TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

DIPLOMARBEIT

Zur Erlangung des akademischen Grades Mag.rer.soc.oek

Anforderungsverfolgung in der Modellbasierten Softwareentwicklung

ausgeführt am Institut für

Softwaretechnik und Interaktive Systeme (IFS)

der Technischen Universität Wien

unter der Anleitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Gerald Futschek

durch

Bakk. Nikolaus Marek

Freyenthurmgasse 18/2/8, 1140 Wien

Wien am 16. Juni 2007

Inhaltsverzeichnis:

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

<Ort, Datum> <Eigenhändige Unterschrift>

Danksagung

Ich möchte mich bei Ao. Univ. Prof. Dipl.-Ing. Dr. techn. Gerald Futschek für die Anregungen und Unterstützung bei der Erstellung dieser Arbeit bedanken. Ganz besonderer Dank gilt meinen Eltern Burgi und Werner Marek die mich Zeit meines Lebens nicht nur finanziell, sondern auch moralisch unterstützt haben und somit mein Studium und diese Arbeit ermöglicht haben. Außerdem möchte ich mich bei Christiane Tiwald für die Unterstützung in den letzten Jahren meines Studiums bedanken.

Abbildungen und Tabellen:

Abstract

Modelle haben in Softwareentwicklungsprozessen sehr stark an Einfluss gewonnen und sind aus der Entwicklung von IT Systemen und deren Komponenten nicht mehr wegzudenken. Der Fokus im Rahmen von Entwicklungsprojekten verschiebt sich zunehmenden in die Erstellung von beschreibenden High Level Dokumenten, aus welchen in weiterer Folge automatisiert Code generiert wird. Diese Dokumente beschreiben Anforderungen einerseits durch Text und andererseits durch Modelle. Mittels Transformationen, die gewissen Regeln folgen und welche auf die Modelle angewandt werden, entstehen Beschreibungen mit einem höheren Detaillierungsgrad, aus welchen sich Quellcode genieren lässt. Die Object Management Group (OMG) definierte die Model Driven Architecture (MDA) als einen Standard zur Softwareentwicklung. Die durchgängige Verfolgung von Anforderungen durch einen MDA Entwicklungsprozess bzw. die weiterführede Möglichkeit eine einzelne Anforderung bei einer Vielzahl von Dokumenten mit graphischem und textlichen Inhalt bis in ein Code Files sicher zu stellen, stellt das Kernthema dieser Arbeit dar.

Es werden die Konzepte der MDA und der Anforderungsverfolgung erläutert und in weiterer Folge die theoretischen Ansätze miteinander verknüpft, um einen Modell zur Anforderungsverfolgung zu definieren. Zusätzlich wird auf existierende Möglichkeiten der Verfolgung von Anforderungen in der modellgetriebenen Softwareentwicklung eingegangen. Dabei stehen die System Modellierungs- Sprache SysML und die Unified Modeling Language (UML) im Vordergrund.

Im Weiteren wird auf den Ansatz des Quality Function Deployments (kurz QFD) eingegangen und dieser erläutert. Basierend auf den Erkenntnissen aus den im QFD angewandten Techniken und Methoden, zeigt diese Arbeit anhand eines durchgängigen Beispiels, wie die Tabellen des QFD dazu verwendet werden können, eine durchgängige Verfolgung von Anforderungen in einem modellbasierten Softwareentwicklungsprozess zu gewährleisten.

Neben der Entwicklung dieses Ansatzes wird ein detaillierter Ausblick auf die Entwicklung der MDA gegeben und insbesondere auf die Anforderungsverfolgung eingegangen.

Abstract in English:

In the last decade models have gained more and more influence in the development of software. Nowadays it is really unusual to realise software development projects without the usage of graphical modelling languages. The focus in this area moves more and more towards the design of high level documents, which are used as a baseline for automated code generation. Those documents describe requirements on a low level in a textual form and on a higher level via the syntax of modelling languages. Transformations are used to transfer models from a lower level to a more detailed level and in the end to generate source code. This process of developing software, called the Model Driven Architecture MDA, was standardized by the Object Management Group in the year 2003. The tracing of requirements throughout a MDA development process, especially the tracing of single requirements through a large number of models plus documents to verify the realisation of this requirement is described in this paper.

In the first part the basics of MDA and Requirements Tracing (RT) are explained and theoretically combined. Additionally the actual concepts of requirements tracing in model based software development are explained. The modelling languages SysML (System Modelling Language) and UML (Unified Modelling Language) and their possibilities in the usage of tracing requirements are explained and challenged.

In the latter section the approach of quality function deployment QFD is addressed and explained. A short history is given and the usage in software development projects is defined. Together with the cognition and methods from prior sections an approach to use quality function deployment tables and techniques to trace requirements and to ensure that defined requirements are realised in a model based software project, is developed and presented by an example.

In this document, you will find a detailed outlook on how the MDA will change current software industries and job roles in those industries, next to an outlook in which direction requirements tracing and the techniques used, will develop.

1 Kapitel Einleitung:

1.1 Ausgangssituation:

Nachfolgende Graphik zeigt ein Netzwerk von entstehenden Dokumente in einem Softwarenentwicklungsprozess. Ausgehend von einem Geschäftsprozess werden durch die einzelnen Schritte der Anforderungsanalyse Anforderungen definiert, welche durch Modelle erläutert und in das System eingebettet werden.
Die definierten Anforderungen werden in weiterer Folge verfeinert und in mehreren technologiespezifischen Modellen und Dokumenten festgehalten. Bis zu diesem Zeitpunkt sind die Abhängigkeiten der einzelnen Dokumente zueinander noch recht nachvollziehbar. Wie die Grafik zeigt, wird eine Verfolgung einer einzelnen Anforderung auf den darunter liegenden Ebenen des Entwicklungsprozesses bereits sehr komplex. Viele Dokumente, viele Relationen und Abhängigkeiten, bis hin zur realisierten Klasse, Methoden und den dazu gehörigen Testfällen, erschweren das Verständnis selbst in klar definierten Entwicklungsprozessen. Es ist extreme Komplexität gegeben welche in den meisten Fällen zu Inkonsistenz und Strukturverlust führt, sofern die Verantwortlichen nicht genaueste Richtlinien und Mechanismen verfolgen. Aufgrund dessen ist auch die Kommunikation unter den einzelnen Verantwortlichen, in der Graphik in orange gehalten, extrem wichtig und von großer Bedeutung für Projekte. Ebenso wird ersichtlich dass Testfälle und Dokumentation welche in vielen Fällen erst gegen Ende des Entwicklungsprozess erstellt werden, nur sehr schwer zur Verifikation einzelner Anforderungen herangezogen werden können, da die Abhängigkeiten sehr schwer nachvollziehbar sind. Im nächsten Unterpunkt wird erläutert welche Motivation diese Arbeit in sich birgt und warum aus jener beschriebenen Ausgangssituation ein strukturierter Prozess zur Anforderungsverfolgung in der Softwareentwicklung entstehen soll.

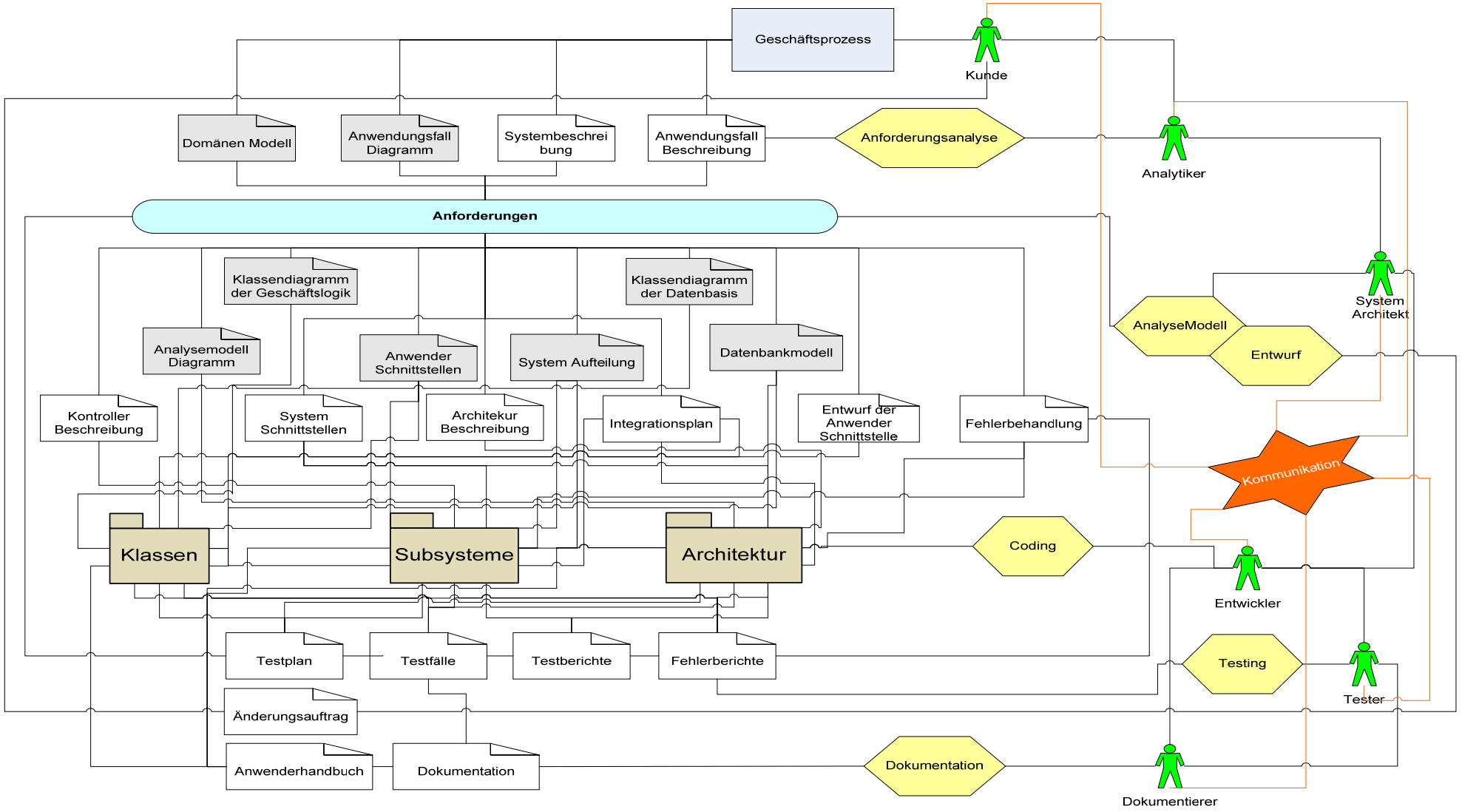


Abbildung 1: Dokumentennetzwerk des SE Prozesses

1.2 Motivation für diese Arbeit

Anforderungsverfolgung dient dazu, Requirements durch den gesamten Entwicklungsprozess zu verfolgen. Dies umfasst die Definition der Anforderungen, deren Realisierung im Quellcode bis hin zur Überprüfung mittels Testfällen. Anforderungsverfolgung begründet sich darin, dass sich Anforderungen ändern können, weitere hinzukommen können oder Konflikte unter verschiedenen Anforderungen entstehen. Das Projektteam muss in diesem Fall oft umfangreiche Änderungen in einer Vielzahl von Artefakten oder in der Implementierung vornehmen. Das Ziel hierbei ist es durchgängige Konsistenz im Projekt zu (er-)halten bzw. diverse Prozess Qualitätsstandards, wie z.B. CMMI, zu erfüllen. Zusammenfassend können folgende Punkte als die Motivation für Anforderungsverfolgung genannt werden:

- Fortschrittsverfolgung
- Kostenoptimierung
- Anforderungsänderungen
- Anforderungskonflikte
- Standards (z.B. CMMI / **S**oftware **P**rocess **I**mproovement)
- Management von Testressourcen

Die nachfolgende Graphik zeigt die Abhängigkeiten von drei der wichtigsten Faktoren in einem SE Projekt : Verändert man eine der Komponenten im Dreieck so wirkt sich das auf die anderen Komponenten aus:

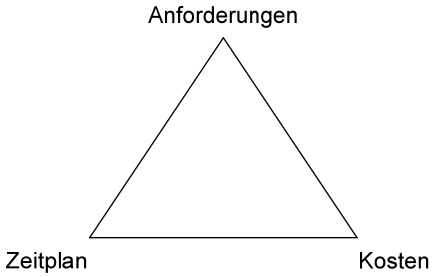


Abbildung 2 Kosten, Zeitplan Anforderungen [Davis 93]

Modelbasierte Architektur, umfasst die Verwendung von Modellen und Codegeneratoren um den Entwicklungsprozess zu beschleunigen, und die Qualität der entwickelten Software zu verbessern. Um dieses Ziel zu erreichen werden Modellierungssprachen wie UML oder SysML eingesetzt.

Der Aufwand für manuelle Anforderungsverfolgung ist sehr hoch. Auf Grund dessen wird sie in einer Vielzahl von Projekten vernachlässigt. Ein Grund dafür ist, dass die zur Verfolgung notwendigen High Level Dokumente nicht oder nur unzureichend erstellt werden und somit die Basis zur Anforderungsverfolgung fehlt.
Auf Grund dessen soll in dieser Arbeit folgende Thesen behandelt werden:

- Die modellgetriebene Entwicklung unterstützt Anforderungsverfolgung auf Grund von umfangreichen Modellen in verschiedenen Abstraktionsgraden. Kann man diese Aussage verifizieren bzw. durch das Zusammenführen der beiden Ansätze Anforderungsverfolgung und modellbasierte Entwicklung ein einfaches Modell für Anforderungsverfolgung definieren?

- Können Anforderungen durchgängig verfolgt werden? Existiert eine Technologie um eine komplette durchgängige Anforderungsverfolgung zu erhalten. Wie werden Abhängigkeiten zwischen Anforderungen veranschaulicht und nachgezogen?

- Kann man den Ansatz des Quality Function Deployments in den Bereich der Anforderungsverfolgung bei modellbasierten Entwicklungen einfließen lassen und wie gut kann man Abhängigkeiten einzelner Modelle und die darin festgehaltenen Kundenanforderungen definieren?

1.3 Zielsetzungen dieser Arbeit:

Auf den nachfolgenden Seiten werde ich die Motivation für die einzelnen Teilbereiche dieser Arbeit erläutern. Das nachfolgende Kapitel „Relevante Literatur“

stellt die einzelnen Themenbereich vor, welche im Kapitel 3 Anforderungsverfolgung in der modellbasierten Softwareentwicklung zusammengeführt werden.

Es soll ein Modell für Anforderungsverfolgung in modellgetriebener Entwicklung entstehen. Das Modell soll graphisch dargestellt werden und eine durchgängige Linie von Geschäftsprozess bis hin zu realisiertem Code bilden. Dabei soll auch auf beteiligte Personen und Dokumente im Entwicklungsprozess eingegangen werden, wobei eine Verknüpfung mit Anforderungen und deren Nachverfolgung durch den Entwicklungsprozess entstehen soll. Die Einzelnen Modellebenen und die darin relevanten Sichten sollen anhand von einzelnen Beispielen erläutert werden. Anforderungsverfolgung und Funktionalität stehen hierbei im Vordergrund.

Neben der analytischen Betrachtung von Anforderungsverfolgung und der Beschreibung dazu verwendeter Technologien in der modellbasierten Softwareentwicklung, wird im Kapitel 3 weiters ein Ansatz vorgestellt, welcher die Methoden des Quality Function Deployments zur Anforderungsverfolgung adaptiert. An einem Fallbeispiel werden die Möglichkeiten erläutert und graphisch dargestellt. Die Intuition dabei ist es Software Quality Function Deployment als einen Prozess in die modellbasierte Entwicklung einzubinden und durch das festhalten von Relationen zwischen Anforderungen, Modellen, Quellcode und Testfällen die Anforderungsverfolgung zu gewährleisten

1.4 Motivation für Model Driven Architecture (MDA)

Traditionelle Software Entwicklung:

Software Entwicklung ist zeitintensiv und benötigt einen hohen personellen Aufwand. Mit jeder neuen Technologie müssen bestehende Softwaresysteme angepasst werden. Systeme werden in den meisten Fällen auf verschiedenen Plattformen realisiert und müssen mit anderen Systemen kommunizieren. Wir stehen also vor dem Problem der sich ewig ändernden Anforderungen an das System und die Software, welche auf dem System implementiert ist.

Das Problem der Produktivität

In momentan aktuellen Softwareentwicklungsprozessen trifft man in den meisten Fällen auf folgende Phasen:

1. Konzeptualisierung und Anforderungsdefinition
2. Analyse und funktionale Beschreibung
3. Design
4. Coding
5. Test
6. Einsatz

In Phase 1 -3 entstehen Dokumente, hauptsächlich Text und Diagramme die die Anforderungen beschreiben. Die Anzahl der hier produzierten Artefakt ist oft eindrucksvoll jedoch verlieren diese Dokumente ab Phase 3, dem Design, sehr schnell ihren Wert. Sobald die Codierungsphase beginnt wird die Abweichung zwischen Dokumenten und dem Code oft merklich größer und Inkonsistenz droht. In den meisten Fällen werden Änderungen nur noch in Codeebenen umgesetzt und nicht, wie es eigentlich passieren sollte, ebenso in den Anforderungsdokumenten geändert. Dies ist auf die Mangel an Zeit zurückzuführen, da das updaten und anpassen von „High Level Dokumenten“ langwierig und zeitraubend ist. Nun stellt

sich die Frage, warum wir uns bei einer MDA Entwicklung die Zeit dafür nehmen diese „High Level Spezifikationen“ zu erstellen.

Im Zusammenhang mit dieser Problematik wurde die Idee des Extreme Programmings (XP) immer beliebter. Dies liegt daran dass der Ansatz des XP darin liegt, dass Code der treibende Faktor der Softwareentwicklung ist und die einzigen wirklich produktiven Phasen im Entwicklungsprozess das Coding und die darauf folgende Testphase ist.

Solange ein und dasselbe Team an einem Softwareprojekt arbeitet ist es auch kein Problem Änderungen oder Erweiterungen einzubeziehen, da genügend Wissen über bereits implementierte Teile oder das gesamte System vorliegt. In den meisten Fällen treten die ersten Probleme nach der Auslieferung einer Software auf. Das Projektteam zerstreut sich und oft sind es neue oder andere Mitarbeiter die vor den Problemen der Wartung, Änderungen, Funktionalitätserweiterung oder Ähnlichem stehen. Bei mittleren bis großen Projekten, wo die Anzahl der Codezeilen immens ist und deren Dokumentation auf Grund oben bereits erwähnter Tatsachen oft als unzureichend bezeichnet werden kann, führt das zu extremen Zeitverzögerungen auf Grund des umfangreichen Reengineerings.
Auf Grund dessen fordert Alistar Cockburn in seiner Arbeit über Extreme Programming [Cockburn 2002] das Vorhandensein von so genannten „Markern“ in Form von Text oder High Level Dokumenten. Ohne diese meint er wäre man im Code vieler Programme buchstäblich verloren.

Das Problem der Portabilität

Die IT Branche hat den Ruf extrem kurze Technologielebenszyklen vorzuweisen. Dies spiegelt sich natürlich ebenso in der Softwareentwicklung wieder. Sich sehr schnell entwickelnde Technologien wie XML, SOAP, .NET, JSP, UML, J2EE, Flash und Web Services haben im letzten Jahrzehnt massiv dazu beigetragen die Standards in der Softwareentwicklung zu verbessern. Andererseits wächst der Druck auf die jeweiligen Firmen diesen Technologien Folge zu leisten. Neben diversen technologiespezifischen Gründen nimmt die betriebswirtschaftliche Sichtweise ebenso einen massiven Stellenwert ein. Wenn Unternehmen mit der neuesten Technologie entwickeln und somit höchsten Anforderungen gerecht werden stehen die Chancen besser lukrativere Aufträge zu akquirieren und somit gewinnbringender auf dem Markt zu agieren. Jedoch ist die Kehrseite, dass mit dieser Strategie die Investitionen in vorhergehende Technologien schneller an Wert verlieren .

Das Problem der Interoperabilität

Interoperabilität ist eine der wichtigsten Anforderungen an heutige Systeme. Eine Vielzahl von vor allem verteilten Systemen greifen in den meisten Fällen auf ein breites Spektrum an verwendeten Technologien zurück um die geforderte Performance zu erfüllen. Der Anwender interagiert oft mit einem HTML, JSP oder ASP basierenden Web Interface, welches wiederum die Informationen an ein dahinter liegendes System weiterleitet, wo die Daten mittels EJB weiterbearbeitet werden und somit wiederum relationale Datenbanken zum Einsatz kommen müssen. Die Existenz von monolithischen Systemen ist nicht mehr berechtigt was die Tatsache mit sich bringt, dass das Adaptieren oder Ändern von Systemen erleichtert wurde. Die einzelnen Module, welche die beste Technologie für die Problemstellung mit sich bringen, werden verwendet, sofern die Interoperabilität gewährleistet ist.

Das Wartungs- und Dokumentationsproblem:

Um es auf den Punkt zu bringen: Dokumentation von Softwaresystemen war und ist immer schon ein Schwachpunkt im Entwicklungsprozess. In den meisten Fällen wird Dokumentation im Nachhinein erledigt und nicht entwicklungsbegleitend, wie es eigentlich passieren sollte. Das Problem hierbei liegt darin, dass die Entwickler die Erstellung von Code als ihre Hauptaufgabe sehen und somit die Erstellung der Dokumentation bestenfalls zweitrangig wird. Darin liegt auch der Ursprung, dass die Qualität der erstellten Dokumentation oft zu wünschen übrig lässt. Mit jeder Änderung im Code muss auch die Dokumentation manuell geändert werden – passiert das nicht ist die Dokumentation nicht vollständig, und somit auch wertlos.

Die Entwickler sind eigentlich dazu angehalten Software zu entwickeln, welche leicht wartbar und auch auf Grund oben angeführter Probleme änderbar ist – dies ist möglich – allerdings nur mit einer vollständigen Dokumentation die auch für Programmierer nachvollziehbar ist, welche nicht Teil des Entwicklungsteams waren. Ein erster Lösungsansatz ist hierbei die Dokumentation automatisch neben dem Codierungsprozess zu generieren. Diverse Sprachen wie z.B. Java ermöglichen diese automatisierte Dokumentation. Jedoch deckt man hierbei nur den Bereich der Low–Level Dokumentation ab. Die eigentliche High – Level Dokumentation, welche Architektur und Systemverteilung erklärt und meist in Modellen und auf Text basierenden Beschreibungen vorliegt, fehlt. Diese ist bei der in heutigen Systemen auftretenden Komplexität ein absolutes Muss.

Quellenangabe f. d. Kapitel [Kleppe 03]

1.5 Motivation für Anforderungsverfolgung , Requirements Tracing (RT):

Die Spezifizierung von Anforderungen (Requirements), ist in Softwareprojekten die wichtigste Tätigkeit bzw. der wichtigste Subprozess auf welchem der gesamte Entwicklungsprozess aufbaut. Die zu Beginn eines Projekts abgesteckten Anforderungen, basierend auf Kundenanforderungen und Systemanforderungen, gilt es umzusetzen und zu implementieren. Fehler bei der Spezifikation sind extrem teuer in der Behebung und oft der wirtschaftliche Untergang eines Entwicklungsprojektes. Um dies zu vermeiden bietet RT die Möglichkeit, dass durch die Allokation aller Produktanforderungen in einer frühen Phase des Entwicklungsprozesses die Kosten durch lange Wartezeiten auf Testergebnisse von Integration und Systemtests, deren Fehler korrigiert werden müssen, bis um den Faktor 30 reduziert werden können.

Projektmanager und das Entwicklungsteam versuchen mittels genauer Spezifikation und der Beobachtung von Abhängigkeiten zwischen einzelnen Artefakten, Konflikte unter Anforderungen frühzeitig zu erkennen, und somit auf mögliche Änderungen rechtzeitig reagieren zu können. Ein nicht unwesentlicher Aspekt, der als Motivation für diese Tätigkeiten angesehen werden kann, ist einerseits eine Kostenoptimierung des Entwicklungsprozesses und andererseits die Erfüllung von Qualitätsstandards wie Beispielsweise CMMI (Capability Maturity Model Integration).
Bei diesem in 5 Ebenen unterteilten Standard ist es um auf Level 3 zu gelangen verpflichtend RT durchzuführen. Ähnliche Verpflichtungen gelten für Softwareunternehmen die Regierungskunden bedienen. Das Amerikanische Verteidigungsministerium (DoD, Department of Defense) fordert in seinem Standard 2167A Anforderungsverfolgung.
[M.C. Paulk, B. Curtis, M.B. Chrissis, C.V. Weber, 93] [Ramesh et al. 95]

Neben diesen Gründen steht auch der Aspekt der Überprüfung von Anforderungen im Vordergrund. Anforderungsverfolgung hilft hierbei die Verifikation von Anforderungen im Design (Modell), Code oder Testfällen und Testprozeduren zu forcieren. Durch diesen Prozess der Nachverfolgung wird sichergestellt, dass nur benötigte Anforderungen auch umgesetzt werden. RT ist keine Option sondern

essentiell um Kundenzufriedenheit zu gewährleisten. Durch RT wird ein dokumentiertes Mittel bereitgestellt, welches dem Kunden bescheinigt, dass alle Anforderungen erfüllt wurden und das Produkt einsatzbereit und fertig ist.
[Ramesh et al. 95]

Auch das Faktum, dass während internen und externen Audits die Projekte besser betreut werden können und transparenter sind , da jederzeit die Daten aus dem RT Prozess abrufbar sind und somit der Projektfortschritt ideal nachvollzogen werden kann, ist von Bedeutung. Somit ist es für Projektmanager einfacher ihr Projekt zu kontrollieren und den Projektfortschritt gegenüber dem Kunden und internen Instanzen zu rechtfertigen. Zur Zeit passiert das in vielen Projekten nur mittels Projektmanagement Tools, welche die anfallenden Kosten in Projekten anführen und somit mögliche Probleme mit z.B. Mitarbeiterauslastung oder Kostenprobleme aufzeigen.

Besonders begründet ist RT wenn es um die Optimierung von Änderungsmanagement (Change Management) geht. Auf Grund der Tatsache, dass jede Anforderung genau nachgezogen wird, ist es wesentlich einfacher Änderungen am System vorzunehmen, da genauestens dokumentiert ist, welche Anforderung, wo und wie implementiert wurde. Die damit entstehende, aktuelle Dokumentation ist ein angenehmer Nebeneffekt, welcher im Rahmen der Wartung als äußerst wichtig anzusehen ist. (siehe Kapitel Motivation für MDA)
Weiters ist die Tatsache dass durch Änderungen der Anforderungen in Folge auch geänderte Tests vollzogen werden müssen, in diesem Zusammenhang von großer Bedeutung. Somit wird das Management von Testressourcen durch den Einsatz von RT maßgeblich unterstützt.
Eine detailliertere Definition von Anforderungsverfolgung wird unter dem Relevante Literatur / Anforderungsmanagement gegeben.

2 Kapitel Relevante Literatur:

2.1 Modellbasierte Softwareentwicklung / Model Driven Architecture (MDA)

2.1.1 Definition von Modellen und Metamodellen

Bevor auf den konkreten Prozess der MDA Entwicklung eingegangen wird, ist es zunächst notwendig einen Rahmen für den Begriff Modell in der Softwareentwicklung festzulegen.

