Modellgetriebene Softwarenentwicklung mit Open–Source Komponenten im industriellen Einsatz: Ein Erfahrungsbericht

Björn Hansen[1], Gisela Stumm[1], Olaf Zukunft[2]

[1] Werum Software & Systems AG, Wulf–Werum–Straße 3,
21337 Lüneburg, {nachname}@werum.de,
[2] HAW Hamburg, Department Informatik, Berliner Tor 7,
20099 Hamburg, zukunft@informatik.haw-hamburg.de

Abstract: Die im kommerziellen Einsatz befindliche Softwareproduktlinie „HyperTest" realisiert auf J2EE basierende Informationssysteme zum Meßdatenmanagement. In diesem Beitrag werden Erfahrungen geschildert, die bei der Einführung eines modellgetriebenen Softwareentwicklungmodells für „HyperTest" gemacht wurden. Die Nutzung von Open–Source Komponenten erweist sich als ein kritischer Erfolgsfaktor bei der inkrementellen Werkzeugeinführung.

1 Einleitung

In der klassischen Softwareentwicklung dienen Modelle häufig allein zu Dokumentationszwecken. In der industriellen Praxis sind sie — soweit existent — häufig Eins-zu-Eins-Abbildungen des Quellcodes und daher lediglich schwach abstrahierend.

Der Einsatz modellgetriebener Softwareentwicklung wie etwa durch die Model Driven Architecture (MDA) der OMG [Obj01, Fra03] definiert, verspricht Effizienzsteigerungen gegenber herkömmlichen Entwicklungsprozessen. Die Vision der MDA ist ein modellbasierender Entwicklungsprozess, in dem das Softwaresystem zunächst plattformunabhängig modelliert wird. Anschließend soll das Modell um plattformspezifische Eigenschaften angereicht werden und so für verschiedene Plattformen ausführbare Software generierbar sein, die die Semantik der Anwendung über das Modells abbildet. Für die Umsetzung des modellgetriebenen Softwareentwicklungsansatzes sind verschiedene Open–Source und Closed–Source Werkzeugumgebungen verfügbar. Solche Werkzeugumgebungen sind kritisch für die Einführung des neuen Entwicklungsansatzes, da die Generierung großer Anwendungsteile einen der wirtschaftlich attraktivsten Aspekte der MDA darstellt.

Die Nutzung der modellgetriebenen Softwareentwicklungsverfahren erfordert in der industriellen Praxis einige Anpassungen sowohl bezüglich des bisher verwendeten Softwareentwicklungsprozesses als auch der eingesetzten Werkzeuglandschaft. Diese werden ebenso wie die Werkzeugauswahl im weiteren beschrieben.

2 Grundlagen

2.1 Die Softwareproduktlinie „HyperTest”

HyperTest ist eine kommerzielle auf J2EE basierende Closed–Source Softwareproduktlinie (SPL), die von der Firma Werum AG entwickelt wurde und von verschiedenen Kunden — jeweils spezifisch angepasst — eingesetzt wird. Dem Charakter einer SPL folgend existiert in HyperTest eine kundenunabhängige Kernfunktionalität, die für jeden Kunden spezifisch erweitert und angepasst wird. Die Kernfunktionalität ermöglicht eine effiziente Verwaltung großer Mengen technischer Daten, auf die flexibel und langfristig zugegriffen werden soll. Die kundenspezifischen Erweiterungen umfassen u.a. die Benutzungsoberfläche, die Navigationsstruktur zum Zugriff auf die Daten und die konkreten Datenstrukturen selbst. Derzeit wird die Weiterentwicklung des Softwarekerns und der kundenspezi-

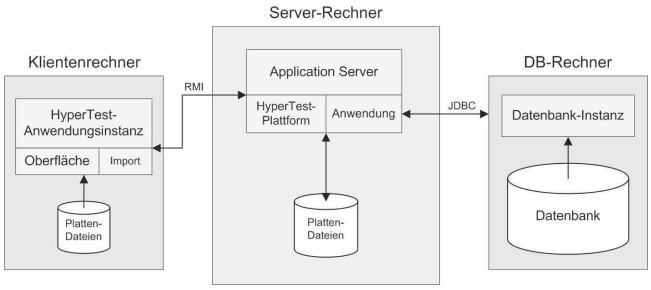

Abbildung 1: Überblick über HyperTest

fischen Erweiterungen in einem traditionellen Softwaremodell vorgenommen.

2.2 Modellgetriebene Softwareentwicklung

Ziel der modellgetriebenen Softwareentwicklung ist es, den Softwareentwicklungsprozess auf eine höhere Abstraktionsebene zu heben. Besonders im Umfeld verteilter Client-Server-Anwendungen ist ein sehr hoher Komplexitätsgrad erreicht worden, der die Entwicklung fehlerbehaftet und unübersichtlich werden lässt, wodurch die Softwarequalität sinkt und Dauer und Kosten des Entwicklungsprozesses ansteigen.

