Evaluation von Transformationsmaschinen in der modellbasierten Qualitätssicherung

Mario Friske, Konrad Hilse
Fraunhofer FIRST, Kekuléstraße 7, D-12489 Berlin
{mario.friske,konrad.hilse}@first.fhg.de

Abstract: Anhand eines Beispiels aus der modellbasierten Qualitätssicherung evaluieren und vergleichen wir vier Eclipse-basierte Transformationsmaschinen. Über einen flexiblen Mechanismus binden wir diese in ein Testwerkzeug ein und setzen eine sprachunabhängig beschriebene Beispieltransformation in den einzelnen Transformationssprachen um. Anschließend diskutieren wir deren Eignung anhand eines von uns erstellten Kriterienkataloges.

1 Einleitung

Durch den Einsatz generativer Techniken lassen sich in der Entwicklung und beim Test von Software Zeit- und Kostenersparnisse erzielen. Daraus resultiert eine große Popularität der korrespondierenden Paradigmen der *modellbasierten Entwicklung* und des *modellbasierten Tests* in der aktuellen Forschung und Entwicklung. Obwohl mit der Model-Driven Architecture (MDA) [Obj03] schon seit einiger Zeit eine Vision vorliegt, sind die Arbeiten an den zugehörigen Standards noch immer nicht vollständig abgeschlossen. Seit kurzem sind auch erste Frameworks zur modellbasierten Entwicklung, wie das Eclipse Modeling Framework (EMF) [BEG+03], und darauf basierende Transformationsmaschinen [Dem05, JK05, LS05, Bor05] verfügbar. Es stellt sich die Frage, inwieweit die aktuell verfügbaren MDA-basierten Transformationssprachen und -werkzeuge trotz der noch nicht abgeschlossenen Entwicklung schon zur Softwareentwicklung und -qualitätssicherung einsetzbar sind.

Ein Problem bei den modellbasierten Ansätzen ist es, dass automatisierte Transformationen formale Modelle erfordern. Anforderungen liegen oft jedoch nur in Form von strukturiertem Text vor. Eine Möglichkeit, diese zu formalisieren, ist die interaktive Aufbereitung. Mit dem *Use Case Validator* (UCV) [FP05] haben wir ein Werkzeug entwickelt, mit welchem textuelle Anwendungsfallbeschreibungen interaktiv formalisiert werden können, um sie dann als Ausgangspunkt für automatisierte Transformationen zu nutzen.

Dieses Papier ist wie folgt gegliedert: Im folgenden Abschnitt geben wir zunächst einen kurzen Überblick über den UCV. In Abschnitt 3 werden die von uns evaluierten Transformationswerkzeuge für die Eclipse-Plattform [Ecl] vorgestellt und unsere Evaluationsergebnisse präsentiert und diskutiert. Im letzten Abschnitt geben wir eine Zusammenfassung und einen Ausblick.

2 Modellbasierte Qualitätssicherung mit dem Use Case Validator

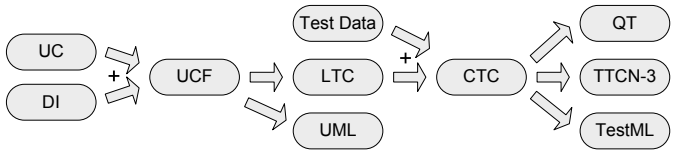

Abbildung 1: Übersicht der Transformationen des Use Case Validator

Im UCV sind die in Abbildung 1 dargestellten Transformationen vorgesehen. Zunächst werden *textuelle Anwendungsfallbeschreibungen* (Use Cases, UC) interaktiv aufbereitet, wozu *Entwurfsinformation* (Design Information, DI) genutzt wird [FS05]. Ergebnis dieser Formalisierung sind *formalisierte Anwendungsfallbeschreibungen* (Use Case Formalization, UCF) [FP05]. Mittels automatisierter Transformation sollen diese nun weiterverarbeitet werden. Einerseits sollen sie in grafische Repräsentationen, wie beispielsweise *Aktivitätsdiagramme* (UML) [Obj04], überführt werden. Anderseits sollen nach dem in [Fri04] skizzierten Verfahren *logische und konkrete Testfälle* (Logical and Concrete Test Cases, LTC und CTC) generiert werden. Abschließend sind diese in ausführbare Testskripte zu überführen. Geplant ist die Generierung von Skripten für *QuickTest Professional* (QT) [Mer], *TTCN-3* [ETS03] und *TestML* [GCF$^+$06].

