Software-Entwicklung mit Open Source Werkzeugen – Der GeneSEZ-Ansatz

Tobias Haubold, Wolfgang Golubski, Oliver Arnold, Georg Beier, Gerrit Beine

Westsächsische Hochschule Zwickau
Fakultät Physikalische Technik / Informatik
Dr.-Friedrichs-Ring 2a
08056 Zwickau
toh@fh-zwickau.de
golubski@ fh-zwickau.de

Abstract: Bei der Softwareentwicklung werden häufig eine Vielzahl von Werkzeugen z.B. zum Anforderungsmanagement und zur Modellierung oder auch Bug-Tracker sowie Build-Server eingesetzt. Zur effektiven Softwareentwicklung ist ein gutes Anforderungsmanagement unverzichtbar. Bei agilen Vorgehensweisen werden häufig Modelle zur Architekturdokumentation verwendet, die auch als Ausgangspunkt für die teilautomatisierte Softwareentwicklung dienen können. Dieses Paper stellt den GeneSEZ-Ansatz vor, mit dem durch entwickelte Open Source Werkzeuge Anforderungsbeschreibungen erfasst und Modelle teil-automatisiert in Quellcode transformiert werden können.

1 Motivation

Die Kosten der Softwareentwicklung machen heute immer noch einen großen Teil der Gesamtkosten der Entwicklung von Softwareprodukten aus. Seit mehr als drei Jahrzehnten werden enorme Anstrengungen in Forschung und Entwicklung unternommen, um diesen Kostenfaktor weiter zu reduzieren.

Ziel des hier vorgestellten GeneSEZ-Projektes [GEN] (Generative Software Engineering Zwickau) ist, die Softwareentwicklung als Ganzes effizient zu gestalten. Dabei wurde eine agile Vorgehensweise zur Anforderungsermittlung mit geeigneter Toolunterstützung sowie ein Framework zur Codegenerierung aus UML-Modellen entwickelt. Aus den Modellen wird teilweise oder vollständig ausführbarer Programmcode (z.B. Java, C#, PHP etc.) generiert und so die Entwicklung komplexer Software vereinfacht.

Die im Projekt entstandenen Werkzeuge und Methoden, die allesamt als Open-Source-Werkzeuge zur Verfügung stehen, ermöglichen eine schnelle, qualitativ hochwertige und wirtschaftliche Entwicklung von Systemen für verschiedene Anwendungsbereiche.

Eine Herausforderung bei der Anforderungsermittlung ist das Verwalten der Anforderungen. Bei komplexeren Fragestellungen wird eine Vielzahl von Dokumenten

erstellt, die meistens auch noch miteinander verknüpft sein müssen. Daher ist eine elektronische Unterstützung unabdingbar. Dies ist durch ein geeignetes Wiki-System [ABG08], das auf der Basis des bekannten Volere-Prozesses [RR06] und der Volere-Templates entworfen wurde, realisiert.

Die Object Management Group (OMG) [OMG] propagiert eine Vorgehensweise zur rationellen, teilautomatisierten Softwareentwicklung auf der Basis formaler UML-Spezifikationen, die sogenannte Model-Driven Architecture (MDA) [MDA]. Im Gegensatz zu dem Ziel der OMG, Interoperabilität zwischen Softwareentwicklungswerkzeugen und Standardisierung von Modellen für Anwendungsbereiche zu erreichen, zielt der hier dargestellte Ansatz (MDSD – Model-Driven Software Development) auf die Bereitstellung von praktisch einsetzbaren Komponenten und Werkzeugen für den Softwareentwicklungsprozess.

Kapitel 2 betrachtet die Analyse und das Management von Anforderungsbeschreibungen, während Kapitel 3 auf die teilautomatisierte Softwareentwicklung mit Modellen eingeht. Anschließend werden verwandte Arbeiten sowie die bisherigen Erfahrungen betrachtet.

2 Anforderungsermittlung und -Management

Das korrekte Ermitteln der Anforderungen eines zu entwickelnden Produktes stellt heute immer noch eine sehr anspruchsvolle und kreative Tätigkeit dar. Basierend auf langjährigen Erfahrungen in der Praxis haben sich verschiedene Herangehensweisen entwickelt, die situationsbedingt erfolgreich eingesetzt werden können. Unabhängig von der Art der Erfassung sowie der Struktur von Anforderungen ist eine elektronische Unterstützung durch ein geeignetes Werkzeug unerlässlich. Dies kann mit Hilfe eines Wiki-System realisiert werden.