Ein Modell ist eine Abstrahierung der Realität oder eines real existierenden Gegenstandes. Es beschreibt eine Gesamtheit oder einen Teilausschnitt und ist nicht immer eine exakte Abbildung dessen was es darstellt, sondern bietet z.B. die Möglichkeit den Fokus auf verschiedene Teilgebiete zu lenken.
[Kleppe 03]

Modelle bedürfen Modellsprachen in welchen sie geschrieben bzw. modelliert werden. Die Variationen an Modellsprachen sind vielfältig und reichen von normalem Text bis zu UML oder einer Programmiersprache wie Java. Normaler Text kann von Maschinen nicht gelesen werden, weswegen bei MDA ein Rahmen für eine wohl definierte Sprache angegeben ist um das notwendige Regelwerk für automatisierte Modelltransformationen zu spezifizieren.

Definition:
„ Ein Modell ist eine in einer wohl definierten Sprache geschriebene Beschreibung eines (Teil) Systems [Kleppe 03]

Eine wohl definierte Sprache ist eine Sprache welche in einer wohl definierten Form (Syntax) und Inhalt (Semantik) für die automatisierte Interpretation durch einen Computer geeignet ist.“ [Kleppe 03]

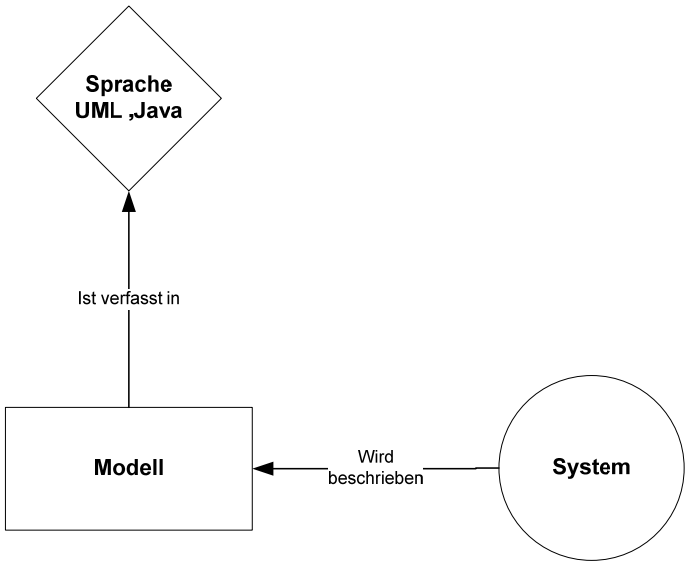


Abbildung 3 Definition von Modell [Kleppe 03]

Nachdem wir nun eine für diesen Bereich eindeutige Definition des Begriffs Modell gegeben haben, können wir einen weiteren wichtigen Aspekt der MDA behandeln. Es geht um Metamodelle.

Die Entstehung von Metamodellen basiert auf der Tatsache, dass Modellierungssprachen in vielen Fällen nicht textbasiert sind. In den meisten Fällen wie z.B. UML ist es eine graphische Syntax die zum Einsatz kommt. Diese graphische Syntax muss jedoch maschinenlesbar sein, denn sonnst wäre die Bedingung einer wohl definierten Sprache nicht erfüllt. Der Mechanismus, welcher zur Beschreibung solcher Sprachen zum Einsatz kommt, wird in der MDA als Metamodellierung bezeichnet.

„Ein Metamodell ist ein Modell welches Informationen über einen oder mehrere Aspekte eines anderen Modells oder eine Menge von Modellen abbildet.“ [Blah 93]

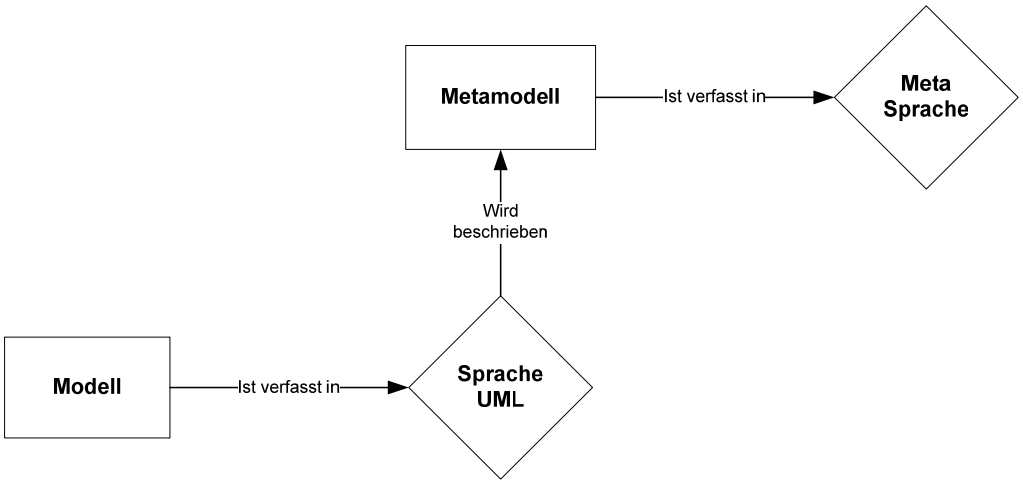


Abbildung 4 Modelle , Sprachen, Metamodelle, Metasprachen

Diese Definition reicht jedoch im Sinne der MDA noch nicht aus, weswegen folgende Definition in diesem Kontext unverzichtbar ist

„*Ein Modell ist – bezogen auf die Metaebene eines anderen Modells dann ein Metamodell, wenn es in einer Abstraktionsbeziehung im Sinne eines höheren Abstraktionsgrades zu diesem steht, Aussagen über das andere Modell (und nicht dessen Gegenstandsbereichs) macht und die Modellelemente des anderen Modells als Instanzen der Metamodellelemente des Metamodells verstanden werden können.*“ [Petratsch, Meinberg 06 S.48]

Jedes Element eines Modells muss durch das Metamodell welches die Sprache die benutzt wird definiert, beschreibbar sein. Am Beispiel UML lässt sich dies veranschaulichen: Es besteht die Möglichkeit Klassen, Attribute, Assoziationen, Zustände, Aktionen usw. zu verwenden um ein System zu beschreiben. Wäre nun z.B. die Metaklasse Interface nicht im Metamodell enthalten, so wäre es nicht möglich ein solches in UML zu definieren.

Der Kreislauf der Metamodellierung und den damit verbundenen Ebenen ist endlos, denn eine Metasprache kann wiederum durch ein Metamodell, welches in einer anderen Metasprache formuliert ist, beschrieben werden. Auf Grund dessen sind vier Ebenen im MDA Prozess von der Open Management Group spezifiziert:

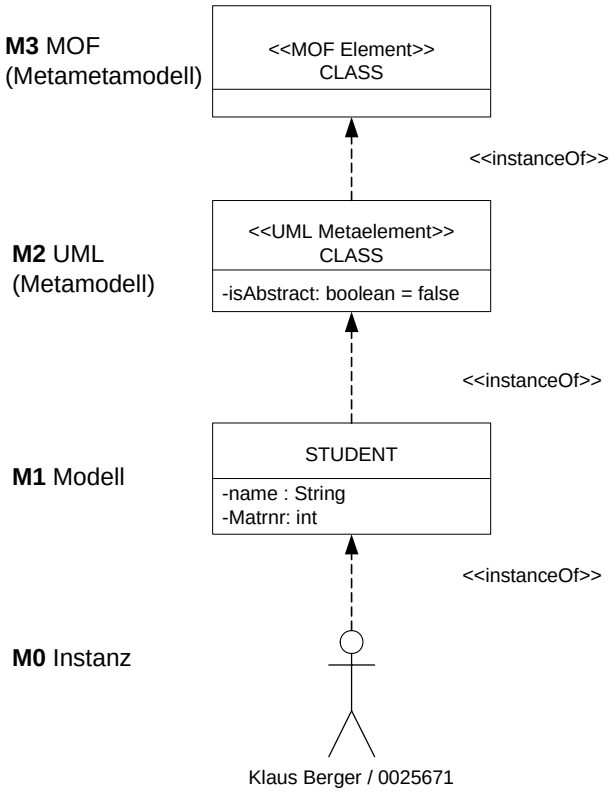


Abbildung 5 Die 4 Modell Ebenen der OMG [adaptiert von [OMG/ MOF]

M0: Instanz

Beschreibt das laufende System in welchem eine Instanz, wie in diesem Fall der Student Klaus Berger existiert, der über die Matrikelnummer 0025671 eindeutig identifizierbar ist.

M1: Modell

Beschreibt das Modell in welchem die Instanzen mit ihren Eigenschaften repräsentiert werden. Dies kann ein UML Modell sein, in welchem die Eigenschaften des Studenten als Matrikelnummer und Name repräsentiert werden. Es erfolgt also eine Kategorisierung der auf MO existierenden Instanzen. Somit ist auch spezifiziert wie die Instanz auf M0 auszusehen hat. Eine Instanz die durch Name und Sozialversicherungsnummer auf M0 beschrieben wird, wäre nicht brauchbar und müsste auf M1 durch eine andere Klasse (z.B. Person) beschrieben werden.

M2: Metamodell

Beschreibt die Klasse welche auf M1 spezifiziert wird. Man spricht ab M2 vom Metamodell da die Sprachen der OMG auf dieser Ebene definiert werden. UML wurde bereits erwähnt. CWM (Common Warehouse Metamodel) wäre eine weitere dieser Sprachen, welche insbesondere für die in Data Warehouse Prozessen auftretenden Daten, Transaktionen u.a. als Beschreibung genutzt wird.

M3: MetaMetamodell

Das Modell der Ebene M2 wird als Metametaebene bezeichnet. Auch hier findet die gleiche Repräsentierung der Elemente als <<instance of>> statt. Die MOF (Meta Object Family) Element Class beinhaltet also z.B. die Definition einer UML Klasse oder eines UML Attributs. Die Ebene M3 definiert das Konzept welches benötigt wird um auf Ebene M2 die Elemente zu definieren.
[Petratsch, Meinberg 06]

2.1.2 Das MDA Grundmodell:

In folgender Graphik wird das Konzept von modelbasierter Entwicklung veranschaulicht:

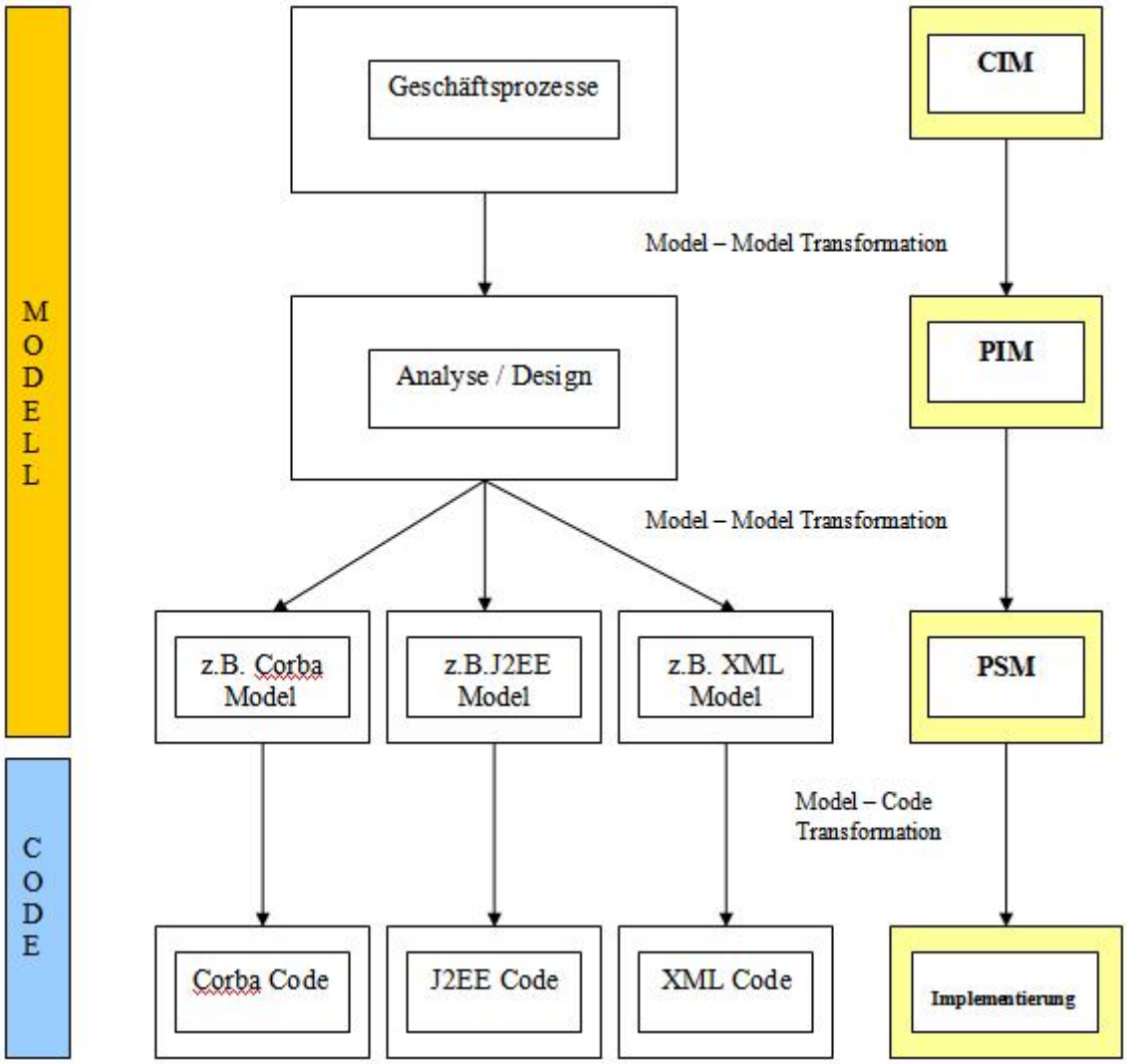
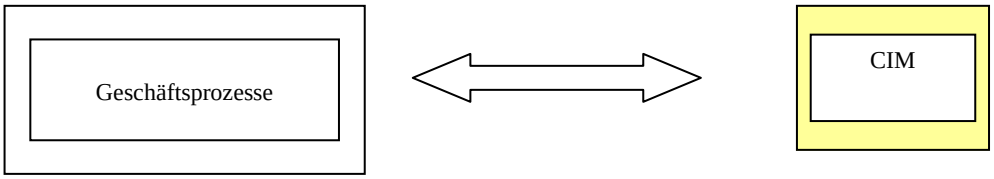


Abbildung 6 MDA Entwicklungsprozess [Adaptiert aus Kleppe 03]

Im nachfolgenden Abschnitt wird auf die einzelnen Ebenen, den dazugehörigen Rollen und Artefakten eingegangen:

Die Geschäftsprozesse und das Computer Independent Model (CIM)



Ziel dieser Ebene ist es Anforderungen zu definieren, welche für die Umsetzung der Geschäftsprozesse in der Software notwendig sind. Die zu automatisierenden bzw. zu unterstützenden Geschäftsprozesse müssen identifiziert und abgebildet werden.

Die Anforderungen für das System werden in einem Computer Independent Model (CIM) festgehalten um die Situation zu beschreiben in welcher das System benutzt wird. Dieses Model wird in manchen Fällen auch als Domänen Modell oder Business Modell bezeichnet und gibt weder Einblick über die verwendeten Technologien, noch darüber wie Daten durch das System verarbeitet oder transformiert werden. Meistens ist dieser Typ von Modell unabhängig von der Implementierung der Software. Es wird hier mehr Wert darauf gelegt, wie das System in das Umfeld für welches es entwickelt wird, eingebettet ist. Sinn und Zweck ist es exakt zu spezifizieren welche Aufgaben zu erfüllen sind. Dies ist nicht nur notwendig um das zu lösende Problem zu verstehen sondern auch um ein gemeinsames Vokabular für die Verwendung in anderen Modellen zu schaffen. In der MDA Spezifikation des CIM sollten die Anwenderanforderungen so abgesteckt werden dass sie auf die darunter liegende Ebene (PIM) verfolgt werden können und dies vice versa möglich ist.

[OMG 01]

Rollen auf dieser Ebene:

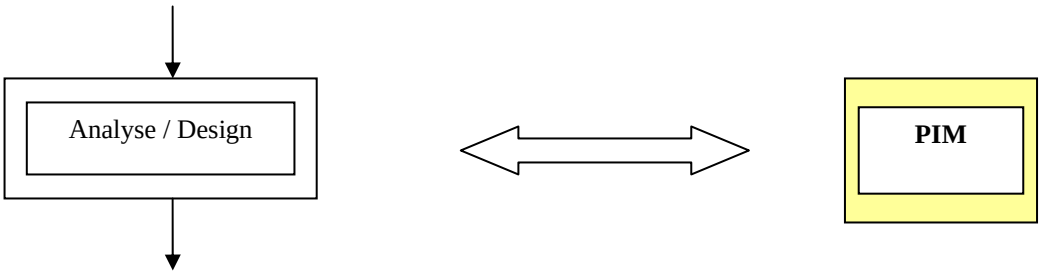
In dieser Ebene des MDA Entwicklungsprozesses sind in erster Linie Manager, Geschäftsprozessinhaber und Analytiker beteiligt. Für den Bereich der Anforderungsverfolgung ist es meines Erachtens notwendig den für das Projekt verantwortlichen Qualitätssicherer hinzuzuziehen um von Anfang an bereits darauf zu achten, dass notwendige Dokumente oder zu erfüllende Standards beachtet und eingehalten werden.
[OMG 02]

Modelle dieser Ebene:

Die angeführten Modelle entstehen im Rahmen des Prozesses und fließen in die ebenenspezifischen Anforderungen ein. Die hier angeführten Modelle sind mögliche Modelle und sollen ein Framework im Rahmen der MDA entwicklung definieren. Sie sind je nach Softwareprojekt in unterschiedlich starken Ausprägungen vorhanden. Auf der Eben des CIM sind das:

- **Business Use Case Modell**
  Beschreibt das Geschäftsumfeld in Form von Use Cases.
- **Enterprise IT Modell**
  Beschreibt die IT Systemlandschaft des Kunden, Anwenders um die Systeme gegebenenfalls einzubetten und mögliche Schnittstellen zu identifizieren.
- **Business Process Modell**
  Geschäftsprozessmodel zur Identifizierung von Prozessen und Akteuren.
- **Organisations- Modell**
  Hierbei liegt der Fokus auf Umsetzung des Projekts aus struktureller Sicht.
- **Business Rules and Constraints**
  Definition von Regeln, und Terminen aus Qualitätssichernder Sicht.
  Modellierung von Anforderungen als Qualitätsaspekt um deren Verfolgung durch die verschiedenen Abstraktionsebenen in der MDA Entwicklung sicherzustellen.

Analyse, Design und das Plattform Independent Modell (PIM):



Auf Basis des CIM wird der modelbasierte Ansatz nun verfeinert. Sinn und Zweck dieses ersten Basismodells ist es Business Regeln und Funktionalität zu erstellen ohne diese durch technische Details oder Technologieaspekte zu verzerren. Die fehlenden technologischen Komponenten auf der ersten Ebene eines Plattform Independent Model (PIM) ist in diesem Sinne wünschenswert, da sie für Anwender oder Entwickler eine Erleichterung in Sachen Verständnis, Überblick und Vollständigkeit der Funktionalität eines zu entwickelnden Systems bietet. Ein weiterer Grund für Technologieverzicht in diesem Modell stellt die Tatsache dar, dass das Modell seinen Wert für Jahre behält und nicht auf Erneuerungen im sich schnell wandelnden technologischen Bereich eingegangen werden muss, sondern lediglich eine Anpassung an geschäftsspezifische Veränderungen durchgeführt werden muss. Diese Aussage ist natürlich nur dann gültig wenn sich der Geschäftsprozess nicht verändert.

PIMs können verschiedene Ebene vorweisen. In der Verfeinerung des PIM Basismodells können erste technische Details auftreten. Plattformspezifische Details werden jedoch weiterhin vermieden. Konzepte wie z.B. Persistenz, Sicherheitsstufen oder Konfigurationsinformationen welche möglicherweise bereits in dieser Ebene einfließen können, dienen in den meisten Fällen dazu dass das Modell spezifizierter wird, um es leichter in die nächste Ebene zu transformieren. Bei der Umsetzung und Entwicklung eines PIMs kommt in den meisten Fällen bereits eine Modellierungssprache zum Einsatz. Im Großteil der Fälle wird UML (Unified Modelling Language) bzw. UML2.0 eingesetzt.

[OMG/ Developing in OMG's Model-Driven Architecture]

Rollen auf dieser Ebene:

Neben den Rollen aus der Ebene des CIM kommen hier Systemarchitekten hinzu welche in der Lage sind von Maschinen lesbare UML Modelle zu konzipieren. Dabei handelt es sich in den meisten Fällen um Aktivitätsdiagramme und Use Cases. Die Dokumente basieren auf den in der darunter liegenden Ebene definierten Geschäftsprozessen und Anwenderanforderungen. Die Erstellung dieser Modelle ist auf Grund der meist komplexen Gegebenheiten nicht automatisierbar und dient in erster Linie der Kommunikation und Visualisierung der Anforderungen.

[OMG 02]

Mögliche Modelle dieser Ebene:

- **System Use Case Modell**
  Systembeschreibung in Use Case Form. Stellt eine Systemübersicht mit Anwendern und Akteuren dar um Funktionalität und Anwendung zu veranschaulichen.

- **System Behaviour Model**
  Systemverhaltensmodell welches die Anforderungen an das dynamische Verhalten eines Systems spezifiziert.

- **Data/Object/Analysis Model**
  Ein erstes Datenmodell um die zu verarbeitenden Daten zu analysieren.

- **Preliminary Deployment Model**
  Erstes Vorläufiges Einsatzmodell für das System.

- **Non functional Requirements Model**
  Spezifizierung nicht funktionaler Anforderungen in einem Model. Insbesondere aus Sicht der Anforderungsverfolgung ist es wichtig ein explizites Anforderungsmodel funktionaler als wie auch nicht funktionaler Anforderungen zu erstellen.

Das Plattformspezifische Modell und die Implementierung:

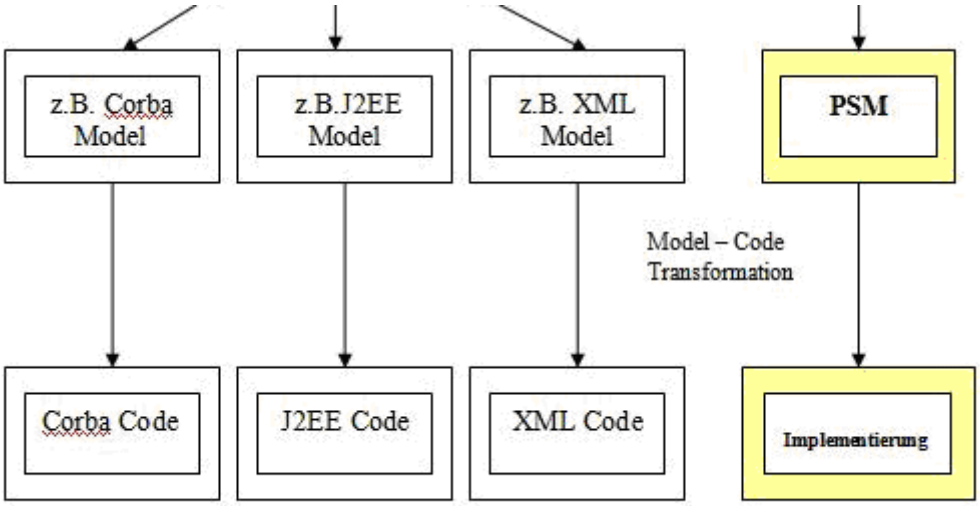


Abbildung 7 PSM / Implementierung

Die auf der Ebene des CIM und des PIM definierten Anforderungen und Systemeigenschaften werden auf der PSM Ebene um technologiespezifische Details erweitert. Dadurch kommt es zu einer Aufteilung eines Modells in plattformkonforme Einzelmodelle welche die zur Umsetzung notwendigen technischen Details definieren. Es werden Subsysteme definiert und deren Anforderungen spezifiziert.

Um das plattformunabhängige Modell in ein plattformspezifischen zu transformieren muss eine Entscheidung über die verwendeten Technologien oder Plattformen getroffen werden. Es ist bei heutigen Anwendungen in der Regel möglich bzw. üblich, verschiedene Applikationen auf der dafür geeigneten Plattform zu realisieren und diese dann in einem System zu vereinen. Falls es keine Notwendigkeit für die Anwendung unterschiedlicher Technologien gibt, kann das plattformspezifische Modell auch nur auf einer Plattform realisiert werden.

Die einzelnen PSM Modelle dienen als Spezifikation für die Modell- Code Transformation. Hierbei kommen Codegeneratoren bzw. durchgängige MDA Tools zum Einsatz welche aus den in den Modellen herausgearbeiteten Spezifikationen Quellcode erzeugen.

Rollen auf diesen Ebenen:

In dieser Phase der MDA kommen Softwaredesigner, Softwareentwickler und Tester zum Einsatz, die gemeinsam mit den Rollen der vorhergehenden Ebenen die PSMs bestimmen.
Die Rolle der Entwickler hat sich durch die modelbasierte Entwicklung insofern verändert dass durch automatische Codegenerierung wie das bei MDA der Fall ist, ihre Hauptaufgaben darin bestehen die notwendigen Abstraktionsgrade für die Modellierungen zu definieren, umzusetzen und weiters generierten Code zu verfeinern bzw. nicht generierbaren Code zu erstellen.
Dies bietet insofern die Möglichkeit die Expertenrolle dieser Mitarbeiter herauszuheben da sie nicht mit der Basis der Codegenerierung beschäftigt sind sondern sich spezifischen Problemlösungen widmen können welche oft zu Projektverzögerungen führen können.
[OMG 02]

Modelle des PSM:

- **Service & Interface Modell**
  Plattform- bzw. Technologie – spezifisches Service und Schnittstellenmodell
- **Architecture Behavior Modell**
  Modell zur Beschreibung des gesamten Softwarearchitekturverhaltens – Die Frage wie interagieren Subsysteme im System und wie verhält sich das gesamte System auf Grund der Einzelsysteme soll mittels eines solchen Modells beantwortet werden.
- **Architecture Modell**
  Graphische Veranschaulichung der Softwarearchitektur.
- **Conceptual Deployment Modell**
  Konzeptuelles Einsatzmodell des Systems.Quality Architecure Modell
  Qualitätsmanagementmodell zur konkreten Festsetzung wie QM in der Anwendung der verwendeten Softwarearchitektur zum Einsatz kommt.