Da der UCV unter Verwendung des EMF realisiert wurde, stellt sich die Frage nach geeigneten Werkzeugen, welche in den UCV integriert werden können und in der Lage sind EMF-Modelle automatisiert zu transformieren. Im Folgenden werden wir verfügbare Werkzeuge anhand einer der zu automatisierenden Transformationen des UCV evaluieren. Als Beispieltransformation verwenden wir dafür die Generierung von UML-Modellen aus formalisierten Anwendungsfallbeschreibungen.

3 Evaluation von EMF-Transformationsmaschinen im UCV

Momentan sind mehrere EMF-basierte Transformationsmaschinen verfügbar. Das kommerzielle Werkzeug **Borland Together Architect 2006** enthält eine Transformationsmaschine für eine Teilmenge von QVT [Bor05], im Folgenden Borland-QVT genannt. Die **Atlas Transformation Language (ATL)** [JK05] entstand 2005 als Antwort auf den QVT Request for Proposal (RFP) [Obj02] und ist als Teil des Projekts *Generative Modeling Tools* (GMT) [Ecl] frei verfügbar. **Tefkat** ist eine vom DSTC und der Universität von Queensland entwickelte Transformationssprache [LS05]. Sie basiert auf dem QVT-Vorschlag von DSTC und IBM [DST03] und ist als Open-Source-Projekt verfügbar. Das **IBM Model Transformation Framework (MTF)** [Dem05] ist ein im Rahmen des QVT RFP [Obj02] entwickelter Prototyp und ist für Evaluationszwecke frei verfügbar.

Um diese Transformationswerkzeuge in den UCV zu integrieren, haben wir einen Erweiterungspunkt definiert. Für die einzelnen Werkzeuge wurden Wrapper erstellt und diese in den UCV eingebunden.

Da sich noch keines dieser Tools als Standardwerkzeug etabliert hat, haben wir als Beispieltransformation die Generierung von UML-Modellen aus formalisierten Anwendungsfallbeschreibungen in allen vier Transformationssprachen umgesetzt und die Transformationsmaschinen evaluiert. Dazu haben wir in Anlehnung an [GAS+05] Evaluationskriterien definiert. Nachfolgend beschreiben wir unsere Erfahrungen bei der Umsetzung anhand dieser Kriterien:

Sprachspezifische Kriterien: Keine der betrachteten Sprachen erfüllt alle Anforderungen an den QVT-Standard [Obj02]. So kennt z. B. nur ATL sowohl imperative als auch deklarative Konstrukte; Tefkat, ATL und Borland-QVT sind nicht bidirektional, und keine der Sprachen unterstützt Modularisierung und Vererbung in vollem Umfang.

Implementierung: Die untersuchten vier Implementierungen sind noch im Stadium eines Prototypen. Bei der Umsetzung der Beispieltransformation kam es zu verschiedenen Problemen, z. B. übersetzt der Compiler von Borland-QVT syntaktisch korrekten Quellcode teilweise fehlerhaft. Insbesondere die Fehlersuche gestaltete sich schwierig, da alle vier Implementierungen oft schwer verständliche Fehlermeldungen liefern. Zusätzlich verfügen Tefkat und MTF über keine ausgereiften Debugger.

