Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminar Ringvorlesung

Julia Pliszka

Modernisierung von Legacy Anwendungen unter Verwendung von MDA-Konzepten

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Julia Pliszka

Modernisierung von Legacy Anwendungen unter Verwendung von MDA-Konzepten

Ausarbeitung im Rahmen der Veranstaltung Seminar und Ringvorlesung
im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Abgegeben am 28. Februar 2009

Inhaltsverzeichnis

1 Einführung **4**

2 Legacy Systeme **5**

3 Vorgehensweisen von Legacy Systemen **7**
3.1 Vorgehensweisen 7
3.2 Bewertung 8

4 Modell Driven Architecture **11**
4.1 Grundlagen 11
4.2 Migration unter Verwendung von MDA 12

5 Fazit und Ausblick **14**

Literaturverzeichnis **16**

1 Einführung

Altsysteme, auch unter dem Namen Legacy Systeme bekannt, sind ein weit verzweigtes Gebiet der Informatik. Jeder einzelne Architekt, Projektleiter, Entwickler usw., der auf einige Jahre Berufserfahrungen zurück blickt, wurde bestimmt bereits mit Legacy Systemen konfrontiert. Die Herausforderung ist die Handhabung der Legacy Systeme. Ziel der Masterarbeit ist es, einen Ansatz zur automatisierten Modernisierung von Legacy Systemen zu finden. Da in der Informatik bereits ein automatisierte Softwareentwicklungskonzept zur Verfügung steht, wird dieses in der Masterarbeit eingesetzt. Hierbei handelt es sich um das MDA-Konzept.

Als erstes wird dem Leser näher gebracht, was genau Legacy Systeme sind, und welche großen Problematiken sie mit sich bringen. Des weiteren werden die möglichen Handhabungen bzw. Vorgehensweisen von Legacy Systemen vorgestellt. Die Verfasserin diskutiert jede einzelne Vorgehensweisen im Detail und präsentiert alle Vor- und Nachteile. Die Debatte wird abgeschlossen durch die Wahl der geeigneten Vorgehensweise für die vorliegende Problemstellung.
Im Anschluss wird das MDA-Konzept vorgestellt. Nach der MDA Einführung folgt eine Bezugsherstellung des MDA-Konzepts auf die Masterthesis, wobei versucht wird, die ausgewählte Modernisierungsvorgehensweise kooperativ zu vereinigen, um den größtmöglichen Nutzen zu erzielen.
Die Ausarbeitung wird abgeschlossen mit der Abgrenzung der Masterthesis von anderen Themenbereichen und einen ausführlichen Ausblick. Der Ausblick beinhaltet eine Übersicht über die Fragestellungen, mit denen sich die Masterarbeit beschäftigen wird. Zudem beschreibt der Ausblick Chancen sowie Risiken, die derzeit erkennbar sind.

2 Legacy Systeme

Wann ist eine Anwendung Legacy? *„Eine Software von so schlechter Qualität,dass sie sich nicht mehr pflegen lässt, unabhängig von der Implementierung. Eine Software, die auf einem proprietären oder überholten System implementiert ist, unabhängig von ihrer Qualität. Eine Software von so schlechter Qualität, dass sie sich nicht mehr pflegen lässt, die auf einem proprietären oder überholten System implementiert ist.“* [Scholz 2007]Dieser Abschnitt definiert in wenigen Worten den Kern einer Altanwendung. An dieser Stelle soll jedoch der Begriff aus verschiedenen Perspektiven betrachtet werden. Aus betriebswirtschaftlicher Sicht wird üblicherweise der Zeit- und somit auch der Geldaspekt in den Vordergrund gestellt. In diesem Zusammenhang wird auf vier wesentliche Standpunkte eingegangen.
Business Value: Ist die Unterstützung von Geschäftsprozessen mit zu viel Zeit und Geld verbunden, so wird von einem Legacy System gesprochen.
Flexibilität: Ebenso handelt es sich um ein Altsystem, wenn anfallende Änderungen an bereits existierenden Funktionalitäten oder Komponenten nur unter überproportionalem Zeit- und Geldeinsatz durchführbar sind.
Entwicklungs-Effizienz: Die Erneuerung oder die Erweiterung des Systems durch die Umsetzunge neuer Funktionalitäten sollte mit angemessenem Zeit und Geldaufwand machbar sein, um nicht den Status Legacy zu erhalten.
Betriebssicherheit: Ein Indiz für ein Altsystem ist, dass nicht sichergestellt werden kann, dass ein System auf mittlere Sicht betriebstauglich sein wird.

