

UML 2 und MDA

Markus Völter, voelter@acm.org, www.voelter.de

Modellgetriebene Entwicklung und MDA

Model-Driven Architecture (MDA) ist der Ansatz der OMG zur Modellgetriebenen Softwareentwicklung (MDSD). Modellgetriebene Softwareentwicklung [SV05] zielt auf die automatische Erstellung von Programm Quellcode aus domänenspezifischen Modellen. Um dies zu ermöglichen, sind drei Dinge essentiell:

- Für die betreffende Domäne müssen passende Modellierungssprachen zur Verfügung stehen – natürlich mit entsprechendem Toolsupport.
- Diese damit erstellten Modelle müssen dann mittels Transformationen bzw. Codegenerierung in ausführbaren Quellcode überführt werden. Auch hier sind Tools essentiell.
- Damit das ganze effektiv funktioniert braucht man auch eine ausgereifte Plattform, für die man den auszuführenden Code generiert.

Dieses Vorgehen hat einige bestechende Vorteile: Fachliche Aspekte (im Modell) und technische Aspekte (in den Transformationen) sind gut getrennt, ein gewisses Maß an Plattformunabhängigkeit wird erreicht. "Langweilige", stereotype Programmieraufgaben werden durch Transformationen automatisiert. Fachexperten können bei der Modellierung vernünftig mitarbeiten. Und die Qualität der Codes wird verbessert, da Generatoren keine Flüchtigkeitsfehler machen.

Wie oben erwähnt, ist die MDA ein OMG Standard zum Thema Modellgetriebene Softwareentwicklung. Der Fokus liegt dabei – wie immer bei der OMG – auf der Unabhängigkeit von Realisierungsplattformen bzw. deren Interoperabilität. Die Unabhängigkeit von Realisierungsplattformen wird durch die Beschreibung der Fachlichkeit im Modell (frei von technischen Aspekten) erreicht. Die OMG geht aber noch einen Schritt weiter: Angestrebt wird auch die Interoperabilität der Tools für Modellgetriebene Entwicklung. Dies erfordert also insbesondere die Standardisierung von Modellierungs- und Transformationssprachen.

Die Rolle der UML

Die Modellierungssprache die die OMG im Rahmen der MDA zu verwenden gedenkt ist – wen wundert's – UML. Wichtig ist es zu verstehen, dass eine für MDSD/MDA geeignete zwei Ansprüchen genügen muss:

- Sie muß erstens die Konzepte der betreffenden Domäne repräsentieren können, und muss

- Zweitens formal und rigide definiert sein, sonst kann sie nicht automatisch mittels Transformationen in ausführbaren Code überführt werden.

Die erste Anforderung ist in der UML *per se* nicht erfüllt – die Domäne der UML ist OO-Softwareentwicklung, nicht irgendeine Fachlichkeit. Die UML bietet allerdings mittels Profilen die Möglichkeit, an eine bestimmte Domäne angepasst zu werden. Im Rahmen der MDA kommen also in erster Linie entsprechende UML Profile zum Einsatz.

Die zweite Anforderung war bei UML 1.x auch nur sehr bedingt erfüllt. Deswegen war eines der wichtigsten Ziele der UML 2 die stärkere Formalisierung und die Bereinigung semantischer Widersprüche bzw. Schwachstellen. Das Ziel wurde mit UML 2 auch erreicht – was man unter anderem daran erkennen kann, dass die UML nun komplett mittels MOF und OCL definiert ist. MOF ist das Metametamodell der OMG und dient damit zur Definition von Modellierungssprachen. OCL ist eine textuelle Sprache zur Definition von Constraints die nicht mittels der grafischen Syntax der MOF bzw. der UML ausdrückbar sind.

Die UML hat aber in Version zwei aus Sicht von MDSD/MDA noch weitere wichtige Features hinzubekommen. Besonders erwähnenswert sind beispielsweise

- Composite Structure Diagramme, mit denen sich hierarchische Strukturen vernünftig darstellen lassen,
- Oder Action Semantics, mit denen man das Verhalten von Modellinstanzen fast wie in einer 3GL beschreiben kann, natürlich integriert mit dem restlichen Modell.

Fazit

Die Praxis zeigt, dass sich die UML 2 als Basis für MDSD und MDA recht gut eignet, auch wenn viele Tools noch weit davon entfernt sind, das Potential in dieser Hinsicht voll auszuschöpfen. Es ist mir allerdings wichtig darauf hinzuweisen, dass man trotzdem nicht alle Aspekte eines Systems mit der UML beschreiben sollte; das prägnanteste Beispiel hierfür sind GUIs bzw. GUI Layout. Man wird sicherlich nicht auf die Idee kommen, diese mittels UML zu modellieren. Wie immer ist es also nützlich, über den Tellerrand (hier: UML) hinaus zu schauen.

Referenzen

[SV05] Stahl, Völder, Modellgetriebene Softwareentwicklung – Techniken, Engineering, Management, dPunkt, 2005