Semesterarbeit

Plattformübergreifende Entwicklung mithilfe modellgetriebener Methoden und Technologien

Semesterarbeit

im Fach Aktuelle Themen der WI
im Studiengang Wirtschaftsinformatik
der Hochschule für Technik und Wirtschaft Berlin

vorgelegt von: Mathias Slawik
Achillesstr. 44
13125 Berlin
Matrikel-Nr.: 517918

Prüfer: Tom Eckart

Abgabetermin: 14.01.2011

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Mithilfe modellgetriebener Entwicklungsmethoden können in diversen Anwendungsgebieten gegenüber herkömmlichen Methoden der Softwareentwicklung Vorteile erzielt werden.

Diese Semesterarbeit soll die Methode der modellgetriebenen Softwareentwicklung erläutern und einige Technologien vorstellen, die modellgetriebene Entwicklung unterstützen. Darüber hinaus werden die Chancen und Risiken modellgetriebener Softwareentwicklung aufgezeigt, sowie eine Anwendung der Methoden auf plattformübergreifende Entwicklung anhand eines Praxisbeispiels erörtert.

Für eine ausführlichere Darstellung der Methoden und ein weiteres Anwendungsgebiet, möchte ich auf meine Bachelorarbeit [1] verweisen. Das dargestellte Praxisbeispiel wird voraussichtlich als Masterarbeit ausgearbeitet werden.

[1] (Slawik, 2008)

2 Methode & Technologie

Dieses Kapitel erläutert die Methodik der modellgetriebenen Entwicklung und stellt das Eclipse Modeling Project vor – eine Sammlung von Werkzeugen, mit Hilfe derer modellgetriebene Entwicklung vonstattengehen kann.

2.1 Methode: Modellgetriebene Entwicklung

Die zentralen Aspekte modellgetriebener Entwicklung lassen sich wie folgt zusammenfassen:

1. Die Rolle der Modelle entspricht der Rolle des Programmcodes.

   Bei der modell*basierten* Entwicklung werden Modelle (z.B. UML[2], BPMN[3]) in der Regel als Dokumentation und Spezifizierung eines Softwaresystems verwendet.

   In der modell*getriebenen* Entwicklung werden Modelle als Grundlage der Generierung von Programmcode und damit großer Teile des Softwaresystems verwendet.

   Damit spielen Modelle bei der modellgetriebenen Entwicklung die gleiche Rolle wie der Programmcode und können daher als Laufzeit-Instanz des Modells betrachtet werden.

2. Verwendung domänenspezifischer Metamodelle (Sprachen).

   Bei der modell*basierten* Entwicklung finden solche Modelle Anwendung, die generischer Natur sind – beispielsweise „(Software-)systeme“ (UML), „Geschäftsprozesse“ (BPMN) oder Data-Warehouse – Systeme (CWM[4]).

   In der modell*getriebenen* Entwicklung werden domänenspezifische Metamodelle und Sprachen verwendet, die auf den jeweiligen Anwendungsfall bezogen sind, beispielsweise „REST Web-Services“ oder „Typo3 Extensions“.

[2] (Object Management Group, 2010a)
[3] (Object Management Group, 2010d)
[4] (Object Management Group, 2010c)

Dieser Bezug auf eine konkrete Domäne sorgt für eine höhere Griffigkeit der Modellierungssprache und einer besseren Verständlichkeit des Modells durch den Fachanwender. Die Verwendung eines einheitlichen Metamodells ermöglicht die Transformation unterschiedlicher domänenspezifischer Modelle untereinander.

3. Generierung von Software aus dem Modell

   In der modell*basierten* Entwicklung wird das Softwaresystem in der Regel von Programmierern auf Basis der Spezifikation (z.B. UML- und BPML-Diagramme, Pflichtenhefte, Architekturmodelle, etc.) manuell implementiert.

   In der modell*getriebenen* Entwicklung werden große Teile des Softwaresystems (z.B. Programmcode, Dokumentation, Schnittstellenbeschreibung) aus den Modellen generiert und nur ein kleiner Teil manuell implementiert.

   Voraussetzung dafür sind Werkzeuge, die eine flexible Codegenerierung ermöglichen, wie beispielsweise die des Eclipse Modeling Projects.

2.2 Definition: Meta-Ebenen

Grundlage modellgetriebener Softwareentwicklung sind mehrere, hierarchisch über Instanziierungsbeziehungen verknüpfte Modelle. In der Regel lassen sich innerhalb dieser Modellhierarchie vier Ebenen identifizieren, die fortlaufend von M0 bis M3 bezeichnet werden.

