Durchgängige Modularität in der modellgetriebenen Entwicklung domänenspezifischer Modellierungssprachen mit Hilfe aspektorientierter Programmierung

Marco Mosconi

Technische Universität Berlin
Fachgebiet Softwaretechnik
mosconi@cs.tu-berlin.de

Abstract: Das hier vorgestellte Promotionsvorhaben hat zum Ziel, eine durchgehend modulare Entwicklung von domänenspezifischen Modellierungssprachen und deren Implementierung in Werkzeugen zu ermöglichen. Dazu werden Modularisierungskonzepte auf Metamodell-Ebene analysiert, erweitert und mit Hilfe aspektorientierter Mechanismen in die Implementierung übertragen. Das Ergebnis soll ein Framework für die Entwicklung von Modellierungssprachen im UML 2 Umfeld sein, welches insbesondere die Erweiterbarkeit von Werkzeugen und deren Integrationsfähigkeit verbessert. Der Ansatz wird schließlich am konkreten Beispiel einer Modellierungssprache für *Object Teams* angewendet und evaluiert.

1 Einleitung

Im Bereich der modellgetriebenen Softwareentwicklung spielen domänenspezifische Modellierungssprachen (*DSML*) eine große Rolle, wenn es darum geht, auf den verschiedenen Abstraktionsebenen, wie sie z.B. in *MDA* definiert sind, spezifische Sichten zu unterstützen.

Eine DSML kann abhängig von Umfang und Anforderungen entweder neu entwickelt werden oder als Erweiterung einer bestehenden Sprache. In beiden Fällen hilft ein Metamodell-basierter Ansatz, bei dem die Sprachen auf einem gemeinsamen Meta-Metamodell wie z.B. MOF basieren. Eine Modellierungssprache besteht typischerweise aus den Komponenten abstrakte und konkrete Syntax, statische und operationale Semantik, sowie zugehörigen Werkzeugimplementierungen. Für diese Komponenten sind Qualitätseigenschaften wie Erweiterbarkeit, Wartbarkeit, Wiederverwendbarkeit und Evolutionsfähigkeit entscheidend, die durch geeignete Modularisierungs- und Erweiterungsmechanismen verbessert werden können. Solche Mechanismen werden auf Metamodell-Ebene in MOF/UML 2 u.a. durch *Package Merge*, Redefinition und Subsetting, sowie Profile bereitgestellt. Die Erweiterung und Verbesserung dieser Mechanismen ist Gegenstand aktueller Forschung [Sch06, Zit06]. Zur Vereinfachung der ansonsten sehr umfangreichen Werkzeugentwicklung entstanden zudem Ansätze wie Eclipse GMF, die basierend auf Metamodellen grafische Editoren erzeugen.

2 Problembeschreibung

Während bei der Metamodellierung in MOF/UML 2 mit Profilen und Package Merge bereits Mechanismen zur Verfügung stehen, Metamodelle modular zu entwickeln und zu erweitern, fällt dies bei der Implementierung, insbesondere von grafischen Editoren, nach wie vor schwer. Wie in Abbildung 1 dargestellt, kann eine auf Basis von Package Merge erstellte DSML existierende Implementierungen einzelner Pakete in der Regel nicht wiederverwenden, wie es im Metamodell der Fall ist. Die Implementierungen und Werkzeuge einer DSML können hier nicht alle Erweiterungs- und Integrationsszenarien vorhersehen. Dies führt dazu, dass generierter Code modifiziert werden muss und Komponenten nur per „copy&paste“ wiederverwendbar sind, was Erweiterbarkeit und Wartbarkeit erheblich behindert (vgl. [HM07]).

