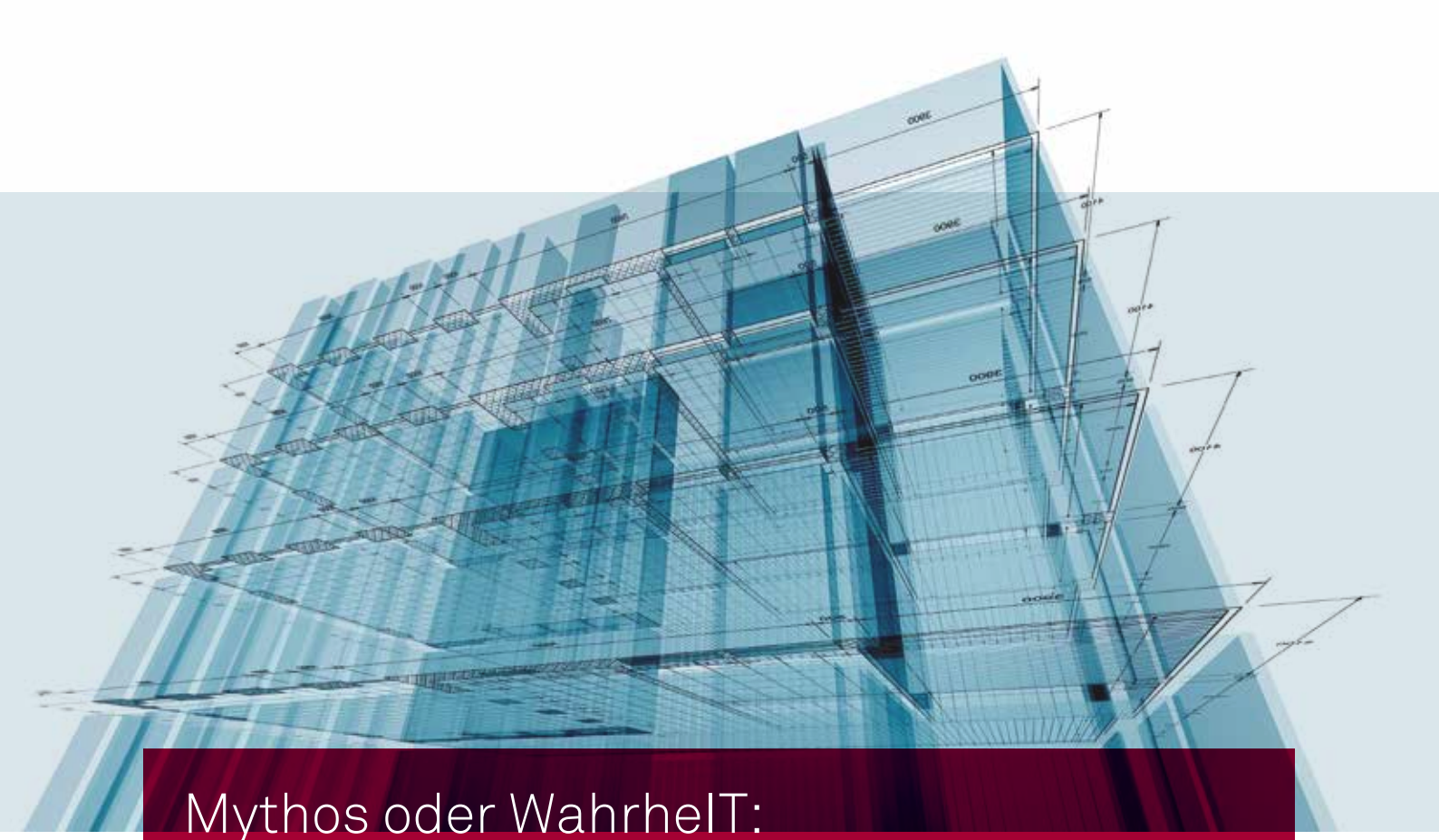

Mythos oder WahrheIT: MODEL-DRIVEN ARCHITECTURE (MDA)

