Ansatz zur Metamodellierung mit Verhalten und dessen Anwendungen

Michael Soden, Hajo Eichler
Humboldt Universität zu Berlin
Unter den Linden 6, 10099 Berlin
soden,eichler@ikv.de

Abstract: Metamodellierung bietet ein adäquates Mittel für die Definition verschiedenster Artefakte in Softwareentwicklungsprojekten. Um eine vollständige, ausführbare Spezifikation von Softwaresystemen und Plattformen zu erhalten, bedarf es Metamodellen mit Verhaltensbeschreibungen. Im Rahmen dieses Forschungsvorhaben werden verschiedene Erweiterungen der Meta Object Facility (MOF) der Object Management Group (OMG) erarbeitet und auf unterschiedliche Phasen im Softwareentwicklungszyklus angewendet. Das dabei entstehende Metamodellierungsframework mit Verhaltensbeschreibung, Instanziierungskonzept, Szenarien- und Datenaufnahme, etc. zielt auf Verbesserungen bei der Anforderungsanalyse, frühst möglicher Validierung durch Simulation und der nahtlosen Integration des Testens ab.

1 Motivation

Ausführung und Simulation von Modellen sind seit Jahrzehnten etablierte Techniken der Softwareentwicklung. Während die Idee der modellgetriebenen Softwareentwicklung (z.B. die MDA Initiative der OMG) verschiedenste Bereiche der Informationstechnologiebranche durchdrungen hat, insbesondere den der Eingebetteten Systeme, wird ein Großteil heutiger Unternehmenssoftware (noch) *nicht* modellorientiert entwickelt im Sinne von automatisierten Modelltransformationen zwischen Prozessphasen, integrierten Werkzeuglandschaften, Nachvollziehbarkeit (Traceability) oder vollständiger Quellcode Generierung. Zudem finden Techniken der Systemanalyse wie beispielsweise die Simulation keine Anwendung im Rahmen der MDA. Wir sehen dafür mehrere Ursachen. Zum einen bieten die meisten Programmiersprachen vergleichbare Abstraktionsmechanismen wie Modelle und kommen mit oft mächtiger Werkzeugunterstützung daher, die oftmals für modellbasierte Spezifikation und Manipulation fehlt. Weiterhin existieren typischerweise starke Abhängigkeiten des modellierten Systems zur Ausführungsumgebung (z.B. Bibliotheksfunktionalität, Frameworks, etc.), weshalb Modellierungssprachen häufig nur zu Dokumentationszwecken und nicht zur präzisen, d.h. automatisiert weiterverarbeitbaren Beschreibung der Systemstruktur und des Systemverhaltens dienen.

Aus unserer Analyse leiten wir für unser Forschungsvorhaben mehrere Ziele ab:

- Bereitstellen geeigneter Modellierungstechniken zur Ausführbarkeit von Modellen, sprachunabhängig auf der Basis von MOF Metamodellen

- Untersuchen von Systemcharakteristiken durch dynamische Analyse bei der Ausführung von Verhaltensmodellen während der Spezifikationsphase
- Lösen von Integrationsproblematiken durch sprach-/metamodellübergreifende Simulation
- Einbeziehen von Teilen des echten Systems in die Simulation
- Sammeln von Simulationsszenarien und Daten für das Testen des Systems
- Vergleichbarkeit von Verhalten in verschiedenen Plattformen durch Verhaltensdefinition in Metamodellen

Basierend auf diesen Zielsetzungen ergaben sich für uns zwei Teilbereiche einer gemeinsamen Arbeit die jeweils in den Abschnitten 3.1 und 3.2 kurz umrissen werden. Die Grundlage bietet ein gemeinsam entwickeltes Modellierungsframework, das in Abschnitt 2 kurz dargestellt wird.

2 Verhaltensmetamodellierung

Die Meta Object Facility Spezifikation [OMG03] der OMG ermöglicht das Erstellen von Metamodellen - Modelle zur Definition von Datenstrukturen. M3Actions (siehe [HES], [SE07] und [SF07] für eine detailierte Sprachdefinition und Beispiele) entwickelt diesen Gedanken der übergreifenden Sprache weiter und reichert MOF um folgende Aspekte an:

- Eine an UML Actions/Activities angelehnte Sprache zur Definition von *operationaler Semantik* [Plo81] zur Beschreibung von Verhaltensabläufen
- Ein explizites *Instanziierungskonzept*, dass die Aussagemöglichkeiten der strikten Metamodellierung erweitert (vgl. [AK01])
- Integration der Object Contraint Language (OCL [OMG05b]) zum Definieren von Modellabfragen

Dieses Metamodellierungsframework ist bereits prototypisch implementiert. Derzeit findet eine Spezialisierung und Erweiterung des Frameworks in verschiedene Richtungen statt, u.a. Importierung von Quellcode durch Abbildung von EBNF Grammatiken auf Metamodelle mittels QVT [OMG05a] (vgl. [AP03]), Szenarien-/Datenaufnahme und Interaktion mit dem zu entwickelnden System.