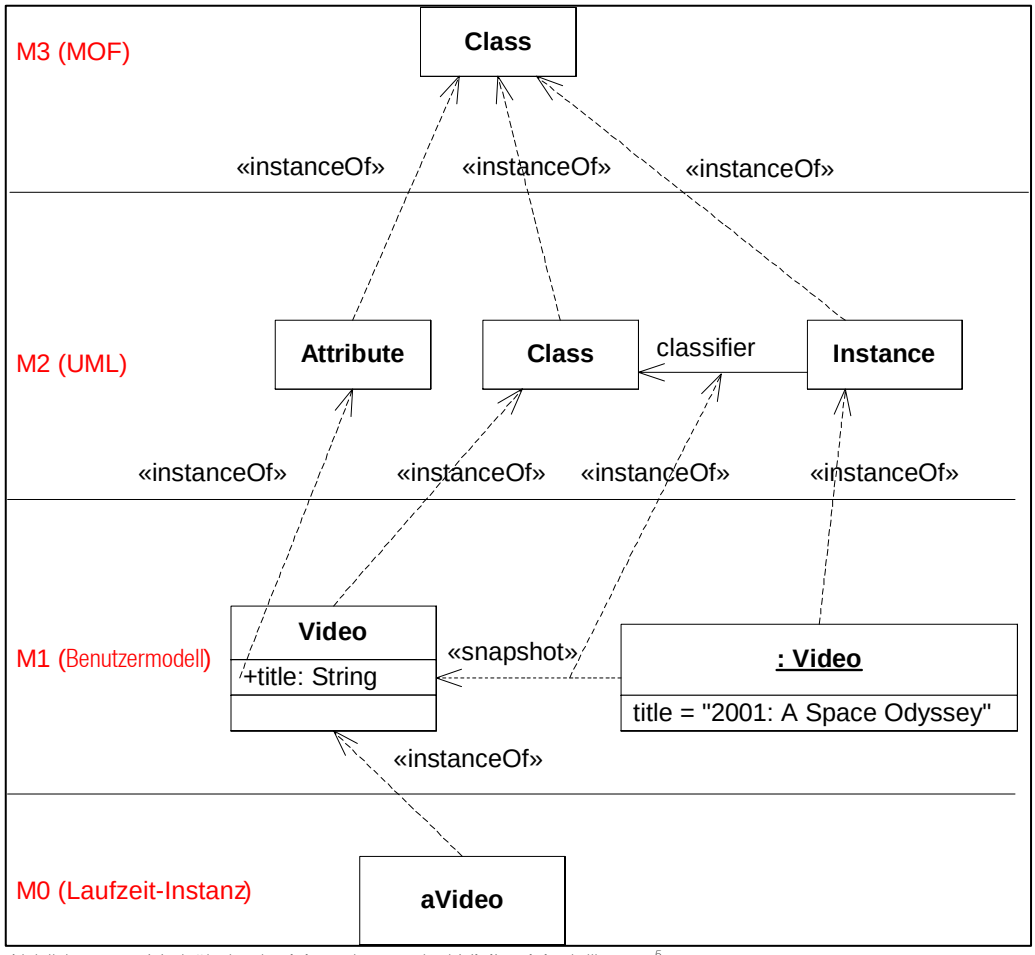

Abbildung 1 - Verhältnis der Metaebenen bei UML - Modellierung[5]

Die Oberste Ebene (M3) stellt das Meta-Metamodell dar. Dieses definiert die formale Syntax und Semantik aller erbenden Modellelemente der darunterliegenden Ebenen. Die am häufigsten verwendete M3-Ebene ist die Meta Object Facility[6] (MOF) der OMG, welche als Ecore[7] eine Referenzimplementierung im Eclipse Modelling Project[8] erfährt und das Meta-Metamodell der UML und des CWM darstellt.

[5] Abgewandelt aus (Object Management Group, 2010b S. 19)
[6] (Object Management Group, 2010e)
[7] (IBM Corporation et al., 2010)
[8] (The Eclipse Foundation, 2011c)

Unter der M3-Ebene befindet sich die M2-Ebene, das Metamodell. Dieses stellt ein domänenspezifisches Modell in der Sprache der M3-Ebene dar. Für die Domäne „(Software-)System“ wäre dies die UML, welche Instanzen der M3-Elemente, wie „Akteur“, „Anwendungsfall“ oder „Aktivität“ enthält. Bei modellgetriebener Entwicklung würde hier das domänenspezifische Modell angesiedelt sein, z.B. in der Domäne „Web-Services“ die Elemente „Service“, „Methode“ oder „Parameter“.

Die M1-Ebene legt fest, in welcher Form das Metamodell (M2) in lauffähige Software (M0) umgewandelt werden soll. Dies geschieht, indem M2-Elemente instanziiert und dadurch konkretisiert werden, beispielsweise in der Domäne „Web-Services“: „Service: Send SMS“, „Methode: Send Flash SMS“, „Parameter: Telefonnummer“ und „Parameter: Textnachricht“.

Die Laufzeit-Instanz wird als M0-Ebene beschrieben. Diese kann manuell implementiert sein, wie beispielsweise bei Software die durch die UML modelliert wird. Bei modellgetriebener Entwicklung wird die Laufzeit-Instanz größtenteils aus dem Modell (M1) generiert. So könnte in der Domäne „Web-Services“ die Laufzeit-Instanz aus SDKs für die Programmiersprachen Java, .NET und PHP bestehen, welche die beschriebenen Web-Services nutzbar machen.

2.3 Technologie: Eclipse Modeling Project

Das Eclipse Modeling Project[9] ist ein Projekt der Eclipse Foundation, welches Softwarewerkzeuge auf Basis der Eclipse Entwicklungsumgebung bereitstellt, welche modellgetriebene Entwicklung unterstützen.