Aus technischer Sicht handelt es sich um ein Legacy System, wenn veraltete Technologien die Grundkonstruktion bilden, und Komponenten oder die Infrastruktur geprägt sind durch ein/e überholte/s:

- Datenhaltung
- Prozessmodell
- User Interface
- Middleware
- Hardware-Plattform
- Betriebssystem

- Entwicklungsumgebung.

Aus der Perspektive der Strategie wird das Legacy identifiziert als ein System, welches auf längere Sicht nicht in der Lage sein wird, seine Aufgaben zu erfüllen [Zoufaly 2002].

Was den einen oder anderen Leser in Erstaunen versetzen wird ist, dass auch Systeme in ihrer eigenen Entwicklungsphase zu Legacy Anwendungen mutieren können. Dieses Phänomen tritt in den meisten Fällen bei der Stabilisierungsphase auf, nämlich dann, wenn das System das erste Release ausliefert, und eine Flut von neuen Anforderungen entsteht. In dieser Phase, wie aber auch in jeder andren Entwicklungsphase, existiert ein Interessenkonflikt zwischen Sponsoren, Anwendern und den Entwicklern. Das Hauptinteresse der Sponsoren ist es, das Time-to-Market auf ein Minimum zu beschränken. Der Anwender sieht den Nutzen in einer Vielzahl von neuen Funktionalitäten, die für ihn eingebaut werden sollen. Der Entwickler dagegen zielt auf eine Entwicklung bzw. Weiterentwicklung eines hochqualitativen Softwaresystems ab, welches er lediglich durch entsprechenden Aufwand und klar definierte Anforderungen erreichen kann.

Wie wohl schwer zu verkennen ist, stehen die Interessen der Sponsoren an erster Stelle. An zweiter Stelle folgen die Anwender, weil sie zukünftig das System nutzen werden. Die letzte Position nimmt der Entwickler ein. Bei einem unqualifizierten Management, das sich dieser Hierarchie beugt, wird die Anwendung zu einem Altsystem mutieren. Denn die Vorgabe, das Time-to-Market auf ein Minimum zu reduzieren, zwingt die Entwickler, unsauber zu programmieren, was zu einem unwartbaren Quellcode führt, der nicht genügend getestet wurde. Auch die unaufhaltsame Flut an neuen Anforderungen von Seiten der Nutzer beschleunigt die Mutation zu einem Altsystem. Durch unnötige und übermäßige Neuanforderungen wird das System kompliziert und mit einem unbenutzen Quellcode überfrachtet. Das Überladen des Systems mit Wunscheskapaden von Nutzerseite wird in Fachkreisen auch als „Featuritis“ bezeichnet. Aber auch die Entwickler selbst sind nicht davor gefeit, während der Stabilisierungsphase das System zu gefährden. Besonders in dieser Phase versuchen die Entwickler, alle technischen Kompromisse zu beseitigen, und noch schnell den neusten Stand der Technologien in das System zu integrieren. Dies kann als Technologienüberladungen ebenso schädigend sein. Wie zu sehen ist, handelt es sich bei Legacy Systeme nicht nur um vor Jahren entwickelte Anwendungen, sondern ebenso um Systeme, die aktuell entwickelt werden, und schon erste Anzeichen von Legacy Systemem aufweisen. Ein IT-ler kann an ein paar einfachen Aspekten erkennen, ob sein momentan zu entwickelndes System den Status “legacy“ annimmt.

- Es fehlen schon jetzt fachliche Wissensträger.
- Es existieren viele Codestellen an denen sich keiner herantraut.
- Die Weiterentwicklung wird immer langsamer [Baron 2008].