3 Anwendung von Verhaltensmetamodellierung

Der aufgezeigte Ansatz der kombinierten Verhaltensbeschreibung und Metamodellierung soll auf 2 spezielle Phasen der Softwareentwicklung angewandt werden, um so den Nutzen

und Mehrwert des Verfahrens darzustellen. Zum einen soll der Ansatz mit Techniken der Modellsimulation ergänzt werden, um Informationen über die Spezifikation zu erhalten. Zum anderen soll erörtert werden, wie das Testen durch die Anwendung der Verhaltensbeschreibung in Metamodellen vorangetrieben werden kann.

3.1 Simulation

Diverse Simulationssprachen und Frameworks existieren um physikalische Prozesse wie Teilchenbewegungen, Geschäftsabläufe, Produktionsprozesse usw. zu analysieren. Die grundlegende Idee dieser Arbeit ist das Anwenden dieser Simulationstechniken auf die Softwareentwicklung selbst, speziell auf Modelle die über MOF Metamodelle definiert sind. Primäres Ziel ist die (sprachunabhängige) Validierung der Korrektheit von Spezifikationen, durch die Auswertung von Modellzuständen während und nach der Modellausführung. Hierbei sei erwähnt, dass — durch die Konzeption über Metamodelle — der Ansatz für alle Sprachen einsetzbar ist, die eine operationale Semantik besitzen, d.h. sowohl Programmiersprachen als auch Modellierungssprachen wie UML, Domain Specific Languages (DSLs), QVT, etc.

Zu diesem Zweck wurde zunächst eine Verhaltensbeschreibungssprache zur Definition von operationaler Semantik im Metamodellierungsframework konzeptionell erarbeitet und realisiert, die das Ausführen von Modellen erlaubt (vlg. Abschnitt 2). Dies war die Voraussetzung zu Analysen des Laufzeitverhaltens, wie z.B. Engpassanalyse, Häufigkeit der Aufrufe von Objekten sowie kausale und zeitliche Aspekte in Modellen. Derzeit wird das Framework erweitert, um Simulationsszenarien und Daten aufzunehmen und wiederzugeben.

In einem nächsten Schritt sollen Teile eines echten Systems mit in die Simulation einbezogen werden, um die Abstraktionslücke zwischen Modell und Produkt zu schließen. Basierend auf den gewonnen Ergebnissen soll anschließend der Ansatz evaluiert und verglichen werden, um das Potential der Techniken bewerten zu können. Unsere Hoffnungen konzentrieren sich auf die einfachere Anwendbarkeit des Ansatzes als formal mathematische Methoden (z.B. Abstract State Machines [YG04][ITU00]), generische Erstellung von Performance Indikatoren und Analyseroutinen für die Modellspezifikation, qualitative Bewertungen von Modellen, Ableiten von sprachspezifischen Simulationsumgebungen aus der Sprachdefinition, usw.

3.2 Testen

Ein zentrales Thema von Forschungsarbeiten im Bereich des Softwaretestens ist die (nahtlose) Integration von Testtechniken in Softwareentwicklungsprozesse. Dieses Bestreben gesehen im Zusammenhang mit der Einführung des Metamodellierungsansatzes bei der Entwicklung von Unternehmenssoftware, führt zu einer engeren Verzahnung von Test und Entwicklung. Einer der entscheidensten Bestandteile von Testfällen ist die Beschreibung

von Abläufen, die bei rein statischer Definition in Metamodellen heutzutage fehlt. Die in Kapitel 2 eingeführten Techniken sollen diese Lücke schließen und damit ermöglichen, modellbasierte Verhaltensbeschreibungen für das Testen des Softwaresystems einzusetzen.

Bei dieser Anwendung sollen verschiedene Aspekte im Umfeld der Softwaretests untersucht werden, die gemeinsam die Vereinfachung der Erstellung einer Testspezifikation aus vorhanden Systeminformationen als Ziel verfolgen. Unter anderem soll untersucht werden, (1) wie gesammelte Daten beim Simulieren für Testfälle wiederverwendet werden können, (2) inwiefern Simulationsabläufe in Testfälle überführt werden können, (3) wie beim Importieren bereits entwickelter Softwarekomponenten, Testtechniken wie zum Beispiel so genannte *Mockup*-Objekte in der Modellwelt realisiert werden können, (4) welche Verhaltensspezifikationen notwendig sind, um Test-spezifische Ausdrücke (wie z.B. Vergleich von Abläufen) zu definieren und (5) welche Fehlerklassen mit dem Verfahren aufgedeckt werden können.