Herausstellende Merkmale des Eclipse Modeling Projects sind:

1. Komplettes Software-Ökosystem von untereinander kompatiblen Modellierungstools

   Das Eclipse Modeling Project enthält zahlreiche Software-Werkzeuge für unterschiedliche Aufgaben, u.A. im Bereich der modellgetriebenen Softwareentwicklung. Durch einheitliche Basistechnologien (Eclipse IDE und ECore) sind alle Werkzeuge untereinander kompatibel.

[9] (The Eclipse Foundation, 2011c)

Darüber hinaus finden sich in der Vielzahl der Software auch unterschiedliche Lösungen für das gleiche Problem (z.B. unterschiedliche Templatesprachen: XPand[10] / Jet[11]), sodass größere Freiheiten in der Wahl der Werkzeuge entstehen.

2. Freie Software (EPL)

   Die Werkzeuge des Eclipse Modeling Projects stehen unter einer freien Softwarelizenz (EPL[12]) teilen damit die Vorteile von freier Software[13].

3. Umsetzung offener Standards

   Offene Standards unterstützen die Interoperabilität der Software-Werkzeuge des Eclipse Modeling Projects.

   Da beispielsweise die Meta-Modelle im XMI[14] – Format vorliegen, können diese auch mit anderen UML-Tools ausgetauscht werden, wie beispielsweise dem Enterprise Architect[15], Innovator[16] oder der ARIS Plattform[17].

In den folgenden Kapiteln werden die wichtigsten Elemente des Eclipse Modeling Projects vorgestellt.

2.3.1 EMF (Core)

Das Eclipse Modeling Framework (EMF) stellt die grundlegenden Technologien für alle anderen Software-Werkzeuge des Eclipse Modeling Projects bereit. Es wird seit 2001 entwickelt und kann nicht zuletzt durch den Initialautor, die IBM, als stabile und ausgereifte Technologie angesehen werden.

Die wesentlichen Bestandteile von EMF (Core) sind:

- Ecore

  Ecore[18] ist das Meta-Metamodell aller Softwarewerkzeuge des Eclipse Modeling Projects und stellt eine Referenzimplementierung der Meta Object Facility[19] (MOF) der OMG dar.

[10] (Irawan, et al., 2011)
[11] (The Eclipse Foundation, 2011b)
[12] (The Eclipse Foundation, 2004a)
[13] Siehe auch (The Free Software Foundation, 2010)
[14] (Object Management Group, 2007)
[15] (SPARX Systems, 2010)
[16] (MID GmbH, 2011)
[17] (Software AG, 2010)

Die Hauptelemente von Ecore gestalten sich wie folgt:

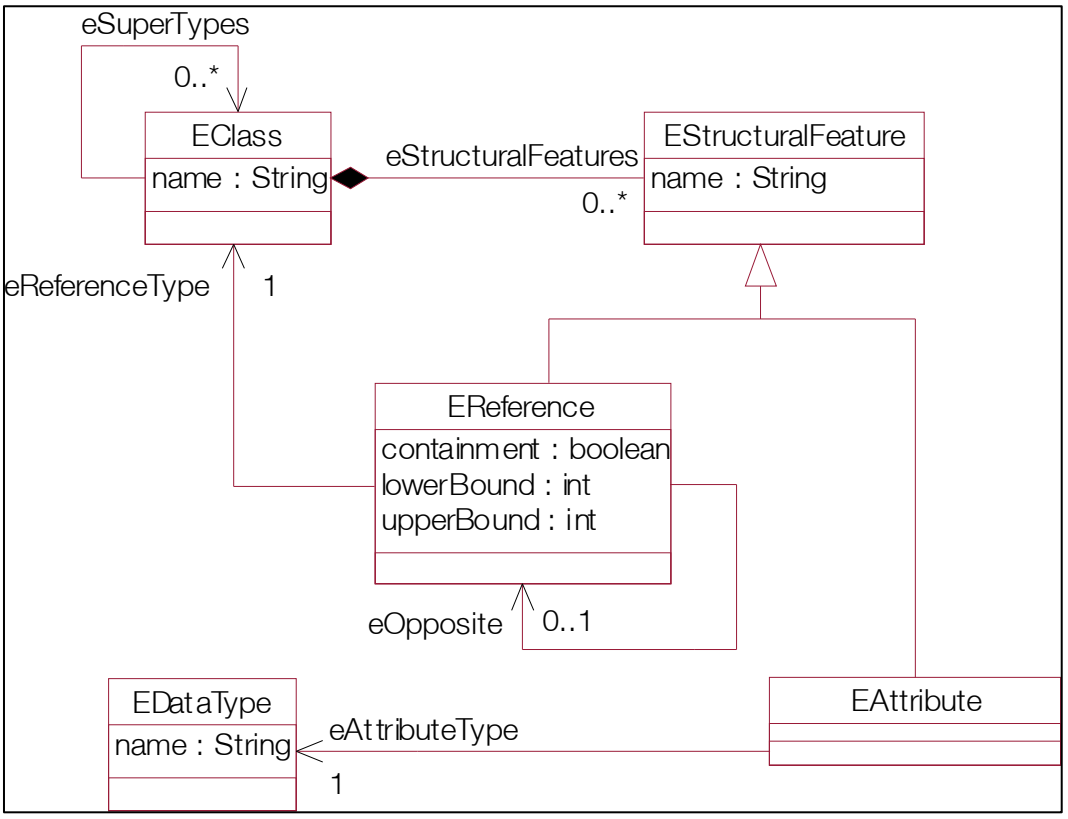

Abbildung 2 - Ecore-Metamodell (Auszug)[20]