Wiki-Systeme als einfach zu bedienende und oftmals frei verfügbare Werkzeuge für die kollaborative und (oftmals räumlich) verteilte Arbeit an der evolutionären Erarbeitung von Wissen erfreuen sich immer größerer Beliebtheit. Dies schließt immer häufiger auch den professionellen Einsatz durch Unternehmen ein. Weit verbreitet ist auch die Nutzung von Wiki-Systemen im Rahmen von Softwareentwicklungsprojekten, allerdings eher in der Realisierungsphase. Auf der Basis des Volere-Prozesses und der Volere-Templates [RR06] ist ein Twiki entstanden. In der Twiki-Lösung [ABG] wird für jede Anforderungsspezifikation ein eigenes Web (eine Verwaltungseinheit in einer TWiki-Installation bestehend aus einer Reihe verlinkter Webseiten) angelegt. Diese basieren jeweils auf einem so genannten Template Web, das die grundlegenden Strukturen enthält, die für eine auf dem Volere-Template basierende Anforderungsspezifikation notwendig sind.

Wichtig für die Anwendbarkeit von TWiki für die Anforderungsermittlung ist die Möglichkeit, strukturierte Daten miteinander zu verknüpfen. Dies geschieht, indem die Inhalte einer TWiki Form dynamisch über Abfragen an das TWiki-System generiert werden.

Basierend auf den Ergebnissen der Anforderungsermittlung wird die Realisierung des Software-Produktes durchgeführt.

3 Das GeneSEZ-Generator-Framework

In der MDA/MDSD-Vorgehensweise zur Softwareentwicklung werden erstellte Anwendungsmodelle automatisch durch Modelltransformationen in Programmcode überführt. Bei den Modellierungssprachen für MDSD-Plattformen sind UML [UML09] sowie Domain Specific Languages (DSLs) weit verbreitet. Die UML stellt objektorientierte Modellierungskonzepte zur Verfügung und bietet eine gute Abstraktionsebene zur Programmcodegenerierung. Leider entwickelte sich die UML mit einer steigenden Anzahl von Modellierungskonzepten zu einer aus folgenden Gründen für Modelltransformationen eher ungeeigneten Modellierungssprache:

- sie wurde sehr komplex [Th03,WM07a,WM07b]
- sie enthält für die Programmcodegenerierung uninteressante Informationen
- die UML Spezifikation enthält keinen Überblick des UML-Metamodells, aus dem hervorgeht, wie auf bestimmte Informationen zugegriffen werden kann
- sie enthält Informationen über die grafische Darstellung von Modellelementen, z.B. den Bildern von Stereotypen
- u.a. [MB02,Ba07] zeigen auf, dass nur ein Teil der UML sinnvoll für MDA/MDSD-Plattformen nutzbar ist.

Neben UML gibt es DSLs, die Konzepte eines Anwendungsbereiches oder technische Aspekte zur Beschreibung von Modellen nutzen. Dabei sind vor allem die fachlichen DSLs interessant, da die Modelle mit den Experten der Anwendungsbereiche diskutiert werden können. Jedoch vergrößern diese DSLs den Unterschied zwischen Modellierungskonzepten und den Konzepten der Programmiersprachen.

Die Idee bei GeneSEZ ist, nicht das Metamodell der Anwendungsmodelle für die MDSD-Plattform zu nutzen, sondern ein knappes und für die objektorientierte Quellcodegenerierung geeignetes Metamodell zu erstellen. Die MDSD-Plattform wird somit zu einer stabilen, wiederverwendbaren und investitionssicheren Basis für Modelltransformationen zur Quellcodegenerierung, die unabhängig von Modellierungswerkzeugen ist. Abbildung 2 zeigt die MDSD-Plattform [Ha09] schematisch mit den folgenden charakteristischen Konzepten:

- das GeneSEZ-Metamodell [HBG09] als zentraler Bestandteil stellt Informationen in aufbereiteter Form für Modelltransformationen zur Verfügung
- Modelladapter zur Anbindung von Modellierungswerkzeugen, indem Anwendungsmodelle in GeneSEZ-Modelle transformiert werden
- die GeneSEZ-Komponenten als gemeinsame Basis für die Erzeugung von Quellcode für verschiedene Programmiersprachen
- Plattformprojekte mit Skripten und Templates für die jeweiligen Programmiersprachen sowie Bibliotheken und Frameworks zur Quellcodegenerierung

