Maximilian Eibl, Martin Gaedke. (Hrsg.): INFORMATIK 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 2589

TUT03 MDA@Work – Prototyping mit MDA

Siegfried Nolte

Abstract: Die Model Driven Architecture ist ein Konzept der Object Management Group für einen geschlossenen modellgetriebenen Softwareentwicklungsansatz, bei dem in einem iterativen Vorgehen permanent Modellierung und Modelltransformation zur Anwendung kommen. In diesem Beitrag soll auf diese Weise exemplarisch ausgehend von einem einfachen UML Klassendiagramm Schritt für Schritt eine prototypische Anwendung entwickelt werden. Ziel dabei ist, auf manuelle Codierung so weit wie möglich zu verzichten. Zum Einsatz kommen die von der OMG spezifizierten formalen Sprachen und Konzepte UML, QVT, M2T; das Ergebnis ist eine Java-Applikation.

Keywords: Modellgetriebene Softwareentwicklung, Modeldriven Architecture, formale Modellierung, Modeltransformation, Model-nach-Text Transformation, Prototypgenerierung

MDA@Work – Motivation und Vision

Mit den Veröffentlichungen zum Thema MDA [MDA14], erstmalig im Jahre 2001, wurde von der Object Management Group (OMG) ein geschlossenes Konzept zur Anwendungsentwicklung auf der Grundlage von formalen Modellen vorgestellt. Dieser Gedanke hat sich im Laufe der Zeit in vielfältige Richtungen ausgebreitet und ent-wickelt, wobei das Schwergewicht oft auf der modellgetriebenen Softwareentwicklung lag, die mehr die Generierung von Softwarekomponenten aus Modellen zum Inhalt hat. MDA jedoch ist mehr, MDA kann mehr, wie in diesem Tutorium gezeigt werden soll. Im Sprachgebrauch der OMG beschreibt MDA den Begriff der *Model Driven Architecture*, der nach Ansicht des Autors pragmatisch im Sinne der „architektur-zentrierten modellgetriebenen Anwendungsentwicklung“ übersetzt werden kann. Im Rahmen dieses Workshops soll das an einem praktischen Beispiel erläutert und begründet werden.

MDA umfasst einen iterierenden Prozess, bestehend aus Modellierung und Modelltransformation bis hin zur Code-Erzeugung oder, im Sprachgebrauch der OMG, *Model-to-Text Transformation*. Dieser Prozess besteht grundsätzlich aus den Phasen

- Modellierung mit formalen Modellierungssprachen wie zum Beispiel der UML [UML15]
- Modelltransformation zum Beispiel mit QVT *Operational Mappings* [Nol10] oder *Relations Language* [Nol09]
- „Modell nach Text“-Transformation zum Beispiel mit der *Model to Text Transformation Language* [M2T08]

In diesem Beitrag soll das Vorgehen vertieft und an einem durchgängigen Beispiel ein systematischer Weg einer modellgetriebenen Anwendungsentwicklung aufgezeigt werden. Das Projekt MDA@Work hat zum Inhalt, einen in Form eines UML-Klassen-

doi:10.18420/in2017_263

diagramms abgebildeten Ausschnitt der realen Welt, im Weiteren als Domäne bezeichnet, soweit zu transformieren und mit Architekturinformationen zu ergänzen, dass letztendlich eine prototypische lauffähige Anwendung generiert werden kann. Dabei steht die Modellierung im Vordergrund. Der Anteil der Programmierung soll so gering wie möglich sein. Das Ziel ist sogar, im Sinne der Entwicklung eines ersten lauffähigen Prototyps einer Anwendung auf Programmierung gänzlich verzichten zu können.

Das Projekt - Vorgehen

Der Entwicklungsweg im Projekt MDA@Work führt über die klassischen Phasen einer Anwendungsentwicklung, die da sind Analyse der Anforderungen und Beschreibung der Domäne, Grobdesign eines Anwendungssystems unter Berücksichtigung von festgelegten Architekturaspekten, Feindesign des Systems unter Berücksichtigung der Implementierungslandschaft und der betrieblichen Infrastruktur, Generierung einer Java-Anwendung. Im Sinne des MDA-Konzeptes sollen diese Phasen bezeichnet werden als

1. CIM *Computation Independent Modeling*: Analyse der Anforderungen und Abbildung der Domäne „wie sie ist“, also losgelöst von jeglicher Überlegung eines Einsatzes von rechnergestützten Systemen.
2. PIM *Platform Independent Modeling*: Dieser Schritt beschäftigt sich im Wesentlichen mit der Analyse des CIM-Modells; hiermit muss das Verständnis über die Gegebenheiten der Domäne einem später Anwendungsentwickler vermittelt werden können. Zudem wird das CIM-Modell mit ersten Aspekten einer gegebenen Anwendungsarchitektur ergänzt.
3. PSM *Platform Specific Modeling*: Ergänzung und Überarbeitung eines vorliegenden Modells unter plattform- und implementierungsspezifischen Aspekten, zum Beispiel Datentypen, Signaturen der Operationen, Festlegen von Schlüsseln etc.
4. IM *Implementation Modeling*: Abschließende Ergänzung des Modells um Standardoperationen, MTL-Transformation eines Prototyps; dieser soll nahezu unmittelbar und ohne weitere Codierung ausführbar sein.

Unterstützt werden die Phasen durch Zwischenschritte CIM2PIM, PIM2PSM, PSM2IM, in denen Modelltransformation und Code-Generierung vorgenommen wird. Hierzu wird eine geeignete domänenspezifische Sprache [Fow11] mit Hilfe eines UML-Profils bereitgestellt. Das Projekt wird ausschließlich mit Mitteln und Werkzeugen aus dem Open Source Umfeld entwickelt, der Eclipse Modeling Workbench [Ecl17], bestehend aus der Eclipse Java-Entwicklungsumgebung, dem Papyrus UML Modeler, der QVT Operational SDK und Acceleo Common MTL.

Literaturverzeichnis

[Ecl17] Eclipse Modeling Project, Release Neon. http://www.eclipse.org/modeling/, 2017

[Fow11] Fowler, M.: Domain-Specific Language. Addison-Wesley, 2011

[Nol09] Nolte, S.: QVT – Relations Language. Springer-Verlag, 2009

[Nol10] Nolte, S.: QVT – Operational Mappings. Springer-Verlag, 2010

[QVT08] OMG, MOF Query/Views/Transformation (QVT) - Version 1.0. http://www.omg.org/spec/QVT/1.0/, 2008

[M2T08] OMG, MOF Model To Text Transformation Language (M2T) - Version 1.0. http://www.omg.org/spec/QVT/1.0/, 2008

[MDA14] OMG, MDA Guide, Rev. 2.0. http://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf, 2014

[UML15] OMG, Unified Modeling Language (UML). http://www.omg.org/spec/UML/2.5/, 2015