Der dargestellte Metamodell-Auszug stellt die geläufigsten Elemente dar:

- EClass (Meta-Klassen)
- Meta-Vererbung (eSuperTypes)
- EStructuralFeature (Strukturelle Eigenschaften einer Meta-Klasse)
- EReference (Meta-Beziehungen, u.A. Enthält-Beziehungen mit containment = true)
- EAttribute (Meta-Attribute)
- EDataType (Meta-Datentypen typisierter (d.h. mit eAttributeType versehener) Meta-Attribute)

[18] (IBM Corporation et al., 2010)
[19] (Object Management Group, 2010e)
[20] (Steinberg, 2008 S. 19)

- EMF.Edit

  EMF.Edit[21] stellt Framework-Elemente bereit, um ECore-Modelle mit den Eclipse Standard-Werkzeugen zu bearbeiten, unter anderem Bearbeitungsfenster und Eigenschaften-Dialogfelder. Dadurch wird eine weitgehende Integration von ECore-Modellen und der darauf basierenden Software in die Eclipse-Plattform erreicht.

- EMF.Codegen

  EMF.Codegen enthält Java Code-Generatoren für Modell-Instanzen und Modell-Editoren. Der generierte Modellcode enthält eine Repräsentation des Modells, sowie zusätzliche Artefakte für die Arbeit mit dem Modell, darunter Klassen und Interfaces, sowie Factories, XMI-Serialisierungsklassen, Validatoren, typsichere Enumerationen und Notifier/Observer für Modell-Instanzen.

- Workspace-Integration

  Darüber hinaus enthält EMF (Core) Werkzeuge zur Integration von EMF-Technologien in Eclipse-Arbeitsbereiche, unter anderem Compare (Modellvergleich), Model Query (Modell-Abfragen) und Model Transaction (Modell-Management).

2.3.2 Vormalige openArchitectureWare – Technologien

Das openArchitectureWare[22] – Framework war ein beliebtes Framework für modellgetriebene Entwicklung. Der Integrationsgrad in die Eclipse-Plattform steigerte sich von Release zu Release, sodass der Sponsor des Frameworks, die itemis AG sich dazu entschloss[23], das Framework unter das Dach der Eclipse-Foundation zu stellen.

2.3.2.1 Typsystem

Grundlage der mittlerweile in das Eclipse Modeling Project aufgegangenen openArchitectureWare – Technologie ist das über die Technologien Xpand, Xtend und Check einheitliche Typsystem, welches die Eigenheiten von dynamischen und statischen EMF-Modellen abstrahiert. Darüber hinaus enthält das Typsystem Datentypen und Methoden für die vereinfachte Arbeit mit Zeichenketten, Listen und weiteren Elementen.

[21] (The Eclipse Foundation, 2004b)
[22] (openArchitectureWare.org, 2009)
[23] (itemis AG, 2009)

2.3.2.2 Xpand

Xpand[24] ist eine der meistverwendeten Templatesprachen (M2T) zur Transformation von EMF-basierenden Modellen in Programmcode.

Xpand ermöglicht die Implementierung von Code-Vorlagen für beliebige, textbasierte Sprachen, z.B. Java, C# oder PHP. Innerhalb von Xpand kann auf in Xtend geschriebene Funktionen zurückgegriffen werden.

2.3.2.3 Xtend

Xtend[25] ist eine funktionale Programmiersprache und kann z.B. verwendet werden, um in Code-Templates wiederkehrende Ausdrücke zu kapseln, beispielsweise Namenstransformationen oder Berechnungen.

Xtend kann darüber hinaus auch zur Modelltransformation (M2M) verwendet werden und stellt durch den funktionalen Programmierstil eine oftmals bessere Alternative bezüglich Lesbarkeit und Wartbarkeit zu manuell implementierten Java-Code oder XSLT-Transformationen dar.

Xtend-Funktionen können auch Java-Methoden aufrufen, sodass vielfältige weitere Technologien über diese Methode innerhalb von M2T verwendet werden können.

2.3.2.4 Check

Check[26] ist eine Technologie, welche es erlaubt, semantische Validierungsregeln für Modelle zu definieren. Diese Validierungsregeln werden wiederum als Xtend-Ausdrücke definiert.

2.3.2.5 MWE

Die openArchitectureWare Workflow Engine, welche mittlerweile als MWE(2)[27] im Eclipse Modelling Project zu finden ist, kann zur Beschreibung von Transformations-Workflows verwendet werden.

2.3.2.6 Xtext

Aus den Erfahrungen mit dem oAW-Typmodell und den Technologien Xpand, Xtend und Check entstand die Technologie Xtext[28].

[24] (Skrypuch, 2011)
[25] (The Eclipse Foundation, 2011g)
[26] (The Eclipse Foundation, 2011a)
[27] (The Eclipse Foundation, 2011f)

Xtext ist ein textuelles Modellierungsframework, welches den Anspruch besitzt, viele Aspekte modellgetriebener Entwicklung auf eine bessere Art zu unterstützen, als die Vorgängertechnologien und stellt aus Sicht der oAW-Entwickler die nächste Evaluationsstufe der oAW-Technologien dar.

Aufgrund der begrenzten Erfahrungen mit Xtext kann ich an dieser Stelle keine Evaluation ausführen.