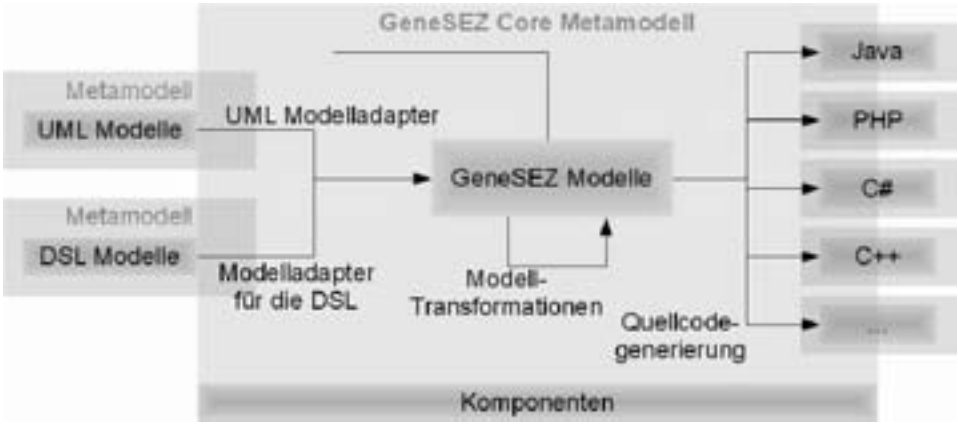

Abbildung 2: Schematischer Überblick des GeneSEZ MDSD Ansatzes

Anwendungsmodelle sind unabhängig von der MDSD-Plattform und können separat entwickelt werden. In iterativen Zyklen können die Modelltransformationen ausgeführt und Quellcode generiert werden. Da meistens der Quellcode nicht sinnvoll vollständig generiert werden kann, gibt es zur Vervollständigung der Implementierung geschützte Bereiche Der manuelle Quellcode bleibt beim erneuten Generieren erhalten. Durch das iterative Vorgehen sind Informationsrückflüsse aus der Implementierung auf das Anwendungsmodell möglich.

Für die Entwicklung von Modelltransformationen ist es hilfreich, dass das GeneSEZ-Metamodell vollständig auf zwei Seiten als Navigationsübersicht ausdruckbar ist.

4 Verwandte Arbeiten

Stanislaw Wrycza et al [WM07a,WM07b] beschreiben die Komplexität der UML2 aus der Perspektive der Lehre. Sie konzentrieren sich auf die Definition einer reduzierten und beschränkten Menge an Modellierungskonzepten und Diagrammen, die sie Light UML nennen. Während Stanislaw Wrycza et al sich auf das Wissen zur Erstellung robuster UML-Modelle konzentrieren, vereinfacht der hier vorgestellte Ansatz die Auswertung von UML-Modellen zur Quellcodegenerierung.

AndroMDA [AND] ist ein MDA-Open-Source-Framework mit Unterstützung für weit verbreitete Architekturen, u.a. Spring, EJB, Hibernate, Struts und .NET. Es wird ein proprietäres, in Java implementiertes UML-Metamodell sowie Velocity als Template-Sprache zur Codegenerierung eingesetzt, welche keine AOP-Fähigkeiten bietet.

Im Gegensatz zu AndroMDA unterstützt das Open-Source-Framework openArchitectureWare (oAW) [OAW] neben UML auch Java- oder EMF-basierte Metamodelle. Für Modelltransformationen stehen die Sprachen Xtend (Modell-zu-Modell) und Xpand (Modell-zu-Text) bereit, die beide Unterstützung für die AOP bieten. Unterstützung zur Quellcodegenerierung bietet oAW nicht. Die beiden Open-Source-

Projekte fornax-Plattform [FOR] und Sculptor [SCU] stellen Modelltransformationen auf Basis des UML-Metamodells bzw. einer DSL für Webanwendungen bereit. Der hier vorgestellte Ansatz basiert auf oAW, da dessen Sprachen für Modelltransformationen gut geeignet sind.

5 Praktische Anwendung und Ergebnisse

Der GeneSEZ-Ansatz wurde im Rahmen eines BMBF-Forschungsprojektes mit zwei kleinen- und mittelständischen Unternehmen entwickelt und in deren realen Projekten eingesetzt. Darüber hinaus wurde er in mehreren Forschungs- und Studentenprojekten sowie in der Lehre weiterentwickelt und angewandt. Die Einsatzgebiete umfassen:

- Entwicklung von Java- sowie EJB3 und Seam-basierter Anwendungen
- Eingebettete Systeme basierend auf Java (Java ME)
- Reengineering und Portierung einer C++ -basierten Software auf C#
- Entwicklung PHP-basierter Webanwendungen

Der Anteil des generierten Quellcodes lag zwischen 50% und 80% Dieser Wert ist abhängig von der Art der Software und der vorhandenen Unterstützung des Generators. Bei eingebetteten Systemen liegt der Wert an der oberen Grenze, während bei Webanwendungen der Wert an der unteren Grenze liegt. Die Ursache ist die aktuell noch fehlende Unterstützung für View-Technologien. Bisherige Tests zeigen auch hier Automatisierungspotenzial.