Modelle der Implementation:

- **Component Interface Model**
  Schnittstellenmodell der einzelnen Komponenten

- **Component Behavior Model**
  Verhaltensmodell der einzelnen Komponenten

- **ComponentModel / Physical Behavior**
  Komponenten Model und physisches Komponentenverhaltensmodell

- **Deployment Model**
  Endgültiges Einsatzmodell

- **Test Model**
  Test Modell mit Testablauf und modellierten Testfällen

Modelltransformationen bei MDA

Wie bereits aus der ersten Graphik hervorgeht, werden im modellbasierten Entwicklungsprozess die ebenespezifischen Modelle verschiedenen Transformationen unterzogen. Im Allgemeinen kann eine Modelltransformation als ein Prozess der Umwandlung von einem Quellmodell in ein Zielmodell beschrieben werden. Beide, Quellmodell und Zielmodell beschreiben ein und das selbe System. Die Transformation beschreibt einen Prozess der Erweiterung. Dem Quellmodell werden Informationen hinzugefügt wodurch aber die Semantik nicht verloren gehen darf.

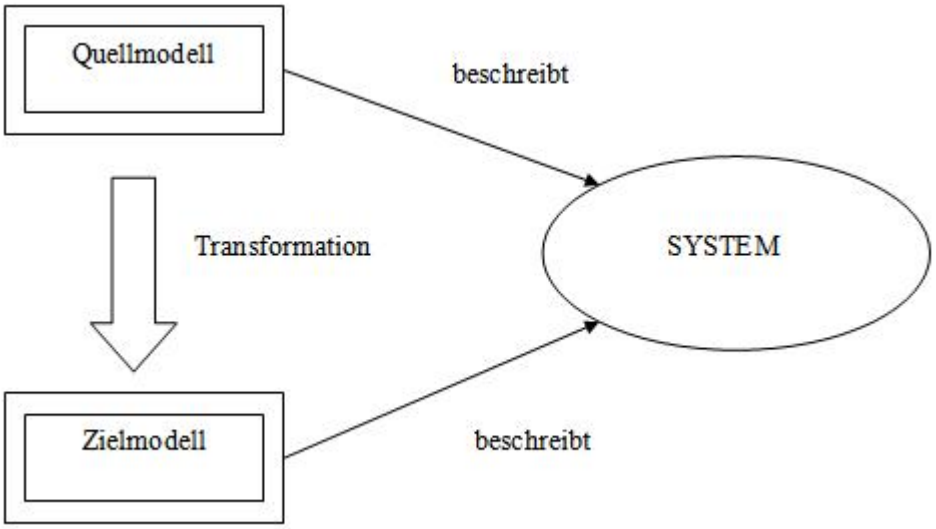


Abbildung 8 Modelltransformation [Kleppe 03]

Im Falle der MDA handelt es sich um einen mehrstufigen Transformationsprozess welcher von einer sehr abstrakten Ebene (CIM) ausgehend durch mehrere Transformationen in eine sehr detaillierte Sicht (PSM) übergeführt wird.

Eigentlich geht der Begriff der Modelltransformation bereits aus der vorhergehenden Erläuterung der einzelnen MDA Modelle hervor jedoch zusammenfassend sei an dieser Stelle nochmals eine Definition gegeben*:*

„Eine MDA Transformation überführt ein plattformunabhängiges Modell in ein plattformabhängiges Modell oder in eine plattformabhängige Implementierung.“
[Petratsch, Meinberg 06]

Die heikle Aufgabe der Transformation wird von einem so genannten Generator übernommen welcher über Transformationsregeln, Profile, Markierungen oder Metamodelle konfiguriert wird. Ein reines Plattformmodell dient hierbei oft als Quellartefakt für die Überleitung des PIM in ein PSM.

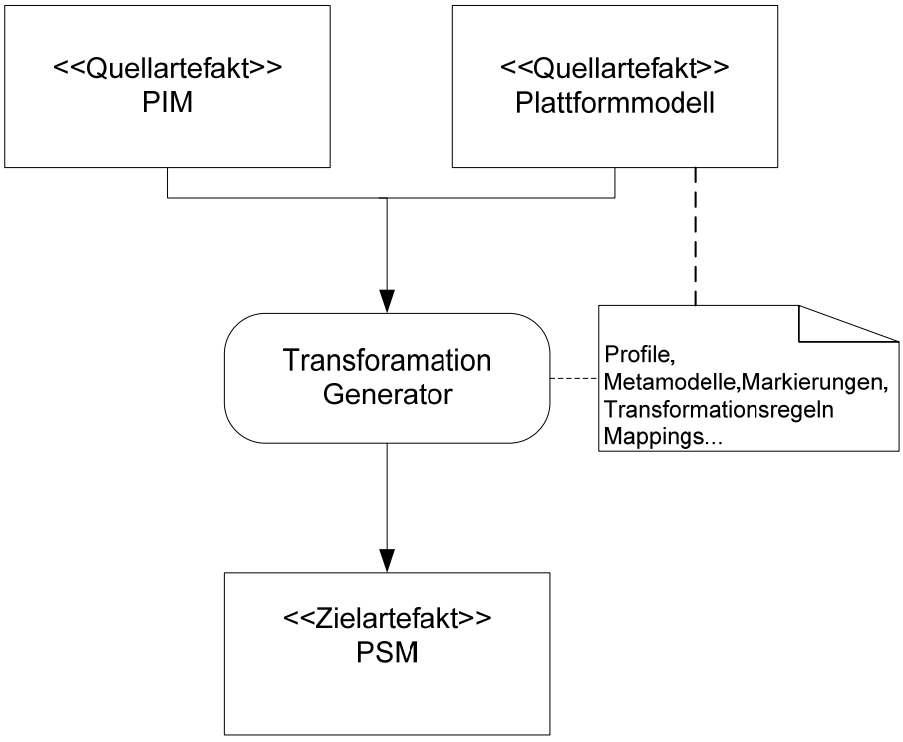


Abbildung 9 Darstellung einer Transformation [Petratsch Mainberg 06]

Arten von Modelltransformationen bei MDA:

Generell können wir bei der MDA Entwicklung zwei Transformationsarten unterscheiden:

- Modell – Modell Transformation
- Modell – Code Transformation

Das MDA Framework bietet diverse Methoden zur Modelltransformation:

Model Type Mapping:

Diese Methode definiert Regeln welche Typen von PIM auf Typen im PSM abgebildet werden. Weiters werden in Form von Instanzwerten Bedingungen definiert, die im PIM gefunden werden müssen.

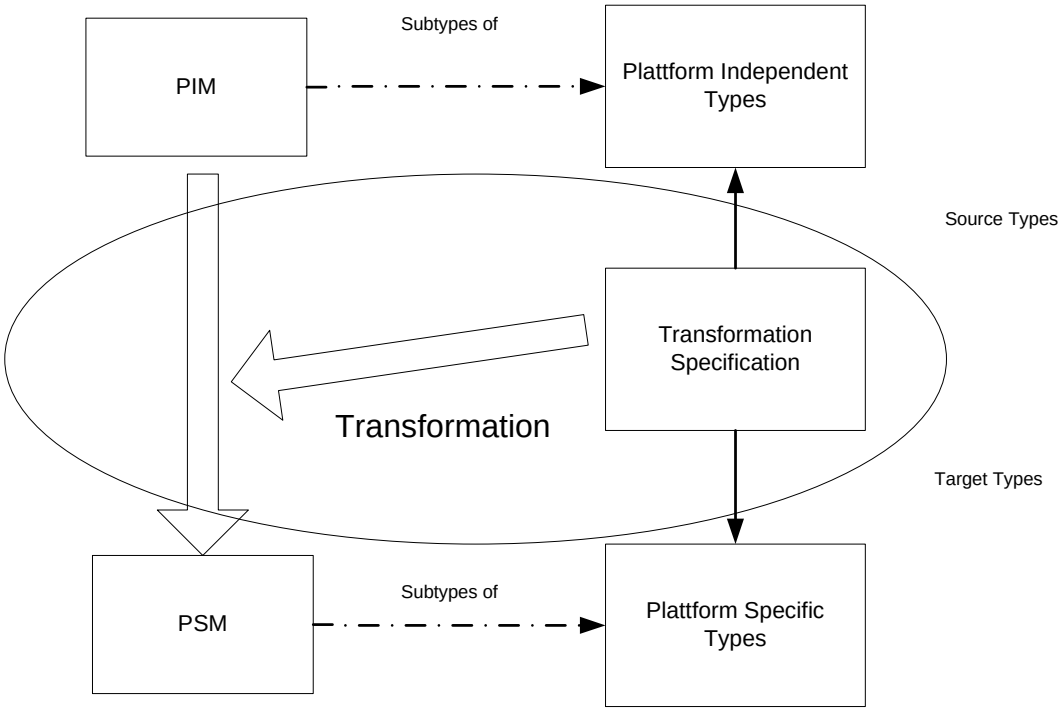


Abbildung 10 Modell Type Mapping [Birchmeier 04]

Model Instance Mapping

Diese Methode ermöglicht es für einzelne Elemente im PIM Definitionen die Transformation betreffen zu definieren. In einem zusätzlichen Marking Model werden die Elemente mit dieser Definition gekennzeichnet. Hierbei ist besonders Augenmerk darauf gelegt dass die in einem Plattform Modell definierten Anforderungen, zum Ausdruck gebracht werden.

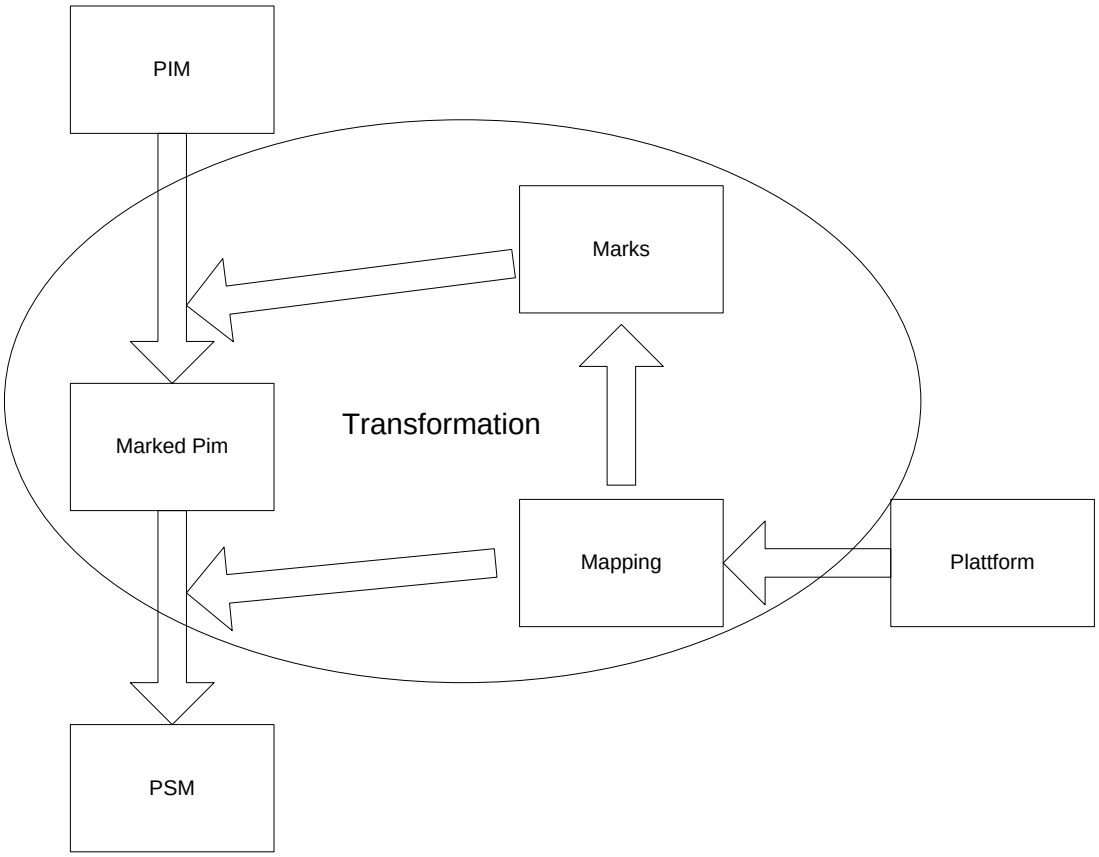


Abbildung 11 Modell Instant Mapping
[Birchmeier 04]

Für den Fall, dass wiederkehrende Anordnungen von Modellelementen im Quellmodell auftreten gibt es die Möglichkeit, so genannte musterbasierte Abbildungen einzusetzen. Diese sind auch unter der Bezeichnung Pattern Mapping geläufig.

In vielen Fällen der MDA Entwicklung wird jedoch eine Kombination von Model Type Mapping und Model Instance Mapping angewandt. Diese Konzepte sind auch in den gängigen Tools zur MDA Entwicklung eingebettet.

Quellen:
[Birchmeier 04] [EWMT 05]

2.2 Anforderungsverfolgung / Requirements Tracing (RT):

Softwareentwicklungsprozesse beinhalten oft eine Vielzahl von komplexen Anforderungen. Diese Anforderungen sind in den seltensten Fällen von Beginn der Entwicklung bis zum fertigen Endprodukt statisch nachvollziehbar, sondern ändern sich im Laufe des Prozesses. Auf Grund dessen, sind die Rollen des Projektmanagements dafür verantwortlich diese Anforderungen zu verwalten. Laut [Maciasek 01] beschäftigt sich das Anforderungsmanagement mit drei großen Kernthmen:

1. Identifizierung, Klassifizierung, Organisation und Dokumentation von Anforderungen
2. Anforderungsänderungen (Management von Prozessen die auf Änderungen der Anforderungen, deren Verhandlung, Validierung und Dokumentation spezialisiert sind.)
3. Anforderungsverfolgung

Um in weiterer Folge auf den Prozess der Anfoderungsverfolgung weiter eingehen zu können, sollen in den folgenden Abschnitten die Grundbegriffe bezüglich dieser Thematik erläutert werden.

Definition von Anforderung:

Der Begriff Anforderung ist den meisten Menschen intuitiv ein Begriff. Das IEEE (Institute of Electrical and Electronics Engineers) veröffentlichte folgende oft zitierte Definition:

"A requirement is (1) a condition or capability needed by a user to solve a problem or achieve an objective; (2)a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents; (3) documented representation of a condition or capability as in (1) or (2)"
[IEEE 610.12 1990]

Eine Anforderung ist (1) eine Kondition oder Fähigkeit die von einem Benutzer benötigt wird um ein Problem zu lösen oder ein Ziel zu erreichen. (2) Eine Bedingung oder Fähigkeit welche von einem System oder einer Systemkomponente erfüllt werden muss um einem Vertrag, einem Standard, einer Spezifikation, oder anderen formal auferlegten Dokumenten gerecht zu werden. (3) Die dokumentierte Repräsentation einer Bedingung oder Fähigkeit wie in (1) oder (2).
Die Anforderungen die an ein System von dessen Stakeholdern gestellt werden sollten eine eindeutige, unmissverständliche Beschreibung dessen sein, zu was dieses System fähig ist und wie dies realisiert wird.

Arten von Anforderungen

Grundsätzlich kann man Anforderungen in funktionelle und nicht funktionelle Anforderungen unterteilen. Stakeholder wie User, Konsumenten Analysten, Entwickler und Manager spezifizieren Anforderungen als funktionell oder nicht funktionell abhängig von dem gewünschten Detailierungsgrad. Je weiter sich die Spezifikation entwickelt desto wahrscheinlicher ist es, dass nicht funktionelle Anforderungen zu funktionellen werden. So kann z.B. eine Anforderung für Ausfallssicherheit sich zu einer Funktion entwickeln welche Fehler in einer detailierteren Form liefert.

2.2.1 Funktionelle Anforderungen:

Mittels der Definition von funktionellen Anforderungen kann zu Beginn des Softwareentwicklungsprozess folgende Frage beantwortet werden: Was wird diese Software machen? Auch hier gibt IEEE eine eindeutige Definition:

„*They define the functions that a software must perform. They describe operations the software or one of it´s components performs on inputs to produce outputs.*“ [IEEE 610.12 1990]

Sie definieren Funktionen die eine Software durchführen muss. Sie beschreiben Operationen die die Software oder eine ihrer Komponenten durchführt um aus Inputdaten Output zu generieren.
Es gibt verschiedenen Blickwinkel unter welchen funktionelle Anforderungen gesehen werden können:

Datensicht:

Im Rahmen dieser Sichtweise liegt der Fokus einerseits auf der Datenstruktur in der Domäne, in welcher die Applikation zum Einsatz kommt, und andererseits in der Verknüpfung dieser mit Regeln und Ableitungen die die erlaubten Zustände dieser Struktur festlegen. Die zentralen Begriffe dieser Sichtweise sind Relationen, Attribute und deren Werte die zusammen die Eigenschaften einer Entität beschreiben. Im Zusammenhang mit der Datensicht werden oft Entity Realtionship Modelle und Konzeptuelle Graphen erstellt.

Funktionelle Sicht:

Die Applikation die entwickelt werden soll wird hier als ein Netzwerk aus Prozessen dargestellt. Damit soll veranschaulicht werden wie die Inputdaten in Outputdaten transformiert werden. Hierbei spielt auch die Verwendung von Speichern eine nicht unwesentliche Rolle. Der Grund dafür liegt darin, dass die zu bearbeitenden Daten, Zwischenergebnisse und transformierten Daten abgelegt werden müssen. Somit

werden auch die Datenflüsse veranschaulicht welche die einzelnen Prozesse in diesem Netzwerk durchlaufen. Zusätzlich kommen bei dieser Ansicht auch noch Entitäten hinzu die zwar nicht im System vorhanden sind aber von außen mit ihm interagieren.

Verhaltenssicht:

Im Rahmen dieser Ansicht geht es darum verschiedene Zustände welche die Applikation einnehmen kann festzuhalten. Dazu gehören auch die Übergänge von einem Zustand in den nächsten und die damit verbundenen Auslösepunkte für den Zustandswechsel. Um dies zu verdeutlichen werden Verhaltensmodelle in den meisten Fällen in Form von Petri Netze oder Zustandsdiagrammen festgehalten.
[Hofmann]

2.2.2 Nicht funktionelle Anforderungen

Nicht funktionelle Anforderungen finden ihren Ursprung in Budgetbeschränkungen, standardisierten Prozessen oder externe Faktoren wie Sicherheitsverordnungen und Qualitätsmerkmale welche nicht diesen Typ Anforderungen fordern.

[Davis 93 / Wieringa 96]

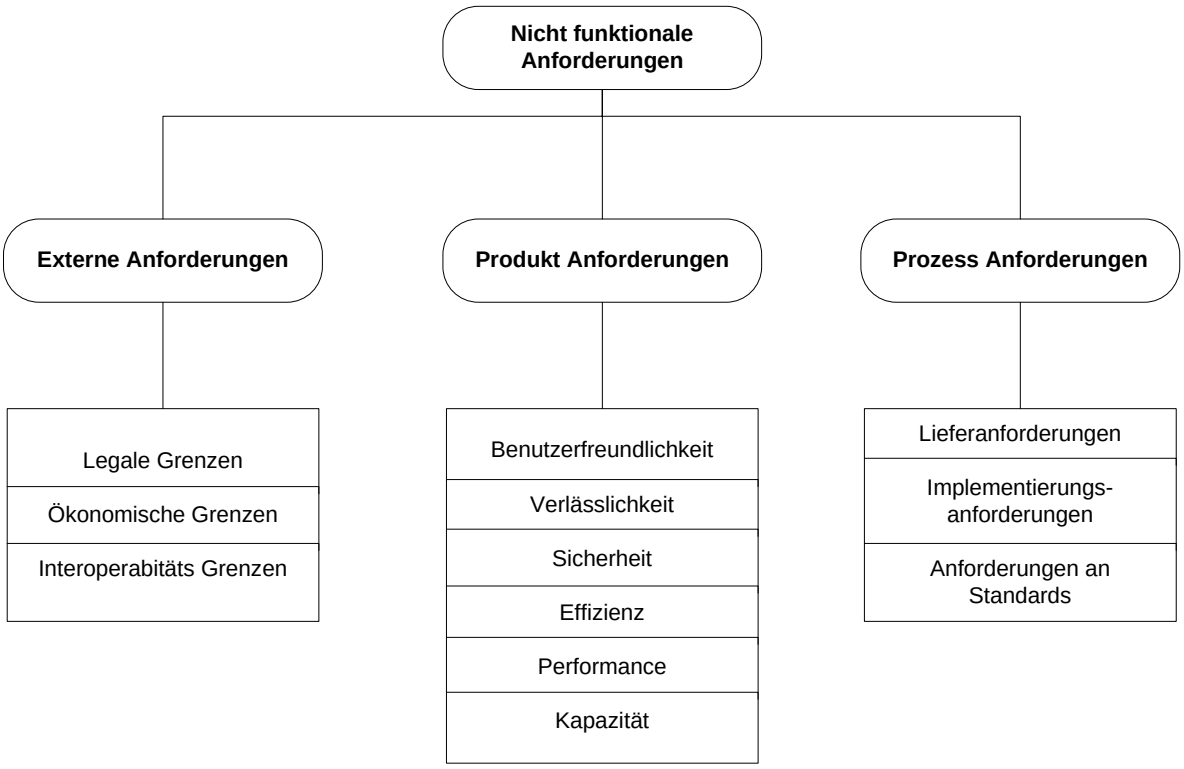


Abbildung 12 nicht funktionelle Anforderungen

Nach der in Abb 11 dargestellten Klassifizierung von Kotonya / Sommerville aus dem Jahre 1998 lassen sich nicht funktionelle Anforderungen in drei Subkategorien unterteilen:

Externe Anforderungen:

Mit Hilfe dieser sollen Grenzen für das zu entwickelnde Produkt erstellt werden, und der Prozess in einen Rahmen gebracht werden. Diese Grenzen entwickeln sich aus der Applikationsdomäne, den organisatorischen Überlegungen und gegebenenfalls aus Regulationen welche von Staat oder Industrie vorgegeben werden. Damit können zum Beispiel standardisierte Prozesse im Gesundheitswesen gemeint sein deren Kennzahlen und Prozeduren in Applikationen, welche auf diesem Sektor agieren und Prozesse automatisieren sollen, als externe nicht funktionelle Anforderungen berücksichtigt werden müssen.
[Kotonya / Sommerville]

Produktanforderungen:

Bei dieser Art von Anforderungen wird die charakteristische Verhaltensstruktur der Software festgehalten. Ein Beispiel wäre eine E – Mail Client der nach klicken des Users auf den Send Button innerhalb einer spezifizierten Zeit eine Verbindung zum Postausgangsserver aufbauen muss. Der IEEE Standard 830-1998 bietet eine Umfangreiche Liste von Produktanforderungen die unter den Gesichtspunkten Performance, Sicherheit, Verfügbarkeit und Benutzerfreundlichkeit definiert sind. Somit kann es passieren dass die Antwortzeit des Postausgangsservers bereits auf Grund von Sicherheitsstandards genormt ist und somit als spezifiziert anzusehen ist.

Prozessanforderungen:

Diese Anforderungen beziehen sich nicht nur auf die zu entwickelnde Applikation selbst, sondern ebenso auf die Organisationsstruktur im Softwarenentwicklungsprozess. Hierbei geht es also auch um die Strukturierung der Rollen unter den Mitarbeitern und den Work Flow zwischen diesen. Ebenso werden hier Richtlinien durch die Erfüllung von Standards wie z.B. ISO 9000 gegeben, die der Entwicklungsprozess zu erfüllen hat sofern der Kunde diese Standardisierung in der zu entwickelnden Software benötigt.

2.2.3 Anforderungscharakteristika

Folgende Charakteristika spezifizieren Anforderungen:

- Korrekt: Stehen die Anforderungen für das was wirklich benötigt wird.
- Konsistent: Die Spezifikation ist konsistent.
- Nachweisbar: Die Anforderungen können getestet werden
- Komplett: Es sind wirklich alle Anforderungen abgedeckt.
- Verständlich: Die Anforderungen sind einfach und klar zu verstehen.
- Eindeutigkeit: Es besteht keine Gefahr der Missinterpretation durch verschiedene Personen
- Modifizierbar: Die Anforderungen können einfach modifiziert werden.

- **Verfolgbar: Die Anforderungen können von Level zu Level nachvollzogen werden.**

[Kontio 1998]

2.2.4 Anforderungsidentifikation

In Softwareentwicklungsprozessen definiert man oft hunderte von Anforderungen welche während dem Prozess verwaltet werden müssen. Folglich muss ein Identifikationssystem eingerichtet werden. Hierbei ist es möglich die Anforderungen in Gruppen zu unterteilen was eine leichtere Handhabung dieser verspricht. Es gibt verschieden Möglichkeiten wie dies umgesetzt werden kann:

- *Ein eindeutiger Identifier:* Eine fortlaufende Nummer welche manuell oder maschinell der jeweiligen Anforderung zugeordnet wird.

- *Eine fortlaufende Nummer in einer Dokument Hierarchie*: Diese Nummer wird unter Berücksichtigung der Position der Anforderung im Dokument vergeben.

- *Eine fortlaufende Nummer in der Anforderungskategorie:* Diese Nummer wird zusätzlich zu einem Namen, welcher die Kategorie der Anforderungen identifiziert, vergeben. (z.B. Kategorie Sicherheitsanforderungen oder Funktionsanforderungen)

Bevor nun im speziellen auf die Verfolgung von Anforderungen in Softwareprozessen eingegangen wird wollen wir noch kurz die Kernaufgaben des Requirements Engineering erläutern:

2.2.5 Der Anforderungsmanagementprozess:

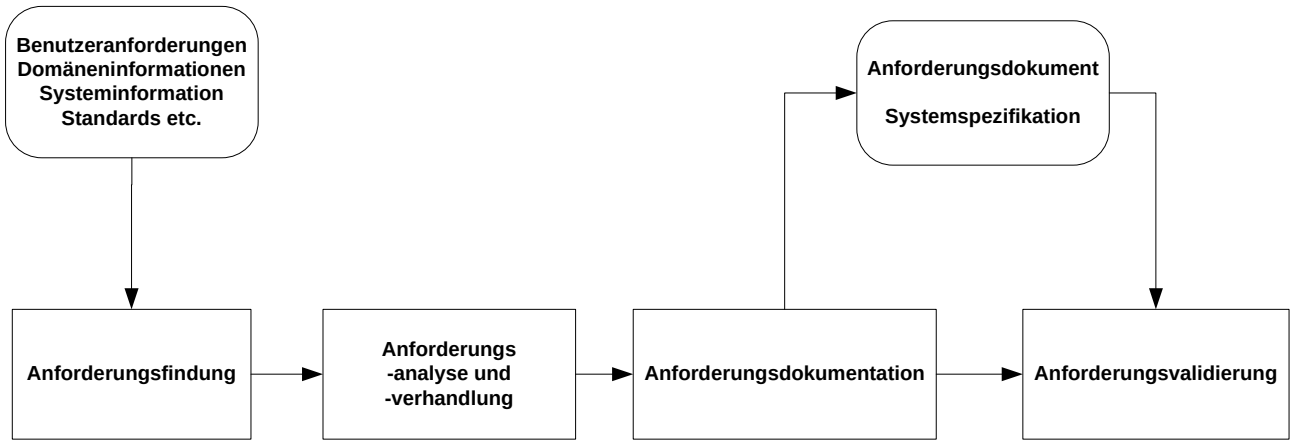


Abbildung 13 Anforderungsmanagement Prozess
[Kotonya / Sommerville 98]

Abb. 12 zeigt einen Anforderungsmanagementprozess wie er in vielen aktuellen Projekten auf Grund der Einfachheit angewandt wird. Die klare Struktur ist leicht zu verstehen, leicht anzuwenden und die Kernthemen sind klar definiert. Im folgenden Teil werden diese Iterationen und die damit entstehenden Artefakte erläutert.