2.3.3 Weitere EMP-Technologien

Im Eclipse Modelling Project findet sich eine Vielzahl an weiteren Technologien[29], größtenteils Inkubatoren in Alpha und Beta-Qualität, aber auch ausgereiftere Projekte, wie EMF (Core) und die vormaligen oAW-Bestandteile.

Besonders hervorzuheben ist noch das Graphical Modelling Project (GMP)[30] welches mit den Bestandteilen Tooling, Runtime, Notation und Graphiti grafische Modellierung innerhalb der Eclipse-Plattform unterstützt.

2.4 Chancen und Risiken modellgetriebener Entwicklung

Die Chancen modellgetriebener Entwicklung sind unter anderem:

- Skaleneffekte

  Insbesondere durch die Abstraktionsmöglichkeit wiederkehrender Entwicklungen und Mehrfachanwendung entwickelter Code-Generatoren lassen sich große Skaleneffekte erzielen.

  Diese zeigen sich auch bei der Weiterentwicklung und Überarbeitung modellgetrieben entwickelter Software und können dadurch zur Kostensenkung und Produktivitätssteigerung im Softwareentwicklungsprozess beitragen.

- Nachvollziehbarkeit des Prozesses

  Durch die Formalisierung des Softwareentwicklungsprozesses, vom Metamodell über die domänenspezifischen Sprachen und die Code-Templates bis zum lauffähigen Programm, wird eine hohe Transparenz des Softwareentwicklungsprozesses hergestellt.

[28] (The Eclipse Foundation, 2011h)
[29] (The Eclipse Foundation, 2011d)
[30] (The Eclipse Foundation, 2011e)

Diese Transparenz steigert die Wartbarkeit der Software, da viele Aspekte, die bei manuell implementierter Software explizit in eventueller Software-Dokumentation erwähnt werden müssen, bei modellgetriebener Entwicklung implizit in den Modellen und den Transformationen vorhanden sind.

- Vereinfachte Validierung des Programmcodes

  Bei idealer Anwendung modellgetriebener Methoden ist aus einem validen Modell erzeugter Programmcode ebenso valide und muss nicht gesondert überprüft werden.

- Hohe „Griffigkeit“ der Modellierung

  Die Verwendung domänenspezifischer Modellierungssprachen ermöglicht eine erhöhte „Griffigkeit“ der Modellierung da das Modell sich, im Gegensatz zu allgemein-abstrahierenden Modellen wie UML oder BPMN, auf das konkrete Domänenobjekt (z.B. „Typo3-Extension Methode“) bezogen werden kann, und nicht auf allgemein-abstrahierende Objekte (z.B. „Operation“) beschränkt ist.

Neben den Chancen birgt modellgetriebene Entwicklung auch Risiken in sich:

- Verhältnisfindung zwischen Generierung und Implementierung schwierig

  Nicht alle Bestandteile einer Software lassen sich mit vertretbarem Aufwand abstrahieren. Der Balancefindung zwischen generiertem Code und manuell implementierter Software kommt bei modellgetriebener Softwareentwicklung daher eine besondere Rolle zu.

  Eine Erhöhung des Anteils manuell implementierten Codes führt zu erhöhtem Pflege- und Wartungsaufwands und verringert die Vorteile modellgetriebener Entwicklung.

  Eine zu starke Abstraktion eventuell singulärer Softwarebestandteile führt durch den fehlenden Skaleneffekt zu einem schlechterem Aufwand-/Nutzen Verhältnis und verringert durch den hohen Abstraktionsgrad die Verständlichkeit des Entwicklungsprozesses und des Modells.

- Anforderung an die Abstraktionsfähigkeit

  Modellgetriebene Entwicklung stellt hohe Anforderung an die Abstraktionsfähigkeit der im Softwareentwicklungsprozess beteiligten Akteure. Dadurch wird die Auswahl an geeigneten Personen für die Entwicklung und Pflege der Modelle und Programme geringer und es erhöht sich das Risiko bei suboptimaler Anwendung der modellgetriebenen Methoden den Produktivitätszuwachs nicht zu nutzen oder – im ungünstigsten Fall – die Produktivität zu verringern.

- Validierung des Metamodells

  Zwar wird im Idealfall aus einem validen Modell auch valider Code erzeugt, jedoch ist die Validierung des Metamodells – der domänenspezifischen Sprache – eine Herausforderung und im Allgemeinen komplexer, als die Prüfung von Programmcode.

- Erhöhte Komplexität

  Durch Modellgetriebene Entwicklung wird die Komplexität des Softwareentwicklungsprozesses, unter anderem durch die verschiedenen Meta-Ebenen und –Modelle, gesteigert. Falls diese Komplexität nicht gehandhabt werden kann, wird daraus eine Produktivitätsreduzierung entstehen und diese kann wiederum die Vorteile modellgetriebener Entwicklung negieren.

Zusammenfassend lässt sich feststellen, dass die Beurteilung modellgetriebener Methoden differenziert vonstattengehen muss. Daher sollte der Einsatz von Modellgetriebenen Methoden erst nach Prüfung von Aufgabe, Umfang, Mitarbeitern und Umfeld entschieden werden, um ein ausgewogenes Verhältnis zwischen Chancen und Risiken herbeizuführen.