Beispielspezifische Kriterien: Die Beispieltransformation ließ sich zwar mit allen vier Sprachen realisieren, allerdings war dies nur mit Tefkat ohne Einschränkungen möglich. ATL und MTF erforderten Änderungen unseres Metamodells. Borland-QVT wiederum konnte das Eclipse-UML2-Plugin [Hus04] nicht als Zielmetamodell verwenden, sondern nur die eigene, vom Standard abweichende Implementierung.

Die Umsetzung der Transformation war mit Tefkat am einfachsten. Borland-QVT erforderte es, dass die deklarative Tranformationsbeschreibung erst in imperativen Code umgesetzt werden musste. ATL war einfach zu benutzen, allerdings erwies es sich als schwierig, von einander abhängige Transformationsregeln zu spezifizieren. Die größten Schwierigkeiten hatten wir, die Transformation mit MTF umzusetzen.

Sonstige Kriterien: Mit Ausnahme von ATL ist nur wenig Dokumentation verfügbar, teilweise gab es auch Diskrepanzen zwischen Dokumentation und Implementierung. Insbesondere für Tefkat waren einige Sprachmerkmale nur aus dem Diskussionsforum zu erschließen. Auch über ein einfaches Tutorial hinausgehende Beispiele sind nur für Borland-QVT und ATL verfügbar.

ATL ist die einzige der drei freien Implementierungen, an der aktiv gearbeitet wird. Tefkat wurde erst im März 2006 in ein Open-Source-Projekt überführt. Das letzte Release von MTF ist inzwischen ein Jahr alt.

Mit allen vier untersuchten Transformationssprachen war das Umsetzen der Beipielstransformation möglich. Es ergaben sich verschiedene Einschränkungen, die teilweise auf die Sprachspezifikation, teilweise aber auch auf die noch nicht vollständige Implementierung zurückzuführen sind. Die Evaluationsergebnisse haben wir in Tabelle 1 zusammengefasst.

	Borland-QVT	ATL	Tefkat	MTF
Sprachspezifische Kriterien				
Modularisierung und Vererbung	−	○	○	○
Bidirektionalität	−	−	−	+
Deklarative Konstrukte	−	+	+	+
Imperative Konstrukte	+	+	−	−
Implementierungsspezifische Kriterien				
Editor	+	○	−	+
Debugging und Fehlersuche	○	○	−	−
Benutzbarkeit des API	○	−	○	+
Beispielspezifische Kriterien				
Transformation vollständig umsetzbar	○	○	+	○
Benutzbarkeit	○	○	+	−
Sonstige Kriterien				
Dokumentation und Beispiele	+	+	−	○
Entwicklungsaktivität	+	+	○	−

Tabelle 1: Evaluationsergebnisse (gut (+), eingeschränkt (○) bzw. nicht erfüllt (−))

4 Zusammenfassung und Ausblick

Wir haben anhand eines Beispiels Transformationsmaschinen in der modellbasierten Qualitätssicherung evaluiert. Als Beispiel diente die Generierung von UML-Modellen aus formalisierten Anwendungsfallbeschreibungen. Über einen von uns entwickelten flexiblen Mechanismus haben wir verschiedene EMF-basierte Transformationsmaschinen in den Use Case Validator (UCV) eingebunden. Es gelang uns mit allen vier betrachteten Werkzeugen, die Beispieltransformation zu realisieren und UML-Modelle zu generieren. Limitierungen der Transformationsmaschinen umgingen wir durch Erweiterung unseres Metamodells. Anhand eines von uns erstellten Kriterienkataloges wurden die Sprachen und zugehörigen Transformationsmaschinen evaluiert.

Es hat sich gezeigt, dass trotz noch nicht abgeschlossener Standardisierung der MDA-Transformationssprache QVT verfügbare Transformationsmaschinen bereits zur Softwareentwicklung und -qualitätssicherung einsetzbar sind. Häufig sind jedoch noch diverse Kinderkrankheiten und nur unvollständige Dokumentation vorhanden. Daraus resultiert ein sehr hoher Einarbeitungsaufwand.