Die Anforderungsfindung:

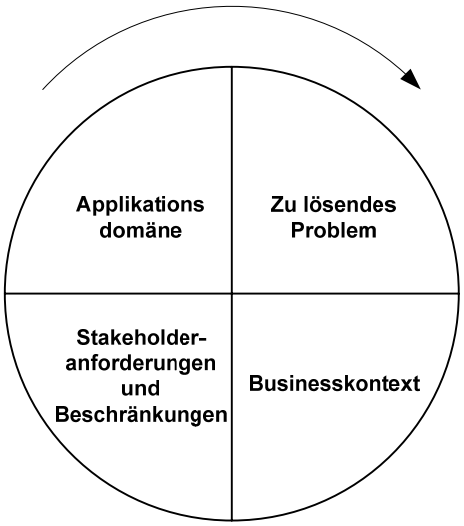


Abbildung 14 Anforderungsfindung [Hofmann 00]

In dieser Phase die bereits zum Entwicklungsprozess gezählt werden kann, werden von den Stakeholdern des Projekts die notwendigen Anforderungen gefunden. Es geht hierbei darum auf die Bedürfnisse aller Stakeholder einzugehen und diese zu definieren. Diese Phase kann durchaus problematische Formen annehmen, da die Lösungsansätze für die Probleme auf Grund von unterschiedlichen Perspektiven meist schwer zu definieren sind. Hierbei ist insbesondere der Standpunkt der Kunden von Bedeutung, welche oft nicht bereit sind, sich auch andere Sichtweisen zu Gemüte zu führen, oder die gegebenen technischen Restriktionen nicht in ihre Sichtweisen mit einbeziehen.

Die wichtigsten Punkte welcher in dieser Phase abgehandelt werden müssen sind einerseits ein gewisses Verständnis für die Domäne zu entwickeln in welcher das System interagieren wird, andererseits die Details der zu lösenden Probleme in dieser Domäne zu definieren und somit in einer frühen Phase der Entwicklung sich Domänenwissen anzueignen und in die Planung einfließen zu lassen. Hinzu kommt auch dass die Definition der Business Ziele welche mit der Software umgesetzt werden sollen in die Planung einfließen muss um bestehende Systeme oder Prozesse zu berücksichtigen. Zusammenfassend geht es darum nach dieser Phase der

Domänenverständnisfindung die speziellen Bedürfnisse der Benutzer und Stakeholder zu definieren, zu verstehen und dann konkrete Lösungen dafür zu kommunizieren.

Die Anforderungsanalyse:

Nach der Phase der Findung geht es darum die definierten Anforderungen zu analysieren. Dabei wird insbesondere darauf geachtet ob bereits Problem unter Anforderungen auftreten um diese sofort zu lösen. Weiters geht es in dieser Phase des Anforderungsmanagements auch darum Konflikte unter einzelnen Anforderungen zu erkennen und zu verhandeln. Sind diese Kernpunkte der Analyse abgeschlossen können Entscheidungen darüber gefällt werden welche Anforderungen nun in Produktion gehen können und welche Anforderungen auf Grund von Problemen wie z.B. mit Standards oder anderen externen Faktoren nicht umgesetzt werden können oder überdacht werden müssen. Hierbei ist selbstverständlich, auch den budgetäre Aspekt in Betracht zu ziehen, um diese Entscheidungen zu unterstützen.

Die Anforderungsdokumentation:

Es muss ein Weg gefunden werden wie die Anforderungen für alle beteiligten Stakeholder im Projekt verständlich dokumentiert werden können. Nachdem neben den im Entwicklungsprozess beteiligten Stakeholdern auch noch externe Rollen davon betroffen sind und diesen nicht zugemutete werden kann sich hochkomplexe IT spezifische Dokumentationsgrundlagen zu erarbeiten, werden Anforderungen meistens in Form von reinem Text festgehalten. Die somit definierten Dokumente dienen dem Entwicklungsprozess als Grundlage. In dieser Phase kommen Anforderungsmanagementtools erstmalig zum Einsatz.

Die Anforderungsvalidierung:

Abschließend müssen die definierten Anforderungen welche in die Dokumentation einfließen validiert werden. Dies dient dazu um die Dokumentation als Spezifizierung der zu erstellenden Software übernehmen zu können. In diesem Sinne ist es eigentlich eine zweite kontrollierende Analysephase, die den Sinn und Zweck

hat, ähnliche Probleme wie Konflikte oder Fehlspezifikationen aufzudecken. Somit wird validiert ob Zusagen für die spezifizierten Anforderungen gemacht werden können. Eine interessante Definition für Anforderungsvalidierung kann in einem Zitat von [Alford 90, Greenspan et al 94] nachgelesen werden: „ *Bei der Verifikation von Anforderungen wird die Spezifikation der Anforderungen auf interne Konsistenz durch mathematische Beweise oder Inspektionstechniken geprüft. Um die Anforderungen zu validieren müssen wir ihre Konsistenz mit der Intension der Stakeholder überprüfen*“

Werden in der Validierung der Dokumentation und den darin enthaltenen Anforderungen Fehler und Inkonsistenzen bemerkt, treten oft neue Anforderungen auf, oder bestehende Anforderungen müssen überdacht werden. Somit muss der komplette Prozess von Anforderungsfindung, -analyse, -verhandlung, -validierung noch einmal durchlaufen werden.

2.2.6 Definition von Anforderungsverfolgung (Requirements Tracing / RT):

“Requirements tracing is the ability to follow the life of a requirement in a forward and backward direction”
[Gottel/ Finkelstein 1994]
´

Anforderungsverfolgung ist die Möglichkeit den Lebenszyklus einer Anforderung sowohl nach vorne als auch nach hinten verfolgen zu können.

2.2.7 Probleme der Softwareentwicklung bezüglich Anforderungen

Während einem Softwareentwicklungsprozess entsteht eine Vielzahl von Dokumenten und Modellen welche die zu entwickelnde Software beschreiben. Diese Dokumente stellen in den meisten Fällen unterschiedliche Sichten auf das System zu unterschiedlichen Zeitpunkten dar und stehen auf Grund dessen in Relation zueinander. Sobald sich eine Anforderung im Spezifikationsdokument verändert, müssen die Änderungen in allen anderen Dokumenten welche mit dieser

Anforderung verknüpft sind, übernommen werden. Somit wird das Entstehen von Inkonsistenz vermieden. Immer wieder kommt es auf Grund von fehlenden Verknüpfungen unter den Anforderungen zu massiven Problemen mit einzelnen Anforderungen.

In den meisten Fällen treten Probleme mit individuellen Anforderungen eines Subsystems auf, weswegen sich das Problem in weitere Anforderungen fortpflanzt. Trotz dessen dass Anforderungsverfolgung bei einer Vielzahl von Projekten durchgeführt wird, sind noch immer viele Probleme mit ihr korreliert, welcher einer Lösung bedürfen.
Eines der offensichtlichsten Probleme bei Anforderungsverfolgung ist, dass die spezifizierten Anforderungen den Wünschen und Erwartungen der Kunden nicht entsprechen. Dies ist einerseits auf die mögliche, oben bereits erwähnte Inkonsistenz zurückzuführen, andererseits auf mangelnde Kommunikation und Missverständnisse zwischen Kunden und Entwicklern. Fehler in der Spezifikation, und somit das Fehlen von vollständigen Anforderungen ist oft nur der Beginn einer Problemkette welche sich negativ auf das Endprodukt auswirkt. Wir werden insbesondere im Themenbereich Quality Function Deployment auf diese Problematik eingehen und einen Lösungsansatz dafür bereitstellen.

Oft werden Anforderungen in Form von Text spezifiziert. Es wird also eine ziemlich einfache Methodik angewandt um komplexe Systeme zu beschreiben. Hierbei kommt es zu dem Fall dass der festgehaltene Text nicht ausreichend formalisiert ist und somit vom Leser falsch interpretiert wird. Es wird in dem Dokument nicht das Bild welches der Verfasser zu spezifizieren versuchte vermittelt, sondern ein differenziertes, welches zu Abweichungen führt.

Ein weiteres Problem das eintreten kann, ist die Tatsache dass keine der beteiligten Rollen, weder Stakeholder noch Entwickler oder Analysten, den kompletten Systemanforderungsumfang versteht. Zum Beispiel kann sich der Stakeholder einer Warenverwaltungssoftware mit der Identifikation von Objekten über deren Strichcode zufrieden geben obwohl jedoch manche seiner Hersteller Waren ohne Strichcode ausliefern. Somit würde sich dies massiv auf das zu entwickelnde System

auswirken – Die Anforderungen wären falsch und das System würde nur einen Teil des zu automatisierenden Geschäftsprozesses abdecken.

Nachdem es bei den meisten Projekten mehr als einen Stakeholder gibt, kommt es auch immer wieder zu Anforderungskonflikten, die auf Grund von mehreren individuellen Anforderungen einzelner Stakeholder entstehen. Somit müssen Kompromisse bez. einzelner Funktionen zwischen einzelnen Stakeholdern gefunden werden die sich wiederum negativ auf andere Anforderungen auswirken. Diese Anforderungen müssen dann im Rahmen einer Anforderungsanalyse konkretisiert werden um eine Lösung auftretender Probleme herbeizuführen.

2.2.8 Requirements Tracing Techniken und Methoden

In [Gotel Finkelstein 94] werden frühe Techniken für RT angeführt:

- Quer Referenzierungs Schemata [Evans 98]
- Schlüsselphrasen und deren Abhängigkeiten [Jackson 91]
- Dokumentvorlagen [IDE 91]
- Anforderungsverfolgungsmatritzen [Davis 90]
- Matrix Sequenzen [Brown 91]
- Hypertext [Kaindl 93]
- Integrationsdokumente [Lefering 93]

Diese Techniken sind nur technische Hilfsmittel um RT durchzuführen und nehmen keinerlei Rücksicht auf Wert einzelner Anforderungen und Kosten. Sie unterscheiden sich hauptsächlich in Quantität und Vielfältigkeit der Information welche verfolgt wird. Weiters bestehen Unterschiede in der Anzahl der Verknüpfungen zwischen den einzelnen Informationen und im Ausmaß mit welchem das RT bei Anforderungsänderungen aufrechterhalten werden kann.

Sieht man sich die verschiedenen Ansätze bezüglich RT an findet man ein Vielzahl die sich in ein Spektrum vom Low- End Model bis zum High – End Modell

erstrecken. Jeder der Ansätze spiegelt eine unterschiedliche Auffassung von RT und somit auch untschiedlichen Detailierungsgraden wieder Um hierbei einen Überblick zu geben werden nun sowohl ein Low End als auch auch ein High- End Modell kurz vorgestellt

[Ramesh, Jarke 2001]

2.2.9 Low- End Traceability Modell

Viele Entwicklungsprojekte geben sich mit Verwendung von Verfolgungstabellen zufrieden. Die Projektbeteiligten erstellen Verbindungen von Anforderungen zu Modell Anforderungen, von Anforderungsspezifikationen zu System Komponenten und versuchen Artefakte zu verifizieren. Jedoch existiert keinerlei explizite Identifikation der Semantik über die Beziehung selbst.
Trotzdem kann gezeigt werden welche Komponenten welche Anforderungen erfüllen und welche Anforderungen auf verschiedene Komponenten abgebildet sind. Somit kann das Projektteam feststellen ob alle Anforderungen vom System erfüllt werden. Es besteht die Möglichkeit dass auch in der Testphase eines Projektes diese Anforderungen auf Testfälle übertragen werden um eine Verifizierung der gegebenen Anforderungen durchzuführen.

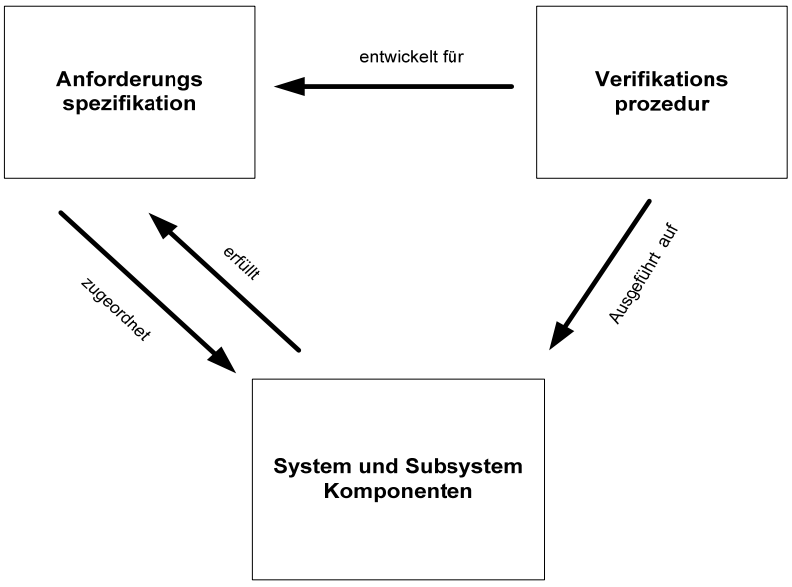


Abbildung 15 Low End Tracability Modell [Ramesh /Jarke 01]

2.2.10 High- Level Traceability Modell:

Im Gegensatz dazu steht das High Level Traceability Modell welches wesentlich umfangreichere Verfolgungsschemata vorzuweisen hat. Weiters wird bei der detailierteren Spezifikation auch die Traceability Information wesentlich umfangreicher behandelt. Rationales Management ist eine zentrale Aufgabe in High –End RT Modellen. Es werden hierbei neben Informationen bezüglich Code Anforderungen und Testfällen auch Informationen über Ressourcen, Entscheidungen, deren Begründungen und Alternativen festgehalten.

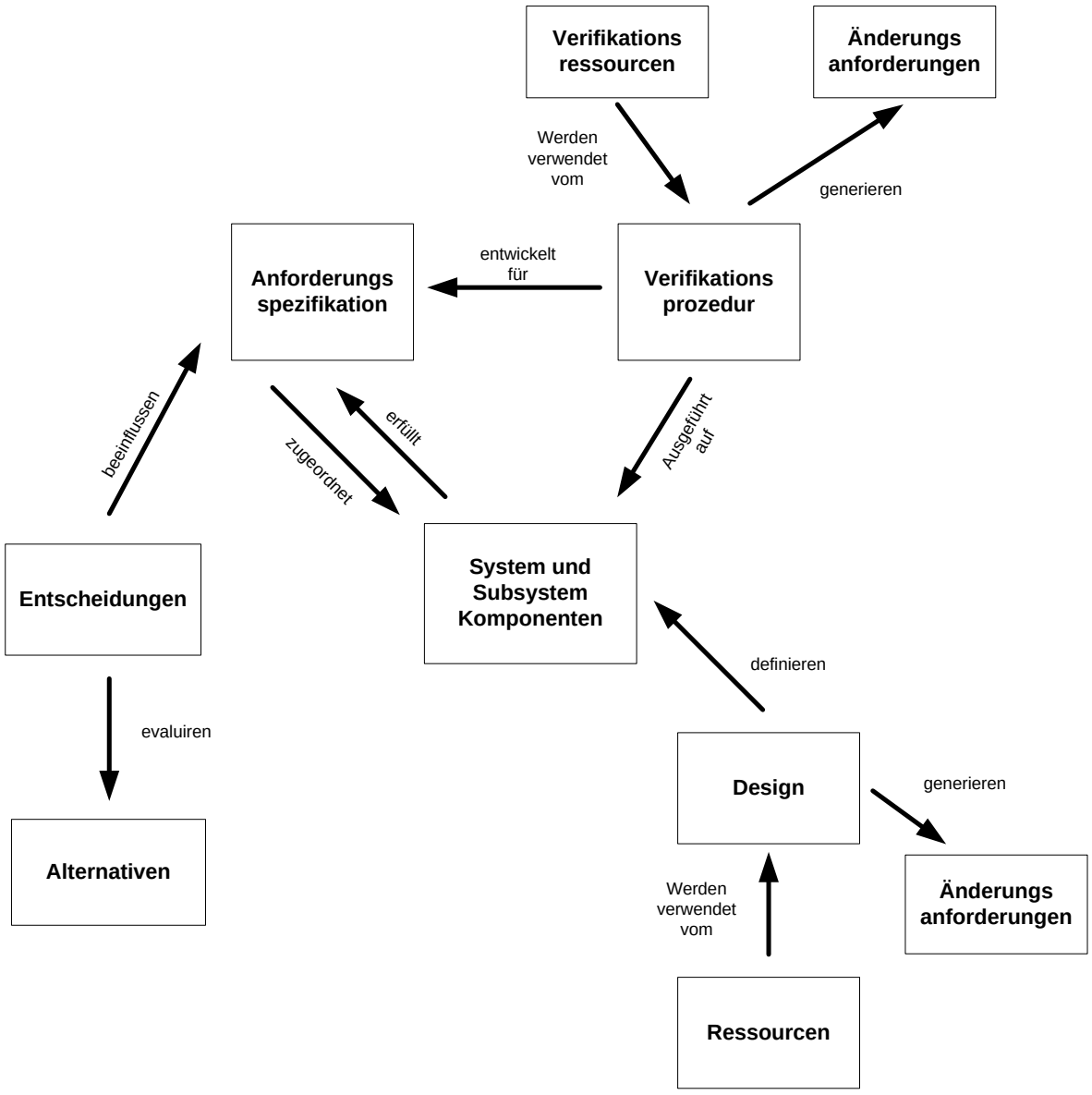


Abbildung 16 High End Tracability Model [Ramesh /Jarke 01]

2.2.11 Modelle in der Praxis:

In den im vorigen Abschnitt erläuterten Modellen wurde das Konzept der Anforderungsverfolgung veranschaulicht. In diesem Abschnitt soll ein Einblick in die vorhandenen Techniken gegeben werden mit welchen Anforderungen dargestellt werden. Neben UML (Unified Modeling Language) bieten auch UP (Unified Prozess) und SADT (Structured Analysis and Design Technique) viele verschiedene Möglichkeiten um vor allem in der Anforderungsfindungsphase zu unterstützen. Szenario Modelle, Datenflussmodelle und Konzeptuelle Modelle kommen bei fast allen Ansätzen zum Einsatz. Besonders bei objektorientierten Projekten wird in den meisten Fällen auf konzeptuelle Modellierung gesetzt. Diese Art von Modell beschreibt Entitäten und Objekte. Bei Datenflussmodellen wird der Verlauf der Daten im System veranschaulicht. Stakeholder, Subsystemen bzw. die transportierte Information und die übermittelten Daten werden somit repräsentiert.

Das wohl am meisten verbreitete Modell ist das Use Case Modell von Jacobsen. Es beschreibt einen Anwendungsfall und definiert die Interaktionen zwischen dem System und den Benutzern. Dabei liegt der Fokus in der Darstellung des Business Goal. Es wird immer nur ein Ablauf oder Prozess dargestellt.

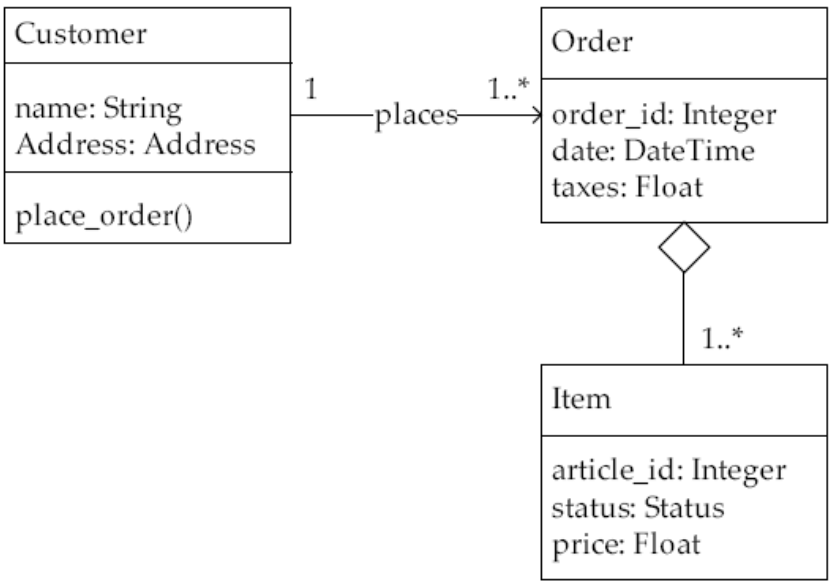


Abbildung 17 Konzeptuelles Modell [Kleppe 03]

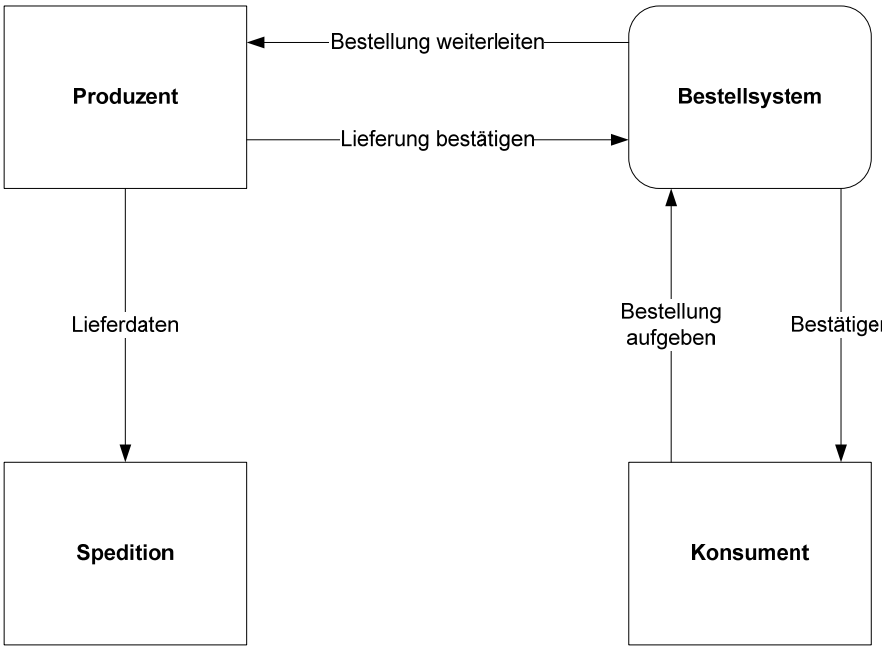


Abbildung 18 Beispiel. für ein Datenflussmodell

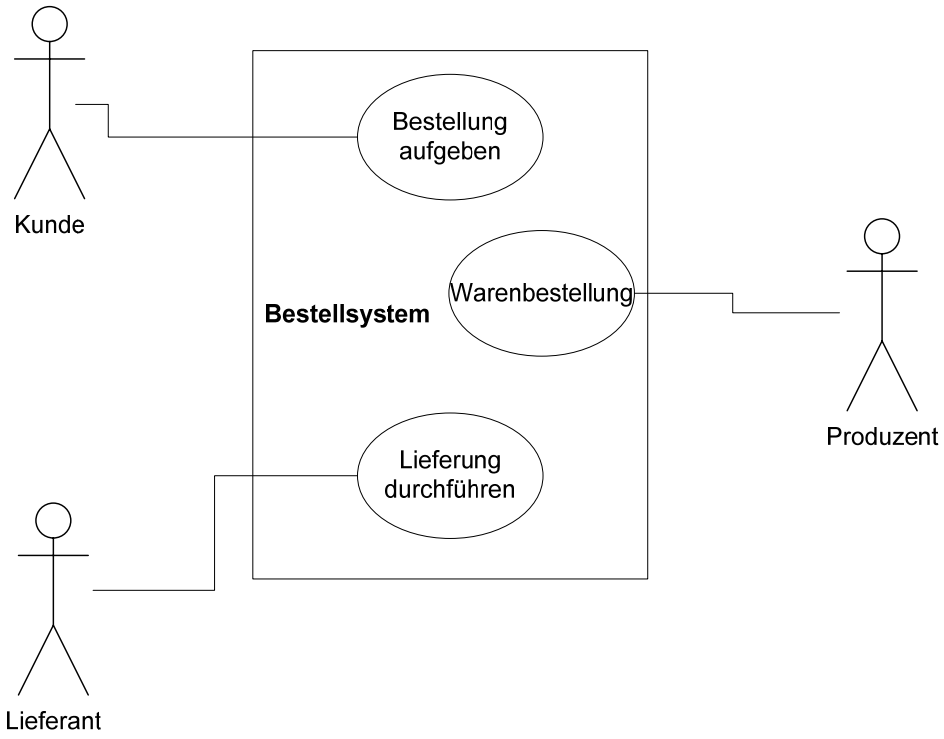


Abbildung 19 Use Case Modell

3 Kapitel Anforderungsverfolgung in der modelbasierten Softwareentwicklung:

Nachdem im letzten Kapitel die Themenbereiche Anforderungsverfolgung und MDA einzeln behandelt wurden, um eine Einführung in die Thematik zu bieten, werden in diesem Kapitel die 2 Teilbereiche zusammengeführt. Konkret wollen wir uns ansehen wie man diese 2 Ansätze verbinden kann um dann in weiterer Folge auf Technologien zur Umsetzung dieser Verknüpfung einzugehen. Eine praxisorientierte Erläuterung anhand von einzelnen Beispielen soll das Verständnis erleichtern.

3.1 Tracability und Konsistenz im Kontext der MDA

Um die Verfolgung von Anforderungen durchzuführen müssen wir noch genauer definieren wie der Begriff Traceability im Sinne der Modelbasierten Softwareentwicklung zu verstehen ist.
In der Literatur bezüglich MDA Traceability, welche zurzeit eher spärlich existiert, wird bei [Shaham-Gafni, Hartman 05] folgende Definition für Traceability gegeben:

„*…a relationship between entities: a set of source entities and a set of target entities. The exact meaning of a relationship depends on the context in which it is used.*“

Bevor wir näher auf diese Definition eingehen möchte ich noch kurz auf den Begriff der Konsistenz eingehen welcher mit der Definition von Traceability in der MDA in Zusammenhang steht. Der Traceability Ansatz den wir unter dem Punkt Anforderungsverfolgung erläutert haben, beschäftigt sich mit der klassischen Nachverfolgung von Änderungen oder Design Entscheidungen, um deren Auswirkungen abschätzen und somit Risiken zu identifizieren und ein gutes Change Management zu unterstützen.
Bei modellbasierter Entwicklung würde das bedeuten, dass man sich mit der Verfolgung von in Modellen entstehenden Anforderungen auf unterschiedlichen Ebenen beschäftigt, während es eigentlich die Modelle sind die die Anforderungen

beschreiben und entwickeln. Somit kann es auch als Aufgabe gesehen werden, die Konsistenz der einzelnen Ebenen im Prozess der Modellerstellung zu überprüfen.

Der Mechanismus der für Anforderungsverfolgung notwendig ist, ist ein wichtiger Punkt in welchem der Unterschied zur Konsistenz liegt. Es wird nämlich ein Werkzeug benötigt um die Verknüpfungen unter einzelnen Modellen und Artefakten festzuhalten. Auch das Definieren und Verwalten der Anforderungen muss durch ein solches Tool ermöglicht sein. Diese können dann gegebenenfalls nachvollzogen werden falls Änderungen durchgeführt werden müssen.