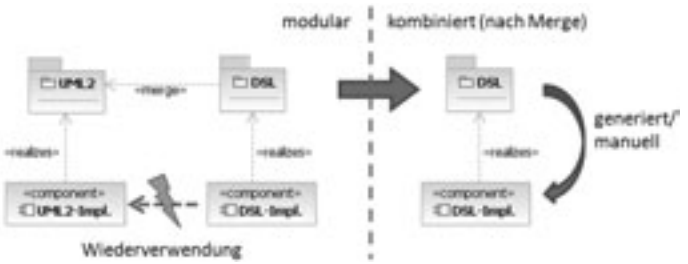

Abbildung 1: Keine modulare Wiederverwendung von DSML-Komponenten

Insbesondere bei umfangreichen Sprachen wie der UML ist es allerdings wichtig, dass eine selektive Wiederverwendung bzw. Erweiterung möglich ist. Für die Metamodellierung stehen dazu Konzepte wie *Package Import* und Merge zur Verfügung. Diese Konzepte finden sich aber in der Implementierung nicht wieder, was dort eine entsprechende Modularisierung erschwert. Die Eclipse UML2 Implementierung realisiert zum Beispiel das gesamte UML 2 Metamodell in „flacher“ Form, d.h. nach Auflösung aller *Package Merge* Abhängigkeiten. Eine Identifizierung der Bestandteile einzelner Metamodell-Pakete ist daher nicht mehr möglich. Auch andere Implementierungen (aMOF [Sch06], MOFLON) lösen Package Merge Beziehungen direkt im Metamodell auf.

3 Ziel und Lösungsansatz

Das Ziel dieser Arbeit ist es, die modulare Entwicklung, selektive Wiederverwendung und Erweiterbarkeit in der modellgetriebenen Werkzeugentwicklung zu ermöglichen und somit die Wartbarkeit und Evolutionsfähigkeit domänenspezifischer Modellierungssprachen und ihrer Werkzeuge zu verbessern. Geeignete aspektorientierte Mechanismen sollen dabei eine modulare Implementierung ermöglichen, die die Modularisierung auf Metamodell-Ebene widerspiegelt.

Die Bestandteile von DSMLs müssen zunächst hinsichtlich ihrer Modularisierbarkeit analysiert werden. Hierbei wird auf existierende konzeptionelle Modularisierungsmechanismen auf Metamodell-Ebene wie *CMOF* [Sch06], UML *Package Merge* [Zit06], Profile und weitergehende Ansätze wie *Theme/UML* [CB05] zurückgegriffen und diese ggf. kombiniert und erweitert. Die Verwendung dieser Techniken und der Methodik wird für unterschiedliche Szenarien am Fallbeispiel beschrieben und validiert.

Ist die modulare Definition einer DSML auf Metamodell-Ebene gegeben, gilt es, diese auch in der Implementierung aufrecht zu erhalten. Im Idealfall sollte die Implementierung auf Werkzeug-Ebene der Modularisierung des Metamodells folgen. Die bestehenden Schwierigkeiten durch mangelnde Flexibilität der generierten Werkzeuge bezüglich Erweiterbarkeit und Wartbarkeit sollen durch den Einsatz aspektorientierter Mechanismen minimiert werden. Die nötige Technologie wird durch *OT/Equinox* bereitgestellt, welches die Implementierung von Aspekt-Plugins mit *ObjectTeams/Java* [HHM07] auf Basis von Eclipse Equinox ermöglicht [HM07]. Hierzu müssen verschiedene Integrationsszenarien entwickelt und untersucht werden, die mit Hilfe der Modularisierungs- und Kompositions-Features von OT/Equinox umgesetzt werden sollen. Insbesondere die Frage nach symmetrischer oder asymmetrischer Komposition sowie der Anteil an generierbaren Artefakten - auch bei den Aspekten - spielt hier eine wichtige Rolle. Die geeigneten Integrationsszenarien werden dann in einem Framework durch generische Komponenten und Code-Generatoren, sowie Muster für die manuelle Integration umgesetzt.

3.1 Eigener Beitrag

Der Beitrag dieser Arbeit besteht in einem konzeptionellen Ansatz zur modularen Definition von domänenspezifischen Modellierungssprachen basierend auf den genannten Mechanismen und einem Framework für die modellgetriebene Entwicklung zugehöriger Implementierungen und Modellierungswerkzeuge auf Basis der Eclipse Plattform (EMF und GMF) und OT/Equinox.