3 Vorgehensweisen von Legacy Systemen

3.1 Vorgehensweisen

Die o.g. Merkmale eines Altsystems bilden zugleich die Schwachstellen von Legacy Systemen. Aus diesem Grund stellt es eine Herausforderung dar, mit Legacy Systemen umzugehen. Mit dieser Thematik beschäftigen sich IT Fachleute schon seit Jahrzehnten, genau genommen, seitdem die ersten Systeme zu Altsystemen erklärt wurden. In der Zwischenzeit sind eine Reihe von Ansätzen, Methodiken und Technologien zur Handhabung von Altsystemen entstanden. Es haben sich vier Vorgehensweisen über die Jahre etabliert:

Standardsoftware
Der Einsatz von Standardsoftware ist die Wiederverwendung eines Systems, welches für einen bestimmten Bereich in der Industrie erstellt wurde. SAP ist einer der größten Standardsoftwareanbieter, der wiederverwendbare Rahmensysteme für zahlreiche Industriebereiche anbietet. Die Systeme werden als Rahmensysteme bezeichnet, weil sie im Grundsatz anpassungsfähig sind. Jedoch ist ihre Flexibilität eingeschränkt, demzufolge müssen die Benutzer des Systems die standardisierten Geschäftsprozesse der Softwaresysteme befolgen. Die individuellen Workflows der einzelnen Unternehmen werden auf Grund dieser Tatsache durch die SAP Standards ersetzt.

Redevelopment(/Re-Engineering)
Im Deutschen wird ebenso der Begriff Neuentwicklung verwendet. Wie der Name bereits andeutet, handelt es sich bei dieser Variante um die Neuentwicklung des Altsystems ohne dessen Verwendung. Hierbei werden wie bei der Entwicklung eines neuen Systems alle Realisierungsphasen von dem Requirements-Management bis hin zum Testen durchlaufen. In diesem Zusammenhang wird der Begriff Re-Engineering eingeführt. Re-Engineering bedeutet, dass unter Verständnis des Altsystems ein neues System in neuer Form rekonstruiert und anschließend die Neuentwicklung des Systems durchgeführt wird. Der Begriff Redevelopment unterscheidet sich vom Begriff des Re-Engineering, bei dem das Altsystem vorab

analysiert wird [Lenhard 2004]. Die Verfasserin warnt vor irreführenden Begrifflichkeiten, da eine umfangreiche Literaturrecherche ergeben hat, dass die Termini Redevelopment, Re-Engineering und ebenso die im nächsten Abschnitt erläuterte Migration homonym verwendet werden. Jedoch verbergen sich unter diesen Begriffen unterschiedliche Vorgehensweisen.

Wrapping
Die Vorgehensweise des Wrapping ist das Umhüllen (abhängig von der Granularität) von Daten, Methoden, Prozeduren, Programmen und/oder Systemen. Die einzelnen Komponenten erhalten Schnittstellen, über die auf sie zugegriffen werden kann. Der innere Kern bleibt unangetastet.

Migration
Unter dem Begriff Migration verbirgt sich eine Reihe unterschiedlicher Migrationsklassifikationen. Als Beispiel sei zu nennen die Hardware-Migration, die Betriebssystem-Migration, die Funktionale-Migration. Diese Ausarbeitung sowie die Masterarbeit beschränken sich auf die Funktionale-Migration. Der Schwerpunkt der Funktionalen-Migration ist das Abbilden existierender Funktionen auf einem neuen System. Das Grundprinzip der Funktionalen-Migration ist die Wiederverwendung eines existierenden Systems. Im Wesentlichen geht es bei der Funktionalen-Migration um die Analyse des vorhandenen Quellcodes, Modellieren des Zielmodells, Transformation zur Zielplattform und Migration der Zielplattform.
In der Ausarbeitung wird unter dem Begriff Migration die funktionale-Migration verstanden.

3.2 Bewertung

Standardsoftware
Standardsoftware wird in vielen Bereichen bereits häufig verwendet, sie bringt jedoch viele Nachteile mit sich. Bei einer Standardsoftware ist das Paradigma nicht die Fragestellung, wie arbeitet der User derzeit, und wie kann der Nutzer in seinem Arbeitsprozess unterstützt werden, sondern, wie arbeiten die meisten Benutzer, oder wie sollten die meisten Benutzer arbeiten. Die Software basiert auf Standardprozessen, die für einen bestimmten Bereich definiert wurden. Dementsprechend muss sich der Unternehmensablauf an die Software anpassen und nicht umgekehrt. Die Entwicklung der Standardsoftware geht ebenso mit der Zeit, und dementsprechend wird versucht, Standardsoftware flexibel zu gestalten, mit dem Ziel, ebenso die Software an die Unternehmensabläufe anzupassen. Die individuelle Anpassung ist jedoch nur beschränkt realisierbar und mit großem Aufwand verbunden. Zwei weitere Nachteile sind zu beachten. Es entstehen Kosten für die Schulung des Personals, die nicht nur auf die Anwender sondern auch das IT-Personal betreffen. Das sind

jedoch nicht die einzigen versteckten Kosten, mit denen zu rechnen ist. Die Lizenzkosten für die standardisierte Software können auf Jahre hoch gerechnet einen enormen Betrag bilden. Fazit: Es ist sinnvoll, auf Standardsysteme zurückzugreifen, wenn die standardisieren Geschäftsprozesse nicht stark von den eigenen Geschäftsprozessen abweichen, und die Kosten verteilt auf die Verwendungsdauer des Systems geringer als die der Neuentwicklung oder Migration sind.