Nach der Definition von Traceability welche wir oben angeführt haben ist diese als die Verbindung zweier Entitäten zu sehen, wobei die Verbindung im Kontext mit der genauen Definition der Verknüpfung zu verstehen ist. Diese Idee beschreibt die genaue Bestimmung der Verknüpfung unter einzelnen modellierten Sachverhalten. Verknüpfungen können z.B. mittels UML, wo die Semantik als Regel den Modellerstellungsprozess definiert, dargestellt werden. Somit wird Tracability auf eine Ebene gebracht wo die Verbindung welche als gegeben modelliert wird, die Funktion der Nachverfolgung mit der gegebenen Semantik abstrahiert. Der positive Effekt der gerade beschrieben wurde wirkt sich ebenso positiv auf die Frage der Konsistenz aus. Die Definition der Regeln zur Beschreibung der Verknüpfungen und Abhängigkeiten einzelner Artefakte kann in das Werkzeug welches wir zur Nachverfolgung und Entwicklung verwenden so eingebaut werden, dass neben ersterer Aufgabe ebenso die Konsistenzüberprüfung auf Grund der gleichen Mechanismen implementiert ist.

Die Aufzeichnung von Regeln im Kontext der Anforderungsverfolgung ist in diesem Zusammenhang noch als mögliche Schwierigkeit zu sehen, jedoch sind auf Grund dieser Überlegungen gegeben, dass man mittels gleicher Mechanismen die Überprüfung von Konsistenz und Abhängigkeiten definieren kann.
Die Verfolgung von Anforderungen profitiert von der Tatsache dass Abhängigkeiten in Modellen dargestellt werden. Semantische Regeln deren Überprüfung auf Grund von Tools möglich ist definieren diese Abhängigkeiten. Ob diese Abhängigkeiten nun halten und die Konsistenz auf den einzelnen Ebenen ebenso gewährleisten, kann

durch diesen Ansatz überprüft werden und somit fließen eigentlich klassische Elemente der Anforderungsverfolgung genauso wie MDA spezifische in das Konzept ein.

3.2 Klassifizierung der Anforderungen im Sinne des MDA Prozess

Unter dem Punkt Arten von Anforderungen wurde eine Klassifizierung in funktionelle und nicht funktionelle Anforderungen vorgestellt. Im Zusammenhang mit dem MDA Entwicklungsprozess wird diese Unterteilung grundsätzlich beibehalten, jedoch bietet sich auf Grund des Prozessverlaufs noch eine genauere Unterteilung an. Wir definieren also:

- Anwenderanforderungen
- Systemanforderungen
- Subsystemanforderungen
- Umsetzung der Anforderungen in Code
- Verifizierung von Anforderungen im Codes durch Testfälle

Diese Arten von Anforderungen entstehen aus der Analyse der Geschäftsprozesse bzw. der einzelnen Subprozesse im MDA Entwicklungsprozess. Die verschieden Dokumente die auf den einzelnen Ebenen spezifiziert werden, beschreiben diese Anforderungen. Die Personen die für die Erstellung der Dokumente und Modelle verantwortlich sind, spielen in diesem Zusammenhang eine wesentliche Rolle, denn sie müssen miteinander kommunizieren um genaue Spezifikationen in verschiedenen Abstraktionsgraden erstellen zu können. Auch die Kommunikation unter den einzelnen Stakeholdern im MDA Prozess spielt im Kontext der Anforderungsverfolgung eine wichtige Rolle. Die Dokumentation von z.B. Emails, (Chat) Meetings oder Besprechungsprotokollen ergänzt die Verfolgung der Anforderungen aus der Sicht der Entscheidungsfindung und erläutern so Änderungen am System.

Den Prozess der schrittweise genauere Definitionen oben gelisteter Anforderungen erstellt, soll in nachfolgende Graphik veranschaulichen.

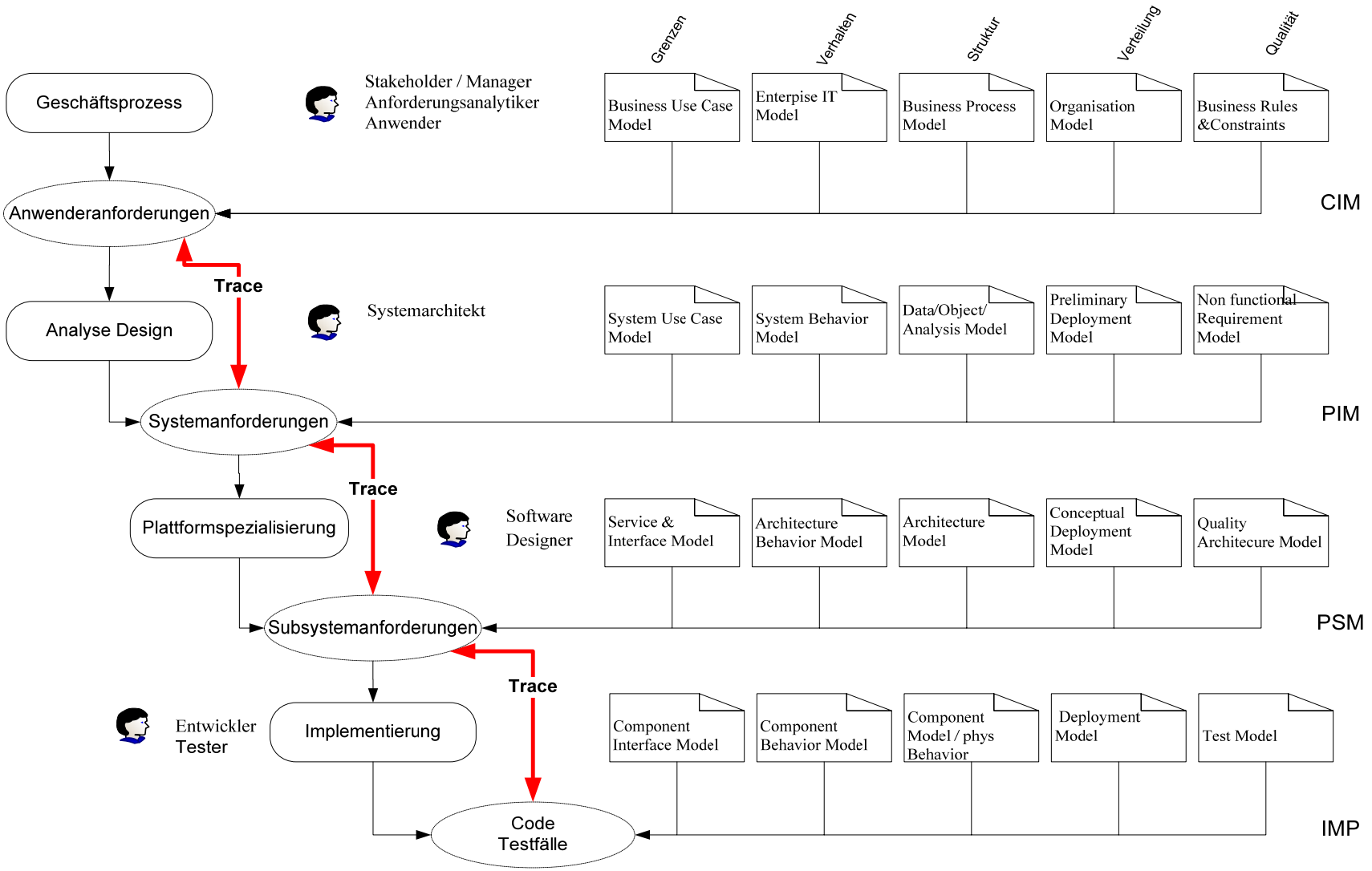


Abbildung 20 MDA Modellframework mit RT Ansatz

3.3 Sichtweisen im Anforderungsverfolgungsprozess:

Die Graphik zeigt was in einem MDA Prozess entsteht und, verfolgt werden soll und ist als Zusammenführung der im Kapitel 2 beschriebenen Themenbereiche anzusehen. Um nun die eigentliche Verfolgung der Anforderungen zu erläutern ist es sinnvoll gegebene Sichtweisen in den Prozess einfließen zu lassen:

Grenzsicht:

Im Rahmen dieser Sichtweise, bezogen auf die Grenzen des zu erstellenden Systems entstehen die Dokumente in der Ersten Spalte oben angeführter Graphik. Darunter sind auf Ebene des CIM Bussiness Use Case , auf Ebene des PIM System Use Case usw. definiert(vgl. Use Case in Kapitel 2/ Requirements Tracing Techniken und Methoden). Die funktionellen Anforderungen eines Systems, im MDA Entwicklungsprozess, werden mit UML im Rahmen des PIM definiert.

Verhaltenssicht:

Dokumente die das bestehende IT System beschreiben oder auf System- oder Architektur- Verhalten eingehen beschreiben diese Ansicht und repräsentieren die 2. Spalte der Dokumente in der Graphik.

Struktursicht:

Das Festlegen von Geschäftsmodell auf CIM Ebene, Daten- und Objekt- Modellen auf Ebene des PIM sowie Architektur, Komponenten oder physisches Verhaltensmodell der Software auf Ebenen des PSM und der Implementierung definieren diese Sichtweise.

Verteilungssicht:

Im Rahmen der Verteilung geht es auf CIM Eben darum, die Organisationsdokumente und Verteilungsmodelle auf den darunter liegenden Ebenen zu definieren. Spalte 3 der Dokumente in der Graphik.

Qualitätssicht:

Unter den Aspekten des Qualitätsmanagement kann Anforderungsverfolgung explizit über alle Abstraktionsebenen modelliert werden. Im Rahmen eines Anforderungsmodells sollen im Sinne dieser Sichtweise funktionelle als wie auch nicht funktionelle Anforderungen modelliert werden. Die Qualitätssicht beschreiben die Dokumente der letzten Spalte welche vom Dokument der Geschäftsregeln und Beschränkungen bis hin zum Test Modell reichen.

3.4 Repräsentation von Anforderungen mit der Unified Modelling Language und Systems Modelling Language:

Wir haben bereits unter Kapitel 2, Definition von Modellen und Metamodellen das Konzept der UML Notation mit den verschiedenen Abstraktionsebenen erläutert. In UML sind zwar keine expliziten Modellelemente für Anforderungen gegeben, auf Grund der Möglichkeit neue Stereotype als Erweiterung einzuführen ist es möglich Anforderungen explizit als solche zu definieren und zu modellieren. Dieser Mechanismus der hier von UML geboten wird ist auf eine Erweiterung der UML Sprache zurückzuführen welche unter dem Namen Systems Modelling Language (im weitern als SysML bezeichnet) der Open Management Group (OMG) bekannt ist

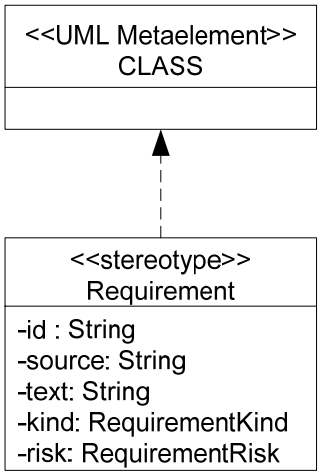


Abbildung 21 UML Stereotype

Die Eigenschaften einer solchen Anforderung werden im Rahmen der Modellierung ebenfalls beschrieben:

-id: ist ein eindeutiger Identifier (siehe Kapitel 2 - Anforderungsidentifikation)
-source: Verweis auf die Quelle der Anforderung (kann ein Dokumenttitel oder ein Stakeholder sein)
-text: Text der die Anforderung charakterisiert und beschreibt.
-kind: Die Art oder Klassifizierung einer Anforderung
-risk: Das bewertete Risiko einer Anforderung
[SysML 05]

Bevor wir hier konkrete Beispiele in Syntax dieser beider Sprachen anführen, erläutern wir kurz den Zusammenhang dieser 2 Sprachen.

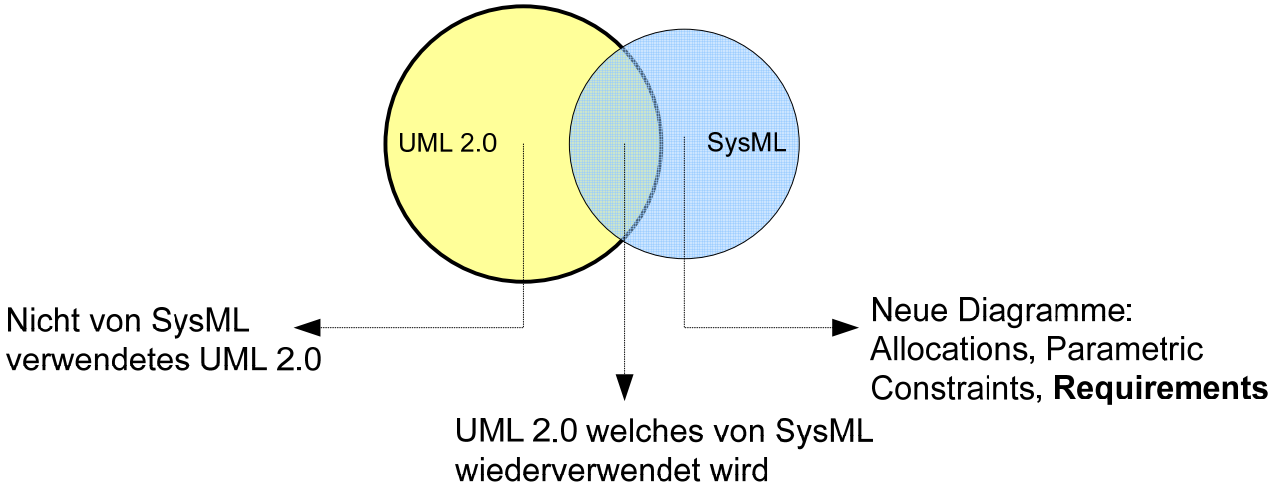


Abbildung 22 UML und SysML [SysML 05]

SysML ist eine graphische auf UML basierende Sprache welche als Open Source Projekt initiiert wurde. Sie benutzt gebräuchliche UML Diagramme und bietet neben diesen Erweiterungen (vgl. Abb. 22). Im Unterschied zu UML ist sie jedoch geringer im Umfang und deswegen leichter zu erlernen, anzuwenden und zu implementieren.

Mittels SysML ist es möglich die Spezifikation, Analyse, das Design, die Verifizierung und die Validation eines Systems vorzunehmen welches aus Hardware, Software, Daten, Personen und Prozeduren besteht. Somit ist SysML als einer der Schlüssel für Modellbasierte Softwareentwicklung anzusehen. Die nachfolgende Graphik zeigt den Umfang der SysML in einer Baumstruktur in welcher die unterschiedlichen Diagrammarten angeführt sind.

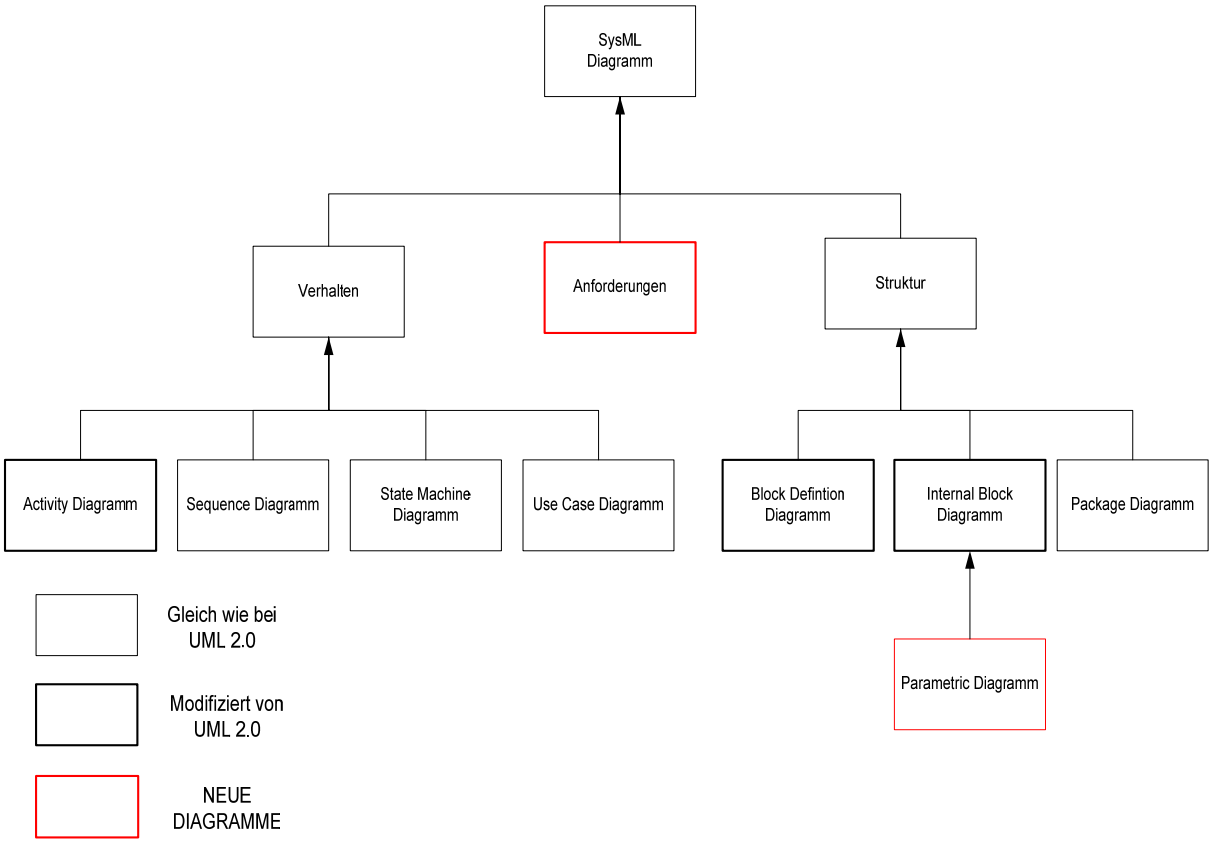


Abbildung 23 SysML Hierarchie [SysML 05]

Werden Anforderungen mit Hilfe standardisierter oder eigener stereotypischen Verbindungen modelliert, besteht die Möglichkeit sie untereinander zu verknüpfen und sie in Analyse, Design bis hin zu Test Modelle zu integrieren. Nachfolgende Graphik zeigt wie eine Anforderung in SysML Notation modelliert werden würde:

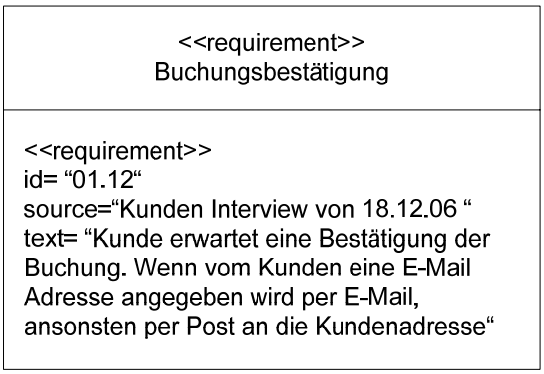


Abbildung 24 SysML Requirement

Die Modellierung von Anforderungen ermöglicht die Integration von in Textform spezifizierten Anforderungen welche mittels Anforderungsmanagement tools verwaltete oder spezifiziert werden. Somit ermöglicht diese Technologie auch eine Integration in den modellbasierten Entwicklungsprozess.

Wir haben im vorhergehenden Abschnitt die einzelnen Sichtweisen in der MDA definiert. Da nun SysML gemeinsam mit UML Möglichkeiten bietet Verbindungen unter Anforderungen zu definieren und dies über alle Ebenen des Entwicklungsprozesses, möchte ich dies auch in Relation mit den Sichtweisen erläutern. Wir führen dazu das Beispiel der Buchungsbestätigung weiter, welches wir bei Abb. 23 herangezogen haben. Es soll wie im Text der Anforderung beschrieben wird eine Buchungsbestätigung gestellt werden, wenn ein Kunde online ein Auto bei einem Autoverleih bucht.

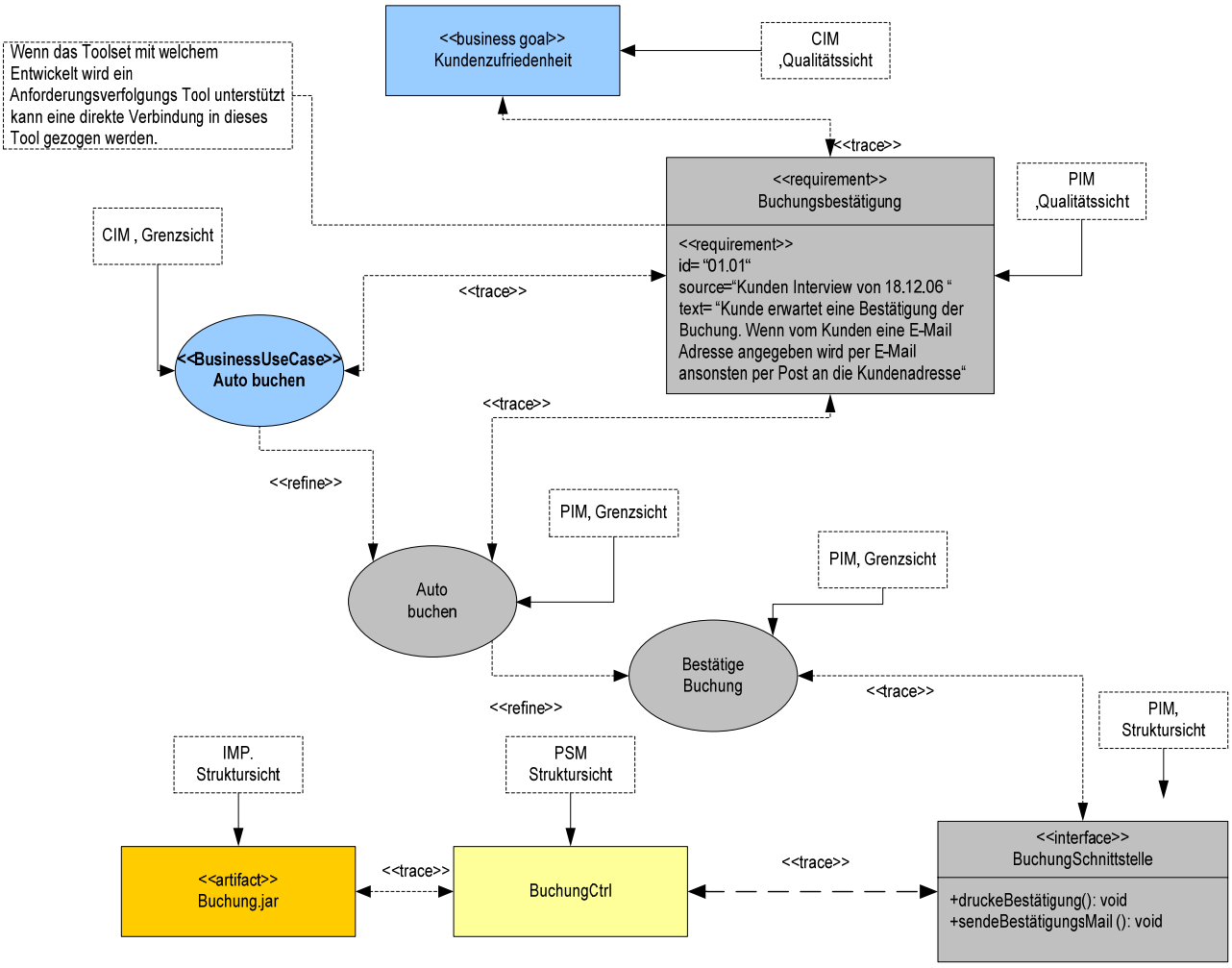


Abbildung 25 Anforderungsverfolgung mit SysML

Wie die Abbildung veranschaulicht werden die einzelnen Ebenen und Sichtweisen in den Prozess miteinbezogen. Das Business Goal Kundenzufriedenheit welches auf Ebene des CIM aus Qualitätssicht definiert ist, wird zum Beispiel durch die Anforderung Buchungsbestätigung erreicht. Diese Anforderung ist im Rahmen der Modellierung funktioneller und nicht funktioneller Anforderungen auf Ebene des PIM auch aus Qualitätssicht festgehalten worden. Das Business Use Case Auto buchen welches auf der Ebene des CIM, im Rahmen der Grenzsichtweise erstellt wurde, ist der Ausgangspunkt, der für die Erfüllung der Anforderung verantwortlich ist. Die Verfeinerung (deswegen wird hier die Verbindung mit <<refine>> bezeichet) der Use Cases und das Überführen dieser auf eine Plattform unabhängige Ebene

(PIM) führt dazu, dass aus einem Business Use Case ein System Use Case entsteht welches ebenfalls aus grenzbezogener Sicht das System beschreibt. Nachdem in diesem die Funktion welche den gesamten Prozess des „Auto buchen“ beschreibt, festgehalten wird, nimmt man eine Verfeinerung vor, welche nur die Funktion „Bestätige Buchung“ beschreibt. Dieser Subprozess wird im Buchungsinterface beschrieben welches in diesem Fall die Struktursicht der PIM Ebene darstellt. Wie man der Graphik entnehmen kann, muss dieses Interface in den PSM Modellen wie Architektur Modelle oder Interface Modell nachvollziehbar sein um schlussendlich die Anforderung Buchungsbestätigung in dem Artefakt Buchung.jar nachweisen zu können.

Es gibt zwei konkrete Punkte die mit dieser Graphik veranschaulicht werden sollen:

Einerseits sind die Pfeile zur Nachverfolgung während des Entwicklungsprozesses mit jeweils 2 Spitzen versehen. Damit soll veranschaulicht werden dass diese Verfolgung in beide Richtungen funktionieren muss um die Definition von Anforderungsverfolgung zu erfüllen. Das heißt im Konkreten, dass die Anforderungen sowohl aus dem Code in die einzelnen Modelle rückverfolgt werden können und ebenso aus den Modellen die Umsetzung im Code nachvollziehbar ist.

Andererseits wird gezeigt dass sich der Prozess der Anforderungsverfolgung nicht nur auf das Nachvollziehen von Elementen auf unterschiedlichen Ebenen des modellbasierten Entwicklungsprozesses bezieht, sondern dass ebenso auch Zusammenhänge zwischen Artefakten innerhalb der einzelnen Detaillierungsebenen gezogen werden müssen. Der Grund darin liegt dass die konkrete Umsetzung einer Anforderung auf einer Ebene modelliert ist welche sich zwar auf der gleichen Abstraktionsstufe befindet, jedoch hier wiederum als Detaillierung dieser gesehen werden kann. Mit dieser Technik in der Verfolgung von Anforderungen ist es möglich die einzelnen Ebenen in sich genauso auf Konsistenz zu prüfen wie den gesamten Prozess.

Die Anforderung welche eine Buchungsbestätigung fordert kann als funktionelle Anforderung gesehen werden. Wie sieht es jedoch aus wenn wir eine nicht funktionelle Anforderung modellieren und diese nachvollziehen wollen.