Der Ansatz ist eine pragmatische Realisierung der MDA/MDSD-Konzepte und entwickelte sich durch Anwendungen in der Industrie, Forschung sowie Studium und Lehre zu einem praktischen Ansatz mit vielversprechenden Ergebnissen:

- Trennung von Modellierungswerkzeugen und MDSD-Prozess durch Einführung eines einfachen Metamodells
- Nutzung von UML sowie DSLs zur Anwendungsmodellerstellung
- Knappes und präzises Metamodell für objektorientierte Software
- Wiederverwendbare und anpassbare Modelltransformationen
- Hohe Investitionssicherheit in Transformationen durch unabhängiges Metamodell
- Hohe Quellcodequalität durch Generierung von sauberen, kompilierbaren, fehlerfreien und geprüften Quellcode gemäß Quellcode-Konventionen / Code-Style
- Unterstützt iterative und agile Softwareentwicklung

Die Aktualität der Modelle zum Quellcode ist ständig gegeben. Da diese Softwaremodelle auch ein Teil der Dokumentation sind, wird die Aktualität dieser nachhaltig verbessert.

Durch das vom Generator erzeugte Assoziationshandling sind die Objektmodelle stets konsistent. Bei bidirektionalen Assoziationen wird die Gegenseite automatisch gesetzt. Bei den Plattformen, bei denen eine selbst erstellte Bibliothek zum Assoziationshandling

eingesetzt wird, ist diese unter LGPL lizensiert, um auch in kommerziellen Projekten den Einsatz zu ermöglichen.

Das einfach strukturierte Metamodell sowie die Unterstützung aspektorientierter Programmierung (AOP) für Modelltransformationen, erlauben eine einfache projektspezifische Anpassung des GeneSEZ-Frameworks.

6 Ausblick

Aktuell werden Strukturmodelle und Zustandsautomaten unterstützt. An der Unterstützung bei der Entwicklung von Tests wird aktuell gearbeitet sowie auch an der stärkeren Einbeziehung von Anforderungsbeschreibungen.

Der GeneSEZ-Ansatz basiert auf Open Source (openArchitectureWare sowie verschiedenen Eclipse-Modeling-Projekten) und ist selbst Open Source. Er ist unter der GPL lizenziert mit Ausnahme der LGPL-lizenzierten Assoziationshandlingbibliotheken. Dadurch kann auch kommerzielle Software entwickelt werden. Lediglich Änderungen am Framework selbst müssen durch die GPL veröffentlicht werden.

Literaturverzeichnis

[ABG08] Arnold, O.; Beier, G.; Golubski, W.: Requirements Management 2.0. In: Proceedings of the IASTED International Conference on Software Engineering (SE 2008), Innsbruck, Austria, ACTA Press, Calgary, Canada, 2008.

[AND] Homepage von AndroMDA, http://www.andromda.org/

[Ba07] Baker, P.; et.al.: Model-Driven Testing: Using the UML Testing Profile. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[FOR] Homepage der fornax Platform, http://fornax-platform.org/

[GEN] Homepage des GeneSEZ-Projektes, http://www.genesez.de/

[Ha09] Haubold, T.; et.al.: The Technical Foundation of the GeneSEZ MDSD Approach. In: New Trends in Software Methodologies, Tools and Techniques: Proceedings of the Eighth SoMeT_09 (Frontiers in Artificial Intelligence and Applications), IOS Press, September 2009.

[HBG09] Haubold, T.; Beier, G.; Golubski, W.: A Pragmatic UML-based Meta Model for Object-oriented Code Generation. In: Proceedings of the 21st International Conference on Software Engineering & Knowledge Engineering (SEKE'2009), Juli 2009.

[MB02] Mellor, S.; Balcer, M.: Executable UML: A Foundation for Model-Driven Architectures. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[MDA] Überblick der OMG MDA-Spezifikationen, http://www.omg.org/mda/specs.htm.

[OAW] Homepage von openArchitectureWare, http://www.openarchitectureware.org/

[OMG] Homepage der Object Managment Group (OMG). http://www.omg.org/.

[RR06] Robertson, S.; Robertson, J.: Mastering the requirements process, second edition Pearson Education, Inc. 2006.

[SCU10] Sculptor, http://fornax.itemis.de/confluence/display/fornax/Sculptor+(CSC)

[Th03] Thomas, D.: Uml - Unified or Universal Modeling Language? Journal of Object Technology, 2(1):7–12, January-February 2003.

[UML09] Die UML Superstructure Spezifikation der OMG 2.2, Februar 2009.

[WM07a] Wrycza, S.; Marcinkowski, B.: Towards a light Version of UML 2.x: Appraisal and Model. Organizacija, 40(4), 2007.

[WM07b] Wrycza, S.; Marcinkowski, B.: A light Version of UML 2: Survey and Outcomes. In Proceedings of the 2007 Computer Science and IT Education Conference, 2007.