Redevelopment/Re-Engineering
Diese Vorgehensweise ist mit hohen Kosten und sehr großem Risiko verbunden. Wie o.g., wird das Softwaresystem neu entwickelt, es werden also alle Implementierungsphasen von dem Requirements-Management bis hin zum Testen durchlaufen, welches mit hohen Kosten verbunden ist. Des weiteren ist es aus psychologischer Sicht oft nicht von Seiten des Unternehmens zu vertreten, ein Altsystem, das den Umfang von vielen Millionen Code-Zeilen besitzt, und somit einen enormen Wert aufweist, einfach zu entsorgen. Nicht nur die Kosten, sondern ebenso das hohe Risiko führen dazu, dass diese Vorgehensweise nur in den seltensten Fällen in der Praxis angewandt wird. Fakt ist, dass die Komplexität des Altsystems gegenüber dem Dokumentationsstand unzureichend ist. Eine über mehrere Jahre ausgereifte Applikation kann durch nicht Know-How-Trägern nicht ohne Weiteres realisiert werden. Diese Problematik sollte man sich immer vor Augen führen.

Wrapping
Der Vorteil von Wrapping ist die Wiederverwendung von bereits ausgereifter Komponenten, die sich über Jahre bewährt haben. Da, wie bereits oben erwähnt, das Hauptmerkmal von Altsystemen ein unwartbarer Quellcode ist, erweist es sich lediglich in zwei Fällen immer noch als sinnvoll, auf diese Vorgehensweise zurückzugreifen. Erster Fall: Das Altsystem besitzt tatsächlich einen qualifizierten Quellcode, der nicht zu einem unwartbaren Quellcode mutiert ist. In diesen seltenen Fällen ist das Wrapping eine sehr effektive Vorgehensweise. Der zweite und auch in der Praxis realistische Fall ist folgender: Der Altcode ist bereits soweit ausgereift, dass er bugfrei ist und neue Anforderungen oder ähnliches nicht direkt den Quellcode betreffen, d.h. der Quellcode muss nicht aufbereitet werden. In diesem Zusammenhang kommt das Wrapping in zwei Bereichen erfolgreich zum Einsatz: beim sogenannten Screen scraping und bei der Kommunikationserweiterung des Altsystems. Screen Scarping bedeutet, dass das User Interface also die graphische Oberfläche für den Benutzer des Altsystem neu erstellt wird. Die Funktionalitäten werden bei dem Legacy System vollständig übernommen, es werden lediglich die In- und Outputs über die neue GUI an die Funktionalitäten des Altsystems übergeben. Nicht nur für Screen Scraping sondern auch für Systeme, die lediglich im Bereich Kommunikation erweitert werden müssen, eignet sich das Wrapping hervorragend. Heutzutage beobachtet man häufiger, dass ausgereifte und bewährte Systeme mit Neusystemen arbeiten müssen. In so einem Fall ist es ausreichend,

die Komponenten des Altsystems in z.B. Web Services Hüllen zu verpacken, damit Sie von andren Systemen verwendet werden können. Fazit: Das Wrapping eignet sich sehr gut für spezielle Altsysteme, bzw. wenn das Altsystem um Kommunikationsaspekte oder das User Interface erweitert werden muss. Wobei nicht zu vergessen ist, dass auch in diesem Bereich das Wrapping schnell an seine Grenzen stoßen kann.