Die OMG hat mit der MDA einen möglichen Ansatz zur Realisierung modellgetriebener Softwareentwicklungsmodelle vorgestellt. Die OMG definiert MDA wie folgt: „The Model Driven Architecture (MDA) defines an approach to IT system specification that separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform.” [Obj01] Die Umsetzung dieses Ansatzes erfordert ein Werkzeug zur Modellierung der Spezifikation und ein weiteres Werkzeug, das im Idealfall aus der Spezifikation die Implementierung des Soft-

waresystems generiert. Weiterhin ist offensichtlich auch im Softwareentwicklungsprozess eine Änderung erforderlich. Der Entwicklungsschwerpunkt liegt noch eindeutiger auf der Spezifikation der Anwendungsfunktionalität und auf der Bereitstellung der oben angesprochenen Werkzeuge.

3 Vorgehen und Ergebnisse

In diesem Abschnitt wird die Nutzung eines modellgetriebenen Softwareentwicklungsvorgehens für HyperTest beschrieben.

3.1 Werkzeugauswahl

Die Entscheidung für ein UML–Modellierungswerkzeug ist industrietypisch aufgrund eines vorhandenes Systems vorgegeben, hat sich aber aufgrund der standardisierten XMI–Schnittstelle zum Modellaustausch auch als weitgehend irrelevant herausgestellt. Neben der Verwendung eines Modellierungstools zur Spezifikation der Anwendungsfunktionalität erfordert ein MDA-konformer Entwicklungsprozess aber insbesondere einen Satz von Laufzeitkomponenten. Diese Laufzeitkomponenten können Bibliotheken, generische Dienste, etc. beinhalten und werden im Folgenden unter dem Begriff “Framework” zusammengefasst. Ein besonders wichtiges Element jedes Frameworks ist die Generator--Komponente. Der Generator dient der Umsetzung der Transformationsregeln. Allen Laufzeitkomponenten gemein ist, dass sie technisch motiviert sind und keinen fachlichen Bezug zu dem zu erstellenden System haben. Es existieren verschiedene freie und kommerzielle Umgebungen und Frameworks, die einen MDA-konformen Ansatz in der Entwicklung unterstützen sollen. Der Umfang der Open-Source und einiger kommerzieller MDA-Frameworks ist sowohl funktional als auch im Verfahren ähnlich. Allgemein wird das in einem XMI/XML-Format vorliegende Modell mittels Transformation auf eine Plattform, häufig direkt in Quellcode, umgesetzt. Diese Frameworks unterstützen eine automatisierte Modellabbildung von plattformunabhängigen Modellen in Code.

Die Spezifikation der Transformationsregeln erfolgt aufgrund der unzureichenden Festlegung der Mappingregeln in der MDA-Spezifikation mit Hilfe einer Skriptsprache. Einsatz finden sowohl existierende Sprachen wie Velocity[1], Python[2] oder XML-Transformation per XSLT als auch Eigenentwicklungen wie die z.B. im openArchitectureWare-Framework eingesetzte Template-Sprache Xpand[3].

Ein sich schnell abzeichnendes Kriterium für die Werkzeugauswahl war die Verfügbarkeit der MDA–Umgebung als Open–Source Komponenten. Hintergrund dafür ist

- die Notwendigkeit, in einem kommerziellen Umfeld die Kosten für die erstmalige

[1] siehe auch: http://jakarta.apache.org/velocity/index.htm (Stand: 24.04.2005)
[2] siehe auch: http://www.python.org/ (Stand: 24.04.2005)
[3] siehe auch: http://architecturware.sourceforge.net/ (Stand: 24.04.2005)

Erprobung einer neuen Technologie wie MDA möglichst gering zu halten,

- der Bedarf, das Generat und den Generierungsprozess prinzipiell uneingeschränkt an die besonderen Bedürfnisse des Unternehmens anzupassen zu können.

Diese Eigenschaften sind mit Closed–Source Komponenten nicht umsetzbar. Die folgende Recherche von Open–Source Komponenten hat ergeben, dass sich das Open-Source MDA-Framework "openArchitectureWare" (oAW) (siehe [OAW05]) für HyperTest eignet. Aufgrund der guten Dokumentation [SV05] wurde es für die Prototypverwendung ausgewählt. oAW stellt Unterstützung bei der Erstellung eines domänenspezifischen Codegenerators zur Verfügung. Die Komponenten des oAW bestehen aus dem Generator–Kern, einem Java–Metamodell der UML, der Template-Sprache Xpand und einigen weiteren Komponenten. Es werden prinzipiell Modelle in beliebiger textueller Notation unterstützt und es kann Programmcode und Dokumentation in jeder beliebigen Sprache erzeugt werden, sofern dazu Anpassungen in Metamodell und Templates vorgenommen werden. Xpand unterstützt Polymorphismus auf Template-Ebene und hebt die variablen Anteile der Templates durch die ausgezeichneten Steuerzeichen $\ll$ und $\gg$ von anderen Sprachen wie Java oder XML und dem fixen Anteil, dem zu generierenden Code, ab. Teil des oAW-Framework ist ein Plugin für die Entwicklungsumgebung Eclipse[4]. Dieses Plugin unterstützt den Entwickler durch Hervorhebungen und Codevorschläge bei der Erstellung der Templates.