Die flexible Architektur des UCV gestattet es, verschiedene Transformationsmaschinen zu verwenden. Falls nötig, können so auch unterschiedliche Transformationsmaschinen kombiniert werden, um Limitierungen zu umgehen. Damit wurde die Voraussetzung zur Umsetzung der verbleibenden komplexeren Tranformationen und somit zum Ausbau des UCV zum vollständigen Testfallgenerator geschaffen. Bis ausgereifte Implementierungen des vollständigen Standards verfügbar sind, beabsichtigen wir, die beiden Open-Source-Produkte Tefkat und ATL einzusetzen.

Literatur

[BEG+03] Frank Budinsky, Ray Ellersick, Timothy J. Grose, Ed Merks und David Steinberg. *Eclipse Modeling Framework*. The Eclipse Series. Addison-Wesley, 2003.

[Bor05] Borland. QVT-Sprachreferenenz. Dokumentation zu Together Architect 2006, 2005.

[Dem05] Sebastien Demathieu. Model transformation with the IBM Model Transformation Framework. IBM, 2005.

[DST03] DSTC, IBM, CBOP. MOF Query/View/Transformation, initial submission, 2003.

[Ecl] Eclipse Foundation. Eclipse. http://www.eclipse.org/.

[ETS03] ETSI. Spezifikation der Testing and Test Control Notation Version 3. ES 201 873 Reihe, Version 2.2.1, 2003.

[FP05] Mario Friske und Holger Pirk. Werkzeuggestützte interaktive Formalisierung textueller Anwendungsfallbeschreibungen für den Systemtest. In A. B. Cremers, R. Manthey, P. Martini und V. Steinhage, Hrsg., *35. GI-Jahrestagung, Band 2*, Jgg. 68 of *LNI*. GI, September 2005.

[Fri04] Mario Friske. Testfallerzeugung aus Use-Case-Beschreibungen. Softwaretechnik-Trends, Band 24, Heft 3, 2004.

[FS05] Mario Friske und Holger Schlingloff. Von Use Cases zu Test Cases: Eine systematische Vorgehensweise. In T. Klein, B. Rumpe und B. Schätz, Hrsg., *Tagungsband des Dagstuhl Workshops „Modellbasierte Entwicklung eingebetteter Systeme“ (MBEES)*. Technische Universität Braunschweig, Januar 2005.

[GAS+05] Roy Grønmo, Jan Aagedal, Arnor Solberg, Mariano Belaunde, Peter Rosenthal, Madeleine Faugere, Tom Ritter und Marc Born. Evaluation of the QVT Merge Language Proposal. Bericht Modelware/I1.2.1, MODELWARE IST Project 511731, 2005.

[GCF+06] Jürgen Großmann, Mirko Conrad, Ines Fey, Alexander Krupp, Klaus Lamberg und Christian Wewetzer. TESTML - A Test Exchange Language for Model-based Testing of Embedded Software. Workshop Advanced Automotive Software and Systems Development: Model-Driven Development of Reliable Automotive Services, San Diego, 2006.

[Hus04] Kenn Hussey. Getting Started with UML2. IBM, Juli 2004.

[JK05] Frédéric Jouault und Ivan Kurtev. Transforming models with ATL. In *Proceedings of Model Transformations in Practice Workshop*, 2005.

[LS05] Michael Lawley und Jim Steel. Practical declarative model transformation with tefkat. In *Model Transformations In Practice Workshop*. Montego Bay, Jamaica, Oktober 2005.

[Mer] Mercury Interactive Corporation. QuickTest Professional. http://www.mercury.com/.

[Obj02] Object Management Group. Request for Proposal: MOF 2.0 Query / Views / Transformations RFP. OMG Dokument ad/2002-04-10, April 2002.

[Obj03] Object Management Group. MDA Guide Version 1.0.1 (omg/03-06-01). http://www.omg.org/mda/, 2003.

[Obj04] Object Management Group. UML 2.0 Spezifikation. http://www.uml.org/, 2004.