Migration
Wenn das Redevelopment zu unakzeptablen Risiken und Kosten führt, die Standardsoftware die ganze Unternehmensstruktur aufwirbeln würde, und das Wrapping untauglich ist, wird auf die Migration zurückgegriffen. Die Migration stellt seit viele Jahren einen Schwerpunkt der Informatik dar, dementsprechend existieren bereits viele Ansätze, Methodiken, Technologien und nicht zu vergessen praktische Erfahrungen, die die heutige Migration stark unterstützen.
Die Migration ist eine hervorragend geeignete Vorgehensweise zur Modernisierung von Altsystemen, jedoch ist zu beachten, dass nicht jedes Altsystem migrationstauglich ist. Ein Altsystem muss bestimmte Voraussetzungen bezüglich seiner Qualität aufweisen, um migriert werden zu können. Die Antwort auf die Frage: Wann ist ein Altsystem hochwertig qualitativ, ist nicht trivial. Es ist nicht ausreichend zu behaupten, ein Altsystem sei migierbarm weil der Quellcode einwandfrei ist. Die Debatte muss sich über mehrere Schichten und Ebenen ausweiten. Es sollten zumindest ebenso die architektonische und die Protokoll- und technologische Ebene ins Visier genommen werden. Erst unter Betrachtung mehrerer Sichtweisen kann ausgesagt werden, ob ein Altsystem migrierbar ist. Im Prinzip kann mit äußerster Vorsicht die Aussage getroffen werden, dass gute modularisierte Altsysteme sich für die Migration eignen und monolithische eher nicht. [Bisbal u. a. 1999, Seite 1-6][Juric u. a. 2000].

Für die automatisierte Modernisierung von Altsystemen eignet sich lediglich die Migrationsmethodik. Da der Ansatz verfolgt wird, auf bestehenden Altsystemen ein Zielsystem automatisiert zu generieren, ist die Basis das bereits existente System. Wie in der Bewertung bereits erwähnt, können nicht alle Altsysteme migriert werden. Aus diesem Grund wird sich die Masterthesis ebenso mit diesen Themenbereich detailliert auseinander setzten müssen. Weitere Ausführungen dazu im Ausblick und in der Masterarbeit.

4 Modell Driven Architecture

Das Ziel einer Migration ist es, die fachlichen Funktionalitäten eines Altsystems auf ein neues System zu transformieren, das den heutigen Ansprüchen genügt. Um die fachlichen Anforderungen von den technischen zu trennen, muss auf einer höheren Abstraktionsebenen gearbeitet werden. Hätte man ein Legacy System auf einer höheren Abstraktionsebene, so gäbe es die Möglichkeit, diese zu einer beliebigen Plattform zu transformieren. Der Ansatz der Modellierung auf einem höheren Abstraktionsniveau ist bereits in der IT vorhanden und verbirgt sich unter dem Namen MDA (Model Driven Architecture).

4.1 Grundlagen

OMG (Object Management Group) spezifizierte 2001 den MDA Standard. Das Grundkonzept von MDA ist die modellgetriebene Softwareentwicklung, welche das Modellieren in den Vordergrund der Softwarerealisierung stellt. Das Prinzip hierbei ist, zuerst das System zu modellieren, um im Anschluss aus dem stabilen Modell den Quellcode generieren zu lassen. Um die technische und fachliche Trennung durchführen zu können, basiert MDA auf drei Modellebenen mit unterschiedlichen Abstraktionsniveaus (auch Viewpoints genannt). Die Abbildung 4.1 skizziert alle drei Ebenen, wobei gilt, je höher die Ebene desto abstrakter.

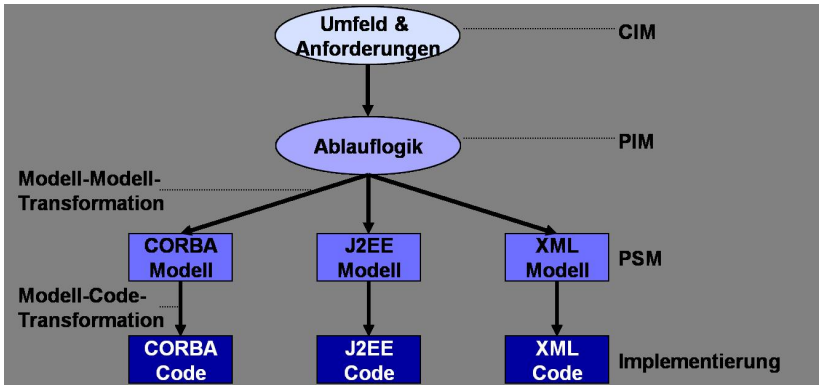

Abbildung 4.1: MDA Modellebenen

CIM Computation Independent Model
Ist die höchste Abstraktionsebene und dient der Erfassung aller Anforderungen und stellt das System aus Sicht der Geschäftsanwendungsfälle dar. Es ist somit unabhängig von der Implementierung des Systems. Das CIM entspricht weitgehend dem Analysemodell und ist in Fachkreisen auch unter dem Namen Domänenmodell oder Fachmodell bekannt.