Eine solche Anforderung wäre z.B. dass die Response Time der Applikation nicht länger als 0,5 Sekunden betragen darf wenn über die graphische Benutzeroberfläche (GUI) eine neue Buchung erstellt wird. Nachfolgende Graphik erläutert wie die Traces in diesem Bereich verfolgt werden können.

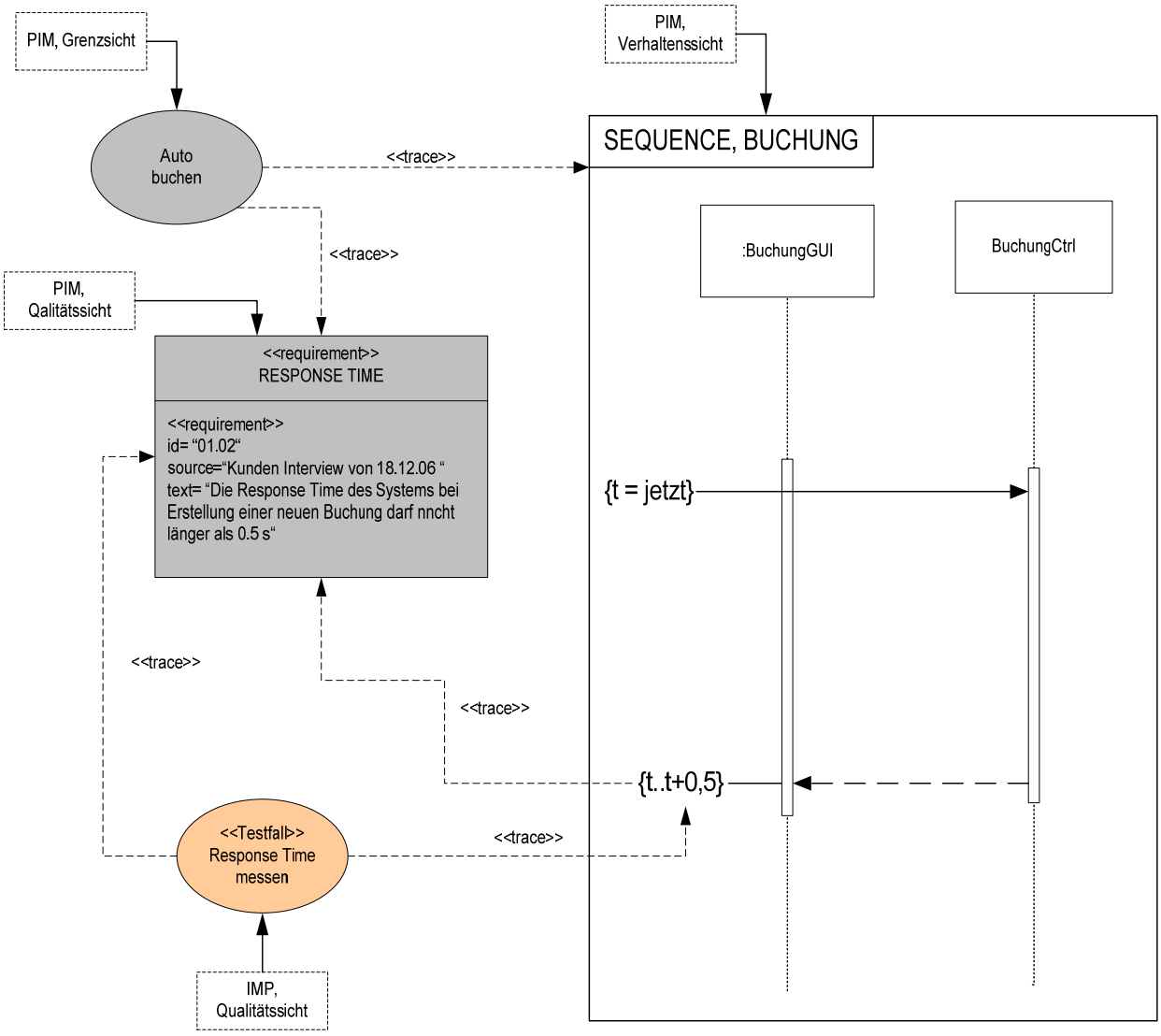


Abbildung 26 Anforderungsverfolgung mit SysML bei nicht funktionellen Anforderungen

Ausgehend vom PIM Auto buchen welches im Rahmen der Grenzsicht erstellt wurde ziehen wir eine Verbindung zu der Qualitätssicht welche die Anforderung Response Time definiert. Im Rahmen des PIM aus Verhaltenssicht wird ein Sequenzdiagramm erstellt welches die Dauer spezifiziert, die bei einer neuen Buchung und somit einer Interaktion zwischen GUI (BuchungGUI) und der Buchungssteuerung (BuchungsCTRL) entstehen darf. Diese Dauer erfüllt die Anforderung. Genauso dient der Testfall Response Time messen aus der Implementierungsebene dafür diese Anforderung zu verifizieren. Somit müssen Traces vom Testfall ebenso zur Anforderung als auch zum Sequenzdiagramm gezogen werden.

SysML bietet neben Möglichkeiten wie sie in den letzten beiden Beispielen erläutert wurden noch weiter Annehmlichkeiten welche den Softwareentwicklungsprozess aus Sicht der Anforderungsverfolgung unterstützen:

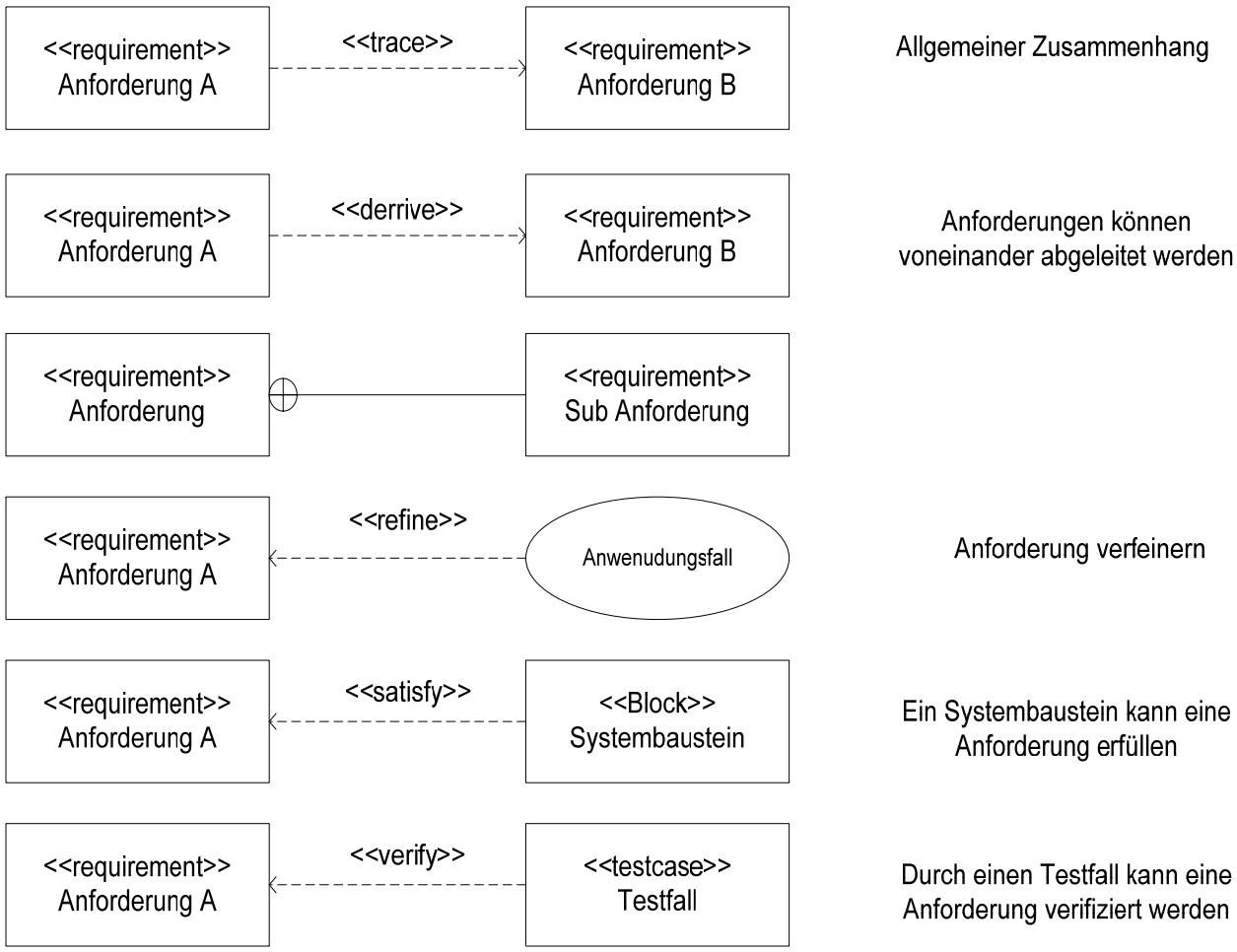


Abbildung 27 SysML Connectors [SysML 05]

Im letzten Abschnitt haben wir erläutert welche Möglichkeiten existieren mittels UML und SysML Anforderungen zu modellieren und zu verfolgen. Die einzelnen Ebenen und Sichten wurden dabei ebenfalls einbezogen um den Zusammenhang zwischen den in Kapitel 2 erläuterten Teilgebieten zu definieren. Wir haben die Möglichkeit komplette Anforderungsbäume zu modellieren in welchen Subanforderungen, abgeleitete Anforderungen, Anforderungen welche aus Anwendungsfällen resultieren, Anforderungen welche durch einen Systembaustein erfüllt werden und Anforderungen welche durch einen Testfall verifiziert werden, dargestellt werden können. Mittels der SysML Notation besteht die Möglichkeit einfache Attribute der in Textform spezifizierter Anforderungen zu modellieren. Für komplexere, wie z.B. operative Anforderungen oder Speicheranforderungen können mittels der oben angegegebenen stereotype Funktion modelliert werden. SysML

bietet die Möglichkeit all diese unterschiedlichen Arten von Anforderungen in einer Graphischen Format, in Baumstruktur oder in tabellarischem Format aufzulisten. Das graphische Format ist unter der Bezeichnung Requirements Diagram bekannt.

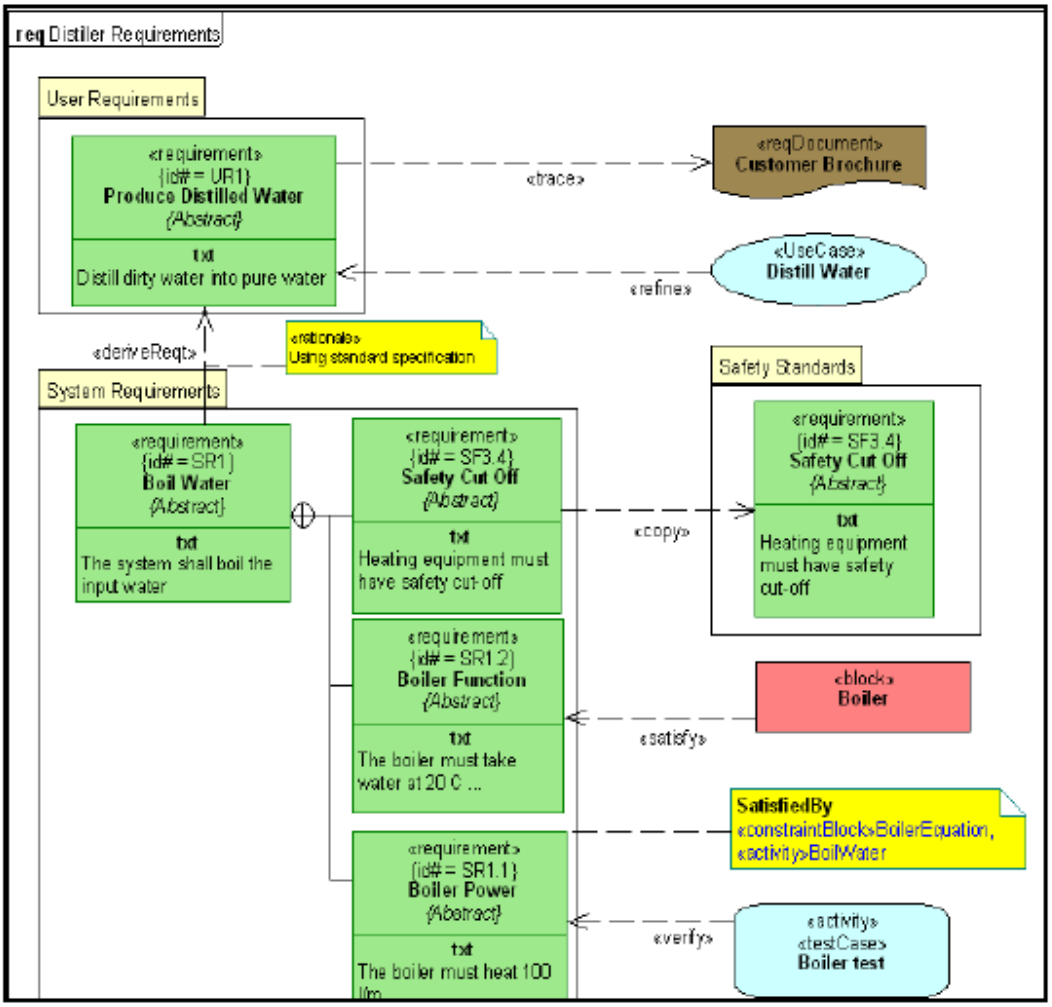


Abbildung 28 Requirements Diagram [Artisan- The SysML]

All diese soeben genannten Elemente können in Modelle welche im Entwicklungsprozess entstehen integriert werden. Diese Tatsache bedarf jedoch genauer Überwachung und Kontrolle welche manuell nicht zu bewältigen wäre. Gerade im MDA Prozess wo automatisierte Transformationen auf die einzelnen Modelle angewandt werden ist dieser Nachverfolgung problematisch. In diesem Zusammenhang müssen Verknüpfungen zwischen Modellierungs- bzw. Entwicklungstools und Anforderungsmanagement Tools gezogen werden welche die komplexe Aufgabe der Verfolgung durch einen MDA Prozess weitgehend erleichtert. Eine vollständige Automatisierung ist jedoch in diesem Kontext sehr

schwer möglich da z.B. auch bei der Definition der Anforderungen im Management Tool manuelle Fehler gemacht werden können, welche nicht oder erst zu spät zum Vorschein kommen. Aus diesem Grund wollen wir den Fokus auf den nächsten Seiten auf Anforderungsmanagement Tools lenken um uns an einem konkreten Tool anzusehen welche Möglichkeiten geboten werden um Anforderungen zu definieren, zu verfolgen, und zu verifizieren. Bevor wir jedoch in diesen Themenbereich eintreten, verifizieren wir noch deren Berechtigung und gehen auf dokumentenbasierte Anforderungsverfolgung ein um einen kurzen Vergleich anzustellen.

3.5 Dokumentenbasierte Anforderungsverfolgung:

Dokumente entstehen im Rahmen eines Entwicklungsprozess in einer oft unkontrollierbaren Vielzahl. Im Zusammenhang mit Anforderungen ist deren Verwaltung ein aufwändiges Problem. Für kleine Projekte bieten sie den Vorteil dass sie einerseits leicht mit dem Kunden auszutauschen sind und weiters die Grundlage für manuelle Anforderungsverfolgung anhand von Matrizen oder anderen manuellen Techniken bieten.

Im Gegensatz dazu steht die Tatsache dass der Kostenaufwand bei einer manuellen Untersuchung dieser Dokumente auf Konsistenz extrem zunimmt. Dies liegt in erster Linie daran, dass der zeitliche Aufwand so hoch ist und somit viele Mannstunden aufgebracht werden müssen. Ebenso ist es aufwändig Änderungen in einer Vielzahl von Dokumenten vorzunehmen, da einerseits Verknüpfungen manuell nachvollzogen werden müssen und somit wiederum das Problem der Inkonsistenz auftritt. Die dadurch entstehende Trägheit im Fortschreiten der Veränderung führt zu einer Art Wasserfallprozess selbst wenn der Entwicklungsprozess nicht als solcher definiert ist. Der massivste Punkt unter den Nachteilen welcher in Zusammenhang mit MDA relevant ist besteht darin dass es nicht oder nur schwer möglich ist die dokumentenbasierte Anforderungsverfolgung mit Modellierungs- Tools zu verknüpfen.

3.6 Toolunterstützung für MDA Anforderungsverfolgung

Wie wir im vorigen Abschnitt angeführt haben überwiegen bei rein auf Dokumenten basierender Anforderungsverfolgung die Nachteile. Wie sieht das mit einem Anforderungsmanagement Tool aus? Ein wichtiger Punkt den ich als ersten anführen möchte ist die Tatsache dass mit einem solchen Tool der Bedarf an einem Aufbewahrungsort einem so genannten Repository für Anforderungen absgedeckt wird. Alle Anforderungen werden zentral verwaltet und abgelegt. Weiters werden Verknüpfungen zwischen Anforderungen und den dazugehörigen Dokumenten gezogen, welche im Falle von, zeigen wo und wie Änderungen vorgenommen werden müssen. Ein ebenso wichtiger Vorteil ist, dass es im Gegensatz zur Dokument basierenden Anforderungsverfolgung möglich ist, über Schnittstellen gemeinsam mit Modellierungstools zu arbeiten und damit einen gesamtheitlchen Rahmen für Anforderungsverfolgung zu schaffen. Außerdem kann ein anforderungsbasierter Entwicklungsprozess explizit durch die Verwendung eines Tools unterstützt werden. Als letzter Vorteil sei noch die Tatsache angeführt, dass in den meisten Fällen der Anwendung, bessere Anforderungen in einer früheren Phase des Entwicklungsprozesses entstehen und es somit zu weniger Anforderungsänderungen, Erweiterungen oder Konflikten kommt.

Dem gegenüber steht das Faktum dass durch die zusätzliche Verwendung eines solchen Werkzeugs auch der Aufwand in der Umsetzung steigt, da die Entwickler und Projektbeteiligten ein weiters Tool managen müssen. Dieser Aufwand steht meiner Meinung nach, jedoch in keiner Relation zur manuellen Anforderungsanalyse. Weiters werden Anforderungen zum Teil auch graphisch definiert. Dies ist zwar Grundlegend dank graphischen Modellierungssprachen kein Problem, jedoch können Anforderungsmanagementools oft besser mit Text umgehen als mit umfangreicher graphischen Information es sei denn es wird eine standardisierte Syntax verwendet wie es bei SysML der Fall ist. Insofern sollte ein Anforderungsmanagement Tool SysML unterstützen um hier die gegebenen Möglichkeiten auszuschöpfen.

3.7 Ein Anforderungsmanagement Tool: Reqtify^{Tm}

Reqtify der Firma TNI Software ist ein Tool zur Anforderungsverfolgung welches ins Besondere gemeinsam mit dem von Artisan Software entwickelten Artisan Studio verwendet wird. Jedoch sind Anbindungen an andere Modellierungswerkzeuge und Requirmentsmanagement Tools kein Problem. Die Verwendung von Reqtify ist auf keinen Softwareentwicklungsprozess limitiert.
Dieses Tool bietet die Möglichkeit Anforderungen zu jedem Zeitpunkt des Entwicklungsprozesses nachzuverfolgen und somit können die oben erläuterten Traces permanent gezogen werden bzw. passiert das bis automatisch. Somit ist es möglich eine durchgängige Projektüberwachung in Sachen Anforderungserfüllung zu gewährleisten

Reqtify arbeitet ohne Datenbanken, was die Wartung und Handhabung massiv erleichtert. Die Grundlage für die Analyse bieten anforderungsbeschreibende Dokumente welche in verschiedenen Dokumenten (Excel^{Tm} Listen, Word^{Tm} Dokumente, Framemaker^{Tm}, reiner Text…) spezifiziert sein können. Das heißt besonders im Kontext mit dem vorhergehenden Abschnitt ist es zwar noch immer Dokumentbasierte Anforderungsverfolgung jedoch fließen ebenso Dateien aus den Entwicklungstools wie z.B. Rose^{Tm} Datein, Artisan Studio^{Tm} Dateien, Stood^{Tm}, Sildex^{Tm} etc...) ein und werden mittels Konverter nach Reqtify integriert. Durch die Verknüpfung mit projektbeschreibenden Graphen, Standardtypen und spezifischen Typen welche ebenfalls in Reqtify integriert werden, können durch die im Tool implementierte Logik Tracability Ergebnisse erzielt werden. Diese Ergebnisse können gefiltert oder in einer gesamtheitlichen Form über das graphische User Interface dargestellt werden. Weiters besteht die Möglichkeit automatisiert Tracability Reports zu erstellen welche dann wiederum als Dokumente exportiert werden können. Nachfolgende Graphik beschreibt diesen Prozess:

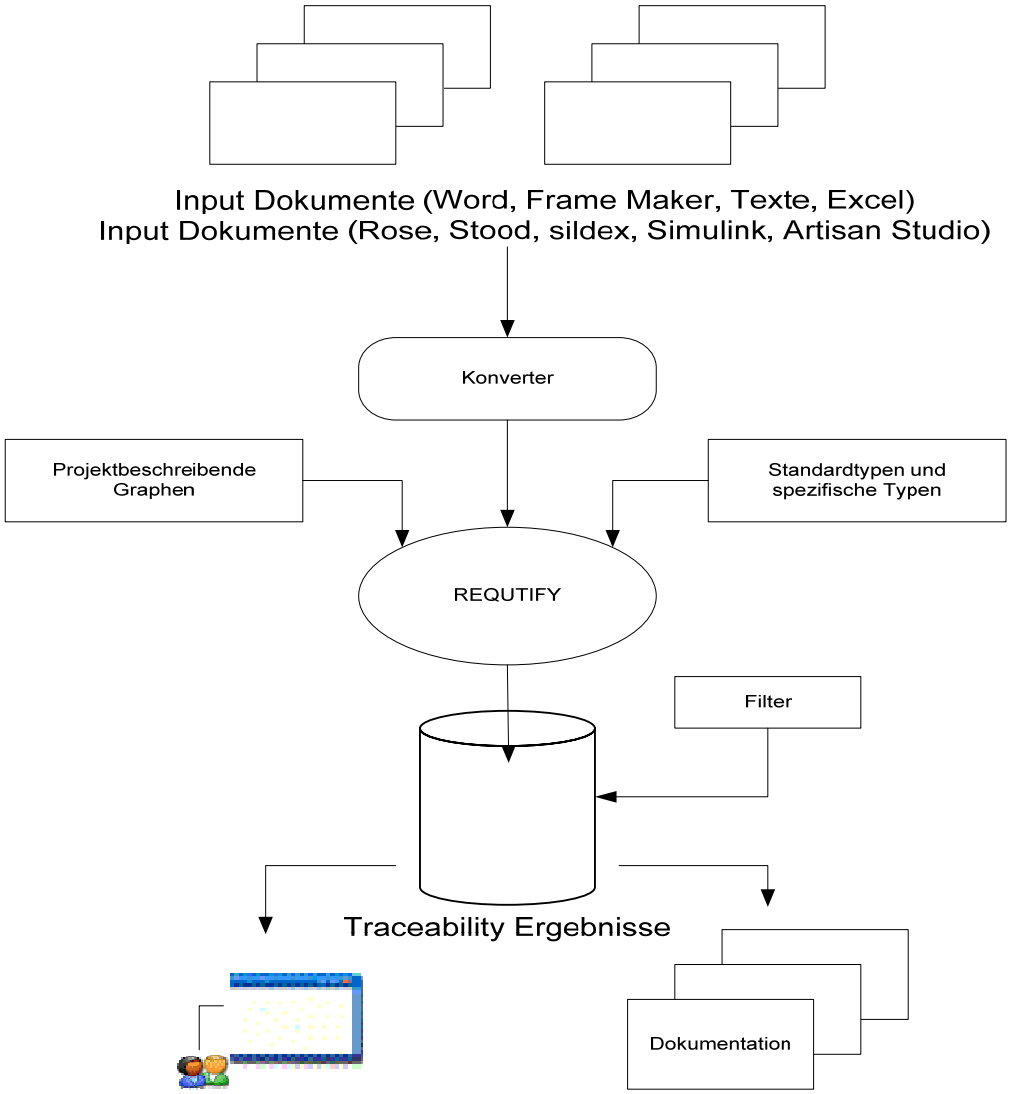


Abbildung 29 Arbeitsweise Reqtify [TNI Software]

Es können verschiedene Dateien in den Anforderungsverfolgungsprozess integriert werden und somit besteht auch die Möglichkeit in bestehenden Projekten im Nachhinein einen Anforderungsverfolgungspozess mit Toolunterstützung zu integrieren. Dies ist ein konkreter Vorteil den dieses Produkt bietet. Der Verzicht auf eine Datenbank kommt durch die Definition einzelner Traceability- Element zustande welche in den Dokumenten auf Grund der Formatierung oder expliziter Tags ausgewiesen werden. Diese Traceability Elemente werden wie folgt definiert:

- Requirement: Definiert durch eine mathematische Formel, einen Text etc...´

- Macro Requirement: Bündelung einzelner Requirements. Änderungen am Macro Requirement wirken sich auf alle in dem Bündel enthaltenen Requirments aus.

- Derived Requirement: Stellt eine Anforderung auf einer darüber liegenden Ebene dar. Das bedeutet es wird durch ein Dokument eingebracht mit welchem Anforderungen abgedeckt werdern.

- Section: Darunter versteht man eine Art Bereich in welchem hierarchische Beziehungen also z.B. Ordnerstrukturen oder Modell Pakete zusammengefasst und miteinander korreliert sind.

- Entity: Mit diesem Element ist die Elementdefinition für eine Analyse welche ausgeführt werden muss gemeint. Dies dient dazu um Anforderungsverfolgungs –Informationen aus dem Inhalt herauszuholen. Damit können Code-Module oder ähnliches gemeint sein.

- Text: Hier ist die Beschreibung eines Elements enthalten

- Link: Dieses Element zeigt keine Coveregeinformationen sondern stellt eine Referenz wie z.B. supported by, issued by... dar

- Referenz: Damit wird die Coverage Beziehung eines Reqirements beschrieben. Diese Beschreibung gilt ebenso für ein Derived oder Macro Requirement.

- Attribute: Diese dienen dazu die Requirements zu vervollständigen und beschreiben diese durch z.B. Kategorisierung.

- Referenzattribut: Mit diesem werden Referenzen beschrieben.

[Reqtify 04]

Ein weiterer Vorteil welcher durch das Tool ermöglicht wird ist die permanente Analyse der Anforderungsabdeckung während des Entwicklungsprozess. Dies ist so zu verstehen dass diese Regeln bereites während der Eingabe oder dem Hinzufügen von Dokumenten zum Projekt bereits verifiziert werden. Dies basiert auf einem implementierten Regelwerk welches die Verknüpfungen überprüft und analysiert. Treten also „nicht definierte Anforderungen“ , „nicht erfüllte Anforderungen“ oder „mehrfach definierte Anforderungen“ auf so wird der Benutzer sofort anhand von Symbolen bzw. Warnungen darauf hingewiesen. Ebenso werden Warnungen bezüglich Formatierungsfehler (falls das Dokument nicht mit Reqtify verarbeitet werden kann) oder Inkonsistenzen im FileSystem ausgegeben.