3 Vorstellung Praxisprojekt

Im folgenden Kapitel soll ein Praxisprojekt vorgestellt werden, welches die Vorteile Modellgetriebener Methoden und die Anwendung der Software-Werkzeuge des Eclipse Modeling Projects aufzeigt.

Da die Aufarbeitung dieser Aufgabe voraussichtlich das Thema meiner Masterarbeit sein wird, möchte ich in einigen Punkten nicht zu weit vorgreifen und beschränke mich auf die wesentlichen Aspekte.

3.1 Vorstellung Developer Garden

Der Developer Garden ist ein Projekt der Deutschen Telekom AG, welches Web-Services über offene Schnittstellen (REST/SOAP) bereitstellt, unter anderem:

- Voice Call (Telefongespräche aufbauen)
- Conference Call (Outbound Telefonkonferenzen einleiten)
- Send SMS
- Send MMS Beta
- IP Location

Der Developer Garden stellt neben den Schnittstellen ebenso SDKs für drei Programmierplattformen bereit: Java, .NET und PHP.

Das Besondere an den Developer Garden Services ist die Nutzung ohne individuelle Verträge und die Kostenkontrolle durch einen Pre-Paid Mechanismus.

3.2 Bisheriger Entwicklungsprozess

Der bisherige Entwicklungsprozess zeichnete sich durch manuelle Implementierung der SDKs durch einen externen Dienstleister aus. Die parallele manuelle Pflege von drei SDKs führte zu hohen Entwicklungskosten und die externe Auftragsabwicklung zu aufwändigen Abstimmungsprozessen. Darüber hinaus gab es Termin- und Qualitätsprobleme, welche die Entwicklung der SDKs beeinträchtigten.

Durch die manuelle Implementierung war außerdem nicht gewährleistet, dass die Dokumentation auf dem Developer Garden der Dokumentation in den SDKs entsprach, dass es einheitliche Konventionen (z.B. bei der Benennung von Variablen) gab und dass aktuelle Technologien verwendet wurden.

3.3 Neuer, modellgetriebener Entwicklungsprozess

Um die Herausforderungen des bisherigen Entwicklungsprozesses zu lösen, entschied sich die Deutsche Telekom die SOAP-SDKs durch modellgetrieben entwickelte REST-SDKs zu ersetzen. Diese Ablösung war im Jahr 2010 mein Hauptaufgabengebiet.

Die Eigenentwicklung führte zu einer Kostenreduktion des Entwicklungsprozesses und der Verkürzung der Abstimmungsprozesse.

Durch die angewendeten modellgetriebenen Verfahren konnte außerdem die Qualität der SDKs gesteigert werden. So entspricht die Dokumentation in den SDKs exakt der Beschreibung der SDKs auf dem Developer Garden, da beide mit derselben Methode generiert werden, durch Code-Generierung ergeben sich einheitliche Konventionen und Verbesserungen in den eingesetzten Technologien führen außerdem zu einer Verbesserung des Laufzeitverhaltens und des Ressourcenverbrauchs.

3.4 Konzept des REST-SDK Generations-Framework

Das REST-SDK Generations-Framework verwendet die Technologien des EMP, um die modellgetriebene Entwicklung der SDKs zu unterstützen:

- Ecore – Metamodelle

  Ecore-Metamodelle legen die Modellelemente und ihre strukturellen Features fest. So enthält das Service-Metamodell Elemente, die Web-Services, REST-Ressourcen und HTTP-Parametern entsprechen.

  Das Unit Test – Metamodell enthält Elemente für der den SDKs beigefügten Unit Tests und das Sample – Metamodell Elemente für die Struktur des Beispielcodes der SDKs.

- Developer Garden – Modellinstanz

  Die Developer Garden – Modellinstanz beschreibt die konkreten Ausprägungen in den domänenspezifischen Sprachen, also die Web-Services des Developer Gardens inklusive REST-Ressourcen und HTTP-Parametern, die Unit Tests der Services sowie Beispiele ihrer Verwendung.

- Code-Templates

  Die in Xpand verfassten Code-Templates steuern die Generierung der SDKs in den drei Zielsprachen Java, PHP und C#.

- Xtend Erweiterungen

  Die Xtend Erweiterungen kapseln häufig oder übergreifend verwendete Ausdrücke.

- Build-Definition

  Die Build-Definition nutzt Ant in Verbindung mit der Modelling Workflow Engine (MWE) um den gesamten Prozess der Generierung der SDKs und der Zusammenführung von manuell implementierten und automatisch generierten SDKs zu steuern.

- Docbook-Repository

  Das Docbook-Repository enthält die Dokumentation aller Services, Methoden und Parametern. Aus dem Docbook-Repository wird sowohl die Dokumentation für das Developer Garden Portal, als auch die Source Code Dokumentation generiert.

  Damit wird sichergestellt, dass keine Inhaltlichen Unterschiede zwischen den Ausgabeformaten Source-Code, HTML und PDF existieren.

Insgesamt wurde ein Verhältnis zwischen Generierung und Implementierung von ungefähr 9:1 (bezogen auf den generierten Programmcode) erreicht:

- Generierung (ca. 90%)
  - Serviceklassen
  - Datenklassen
  - Datenklassen-Factories
  - Unit-Testklassen
  - Sample-Klassen
  - Visual Studio – Projektdateien
  - Code-Dokumentation (Deutsch / Englisch)

- Implementierung (ca. 10%)
    - Elternklassen der Service- und Datenklassen
    - Stylesheets der Dokumentation
    - Workflows und Build-Definitionen (Maven, MSBuild, PHPDoc, etc.)