PIM Platform Independent Model
Die nächste Stufe wird über das PIM (Platform Independent Model) realisiert, das ohne konkrete Eigenschaften über die Plattform das Verhalten des Systems beschreibt. In den PIMs wird das fachliche Wissen (Fachlogik) des Softwaresystems/Anwendung, wie z. B. Geschäftsprozesse oder Fachverfahren, technologieunabhängig erfasst und modelliert. Es entspricht einem Entwurfsmodell ohne Detailinformationen über die verwendete Plattform, bzw. werden implementierungsspezifische Details wie

- Betriebssystem,
- Programmiersprache,
- Datenverwaltungssystem
- etc.

ausgeblendet.

PSM Platform Specific Model
Die letzte Stufe wird mit dem PSM (Platform Specific Model) erreicht,das konkrete Eigenschaften der Zielplattform beinhaltet. Implementierungstechnologien werden definiert, d. h. die technischen Aspekte bezogen auf eine konkrete Plattform. Hierbei handelt es sich um die letzte Modellierungsebene vor der Implementierung, die alle Informationen enthält, die benötigt werden, um den ausführbaren Code zu erzeugen. [Siegel 2005][OMG 2008]

4.2 Migration unter Verwendung von MDA

Der Grundgedanke der Verfasserin ist der, das MDA-Konzept und dessen Vorteile zu verwenden, um ein Altsystem zu migrieren. Das MDA-Konzept in der Praxis anzuwenden, stellt bereits eine große Herausforderung dar, dieses jedoch mit einer Migrationsstrategie zu kombinieren, eine weitere. Wie bereits erwähnt, wird bei dem MDA-Konzept der Quellcode aus einem abstrakten Modell generiert. Bei einem Legacy System existiert leider die abstrakte Modellierung noch nicht. Im Gegenteil wird ein komplizierter, undokumentierter Quellcode

in den meisten Fällen vorliegen. Also muss dementsprechend das MDA-Konzept insoweit erweitert werden, dass aus dem Quellcode des Altsystems ein abstraktes Modell generiert wird und im Anschluss daraus das Zielsystem. Kurz gesagt, aus einem Platform Specific Model (PSM), welches dem alten Quellcode entspricht, wird ein Platform Independent Model (PIM). Durch Erweiterung dieses Modells mit geschäftsspezifischen Beschreibungen und den für das Zielmodell erstellten Transformationsregeln wird ein neues (PSM), welches der Zielarchitektur entspricht, erstellt.
Wie diese kurze Beschreibung vermuten lässt, ist dies ein sehr interessanter und noch wenig erforschter Ansatz, welcher mit vielen Problematiken und Risiken verbunden ist. Details dieser neuen Strategie werden in der Masterarbeit behandelt, jedoch soll diese Ausarbeitung im letzten Kapitel 5 auf entscheide Problematiken hinweisen.