Das openArchitectureWare zugrundeliegende architekturzentrierte Design wird als MDA–Variante unter Verzicht auf den Einsatz plattformspezifischer Modelle verstanden. Stattdessen finden plattformunabhängige Modelle in architekturzentriertem Design Anwendung, die per Transformationsvorschriften direkt in ausführbaren Code übersetzt werden. Die Bindung eines Modells zu der verwendeten Architektur erfolgt durch Interpretation der Elemente einer UML-Erweiterung bei der Generierung. Auf diese Weise können z.B. bestimmte Stereotypen in den Kontext der Architektur gebracht werden. Beispiel: Eine Klasse mit dem ausgezeichneten Stereotyp $\ll$entity$\gg$ erhält im Zusammenhang mit einer EJB–basierten Architektur die Bedeutung einer Entity-Bean mit ihren Remote-Interfaces. Im Kontext einer C++/CORBA-basierten Architektur wäre eine entsprechende Interpretation desselben Stereotyps eine nicht verteilbare C++-Klasse, deren Instanzen mittels objektrelationalem Mapping in einem relationalen Datenbank-System abgelegt werden würden.

Eine Bedeutung im Sinne der Plattform wird den Modellen also auf der Transformationsebene gegeben. Dadurch lassen sich bestehende Modelle auf nahezu beliebige Architekturen abbilden, insbesondere auch auf die J2EE–Plattform von HyperTest. Somit ergibt sich die Möglichkeit der automatisierten Transformation der Modelle in Quellcode, womit auch eine Konsistenz von Modell und Code erzwungen wird. Die Modelle dienen also der Codegenerierung auf der Abstraktionsebene eines plattformunabhängigen Modells.

[4] siehe auch: http://www.eclipse.org/ (Stand: 24.04.2005)

3.2 Anpassung des Open–Source Generatorframeworks

Um ein Modell zu erstellen, müssen in oAW zunächst UML–Profile erstellt werden. Auf diesen basieren die zu erstellenden Referenzmodelle, die die Struktur einer bestehenden HyperTest-Anwendungsinstanz widerspiegeln. Die Modellierung erfolgt mittels eines prinzipiell beliebigen UML–Werkzeuges in Form von Klassen- und Aktivitätsdiagrammen. Die Erweiterungen am openArchitectureWare-Framework erfolgen hauptsächlich in Form von Java-Code. Einzelne Konfigurationsdateien sind in XML-Form verfasst. Für die Definition der Transformationen steht im oAW die Sprache Xpand zu Verfügung.

Die HyperTest–Modelle sind in zwei UML-Packages aufgeteilt. Ein Package spezifiziert die zu erzeugende Anwendungsinstanz in zwei Klassenmodellen. Zur Generierung sind jedoch auch weitgehende Template–Artefakte zu stellen. In unserem Ansatz werden diese nicht händisch, sondern ebenfalls mit oAW erstellt. Das dazu definierte zweite Package beinhaltet die Generierungsvorschriften, modelliert in Klassen- und Aktivitätsdiagrammen. Die Generierungsvorschriften, die die Struktur der Templates zur Codegenerierung definieren, sind in zwei Klassen- und derzeit vier (je nach Bedarf auch mehr) Aktivitätsdiagrammen festgehalten. Diese Nutzung von oAW zur Metamodellierung durch Templategenerierung ist eine weitere Anpassung in der Verwendung von oAW.

4 Bewertung und Ausblick

Die Nutzung des Open–Source Frameworks openArchitectureWare hat gezeigt, dass die Überführung eines Softwareprodukts in eine modellgetrieben weiterzuentwickelnde Softwareproduktlinie auch unter industriellen Randbedingungen möglich ist. Mit Closed–Source Komponenten wäre das Risiko der Erprobung einer neuen Softwareentwicklungstechnologie zu hoch gewesen, da der Zeitaufwand für Produktanpassungen und Kosten nicht tragbar wären. Der vorgestellte Ansatz ist in der Lage, eine Instanz von HyperTest zu generieren. Da derzeit insbesondere das Datenmodell und die Interaktionskomponente modelliert wurden, soll als nächstes untersucht werden, mit welchem Aufwand weitere typische Anpassungsaufgaben in der SPL modelliert und in oAW generiert werden können.

Literatur

[Fra03] Frankel, Davis S. *Model Driven Architecture. Applying MDA to Enterprise Computing*. Wiley Publishing, Inc., 1. Edition, 2003.

[OAW05] *openArchitectureWare Generator Framework*, 2005. URL `www.architectureware.org`.

[Obj01] Object Management Group. *MDA Public Draft*. *OMG Press*, 2001. URL `http://www.omg.org/cgi-bin/apps/doc?ormsc/01-07-01.pdf`.

[SV05] Stahl, Thomas und Voelter, Markus. *Modellgetriebene Softwareentwicklung*. dpunkt Verlag, 1. Auflage, 2005.