfang und in ihrer Umsetzung von MDA nur schwer zu vergleichen. Eine genaue Evaluierung ist daher vor ei-

ner Investition dringend zu empfehlen. Die Unterstützung von UML 2 sollte inzwischen obligatorisch sein. Wer Modelle zwischen verschiedenen Entwicklungswerkzeugen austauschen möchte, sollte auf Unterstützung des XML-basierten Datenformats XMI – möglichst in der Version 2.0 – achten. Unterstützt werden die unterschiedlichsten Zielplattformen: VxWorks, Windows CE, pSOS, QNX, um nur einige zu nennen. Die Qualität des generierten Codes hat sich in den vergangenen Jahren stark verbessert.

Fazit: MDA ist nicht nur für Enterprise-Applikationen von Interesse, sondern auch für eingebettete Systeme und speziell für Automotive Systems mit ihrer Variantenvielfalt. MDA ist keine Revolution, sondern vielmehr eine Weiterentwicklung zu einem höheren Grad an Abstraktion in der Softwareentwicklung. Dabei ist MDA im Grunde nicht wirklich neu, handelt es sich doch um eine Kombination bestehender OMG-Standards. Grundlage bildet die Systemmodellierung mit UML. Durch die Aufteilung in plattformunabhängige und plattform-spezifische Modelle ergeben sich eine Reihe von Vorteilen: Die höhere Qualität von Produkten und Entwicklungsprozeß sowie ein vereinfachter Umgang mit Varianten seien an dieser Stelle noch einmal hervorgehoben. Häufig müssen interne Widerstände bei der Einführung modellbasierter Entwicklungsmethoden überwunden werden. Viele Entwickler teilen immer wieder mit, daß sie sich durch die Modellierung in ihrer Kreativität eingeschränkt fühlen. Es muß aber darauf hingewiesen werden, daß Softwareentwicklung aufgrund der wachsenden Komplexität bald nicht mehr handhabbar sein wird. Dennoch darf MDA auf keinen Fall über die Köpfe der Entwickler hinweg eingeführt werden, sondern nur mit ihnen. Das ist ein langwieriger Prozeß und sollte schrittweise durchgeführt werden. Nur so kann der Weg aus der „Softwarekrise“ gelingen.
(bar)

Weitere Informationen

Bei weiteren Fragen stehen wir Ihnen gerne zur Verfügung.

Ingenieurbüro Barheine
Albstraße 47
76275 Ettlingen

Tel.: 0 72 43 / 52 37-67
Fax.: 0 72 43 / 52 37-68

E-Mail: kontakt@barheine.de

Web: <http://www.barheine.de>