5 Fazit und Ausblick

Die Masterarbeit soll sich im Wesentlichen mit der Modernisierung von Legacy Systemen unter Verwendung von MDA-Konzepten beschäftigen. Da dieses Themengebiet zu umfangreich ist, wird eine Abgrenzung für die Masterthesis festgesetzt. Die Masterarbeit wird sich auf die Generierung eines abstrakten Modells beschränken. Die Basis dafür liefert der Quellcode des Legacy Systems.
Ziel ist es, mit wenig Aufwand automatisiert aus einem vorhandenen System ein vollständiges abstraktes Modell zu generieren. Das abstrakte Modell soll das Fundament zur Generierung der Zielplattform bilden.
Die Verfasserin möchte wie folgt ihre Masterarbeit strukturieren: Die Grundlage der Masterarbeit soll ein Altsystem in einem Unternehmen sein, welches weiterhin in Betrieb ist. Dadurch können die auftretenden Problematiken realistisch betrachtet werden. Eine kleine Testanwendung als Altsystem würde zu verfälschten Ergebnissen führen.
Der erste Schritt hierbei ist die Qualitätsanalyse des Altsystems. In diesem Zusammenhang werden viele Herausforderungen auftreten. Es muss debattiert werden, welche Voraussetzungen ein System erfüllen muss, um den Status qualitativ hochwertig zu erhalten. Die Debatte wird sich auf mehrere Sichtweisen und Ebenen ausweiten müssen, unter anderem auf die Codeebene, die Protokoll- und technologische Ebene sowie die architektonische Ebene. Ebenso muss in Betracht gezogen werden, welche Dokumentationen vorausgesetzt werden müssen, und welche von großen Nutzen für die Überführung in ein abstraktes Modell sind.
Die nächste Herausforderung, die sich stellt, ist die Frage: Wie kann geprüft werden, ob ein System den Bedingungen eines guten Altsystems Genüge leistet? Existieren in diesem Themenbereich bereits Ansätze? Ist lediglich eine händische Analyse denkbar, oder stehen bereits andre Alternativen zur Verfügung?
Anschließend muss die Verfasserin sich mit der gesamten Überführung auseinander setzten. Wie werden die benötigten Informationen aus dem Altcode geparst und repräsentiert? Müssen weitere Informationen hinzugefügt werden ,um ein abstraktes Modell erstellen zu können?In welcher Form soll das abstrakte Modell abgebildet werden? Die Repräsentation des abstrakten Modells muss mit dem MDA-Konzept kompatibel sein; ist es sinnvoll, UML zu nutzen? Es ist zu sehen, dass die Materie der Überführung ein sehr interessantes und komplexes Themengebiet bildet.
Aufbauend auf diesen ganzen Erkenntnissen muss die Verfasserin sich abschließend mit

der Fragestellung befassen, ob es möglich ist, ein abstraktes Modell zu generieren, das MDA kompatibel ist. Wie bereits erwähnt, ist das abstrakte Modell das Fundament für die Generierung des Zielsystems. Aus diesem Grund muss das Modell einen bestimmten Grad an Genauigkeit und Trennschärfe aufweisen. Ebenso ist es wichtig zu wissen, falls Lücken im Modell akzeptabel sind, in welchen Bereichen diese auftreten dürfen, und in welchen Bereichen das Modell auf jeden Fall vollständig sein muss.

Die Masterthesis bildet ein sehr wissenschaftliches und praxisbezogenes Themengebiet. Wie bereits angekündigt, sind viele Hürden zu bewältigen, um ans Ziel zu gelangen. Wird jedoch das Ziel erreicht, oder werden zumindest brauchbare Ansätze erarbeitet, könnte diese Masterarbeit eine Bereicherung für die IT darstellen.

Literaturverzeichnis

Baron 2008
BARON, Pavlo: Legacy Java. 11.Ausgabe. Java Magazin, 2008

Bisbal u. a. 1999
BISBAL, Jesús ; LAWLESS, Deirdre ; WU, Bing ; GRIMSON, Jane: Legacy Information Systems: Issues and Directions. Trinity College Dublin : IEEE, 1999, S. 1–6

Gimnich u. a. 2008
GIMNICH, Rainer ; KAISER, Uwe ; JOCHEN ; WINTER, Andreas: *10 th Workshop Software Reengineering (WSR 2008)*. Bad Honnef : Gesellschaft für Informatik e.V. (GI), 2008. – ISBN 3–88579–220–8

Juric u. a. 2000
JURIC, Matjaz B. ; ROZMAN, Ivan ; HERICKO, Marjan ; DOMAJNKO, Tomaz: Integrating legacy systems in distributed object architecture. University of Maribor : ACM, 2000

Lenhard 2004
LENHARD, Thomas H.: Multi-Discipline-Reengineering / Redevelopment of IT-Infrastructures. ACM, 2004

OMG 2008
OMG (Hrsg.): *OMG Model Driven Architecture*. Version: 2008. `http://www.omg.org/mda/`, Abruf: 19. Februar 2008

Scholz 2007
SCHOLZ, Peter ; GMBH, Soft Service C. (Hrsg.): *Migrieren, integrieren oder neuentwickeln Was tun mit Software-Altlasten (Legacy Systems)?* Version: 2007. `http://www.developer.com/mgmt/article.php/1492531`, Abruf: 18. Februar 2008

Siegel 2005
SIEGEL, Jon: Why use the model driven architecture to design and build distributed applications? ACM, 2005

Zoufaly 2002

ZOUFALY, Federico: Version: 2002. http://www.developer.com/mgmt/article.php/1492531, Abruf: 17. Februar 2008