

# **Konzeptionierung und Implementierung einer Konflikterkennung für Model to Model Transformationen im Open Source MDA-Generator AndroMDA**

Thorsten Pohl

Lufthansa Technik AG  
Weg beim Jäger 192  
22335 Hamburg  
thorsten.pohl@lht.dlh.de

**Abstract:** Die Diplomarbeit beschäftigt sich mit den Möglichkeiten und potentiellen Schwierigkeiten, die durch den Einsatz von Modelltransformationen entstehen können. Insbesondere die Erkennung von Konflikten, die hierbei entstehen können wird untersucht. Im Rahmen der Arbeit wird eine prototypische Lösung für die Validierung von Modellen innerhalb einer Transformationskette entwickelt.

## **Das Vorhaben**

Im Rahmen einer Diplomarbeit wurden Möglichkeiten und potentielle Problemstellungen, die der Einsatz von Model to Model Transformationen mit sich bringen könnten untersucht. Das Auftreten von Konflikten durch die Verwendung von unterschiedlichen Modellen wird hierbei gesondert herausgehoben. Diese Problemstellung wird sowohl allgemein als auch konkret vor der Hintergrund der von AndroMDA verwendeten Technologie beleuchtet. Die Diplomarbeit ist abgeschlossen, die dabei entwickelten Ergebnisse werden in die zukünftige Entwicklung von AndroMDA einfließen.

## **Modellgetriebene Softwareentwicklung und Modelltransformationen**

Die Intention aus Modellen Software komplett oder teilweise zu erzeugen, wird vor allem mit dem Begriff Model Driven Architecture (MDA) verbunden. Richtiger und umfassender ist die Beschreibung als modellgetriebene Softwareentwicklung (MDS). Die verschiedenen Ansätze im Bereich der MDS werden verglichen und der von AndroMDA verfolgte Ansatz wird in die existierende Begriffshierarchie eingeordnet.

AndroMDA plant weiterhin ein pragmatisches und unabhängiges Framework zu bleiben und wird auch Model to Model Transformationen für verschiedene Transformationsengines und Metamodelle unterstützen. AndroMDA wird entgegen der Spezifikation der OMG für MDA keine Änderungen an den Zwischenmodellen der Model to Model Transformationskette erlauben. Diese Herangehensweise wird als Forward-Only-Ansatz bezeichnet und bietet den Vorteil, dass dieser Ansatz sehr gut beherrschbar ist und technisch ausgereifte Transformationsengines<sup>1</sup> zur Verfügung stehen. Andere mögliche Ansätze, wie die inkrementelle Generierung (Änderungen in Zwischenmodellen bleiben erhalten) und die bidirektionale Transformation werden beschrieben und eingeordnet, sind jedoch zum Betrachtungszeitpunkt noch nicht ausgereift genug, um für den Einsatz in Frage zu kommen<sup>2</sup>.

## **Auftreten und Erkennung von Konflikten**

Durch den Einsatz der Model to Model Transformationen können Konflikte in den entstehenden Zwischenmodellen auftreten. Die Entstehung dieser Konflikte wird untersucht und Möglichkeiten ihrer Erkennung werden entwickelt und bewertet.

Konflikte entstehen anders als Fehlmodellierungen durch das Zusammentreffen mehrerer Modellelemente, die möglicherweise durch verschiedene Transformationsschritte erzeugt wurden. Ein Beispiel für das Auftreten eines Konfliktes wäre es, wenn in einem Modell von Java-Klassen durch verschiedene Transformationsschritte zwei Klassen entstehen, die gleich heißen und sich im gleichen Package befinden. Konflikte können auch dann auftreten, wenn alle vom Anwender erstellten Modelle für sich genommen fehlerfrei sind. Die Erkennung dieser Konflikte ist dadurch aufwendig und erfordert es, jedes Modell zu überprüfen. Diese Prüfung sollte keine Annahmen über nachfolgende Transformationen machen und lediglich die Korrektheit des Modells auf seinem Abstraktionsniveau prüfen. Dadurch, dass die Validierung auf jeder Abstraktionsebene ausgeführt wird, kann auch die Erkennung von Fehlmodellierungen auf der jeweils geeignetsten Ebene durchgeführt werden.

Gleichzeitig wird die Kopplung von Metamodellen, zu denen auch die Validierungsregeln zu zählen sind, und von Transformationen weiter reduziert. Diese Kopplung würde sich negativ auf die Wiederverwendbarkeit von Transformationsschritten und die Flexibilität der Generierung auswirken. Für die Validierung der verschiedenen Modelle wird eine prototypische Validierungskomponente entwickelt, die in der Lage ist, die Modelle anhand verschiedener Constraints zu prüfen. Die Definition dieser Constraints lässt sich sowohl in der Object Constraint Language, als auch mithilfe von Javacode umsetzen.

---

<sup>1</sup> Die betrachteten Transformationsengines QVT und ATL werden in [Jou06] verglichen

<sup>2</sup> Eine generelle Übersicht über die Ansätze findet sich im [Fra03], S. 230ff

Für die Prüfung werden jeweils einzelne Modellelemente geprüft. Es zeigt sich jedoch, dass, insbesondere im Falle von Konflikten, mehrere Modellelemente einen Validierungsfehler verursachen. Diese „violating Elements“ werden von der Validierungskomponente im Falle eines Verstoßes ermittelt und gehen dann in die weitere Analyse ein.