3.5 Größter Vorteil: Skaleneffekte

Die größte Produktivitätssteigerung, die sich aus dem modellgetriebenen Ansatz ergibt, ist der Skaleneffekt durch Abstraktion von Service- und Plattformspezifika.

Dieser Effekt wird in den nachfolgenden zwei Szenarien exemplarisch dargestellt:

3.5.1 Szenario 1: Neuer Service

Die Ergänzung eines neuen Services zum Service-Portfolio des Developer Gardens führte beim bisherigen Entwicklungsprozess zur Notwendigkeit, alle für die Umsetzung des Services notwendigen Artefakte in allen Zielsprachen manuell zu implementieren. Dies brachte mit sich einher, dass der Aufwand für die Ergänzung eines neuen Services zu allen SDKs direkt proportional zu der Anzahl der Zielsprachen ist.

Der Vorteil des modellgetriebenen Ansatzes besteht darin, dass der Aufwand bezüglich der Anzahl an Zielsprachen *gleichbleibend* ist, da pro Service nur eine gleichbleibende Anzahl an Änderungen am Modell durchgeführt werden muss.

3.5.2 Szenario 2: Unterstützung einer weiteren Plattform

Die Erweiterung des SDK – Angebots um eine weitere Plattform konnte im bisherigen Entwicklungsprozess nur durch die Implementierung aller Artefakte für alle Services in der zu ergänzenden Zielplattform erreicht werden.

Der Vorteil des modellgetriebenen Ansatzes ist, dass pro Zielplattform eine gleichbleibende Anzahl an Code-Templates implementiert werden muss und dadurch der Aufwand für die Unterstützung einer weiteren Plattform *gleichbleibend* bezüglich der Anzahl der Services ist.

Zusätzlich kann durch Wiederverwendung der vorhandenen Xtend-Funktionen eine weitere Aufwandsverringerung bei der Implementierung neuer Code-Templates erreicht werden.

4 Fazit

Diese Semesterarbeit hat gezeigt, wie die modellgetriebene Softwareentwicklung im plattformübergreifenden Kontext in der Lage ist, Produktivitätssteigerungen zu erwirken.

Darüber hinaus wurde das Eclipse Modeling Project als ein Beispiel für eine Sammlung an Technologien zur Unterstützung modellgetriebener Entwicklung dargestellt.

Zwar ist der Umfang der Darstellung aufgrund der Form einer Semesterarbeit auf das Wesentliche beschränkt, jedoch hoffe ich ein Grundverständnis vermittelt und eine Anregung gegeben zu haben, sich bei Interesse intensiver mit diesem lohnenden Thema zu beschäftigen.

Literaturverzeichnis

IBM Corporation et al. 2010. org.eclipse.emf.ecore (EMF Javadoc). *Eclipse - The Eclipse Foundation open source community website.* [Online] 23. Juni 2010. [Zitat vom: 13. Januar 2011.] http://download.eclipse.org/modeling/emf/emf/javadoc/2.6.0/org/eclipse/emf/ecore/package-summary.html.

Irawan, Hendy, et al. 2011. Xpand - Eclipsepedia. *Eclipsepedia.* [Online] 9. Dezember 2011. [Zitat vom: 13. Januar 2011.] http://wiki.eclipse.org/Xpand.

itemis AG. 2009. openArchitectureWare.org - Why openArchitectureWare moved to Eclipse.org. *openArchitectureWare.* [Online] 20. September 2009. [Zitat vom: 10. Januar 2011.] http://www.openarchitectureware.org/staticpages/index.php/oaw_eclipse_letter_of_intent.

MID GmbH. 2011. Modellierungsplattform Innovator - MID GmbH. *MID GmbH - The modeling company - MID GmbH.* [Online] 2011. [Zitat vom: 13. Januar 2011.] http://www.mid.de/de/produkte/modellierungsplattform-innovator.html.

Object Management Group. 2010d. OMG Business Modeling and Management Specifications. *Object Management Group.* [Online] 26. August 2010d. [Zitat vom: 13. Januar 2011.] http://www.omg.org/technology/documents/br_pm_spec_catalog.htm#BPMN.

—. 2010c. OMG Modeling and Metadata Specifications. *Object Management Group.* [Online] 10. Mai 2010c. [Zitat vom: 13. Januar 2011.] http://www.omg.org/technology/documents/modeling_spec_catalog.htm#CWM.

—. 2010e. OMG's MetaObject Facility (MOF) Home Page. *OMG.* [Online] 1. Oktober 2010e. [Zitat vom: 10. Januar 2011.] http://www.omg.org/mof/.

—. 2010a. UML. *Object Management Group.* [Online] Mai 2010a. [Zitat vom: 13. Januar 2011.] http://www.omg.org/spec/UML/.

—. 2010b. UML 2.3. *Object Management Group.* [Online] 3. Mai 2010b. [Zitat vom: 13. Januar 2011.] http://www.omg.org/spec/UML/2.3/Infrastructure/PDF/.