3.2 Validierung der Ergebnisse

Als Fallstudie für den entwickelten Ansatz und dessen Werkzeuge dient die Anwendung auf die aspektorientierte Programmiersprache Object Teams [HHM07]. Dazu wird im Rahmen der Arbeit eine DSML-Sprachfamilie für die modellgetriebene Entwicklung mit Object Teams als Erweiterung der UML definiert und mit den vorgestellten Techniken die zugehörigen Editoren modellgetrieben entwickelt. Dabei sollen UML2-Anteile in Form von Metamodell und existierenden Werkzeugen weitestgehend wiederverwendet werden. Die Sprachfamilie besteht aus Paketen, die unterschiedliche Sichten oder Abstraktionsebenen realisieren. Dazu gehören Pakete für Struktur und Verhalten, Komponenten in OT/Equinox, Pointcuts, Joinpoint Query Language, sowie erweiterte Feature-Modellierung. Weitere konkrete Einsatzmöglichkeiten wären zum Beispiel die direkte Unterstützung der

UML *compliance levels* durch modulare UML-Implementierung, bessere Integration von Profilen, Metamodellierungs-Werkzeuge für MOF oder SysML-Erweiterungen.

4 Bisherige Arbeiten

Ein Positionspapier [Mos05] motivierte zunächst die Bestrebung, aspektorientierte Mechanismen durchgängig in modellgetriebenen Ansätzen zu verwenden, um auf den verschiedenen Ebenen eine möglichst gute Trennung von Belangen (separation of concerns) zu erreichen. Es folgte die Realisierung eines ersten grafischen Editors für Object Teams, der als UML2 Erweiterung mit Eclipse GMF implementiert wurde. Die Adaptierung von Eclipse-Plugins mit OT/Equinox [HM07] ermöglicht schließlich den beschriebenen Lösungsansatz. Für einige der DSMLs für Object Teams wurden bereits im Rahmen von Diplomarbeiten Vorarbeiten geleistet [Mer07, Wer07]. Dabei wurden Erfahrungen mit der modellgetriebenen Entwicklung und Integration von UML erweiternden Editoren gesammelt.

Literatur

[CB05] Siobhan Clarke und Elisa Baniassad. *Aspect-Oriented Analysis and Design. The Theme Approach*. Addison-Wesley Longman, Amsterdam, 2005.

[HHM07] Stephan Herrmann, Christine Hundt und Marco Mosconi. ObjectTeams/Java Language Definition - version 1.0. Bericht-Nr. 2007/03, Technische Universität Berlin, ISSN 1436-9915, 2007.

[HM07] Stephan Herrmann und Marco Mosconi. Integrating Object Teams and OSGi: Joint Efforts for Superior Modularity. In *Journal of Object Technology, vol. 6, no. 9, Special Issue: TOOLS EUROPE 2007*, Seiten 105–125, October 2007. http://www.jot.fm/issues/issue_2007_10/paper6.

[Mer07] Andreas Mertgen. Modellbasierte Integration von Joinpoint Queries für die aspektorientierte Sprache ObjectTeams/Java. Diplomarbeit, Technische Universität Berlin, Oktober 2007.

[Mos05] Marco Mosconi. Preserving the Separation of Aspects through all Abstraction Levels in Model-Driven Software Development. In *First Workshop on Models and Aspects at ECOOP 2005*, 2005.

[Sch06] Markus Scheidgen. CMOF-Model Semantics and Language Mapping for MOF 2.0 Implementation. In *Proceedings of the 4th MBD and 3th MOMPES Workshop (MBD-MOMPES'06)*, 2006.

[Wer07] Andreas Werner. Modellgetriebene Entwicklung eines erweiterten UML-Editors zur Modellierung von ObjectTeams/Java-Programmen. Diplomarbeit, Technische Universität Berlin, Dezember 2007.

[Zit06] Alanna Zito. UML's Package Extension Mechanism: Taking a Closer Look at Package Merge. Diplomarbeit, Queen's University, 2006.