## **Verarbeitung der erkannten Konflikte und Modellierungsfehler**

Nachdem ein Konflikt erkannt wurde ist es notwendig, ihn für den Anwender so aufzubereiten, dass er ihn verstehen und beheben kann. Ein Aspekt der Aufbereitung ist es, die Fehler für den Anwender mit aussagekräftigen Fehlermeldungen zu beschreiben. Um dies zu erreichen, enthält die Validierungskomponente die erweiterbare Möglichkeit, Templates für die Fehlermeldungen für jeden Constraint zu definieren.

Ein anderer Aspekt ist es, die Zusammenhänge zwischen den Modellen darzustellen. Dies ist von besonderer Wichtigkeit, da die Fehler auch in Modellen mit niedrigerem Abstraktionsniveau auftreten können, in denen der Anwender gar nicht modelliert (siehe Forward-Only Ansatz). Der Anwender muss also in einem anderen Modell Änderungen vornehmen, als dem, in dem der Fehler auftrat. Um ihn dabei zu unterstützen, ist sowohl die Erzeugung einer aussagekräftigen Fehlermeldung als auch die Rückverfolgung der verursachenden Modellelemente in ein dem Anwender bekanntes Modell von großer Bedeutung.

## **Traceability von Modellelementen**

Für die Herstellung der Rückverfolgbarkeit werden verschiedene Möglichkeiten untersucht. Um diese Rückverfolgung zu ermöglichen, müssen Traces zwischen Modellelementen der verschiedenen Modelle gelegt werden. Die Traces zwischen den Modellen können auf verschiedene Arten gelegt werden. Auch ihre Speicherung kann prinzipiell auf verschiedenen Wegen erreicht werden.

Die Untersuchung der verschiedenen Alternativen ergibt, dass die Auffassung dieser Traces als Modell und ihre Erzeugung durch explizite Regeln in der Transformationsvorschrift die meisten Vorteile bietet. Sie kann darüber hinaus vergleichsweise unabhängig von der verwendeten Transformationsengine eingesetzt werden. Diese Unabhängigkeit erfüllt die Anforderung des Frameworks AndroMDA, prinzipiell alle Transformationsengines und Metametamodelle zu unterstützen. Die Traceability der Modellelemente wird neben der Validierung auch für weitere Anforderungen benötigt. Aus diesem Grund ist es notwendig, dass die Umsetzung der Traceability sowohl umfangreiche Funktionalitäten bietet als auch im Einsatz eine angemessene Performance aufweist.

Für die Implementierung dieser Anforderungen wird ein Modell der Traces definiert. Die entwickelte Transformationskette nutzt dieses Modell, um ihre Rückverfolgbarkeit zu realisieren. Jede Transformation kann diese Informationen bereitstellen und in den entsprechenden Modellen speichern. Für die Verwendung der Traceabilityinformationen wird eine Komponente prototypisch entwickelt, die in der Lage ist, für jedes Modellelement die jeweilig erzeugenden Modellelemente zu ermitteln. Die eingesetzte Lösung zur Traceability ist für den Prototypen auf das Metametamodell EMF Ecore beschränkt. Diese Beschränkung kann durch den konsequenten Einsatz des in AndroMDA 4 geplanten Modellrepositoryes gelöst werden. Hierzu muss diese Komponente eine Kompatibilitätsschicht über den Metametamodellen einführen. Diese Einführung ist bereits geplant, da sie auch für weitere Aspekte innerhalb der Entwicklung sinnvoll ist. Die entwickelten Komponenten können zusammen eingesetzt werden, um die Konflikterkennung und Validierung mit der Rückverfolgung sinnvoll zu implementieren.

## **Kritische Betrachtung**

Die verschiedenen Ansätze zur modellgetriebenen Softwareentwicklung werden miteinander verglichen und weitere Einsatzmöglichkeiten von Model to Model Transformationen geprüft. Die Verknüpfung mit der domänenspezifischen Modellierung könnte dazu genutzt werden, um die Erstellung des für die domänenspezifischen Modelle notwendigen spezifischen Generatoren zu vereinfachen.

Die erwarteten Vorteile von Model to Model Transformationen werden vor dem Hintergrund der während der Erstellung der Arbeit gewonnenen Erkenntnisse untersucht. Insbesondere die Wiederverwendbarkeit von Transformationsschritten wird nur dann gegeben sein, wenn dieser Aspekt in der Entwicklung dieser Schritte und der Metamodelle kontinuierlich im Fokus bleibt. Den Metamodellen als Übergabepunkte zwischen den Transformationsschritten kommt hierbei eine besondere Bedeutung zu. Die Weiterentwicklung von AndroMDA und der Einsatz von Model to Model Transformationen sollte nicht dazu führen, dass sich die Einstiegshürde für den Einsatz erhöht. Dieser Erhöhung sollte durch den Einsatz von Konstrukten, ähnlich der aktuellen Metafassaden, entgegengewirkt werden. Ein wichtiger Aspekt ist in diesem Zusammenhang die Verarbeitung der UML in Transformationsvorschriften. Die UML besitzt ein umfangreiches und sehr mächtiges Metamodell und erschwert so ihren Einsatz als Modell in Transformationen.

## **Literatur**

- [Fra03] David S. Frankel. Model Driven Architecture - Applying MDA to Enterprise Computing. Wiley Publishing, Inc, Indianapolis, IN, US, 2003.
- [Jou06] Frédéric Jouault and Ivan Kurtev. On the Architectural Alignment of ATL and QVT. In Proceedings of ACM Symposium on Applied Computing (SAC 06), model transformation track, Dijon, Bourgogne, France, 2006. URL <http://delivery.acm.org/10.1145/1150000/1141561/p1188-jouault.pdf>.