Unsere Kolumne „Mythos oder Wahrheit“ nimmt in dieser Ausgabe mit Model-Driven Architecture (MDA) einen Trend unter die Lupe, der bis in das Jahr 2000 zurückreicht. In den letzten Jahren ist es jedoch still um den einstigen Hoffnungsträger geworden. Allerdings gibt es mehrere verwandte Trends aus jüngerer Zeit, die das Erbe des MDA-Ansatzes aufgreifen. Ein genauerer Blick lohnt sich daher auch heute noch.

| von **ANDREAS RAQUET**

In der modellgetriebenen Architektur wird ein Softwaresystem nicht von Hand entwickelt, sondern zunächst auf fachlicher Ebene modelliert. Daraus werden im Anschluss ein technisches Modell und erst daraus der Programmcode generiert. Der Modellierer konzentriert sich dabei vollständig auf die Spezifikation des fachlichen Datenmodells und der Funktionalität. Der eigentliche Programmcode – oft von wiederkehrenden Codefragmenten und von technologischen Anteilen wie Logging oder Fehlerbehandlung durchsetzt (so genannter boiler-plate code) – wird durch eine entsprechende Software automatisch generiert. Das spart einerseits viel Aufwand gegenüber der manuellen Softwareentwicklung. Andererseits verspricht man sich von der Trennung des fachlichen Modells von der technischen Umsetzung eine weitgehende Plattformunabhängigkeit des Systems.

Obwohl Modellierung und Generierung auch heute noch wichtige Elemente der Softwaretechnik darstellen, muss der eigentliche MDA-Ansatz als weitgehend gescheitert angesehen werden. Uns ist kein größeres Softwareprojekt bekannt, das auch nur annähernd vollständig als MDA implementiert wurde.

WARUM IST DAS SO?

Zunächst basiert MDA auf der Idee, die gesamte Geschäftslogik in Form eines UML-Modells auszudrücken. Da die UML jedoch dazu nicht ausdrucksstark genug ist, muss sie über sogenannte UML-Profile erweitert werden. Zudem korrelieren die unterschiedlichen Diagrammtypen der UML nur lose miteinander. Um damit die Struktur und das Verhalten eines IT-Systems bis auf die Ebene der Ausführung modellieren zu können, wird ein Einsatzkonzept der UML benötigt, das weit über das gängige „Zeichnen von Überblicksdiagrammen" hinausgeht.

Die Erfahrung zeigt, dass insbesondere Fachbereiche schon mit der klassischen UML-Modellierung überfordert sind. Doch MDA kann nur dann funktionieren, wenn die Modellierer die zugrunde liegenden softwaretechnischen Konzepte, die Notation und das oben genannte Einsatzkonzept exzellent beherrschen. Unschärfen in der Modellierung münden sonst automatisch in Softwarefehlern. Ein Fachbereich ohne große software-technische Affinität kann das in der Regel nicht leisten. Stattdessen werden Business-Analysten benötigt, die sich intensiv der Modellierung widmen.

> *„MDA – The Architecture of Choice for a Changing World"*
>
> Heilsversprechen auf der Website der OMG

Haben diese ein (hoffentlich) vollständiges und korrektes Modell erstellt – testen lässt sich das auf der Modellebene nicht –, zeigt sich schon die nächste Herausforderung: die Generierung des Codes. Die Herausforderung besteht dabei nicht in der erstmaligen Generierung des Code, sondern darin, Modell und Code dauerhaft konsistent zu halten. Insbesondere muss es möglich sein,

- Änderungen am Modell in bereits generierten Code zu übertragen und
- Änderungen am Code wieder in das Model einzulesen.

Durch Änderungen an Code und Modell entsteht ein Wechselspiel aus Codegenerierung und Einlesen des Quellcodes in das Modell, das sich in der Realität als kaum beherrschbar erwiesen hat. Immer wieder geht Quellcode beim Wiedereinlesen verloren, oder das Modell wird beschädigt. Selbst eine enge Integration von grafischer Modellierung und textlicher Programmierung in einer gemeinsamen Entwicklungsumgebung konnte dieses Problem nicht lösen. Das erfolgreichste Produkt in diesem Sektor, Borland Together, hat keine Marktrelevanz mehr.

Hinzu kommen auch nach 15 Jahren noch Probleme im praktischen Umgang mit den Modellen: Versionierung und Änderungsverfolgung, Reviews sowie die verteilte Entwicklung an einem gemeinsamen Modell sind, anders als in der realen Entwicklungspraxis, in den Modellierungswerkzeugen oft nur unbefriedigend gelöst.