—. 2007. XMI. *Object Management Group.* [Online] Dezember 2007. [Zitat vom: 13. Januar 2011.] http://www.omg.org/spec/XMI/.

openArchitectureWare.org. 2009. openArchitectureWare.org - Official openArchitectureWare Homepage. *openArchitectureWare.org.* [Online] 2009. [Zitat vom: 13. Januar 2011.] http://www.openarchitectureware.org/.

Skrypuch, Neil. 2011. Eclipse Modeling - M2T - Home. *Eclipse - The Eclipse Foundation open source community website.* [Online] 2011. [Zitat vom: 13. Januar 2011.] http://www.eclipse.org/modeling/m2t/?project=xpand.

Slawik, Mathias. 2008. *Entwicklung und Anwendung eines Integrationskonzepts für das Typo3 CMS unter Verwendung von Methoden der Modellgetriebenen Softwareentwicklung.* Berlin : s.n., 2008.

Software AG. 2010. ARIS Platform | Business Process Excellence. *Business Process Excellence: Improve Business Process Faster with Software AG.* [Online] 2010. [Zitat vom: 13. Januar 2011.] http://www.softwareag.com/de/products/aris_platform/default.asp.

SPARX Systems. 2010. Enterprise Architect - UML Design Tools and UML CASE tools for software development. *SPARX Systems.* [Online] 2010. [Zitat vom: 10. Januar 2011.] http://www.sparxsystems.com/products/ea/index.html.

Steinberg, Dave. 2008. Fundamentals of the Eclipse Modeling. *eclipse.org.* [Online] 17. März 2008. [Zitat vom: 10. Januar 2011.] http://www.eclipse.org/modeling/emf/docs/presentations/EclipseCon/EclipseCon2008_309T_Fundamentals_of_EMF.pdf.

The Eclipse Foundation. 2011a. Check. *Help - Eclipse SDK.* [Online] 2011a. [Zitat vom: 13. Januar 2011.] http://help.eclipse.org/helios/topic/org.eclipse.xpand.doc/help/Check_language.html.

—. 2011b. Eclipse Modeling - M2T - Home. *Eclipse - The Eclipse Foundation open source community website.* [Online] 2011b. [Zitat vom: 13. Januar 2011.] http://www.eclipse.org/modeling/m2t/?project=jet#jet.

—. 2011c. Eclipse Modeling Project. *Eclipse.* [Online] 2011c. [Zitat vom: 3. Januar 2011.] http://www.eclipse.org/modeling/.

—. 2011d. Eclipse Modeling Project. *Eclipse - The Eclipse Foundation open source community website.* [Online] 2011d. [Zitat vom: 13. Januar 2011.] http://www.eclipse.org/modeling/.

—. 2004a. Eclipse Public License - Version 1.0. *Eclipse - The Eclipse Foundation open source community website.* [Online] Februar 2004a. [Zitat vom: 13. Januar 2011.] http://www.eclipse.org/legal/epl-v10.html.

—. 2004b. EMF.Edit Overview. *Help - Eclipse SDK.* [Online] 1. Juni 2004b. [Zitat vom: 13. Januar 2011.] http://help.eclipse.org/helios/index.jsp?topic=/org.eclipse.emf.doc/references/overview/EMF.Edit.html.

—. 2011e. Graphical Modeling Framework. *Eclipse - The Eclipse Foundation open source community website.* [Online] 2011e. [Zitat vom: 13. Januar 2011.] http://www.eclipse.org/modeling/gmp/.

—. 2011f. Modeling Workflow Engine Reference. *Help - Eclipse SDK.* [Online] 2011f. [Zitat vom: 13. Januar 2011.] http://help.eclipse.org/helios/topic/org.eclipse.emf.mwe.doc/help/index.html.

—. 2011g. Xtend. *Help - Eclipse SDK.* [Online] 2011g. [Zitat vom: 13. Januar 2011.] http://help.eclipse.org/helios/topic/org.eclipse.xpand.doc/help/Xtend_language.html.

—. 2011h. Xtext User Guide. *Help - Eclipse SDK.* [Online] 2011h. [Zitat vom: 13. Januar 2011.] http://help.eclipse.org/helios/topic/org.eclipse.xtext.doc/help/index.html.

The Free Software Foundation. 2010. The Free Software Definition. *GNU Operating System.* [Online] 12. November 2010. [Zitat vom: 10. Januar 2011.] http://www.gnu.org/philosophy/free-sw.html.

Glossar

CWM	Common Warehouse Metamodel
EMF	Eclipse Modeling Framework
EMFT	Eclipse Modeling Framework Technology
EMP	Eclipse Modeling Project
M2M	Model-to-Model
M2T	Model-to-Text
MDD	Model Driven Design, Modellgetriebenes (Software-) Design
MDSE	Model Driven Software Engineering, Modellgetriebene Softwareentwicklung
oAW	openArchitectureWare
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
UML	Unified Modeling Language
XMI	XML Metadata Interchange

Abschließende Erklärung

Ich versichere hiermit, dass ich die vorliegende wissenschaftliche Arbeit selbstständig und ohne fremde Hilfe angefertigt und keine andere als die angegebene Literatur benutzt habe. Alle von anderen Autoren wörtlich übernommene Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnenden Ausführungen meiner Arbeit sind besonders gekennzeichnet. Diese Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Berlin, den 14. Januar 2011