Es gibt weiters verschiedene Ansichten mit welchen die Traces graphisch veranschaulicht werden und somit die Zusammenhänge zwischen Anforderungen und deren Umsetzungen analysierbar gemachtwerden. Die graphische Variante ist nicht nur benutzerfreundlich sondern bietet auch Möglichkeiten Kunden oder nicht expliziten Mitgliedern des Entwicklungsteams die Möglichkeit zu geben auf Grund von graphischer Auflösung komplexer Problemstellungen wichtige Indikatoren wie Projektfortschritt oder Umsetzung einzelner Anforderungen zu verdeutlichen.

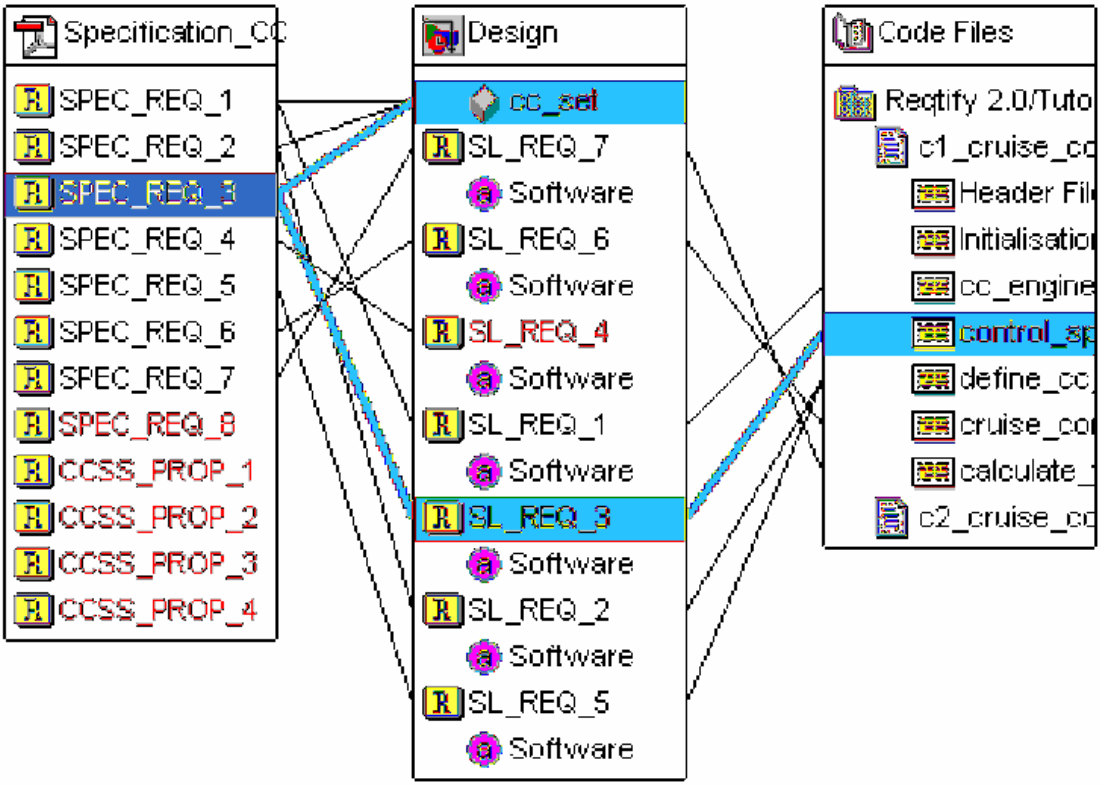


Abbildung 30 Graphische Darstellung von Traces [Reqtify 04]

Graphische Darstellung der Traces

Weitere Ansichten sind der Impact Analysis und der Coverage Mode. Diese 2 Modi bieten mehr Information welche von User selbst verändert werden kann. Damit kann zum Beispiel der Fokus auf einzelne Modelle gelenkt werden, zu welchem dann detaillierte Informationen, in den meisten Fällen durch auf text basierenden Beschreibung zur Verfügung gestellt wird. Realisiert durch drei Frames , einer zur Auswahl des Fokus, einer für forward Coverage, einer für backward Coverage, (vgl. Definition von Anforderungsverfolgung) und durch Prozentuelle Angaben der Anforderungsabdeckung bieten diese Ansichten die idealen Voraussetzungen für Anforderungsverfolgung aus Sicht der im Entwicklungsprozess involvierten Personen.

Ein nicht unwesentlicher Punkt der hier auch erwähnenswert ist, ist die Möglichkeit der schnellen Eskalation von Problemen wenn durch die übersichtliche Darstellung

nicht abgedeckte Anforderungen entdeckt werden. Ebenso in der Behandlung von Change Requests bietet dieses Tool massive Vorteile.

Wir haben ganz zu Beginn dieser Probleme im Bereich der Softwareentwicklung angeführt (siehe Motivation für MDA). Konkret möchte ich hier auf das Dokumentationsproblem eingehen. Reqtify bietet nämlich aufgrund einer Template Unterstützung die Möglichkeit automatisch eine gesamte Projektdokumentation zu erstellen die auf Grund der Einbindung von Modellen und Code wesentlich besser ist als die die man von reinen Entwicklungsumgebungen kennt. Für den Fall das auf Grund der Projektgröße eine umfangreichere oder speziellere Dokumentation notwendig ist besteht die Möglichkeit auf den in Reqtify integrierten Dokument Editor zuzugreifen, mit welchem es ermöglicht wird den Fokus im Rahmen der Dokumentation zu spezifizieren und somit einen detaillierterer Blickwinkel auf die Teilbereiche zu geben. Mittels Drag and Drop Prinzip können Dokumentationsspezifische Teilbereiche in die einzelnen Schriftstücke einfließen und vereinfachen die Erstellung um ein Vielfaches. Natürlich ist im Rahmen dessen auch ein wenig der User und dessen Aufmerksamkeit gefordert, da es leichter dazu kommt relevante Teile nicht zu dokumentieren weil sie in keiner standardisierten abrufbaren Form vorgeschlagen werden. Ebenso umgekehrt, dass es mittels der bereits vorgegebenen Möglichkeiten nicht möglich ist eine Dokumentation zu erstellen. Die Dokumentation bei nachfolgenden Änderungen erfolgt nicht automatisch, jedoch ist der Aufwand um ein vielfaches kleiner als bei einer manuellen Dokumentationserserstellung da Änderungen leichter identifiziert werden können und die notwendigen Maßnahmen initiiert werden können

Standardmäßig sind in Reqtify folgende Dokumentationsarten umgesetzt:

- **Analysis Result:** Hier wird wie oben schon beschrieben der Fokus auf die Projektübersicht gelenkt. Eignet sich sehr gut um den Fortschritt zu dokumentieren oder um zu eskalieren.

- **Project Description:** Wie der Name schon sagt geht es um die Beschreibung des Projekts anhand von Dokumenten und Modellen. Diese Beschreibung ist jedoch nicht sehr detailliert und sagt nicht viel über die Coverage aus.

- Traceability Matrix: Damit werden Zusammenhänge und Beziehungen sowohl einzelner Dokumente und Modelle als auch einzelner Anforderungen in tabelarischer Form zur Verfügung gestellt. Im Rahmen dieser Matrix wird die prozentuelle Coverage der Anforderungen ausgewiesen. **(Im nächsten Punkt wird explizit noch auf Tracability Matritzen eingegangen und der Zusammenhang mit Quality Function Deployment gezogen)**

- **Die Upstream Impact Analysis**: Diese oben ebenso bereites erwähnte Möglichkeit der Dokumentation gibt für explizit betrachtete Anforderungen an durch welche Elemente im Traceability Kontext abgedeckt werden.

- **Die Downstream Impact Analysis:** Im Rahmen dieser Ansicht wird dem Benutzer darüber Aufschluss gegeben welche Element durch die Auswahl abgedeckt werden.

[TNI 03]

In der Nachfolgenden Graphik soll ein kompletter möglicher Reqtify Anforderungsverfolgungsprozess veranschaulicht werden um das Verständnis zu verbessern. Es sind hierbei einzelne Teilausschnitte aus dem Programm selbst integriert um ein Gefühl zu vermitteln welche Funktionalität gegeben ist.

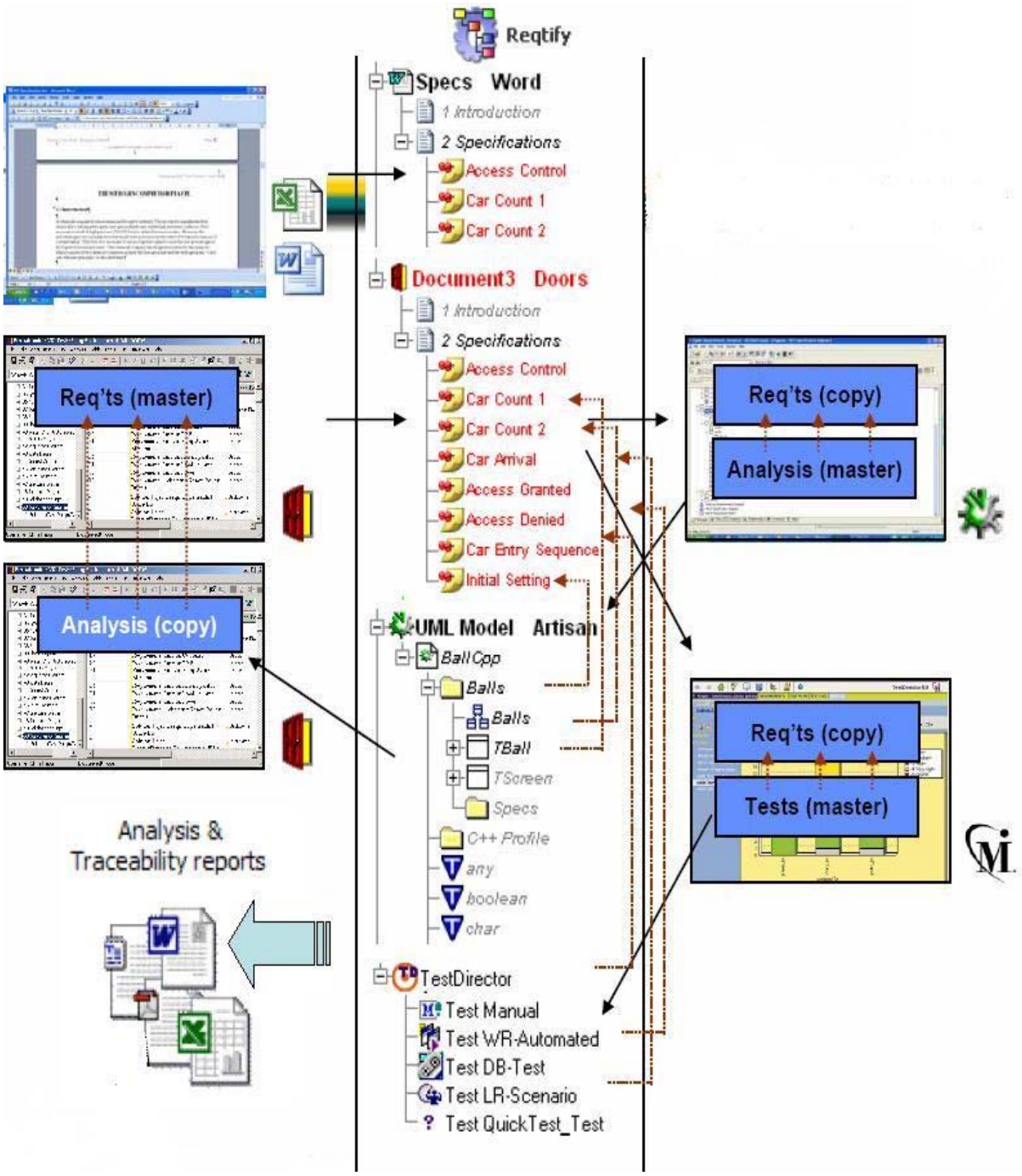


Abbildung 31 Detailansicht Reqtify Anforderungsverfolgung [TNI Software]

Das Beispiel zeigt auf der linken Seite einerseits die Anbindung Reqtifys an die üblichen Office Lösungen und auch die mögliche Verbindung zu Doors als mächtigeres Anforderungsmanagement Tool bzw. Repository. Die Anforderungen können aus beiden Plattformen in Reqtify geladen werden und werden vollständig ohne massiven Aufwand übernommen. Somit sind die nach zu vollziehenden Anforderungen in Reqtify, welches die mittlere Spalte der Graphik repräsentiert.

Die von der linken Seite importierten Anforderungsdokument werdn über Reqtify in die diversen Modelierungstools integriert. Bei den Artisan Studio™ Dateien handelt es sich beispielsweise um konkrete UML bzw. SysML Modelle welche nun in Reqtify mittels der dort abgebildeten Requirement Stereotypen bzw. Requirement Elementen und der darauf basierenden Logik auf die konkreten Anforderungen gemappt werden. Der Test Director™, (rechts unten abgebildet) welcher ebenfalls in Reqtify integrierbar ist bietet nun die Möglichkeit eine Abdeckung der Requirements anhand von bewerteten Testfällen zu garantieren. Dies ist konkret in der Graphik durch die in der mittleren Spalte in orange gezogen Pfeile sichtbar welche hier schematisch die Coverage veranschaulichen sollen.

Schlussendlich besteht dann wie oben bereits erläutert die Möglichkeit den gesamten Verlauf der Anforderungsverfolgung mittels Reqtify erneut nach DOORS™ exportiert werden oder mittels des Dokument Editors als Word™ oder Excel™ Files aus Reqtify transportiert werden um im gegeben Fall konkrete Informationen und Analysen anhand von klar strukturierten Dokumenten zur Verfügung zu stellen.

Reqtify stellt also als relativ kleines Tool eine große Bandbreite an Möglichkeiten der Anknüpfungspunkte zur Verfügung. Schnittstellen zu DOORS™ zum Beispiel ermöglichen die speziellen Funktionen der Verfolgung von Anforderungen auch in größeren Projekten wo bereits eine bestehende Produktpalette benutzt wird zu integrieren und somit das Produktportfolio in Sachen Anforderungsverfolgung zu komplettieren. Wobei jedoch sicherlich Nachholbedarf bei diesem Produkt besteht ist in Sachen Administration bezogen auf Projektkollaboration. Es besteht zum Beispiel nicht die Möglichkeit eines Sicherheitsmechanismus welcher Unbefugte davon abhält sämtliche Projektrelevanten Dokumente einzusehen. Im Besonderen bei der

Entwicklung von Sicherheitsrelevanten Applikationen bei welcher Anforderungsverfolgung eine wesentliche Rolle spielt, ist dies nicht zu verachten. Auch in Sachen Versionsmanagement werden die Änderungen zwar erfasst und erkennbar gemacht jedoch ist dies gerade in Projektteams problematisch da meiner Meinung nach es immer sehr wichtig ist auch zu wissen welche Person eine Änderung vollzogen hat um die Kommunikation in Sachen Projektmanagement nicht zu unterbinden.

Neben der automatisierten Nachverfolgung, welche auf einer Verwendung von explizit modellierten Anforderungen in den einzelnen Modellen basiert, besteht für die Entwickler die Möglichkeit ebenso manuell Traces in die Analyse einzubauen. Durch die Möglichkeit diese Traces ebenso graphisch wie durch drop down Menüs zu erstellen ist hier eine sehr Gute Lösung gegeben. Die einzelnen Dateien oder Sektionen können verknüpft werden und so die Traceability übersichtlich gestaltet werden. Ein großer Vorteil den diese Möglichkeiten bieten liegt auch darin dass erstens die Traceability von sich bereits in der Implementierung befindlichen Projekten ebenso durchgeführt werden kann und zweitens auch bereits fertige Projekte auf Konsistenz der Dokumente und Anforderungen überprüfen kann.

Zusammenfassend kann man sagen das Reqtify ein sehr benutzerfreundliches Tool ist das einen relativ großen Umfang zur Anforderungsverfolgung bietet. Die Anwendbarkeit würde ich auf Grund der fehlenden personenbezogenen Versionsverwaltung bzw. Benutzerverwaltung und dem dadurch erschwerten Mehrpersonenbetrieb eher im Bereich der kleinen bis mittelgroßen Softwareentwicklung ansiedeln bzw. für Teilprojekte. Vorteile liegen sicher in der automatisierten Analyse und den vielen Schnittstellen zu verschiedensten Programmen. Dies ist aus meiner Sicht der Dinge gerade in der heutigen IT Entwicklung ein großer Vorteil da auf Grund der unterschiedlichsten Plattformen und den damit einhergehenden Vielfalt an Entwicklungstools gerade im Bereich der Anforderungsanalyse eine massive Nachfrage für möglichst kompatible Tools besteht. Auf Grund des sehr kompakten Datenumfangs und einer variablen Laufzeitumgebung ist Reqtify ein sehr portables Werkzeug.

Abschließend seien in Bereich der Toolunterstützung auch noch relevante MDA Werkzeuge erwähnt um die Brücke zwischen Anforderungsmanagement und modellbasierender Softwareentwicklung zu schlagen. Es wird hier nur ein kleiner Ausschnitt der am Markt vorhandenen Tools angegeben. Die Reihung dieser birgt keine Präferenzen oder Evaluirung. Viele der Tools wie zum Beispiel der IBM Rational Software Architect oder Rhapsody von der Firma i Logix bieten einerseits integrierte Mechanismen zu Anforderungsverfolgung als wie auch die Möglichkeit Add on Produkte wie auch Requtify eines ist, in die Modellierungssuite zu integrieren.

3.8 Tool Übersicht MDA [MDA Tools 05]

Hersteller	**Artisan**	**I- Logix**	**IBM**	**Borland**
www	www.artisansw.de	www.ilogix.de	www.ibm.com/de/	www.borland.com
Produkt	Real Time Studio 5.0	Rhapsody 6.0	Rational SW Architect	Together Vers 2005
Preis	995 €	k.A	k.A	ab 1500€
Marktbereich	Realtime / Embedded	Realtime / Embedded	Enterprise (J2EE,C++)	Enterprise
Plattform für erstellte Anwendung	Windows Linux JavaVMs RTSj Raven	WindowsXP Linux, Solaris QNX	Windows Linuy Java VM C++	Java JVM 1.3, 1.4, Win32, Linux, MacOS X (C++), .Net (C#, VB.Net)
Frameworks	Realtime Studio Simulation ohne Sourcen	OXF (Object Execution Framework) mit Sourcen	optionale Runtime Umgebungen für J2EE ohne Sourcen	nein
MDA Tool integriert in eine _IDE	nein	eigene IDE, Eclipse geplant	Integration in Eclipse	nein
UML Version	UML 2.0	UML 2.0	UML 2.0	UML 1.5, 2.0
XMI Support	XMI. 1x	XMI. 1x	XMI. 1x	XMI. 1x
UML Profile	SPT, SysML	SPT, Testing Profile SysML	J2EE / EJB	Erikson Penker Business
weitere Diagrammarten	ER für DB, Concurrency, System Architecture	SysML, einbindung UML fremder Diagramme als Bitmaps	ER für DB, Website Navigation	ER für DB , diverse andere
Aufteilung PIM und PSM	über Transformationsregeln	PIM als UML	über Transformationsregeln	1:1 Transformation und anschließende Code Generierung über Patterns
Codegenerator Zielsprache	C, C++, Java, RT Java, Ada95	C, C++, Java, Ada	C++, Java	Java, C++, C#, VB.Net, CORBA IDL
Unterstützung von Forward, Reverse, Roundtrip	Ja	C: Forward und Reverse C++, Java: Forward Reverse und Roundtrip	Ja	Ja
GUI Realisierbar	nicht out of the box	HTML basierender Oberflächengenerator	GUI Builder für Swing, AWT, SWT	nicht out of the box

3.9 Quality Function Deployment

In diesem Kapitel wurde bis jetzt auf technologiespezifischen Möglichkeiten eingegangen, wie Anforderungsverfolgung in einem modellbasierten Entwicklungsprozess umgesetzt werden kann. In den Kapiteln zuvor wurden die einzelnen Konzepte der Anforderungsverfolgung, als wie auch der MDA dargestellt und deren jeweilige Berechtigung erläutert. Schlussendlich geht es im Rahmen von Anforderungsverfolgung darum, schneller mit besserer Qualität zu entwickeln. Dieser Ansatz, welcher in der Softwareentwicklung insbesondere die Qualitätssicherung und Kontrolle umfasst ist noch nicht sehr alt und in den letzen Jahren insbesondere durch die Entwicklung von Standards, mit welchen die Qualität einer Softwareentwicklung definiert werden können, unterstützt worden.

Zeitlich gesehen kam der ersten Artikel bezüglich Qualitätsentwicklung in den 60er Jahren auf. Wenig später darauf brachte die Mitsubishi Heavy Industries Kobe Schiffwerft erste Qualitätstabellen heraus. Diese bildeten die Grundlage für das 1978 erschienene Buch: Quality Function Deployment (QFD): An Approach to Total Quality Control. Die aktuellen Mechanismen des Quality Function Deployments, Kundenanforderungen in den Softwareentwicklungsprozess zu integrieren und diesen regelrecht darauf aufzubauen basieren auf den damals definierten Errungenschaften der Schwerindustrie.

Ich möchte einführend zu diesem Punkt einen Überblick über die Thematik des QFD geben, um dieses dann in weiterer Folge in den MDA und RT Ansatz einfließen zu lassen, mit dem Ziel dadurch Anforderungsverfolgung in eine modellbasierten Softwareentwicklung zu unterstützen.

Mittels QFD werden Methoden geboten, welche aus der sich in den 80er und frühen 90er Jahren, schnell entwickelnden Welt der Industrie kommen. Die Gegebenheiten individuelle Kundenanforderungen in sich schnell ändernden Industriestrukturen überzuführen und dabei qualitativ hochwertige Produkte, mit technischen Innovationen zu entwickeln, ist die Basis auf welcher QFD entstand. Diese Basis

begründete in den 80er Jahren die Existenz vieler Unternehmen. Man kann diesen Qualitätssicherungsansatz auch als die Entwicklung von Entwurfsqualität, welche sich an den Anforderungen von Kunden orientiert, beschreiben. Der QFD Ansatz beschreibt die Definition von Entwurfsanforderungen welche aus Kundenanforderungen entstehen, wichtige Entwurfsziele widerspiegeln und mittels Qualitätssicherungspunkten überprüft werden. Somit entsteht eine Grundlage welche wiederum in die Produktion einfließt. Diese Grundlage kann von Unabhängigen revievt und kontrolliert werden.

Einerseits besteht die Möglichkeit einen Qualitätssicherungsansatz aus der Sicht der Analyse zu definieren. Dabei produziert man ein Produkt oder kopiert gegebenenfalls ein Produkt und versucht dann, durch das Festhalten von Reklamationen entsprechende Verbesserungsmaßnahmen zu treffen, um die Qualität zu steigern. Kundenreklamationen kann man nach dem QFD, in so genannten Ursache – Wirkungs- Tabellen festhalten. Führt man diesen Prozess iterativ durch, kommt man zügig zu Qualitätsverbesserungen. Möchte man jedoch diesen Ansatz umdrehen, also die Qualitätsanforderungen in den Entwicklungsprozess mit einbeziehen und nicht die Qualität eines bestehenden Produkts verbessern oder überhaupt eine gewisses Level an Qualität erreichen, so muss man auf Basis der Anforderungen einen Qualitätsplan entwickeln und versuchen, eine etablierte Entwurfsqualität in den darauf folgenden Produktionsprozess und dessen Schritte einfließen zu lassen.

Der im vorigen Absatz beschriebene Ansatz der Produktentwicklung lässt erste Parallelen zum Softwareentwicklungsprozess der modellbasierten Entwicklung erkennen. Ein nach einem Prozess entwickeltes Produkt welches auf Subsystemen basiert wird durch die Qualität der Subsysteme beschrieben. Diese werden wiederum durch Qualität der einzelnen Elemente des Subsystems definiert welche über im Prozess etablierte Kontrollpunkte sichergestellt werden. Die Basis für das einzelne Element der Subsysteme ist definiert durch die Kundenanforderungen, welche in Qualitätsmerkmale umgesetzt werden müssen. Es entsteht also ein Netzwerk von Beziehungen einzelner Elementen in einzelnen Subsystemen welche das Produkt definieren. Die notwendige Beziehung zwischen Forderungen und Merkmalen

welche hier entsteht kann im modellbasierten Softwareentwicklungsprozess als Anforderungsverfolgung gesehen werden.

Die Übersetzung von Kundenwünschen, welche die Qualität eines Produkts beschreiben, in Anforderungen die für die Entwicklung verwendbar sind, ist notwendig um in weiterer Folge Tabellen von Kundenanforderungen und Qualitätstabellen zu definieren, auf welche ich im nächsten Abschnitt eingehen werde. Zuvor möchte ich jedoch noch ein Beispiel geben, welches die Transformation von Kundenwünschen zu Qualitätsmerkmalen für den Entwurf veranschaulicht. Es wird dazu das Beispiel des Autoverleihs via Internet, welches bereits im Rahmen der SysML Anwendung veranschaulicht wurde, herangezogen um die Vorgehensweise darzustellen:

Formulierung des Kunden	**Umformulierte Daten**	**Maßnahmen**
Ich möchte wissen dass ich das Auto zum angegeben Zeitpunkte benützen kann	Der Kunde ist darüber informiert dass die Buchung nach seinen Angaben erfolgt ist	Eine Bestätigung für den Kunden muss per E-Mail oder per Post ausgegeben werden.
Die Bestellung muss zügig durchgeführt werden können	Die Applikation darf keine langen Ladevorgänge aufweisen	Definiere maximale und minimale Response Time der Applikation

QFD Tabelle, Umformulierung von Kundenwünschen in Maßnahmen
Adaptiert aus [QFD Yoji Akao]

Die Definition der Maßnahmen ergibt im Rahmen der Softwareentwicklung also die Definition der umzusetzenden Elemente. Um nun aus diesen Anforderungen eine Tabelle der Kundenanforderungen zu erstellen muss man diese in primäre sekundäre und tertiäre Kundenanforderungen kategorisieren:

Tabelle der Kundenanforderungen / Erreichen von Kundenzufriedenheit		
Primär	Sekundär	Tertiär
1 Information des Kunden	11 Kommunikationskanäle	111 Info per Mail
		112 Info per Post (Ausdruck)
2 Leichte Bedienbarkeit	21 Graphisches Userinterface	211 Intuitiv zu bedienen
		212 Übersichtlich dargestellt
	22 Shortcuts am Keyboard z.B. Strg C für Copy	221 Standard Shortcuts sind implementiert
		222 Es können individuelle shortcuts erstellt werden

In weiterer Folge müssen die Elemente der Qualitätstabelle erstellt werden. Hierzu werden aus den tertiären Kundenqualitätsanforderungen Qualitätselemente abgeleitet. Die Qualitätselemente werden ebenso in primär, sekundär und tertiär unterteilt. Bei tertiären Qualitätsmerkmalen sollten nur noch messbare Größen spezifiziert sein. Dies ist jedoch insbesondere in der Softwareentwicklung äußerst problematisch, da man Messbarkeit von beispielsweise Usability nur schwer durchführen kann.