Als Ergebnis soll die Frage geklärt werden, ob der dargestellte Ansatz die Idee vom „Testen gegen die Spezifikation“ verwirklichen kann.

4 Verwandte Themen

Es gibt diverse Ansätze zu Verhaltensbeschreibungen, davon seien hier einige aufgezählt, die auch den Metamodellierungsansatz verfolgen: GME[AKL03], XMF[CESW04], Kermeta[Tea], AToM3[ato], MetaEdit+[Met], AMMA[DDR06], MPS[Dmi04], für deren Klassifizierung z.B. die Einteilung aus [FHPS04] hilfreich sein kann. Die meisten dieser Ansätze zielen allerdings nicht auf eine echte Simulationsumgebung mit Testintegration ab, die unserem Forschungsvorhaben zu Grunde liegt.

Literatur

[AK01] Colin Atkinson und Thomas Kühne. The Essence of Multilevel Metamodeling. In *UML'01: Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, LNCS, Seiten 19–33, London, UK, 2001. Springer-Verlag.

[AKL03] Aditya Agrawal, Gabor Karsai und Akos Ledeczi. An end-to-end domain-driven software development framework. In *OOPSLA '03: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, Seiten 8–15, New York, NY, USA, 2003. ACM Press.

[AP03] Marcus Alanen und Ivan Porres. A Relation Between Context-Free Grammars and Meta Object Facility Metamodels. Tucs technical report no 606, Turku Centre for Computer Science, Marz 2003.

[ato] *The Modelling, Simulation and Design lab (MSDL), School of Computer Science of McGill University Montreal, Quebec, Canada: AToM3 A Tool for Multi-Formalism Meta-Modelling. http://atom3.cs.mcgill.ca/index.html.*

[CESW04] Tony Clark, Andy Evans, Paul Sammut und James Willans. *Applied Metamodeling, A Foundation for Language Driven Development*. Xactium, 2004.

[DDR06] Ivan Kurtev Jean Bèvin Alfonso Pierantonio Davide Di Ruscio, Frèric Jouault. Extending AMMA for Supporting Dynamic Semantics Specifications of DSLs. Bericht, Universitïegli Studi dell'Aquila, 2006.

[Dmi04] Sergey Dmitriev. Language Oriented Programming: The Next Programming Paradigm. *onBoard*, (1), November 2004.

[FHPS04] Joachim Fischer, Eckhardt Holz, Andreas Prinz und Markus Scheidgen. Tool-based Language Development. In *Workshop on Integrated-reliability with Telecommunications and UML Languages*, November 2004.

[HES] Markus Scheidgen Hajo Eichler und Michael Soden. A Semantic Meta-Modelling Framework with Simulation and Test Integration. In *Proceedings of the ECMDA Workshop on Integration of Model Driven Development and Model Driven Testing*. IRB Verlag.

[ITU00] ITU-T. *Specification and Description Language (SDL)*, Kapitel SDL formal definition: Dynamic semantics. International Telecommunication Union, November 2000. Z.100 Annex F3.

[Met] MetaCase. *MetaEdit+. http://www.metacase.com.*

[OMG03] OMG. *Meta Object Facility (MOF) 2.0 Core Specification*. Object Management Group, Oktober 2003. ptc/03-10-04.

[OMG05a] OMG. *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Draft adopted Specification, ptc/05-10-02*. Object Management Group, 2005.

[OMG05b] OMG. *OCL 2.0 Specification*. Object Management Group, Juni 2005. ptc/2005-06-06.

[Plo81] G.D. Plotkin. A Structural Approach to Operational Semantics. Bericht, University of Aarhus, Denmark, 1981.

[SE07] Michael Soden und Hajo Eichler. An Approach to use Executable Models for Testing. In *Enterprise Modelling and Information Systems Architectures - Concepts and Applications , Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures*, Jgg. P-119 of *LNI*. GI, 2007.

[SF07] Markus Scheidgen und Joachim Fischer. Human Comprehensible and Machine Processable Specifications of Operational Semantics. 2007.

[Tea] T. Team. *Triskell Meta-Modelling Kernel. IRISA, INRIA. www.kermeta.org.*

[YG04] Wolfram Schulte Yuri Gurevich, Benjamin Rossman. Semantic Essence of AsmL. Bericht, Microsoft Research, 2004.