DAS ERBE VON MDA

Die genannten Schwierigkeiten haben dazu geführt, dass Softwaresysteme, die vollumfänglich auf MDA basieren, heute kaum anzutreffen sind. Was bleibt also vom Hoffnungsträger des frühen 21. Jahrhunderts? Attraktiv bleibt vor allem der Generierungsansatz – in jüngerer Zeit sind zwei Ausprägungen bemerkenswert:

- **Die modellgetriebene Softwareentwicklung („Model-Driven Development", MDD)** generiert Code nicht aus einem UML-Modell, sondern aus einer textlichen Form, einer sogenannten domänenspezifischen Sprache („Domain-specific Language", DSL). Die in der DSL formulierten Ausdrücke stellen ebenfalls ein Modell dar, allerdings kann die – domänenspezifische – DSL sehr viel einfacher gehalten werden als die – universelle – UML. Insbesondere kann sie den Sprachgebrauch des Fachbereichs nachbilden, sodass der Fachbereich keine technische Notation erlernen muss.

- **Die datenmodellgetriebene Generierung** erzeugt Code aus einem Datenmodell. Dieser Ansatz ist vor allem für Anwendungen geeignet, die fast ausschließlich Dienste zur Datenpflege anbieten, wie sie häufig im Bereich der Stammdaten vorkommen. Solche Anwendungen sind tatsächlich durch das Datenmodell weitgehend definiert: Dienste und Masken können daraus abgeleitet werden. Vor allem „Ruby on Rails" und sein Java-Ableger „Groovy on Grails" haben das mit Erfolg demonstriert.

Aber auch die Ausführung grafischer Modelle ist weiterhin von Bedeutung, wenn auch nicht im Bereich der UML. Moderne Business-Process-Management-Systeme (BPMS) modellieren ihr Geschäftsprozesse mittels der BPMN (Business Process Model and Notation). Die Notation wurde – anders als UML – in der Version 2 explizit auf die Formulierung ausführbarer Modelle ausgelegt und wird von Fachbereichen gut akzeptiert. Ob ein Fachbereich damit allerdings ausführbare Prozessmodelle formulieren kann, darf weiterhin bezweifelt werden. Dieser Frage wird eine zukünftige Ausgabe dieser Kolumne nachgehen.

> *Some people think that MDA [...] is nothing more than the Night of the Living Case Tools.*
>
> Martin Fowler

Darüber hinaus haben UML-Modelle natürlich auch weiterhin ihren Platz in der klassischen Softwareentwicklung. Teilweise werden hier auch punktuell Klassenrümpfe und Interfaces generiert. Die Generierung ist aber derart rudimentär, dass dieser Ansatz nicht viel mit MDA zu tun hat. Auch Modelltransformationen findet man immer wieder, beispielsweise beim Erzeugen der XML-Schemata aus UML-Modellen im Rahmen der XÖV-Standardisierung. Aber von MDA sollte man auch hier nicht sprechen.

ZUSAMMENFASSUNG

Ob MDA gescheitert oder erfolgreich ist, liegt wohl im Auge des Betrachters. Klar ist aber, dass der Ansatz in der Praxis nicht in Reinform anzutreffen ist. Dazu waren die beiden wesentlichen Elemente – Modellierung und Generierung – wohl zu schwierig und bisher einfach noch zu unreif. Allerdings haben sie sich in jüngeren Trends manifestiert – sowohl in BPMN als auch in MDD – und haben so auch heute noch ihre Bedeutung. Wir sehen, dass MDA also ein durchaus erfolgreiches Erbe hervorgebracht hat. Offenbar war der Ansatz nicht grundsätzlich falsch, sondern nur vor seiner Zeit – und sicher auch idealisiert.

Ähnliches können wir derzeit auch bei der serviceorientierten Architektur (SOA) beobachten, die letztlich in einer kleineren, bescheideneren Version im Deckmantel der Microservices gerade neu erfunden wird. ●

ANSPRECHPARTNER – ANDREAS RAQUET
Principal IT-Consultant
Public Sector Solutions Consulting
- +49 711 94958-693
- andreas.raquet@msg-systems.com