Tabelle der Qualitätselemente								
Qualitätselemente	primär	Kommunikation		Usability				
	sekundär	Kommunikationskanäle		Menüführung			Mouse Funktionen	
	tertiär	Elektronisch	Physisch	Anwender freundlichkeit	Schriftgröße	Übersicht	Reaktionszeit	Mouse Sensibilität

Nun werden die beiden Tabellen zueinander In Relation gesetzt und um die Korrelation der einzelnen Elemente untereinander erweitert. Es entsteht eine zweidimensionale Matrix:

			Qualitätselemente	Primär	Kommunikation		Usability				
				Sekundär	Kommunikations-Kanäle		Menüführung			Mouse Funktion	
Kundenanforderungen				Tertiär	Elektronisch	Physisch	Anwender freundlichkeit	Schriftgröße	Übersicht	Reaktionszeit	Mouse Sensibilität
Primär	Sekundär	Tertiär									
1 Information des Kunden	11 Kommunikationskanäle	111 Info per Mail			◎						
		112 Info per Post (Ausdruck)				◎					
2 Leichte Bedienbarkeit	21 Graphisches Userinterface	211 Intuitiv zu bedienen					◎	○	◎		
		212 Übersichtlich dargestellt					◎	◎	◎		
	22 Mouse Roll -Over	221 Anzeige von Hilfe bei Berühren eines Buttons mit d. Mouse Zeiger					○		△	◎	◎

Tabelle: Korrelation von Kundenanforderungen und Qualitätstabelle – Adaptiert aus: [QFD Yoji Akao]

Die Korrelation wird durch Symbole dargestellt wobei zu bemerken ist dass ein leeres Feld keine Korrelation darstellt. Das bedeutet dass eigentlich 4 verschiedene Möglichkeiten der Korrelation bestehen:

△ bezeichnet eine schwache Korrelation

○ bedeutet mittlere Korrelation

◎ steht für eine starke Korrelation

Das Zusammenführen von Qualitätstabelle der Kundenanforderungen und die Tabelle der Qualitätselemente ergibt ebenso eine Qualitätstabelle. Diese so entstehende Tabelle ist die Basis auf welcher der Prozess des QFD aufbaut.

„So wie QFD verstanden und umgesetzt wird, ist die Qualitätstabelle ein graphisches Hilfsmittel, mit dem (1) die Strukturen der wahren bzw. endgültigen Qualitätseigenschaften, wie sie entsprechend den wortwörtlichen Aussagen der Kunden gefordert werden systematisch analysiert , (2) die Beziehungen zwischen diesen geforderten Qualitätseigenschaften und bestimmten Qualitätsmerkmalen aufzeigt und (3) die Kundenanforderungen in ergänzende Merkmale umgeformt werden können. Schließlich kann ein Qualitätsentwurf entwickelt werden. “
[QFD Yoji Akao Seite 18 / 2]

Die Idee Kundenanforderungen in das Design eines Produktes zu integrieren, wird in heutigen Softwareentwicklungsprozessen mehr als je zuvor gelebt, jedoch entfernte man sich zunehmend von der matrixbezogenen Vorgehensweise. Die MDA bzw. der allgemeine Trend immer mehr Modelle in den Entwicklungsprozess zu integrieren, oder eigentlich nicht ohne Modelle zu entwickeln forderte Technologien wie SysML mit denen es möglich wird die explizit gegeben Kundenanforderungen in modellbasierter Form im Entwicklungsprozess zu definieren um deren Umsetzung zu verfolgen.

Mit welchem Problem wir uns auch, aber insbesondere im Rahmen der MDA befassen müssen, sind unterschiedliche Subsysteme auf teils verschiedenen

Plattformen und Spezifikationen. Vor allem auch in diesem Bereich bot das QFD einen zum Zeitpunkt seines Entstehens revolutionären Ansatz. Die Einbindung von Subsystemen und deren Qualität ist ein wichtiger Faktor um die Gesamtqualität des Endproduktes zu erreichen. Dazu ist es laut Akao notwendig, die Qualitätsmerkmale der einzelnen Baugruppen (im Fall der SWE Subsysteme) aufzuspalten. Es werden dazu die Subsysteme in ihre Einzelsysteme aufgeteilt. Dies geschieht in Tabellenform. Die Tabelle eines Subsystems beinhaltet die „Subsubsysteme". Weiters ist die Funktion der einzelnen Subsubsysteme als wie auch der Subsysteme festzuhalten. Danach kann man für jedes Subsystem und die darin enthaltenen Systeme Funktionsmerkmale F , Sicherheitsmerkmale S und Qualitätsmerkmale festlegen. Mittels der Funktions- und Sicherheitsmerkmale werden produktionsrelevante Details veranschaulicht welche in der Umsetzung äußerst hilfreich sein können. Für Funktionsmerkmale können gewisse Toleranzgrenzen gesetzt werden – jedoch muss die Funktion eindeutig zwischen diesen Grenzen liegen um die Funktion des gesamten Produktes nicht zu gefährden. Bei Sicherheitsmerkmalen einen Toleranzbereich einzuführen wird nur dann durchgeführt wenn es zum Beispiel bei der Entwicklung eines Mobiltelefons um die gesundheitlichen Richtlinien bez. der Strahlenwerte geht welche einzuhalten sind. Bei der Entwicklung von Software können Standards dafür herangezogen werden, jedoch für definierte Sicherheitsrichtlinien wie SSL Verschlüsselung können keine Toleranzgrenzen eingeführt werden. Schließlich ist es noch notwendig Checkpoints für die Überprüfung der Subsysteme und der Subsubsysteme festzulegen. Nachfolgende Tabelle soll die Entwicklung von Subsystemen mittels QFD Tabellen veranschaulichen:

Autoverleih						
Pay ment	Fahrzeugverwaltung					**Prim**
	Administration			Fahrzeugauswahl		**Sek.**
	+	-	update	Order	Storno	**Tert.**
				◎	△	
	◎	◎	◎	◎	◎	
◎						
◎						
				Ermöglicht es ein Fahrzeug zu bestellen	Ermöglicht eine Fahrzeugbestellung zu stornieren	Funktion der Einzelteile
				Response Time GUI Anbindung	Response Time GUI Anbindung	Merkmale der Einzelteile
				Username / PW Kreditkarten# SSL encoded	Username und Passwort required	Security

Tabelle der Qualitätsmerkmale	
Sekundär	**Primär**
Kommunikationskanal	Kommunikation
Graphical User Interface	Usability
SSL Verschlüsselung	Sicherheit
Signatur	

Es werden einzelne Qualitätsanforderungen mit Subanforderungen verknüpft um diese in weiterer Folge dort zu implementieren. Ebenso wie in der Qualitätstabelle besteht bei Entwicklung der Subsysteme ebenso die Möglichkeit die einzelnen Tabellen miteinander in Korrelation zu setzen.
Siehe Abb. 42

Mit der Erstellung der Subsysteme und der Qualitätsanforderungen für diese wird der erstmalige Entwurf detaillierter dargestellt – auch in diesem Bereich lassen sich Parallelen zur MDA erkennen. Man beginnt den Geschäftsprozess oder besser die entscheidenden Merkmale des Prozesses zu verfeinern, in funktionale Bündel zu fassen, um dann diese durch den Produktionsprozess zu implementieren.

Die hier angeführten Beispiele sind eine Adaption eines sehr frühen industriellen Ansatzes zum QFD aus den 80er Jahren – Ich habe ihn dennoch hier angeführt und adaptiert um einerseits die Möglichkeiten zu veranschaulichen mit einer Industriemethode Softwareentwicklung darstellen zu können und andererseits möchte ich diesen frühen Ansatz auch als Ausgangspunkt für die Integration von QFD in MDA Prozesse zur Anforderungsverfolgung nutzen.

Das QFD sieht eine eine systematische Analyse von Kundenanforderungen vor.
Nach dem QFD erfolgt nach der Produktplanung die Teileplanung, um danach abgeleitete Prozesse zu definieren und weiters die Produktionsplanung abzuschließen.
Nachfolgende Graphik zeigt einen Überblick über die Produktentwicklung mit QFD:

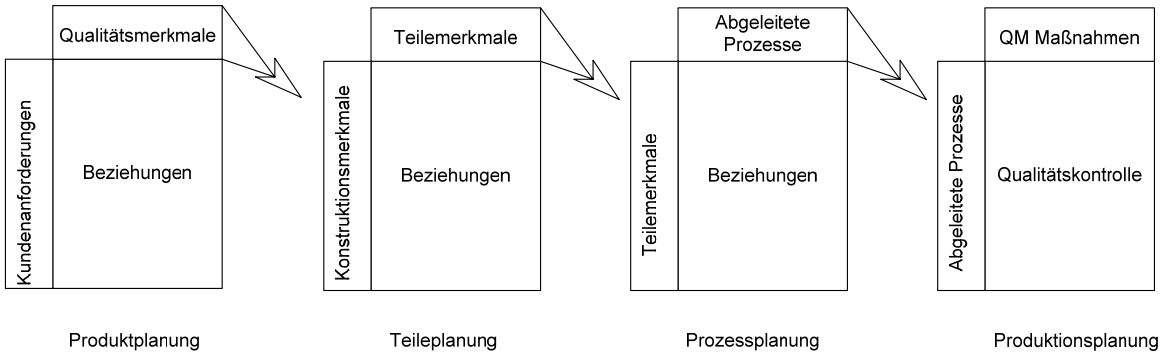


Abbildung 32 Industrielles QFD [Adaptiert aus Crow 96]

Worauf ich nun im Weiteren eingehen möchte, ist einerseits wie man QFD konkret in der Softwareentwicklung einsetzen kann und wie man diesen Ansatz adaptieren muss um gemeinsam mit MDA Anforderungsverfolgung in tabellarischer Form

beschreiben zu können. Weiters soll beschrieben werden ob es eine sinnvolle Integration von Tabellen in den MDA Entwicklungsprozess geben kann.

Die Idee Tabellen in der Anforderungsverfolgung einzusetzen wird hauptsächlich dazu verwendet um Testfälle und Anforderungen in Relation zu setzen. Die Zwischenschritte mit welchen aus Anforderungen Code erzeugt wird, werden dabei jedoch außer Acht gelassen. In der modellbasierten Softwareentwicklung stellt sich dies als Problem heraus, da insbesondere zur Wiederverwendbarkeit und zur Beibehaltung der Modularität die Änderungen verfolgt werden müssen. Die entscheidende Frage ist wie man Modelle, Transformationen, unterschiedliche Technologien und den Prozess in Tabellen integrieren kann dass eine durchgängige Anforderungsverfolgung zur qualitätsfördernden Entwicklung eingesetzt werden kann.

Nachfolgende Graphik beschreibt die Adaption des QFD Prozess für Softwareentwicklung:

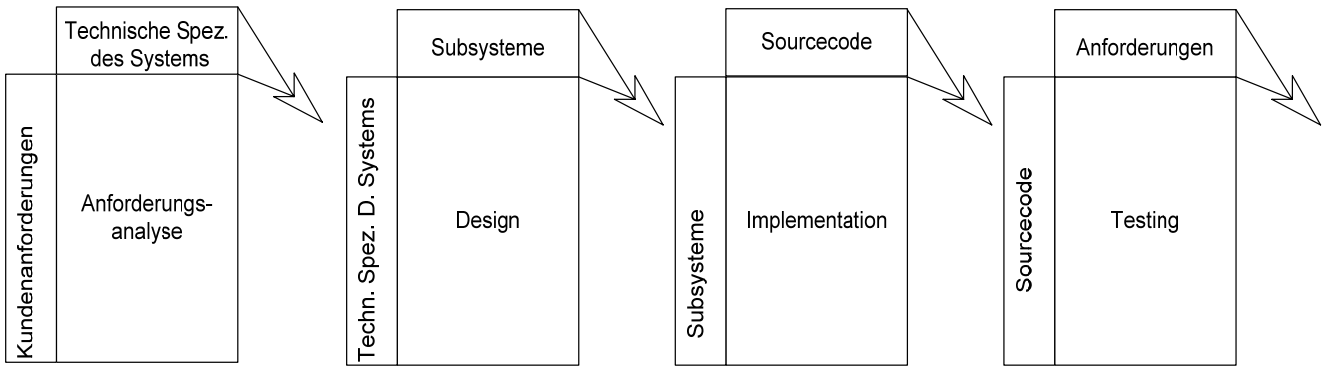


Abbildung 33 SoftwareQFD Framework [Adaptiert nach [Liu. X.F 01]]

Welche Anforderungen müssen im Zusammenhang mit MDA an die Tabellen gestellt werden um Anforderungen zu verfolgen und wie kann man den Prozess für eine MDA spezifische Implementation von Software Quality Function Deployment (SQFD) adaptieren kann: Ein Ansatz dafür wäre auf folgende Fragen genauer einzugehen und nach diesen einen Lösungsansatz zur Anforderungsverfolgung in der MDA zu definieren:

- Von wem oder welchen Dokument kommt die Anforderung?
- Wie wichtig ist die Anforderung? (Anzahl der starken Korrelationen)
- Wo ist die Anforderung dokumentiert?
- Wo ist die Anforderung im Entwurf berücksichtigt?
- Wo wirken sich Änderungen von Anforderungen im Entwurf aus?
- Welche Transformationen werden auf den Entwurf angewandt um die Anforderung zu realisieren?
- Wo ist die Anforderung umgesetzt?
- Welche Technologie wird eingesetzt um eine Anforderung umzusetzen?
- Welche Ergebnisse wurden bei der Überprüfung der Umsetzung erzielt?

Daraus resultieren folgende mögliche Prozessschritte:

1. Aus Geschäftsprozessen müssen Anforderungen identifiziert werden um sie verfolgen zu können. Die Überleitung aus Geschäftsprozess in Anforderungen und in eines oder mehrere Modelle muss aus einer Tabelle ersichtlich sein um die weiteren Schritte verfolgen zu können. Ebenso muss die Definition der Anforderung eindeutig sein. Abhängigkeiten der einzelnen Tabellenelemente werden mittels der Notation aus dem QFD dargestellt.

2. Die Anforderungen von Kunden **und dem** Geschäftsprozess müssen mit den Modellen verlinkt werden. Die Elemente der Modelle oder deren Gesamtheit müssen auch den Anforderungen zugewiesen werden können. Gleichzeitig müssen auch Relationen von Modellen untereinander entstehen können wie z. B. ein System Use Case Diagramm und die damit in Verbindung stehenden

plattformspezifischen Implementierungs- Modelle, welche den Anwendungsfall spezifizieren.

3. Die Elemente der Modelle und deren Verknüpfungen müssen aus den Modellen im Code nachverfolgbar sein. So wie bei beim Reverse Engineering wo aus Source Code Modelle entstehen muss hier die Möglichkeit bestehen diese Elemente verfolgen zu können. Tabellen können hier nur bis zu einem gewissen Grad sinnvoll sein da sie nur Relationen zu Code Files jedoch nicht unbedingt die Relation zu den darin enthaltenen Klassen und Methoden aufzeigen können. Diesen Detaillierungsgrad in einer Tabelle auszuweisen würde wahrscheinlich zu so komplexen Tabellen führen dass die eigentliche übersichtliche Darstellung in dieser Form darunter leiden würde.

4. Die Testergebnisse müssen in Relation mit den Anforderungen überprüft werden.

3.10 Die Verwendung von SQFD Tabellen zur Anforderungsverfolgung in der MDA:

3.10.1 Integration des Geschäftsprozesses:

Dieser Abschnitt beschreibt die Einführung einer Tabelle, welche den Geschäftsprozess der abgebildet werden soll, in Relation mit Kundenanforderungen setzt. Damit wird sichergestellt, dass die Anforderungen auch begründet sind. Werden Änderungen am Geschäftsprozess vorgenommen, wirken sich diese auf die Anforderungen aus, was sich wiederum auf die Modelle, die die Anwenderanforderungen beschreiben einen starken Einfluss ausübt. Somit muss man diese in Relation stellen um die Änderungen verfolgen zu können. Die damit entstehende Analyse des Geschäftsprozesses und die Ableitung von Kundenanforderungen daraus, kann optional gemacht werden, wenn ein solcher Prozess existiert. In jeder Hinsicht ist es aber auch sinnvoll sich über die momentane

Funktionalität des Prozesses bewusst zu werden und danach die Anforderungen an das zu entwickelnde System zu definieren.

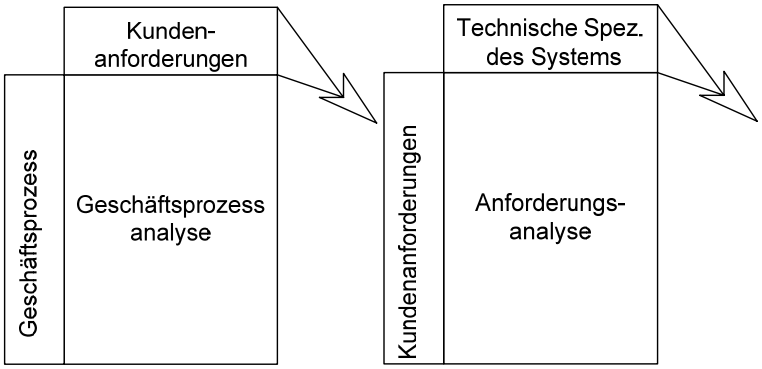


Abbildung 34 Erweiterung des Software QFD Framework

Es bietet sich im Rahmen der Einführung einer Geschäftsprozessanalyse auch die Integration von Modellen in den Prozess des SQFD an. Die Problematik besteht jedoch in der Definition der Korrelation von Inhalten des Modells und der grundsätzliche Beschreibung. Es würde nämlich interessant sein, einen direkten Link zwischen Geschäftsprozess bzw. den einzelnen Element eines Geschäftsprozess und deren modellierten Beschreibungen zu etablieren, um diese mit der gewünschten Kundenanforderung zu verknüpfen. Somit können Relationstabellen von einzelnen Geschäftsprozessschritten und deren Modellen, gemeinsam mit der daraus resultierenden Kundenanforderung etabliert werden. Folgendes Beispiel soll diesen Ansatz veranschaulichen:

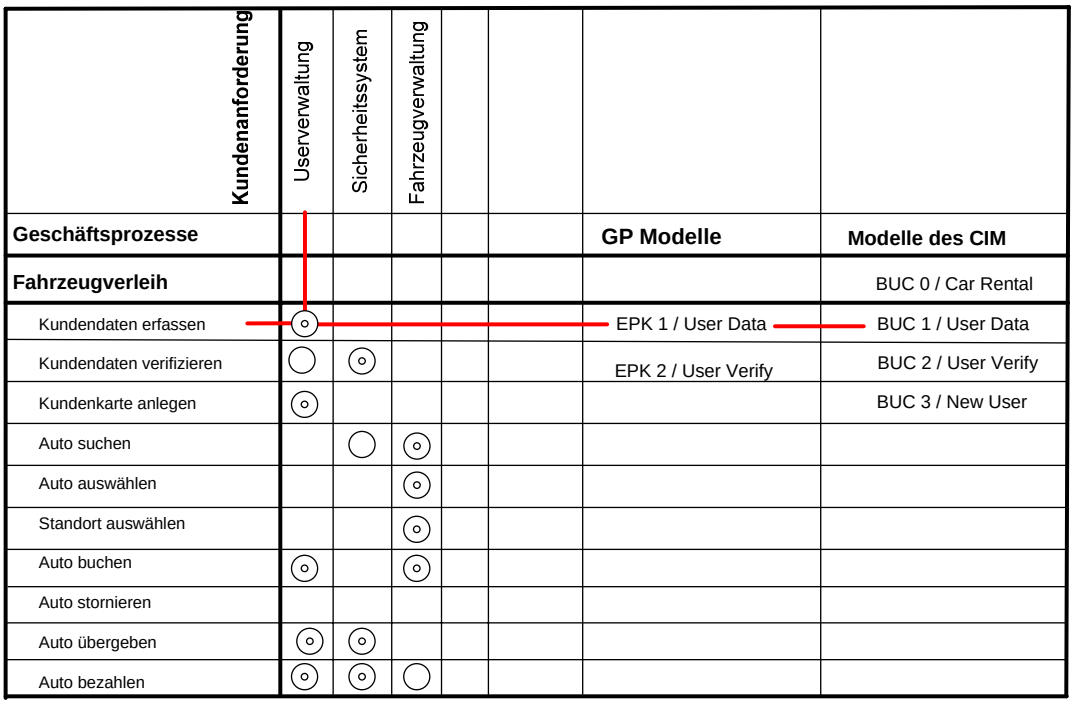


Abbildung 35 Geschäftsprozessanalyse I

Aus der Definition des Geschäftsprozess bzw. der Auflistung seiner Funktionalitäten entstehen Kundenanforderungen, welche durch Modelle veranschaulicht werden. Einerseits gibt es in der Spalte rechts außen der Graphik Geschäftsprozessmodelle und erste Computer Independent Modelle CIM aus der MDA Entwicklung. Das Business Use Case BUC 0 / Car Rental wird durch kein Geschäftsprozessmodell beschrieben. Sehrwohl werden aber BUC 1 / User Data und BUC 2 / User Verify durch beispielsweise Ereignisgesteuerte Prozessketten Diagramme, EPK 1 / User Data und EPK 2 / User Verify, auf Geschäftsprozessebene beschrieben. Es entsteht also eine Matrix mit welcher 2 Tabellen kombiniert werden und Relationen unter den Inhalten der Tabellen gezogen werden. Weiters können aus dem Geschäftsprozess erste Anforderungen abgeleitet werden und gleichzeitig bereits existierende Modelle die den Geschäftsprozess beschreiben, mit daraus entstehenden Modellen in Korrelation gebracht werden

Die Möglichkeit dass ein Geschäftsprozess eine Relation zu mehreren Modellen aufweist, besteht ebenfalls da man in die Felder der Modelle mehrere von diesen eintragen kann. Somit besteht die Möglichkeit 1: N Beziehungen darzustellen und diese gleichzeitig mit Geschäftsprozess und Anforderungen zu verknüpfen. Jedoch

können dann keine expliziten Aussagen über die Relation unter den Modellen gemacht werden. (vgl. Abb. 38)

Im Rahmen der Geschäftsprozessanalyse können auch die Komponenten des IT Systems beispielsweise in tabellarischer Form angeführt werden und somit die Kundenanforderungen detailliert werden. Somit wird ein Überblick über die Vorhandene IT Landschaft gegeben und diese kann in Relation mit den Anforderungen gesetzt werden. Dies ist zum Beispiel sinnvoll wenn eine Applikation einen Webserver benötigt jedoch keiner als solcher konfiguriert ist. Diese Tatsache fließt in die Anforderungen ein und muss somit auch in diversen Modellen dargestellt werden, um diese Komponente in der Architektur zu berücksichtigen. Natürlich wirken Tätigkeiten wie das konfigurieren eines Webservers auch auf die Kosten in einem Projekt aus und fließt somit auch in die Preisgestaltung bzw. in die Aufwandsschätzung eines Projekts ein. Hinzu kommt dass man Risiken im Rahmen einer so detaillierten Geschäftsprozessanalyse erkennen und somit auch in weiterer Folge entschärfen kann.

	Kundenanforderung		Userverwaltung	Sicherheitsaspekte	Fahrzeugverwaltung		Hardwareupdate	Betriebssystem	Leichte Bedienbarkeit	Trainingsmodus		
IT Landschaft											**Definiert in**	**Modelle des CIM**
Webserver			◎	◎	◎		◎				AssetList 1	Enterprise IT Model
Datenbankserver			◎	◎	◎		◎	◎			Asset List 1	Enterprise IT Model
Clients			◎	○			○	○			Asset List 2	Enterprise IT Model
Organisation Personen												
Kunden			◎	◎	◎				◎	△	Kundenkartei	BUC 1 / BUC 2 / BUC 3 ...
Administratoren			◎	○					○	◎	Mitarbeiter Liste	Organisationsmodell BUC 1 / BUC 2 / BUC3
Mitarbeiter			◎	◎	◎				○	◎	Mitarbeiter Liste	Organisationsmodell BUC 3 / New User

Abbildung 36 Geschäftsprozessanalyse II

Die Graphik zeigt, dass durch das Hinzufügen von beispielsweise im Geschäftsprozess beteiligten Personen es leicht zu neuen Anforderungen kommen kann. In diesem Fall wurden die Anforderungen „Leicht zu Bedienen“ da diese für den Kunden sehr wichtig erscheint und „Trainingsmodus“ für Mitarbeiter hinzugefügt. Was ebenso mit dieser Tabelle veranschaulicht wird sind erste Verfolgungen von Elementen in Modelle. Der Mitarbeiter beispielsweise welcher einerseits starke Relationen zur Userverwaltung, zur Fahrzeugverwaltung und zu Sicherheitsaspekten aufweist und laut der Geschäftsprozessbeschreibung in Mitarbeiterlisten definiert ist, wird einerseits im Organisationsmodell aus dem MDA Prozess abgebildet und andererseits in einem oder mehreren Business Use Cases auf CIM Ebene. Es wird also eine Relation vom Mitarbeiter zu den Anforderungen und zu einzelnen oder mehreren Modellen in welchen er Einfluss findet hergestellt.

Sofern in der Entwicklung ein Anforderungsmanagement Tool verwendet wird macht es natürlich Sinn die dort gespeicherten Anforderungen, sei es in Modellform oder in einfacher auf Text basierender Beschreibung, auch in diese Tabellen zu referieren bzw. auch die Tabellen in ein solches Tool zu integrieren, um komplette Anforderungsbäume zu modellieren und in tabellarischer oder hierarchischer Form darzustellen und somit die Brücke zwischen QFD, Anforderungsverfolgung und Modellen zu etablieren.

3.10.2 Die Anforderungsanalyse und technische Produktspezifikationen:

Sinn und Zweck der Geschäftsprozessanalyse ist es wenn ein System neu entwickelt wird oder ein bestehendes System verbessert wird die Frage des „Wie wird es zur Zeit gemacht?“ in erste „Was muss gemacht werden?“ umzusetzen. Die aus dem Geschäftsprozess resultierenden Anforderungen sollen dann gemeinsam mit dem Kunden besprochen und in der Anforderungsanalyse ergänzt werden. Im Rahmen eines solchen Gesprächs besteht dann die Möglichkeit weiter Kundenwünsche hinzuzufügen. Insofern wäre es sinnvoll neben der Geschäftsprozessanalyse gesondert Anforderungen des Kunden zu erheben und diese dann denen aus dem Prozess gegenüberzustellen. In den meisten Fällen werden es die gleichen Anforderungen sein – jedoch kann man dadurch die Anforderungsanalyse qualitativ

hochwertiger gestallten und wie wir bereits diskutiert haben, resultieren aus besser definierten Anforderungen bessere und einfacher zu bewältigende Softwareprojekte. Im Rahmen der Anforderungsanalyse lassen sich weiters die Zusammenhänge der Anforderungen mit den Technischen Komponenten eines Systems definieren. Diese Komponenten sollten wie eingangs beschrieben eindeutig messbar sein.

	Techn. Produktspezifikation	Kapazität Anzahl Fahrzeuge	Kapazität Anzahl User	DB Abfrage kleiner 5sek. Responsetime	Schnittstelle zu Userverwaltung	Schnittstelle zu Payment	SSL Verschlüsselung	Verwendung von XML Files		
Kundenanforderungen									**Modelle des CIM**	**Modelle des PIM**
Userverwaltung			◎	◎		◎	◎	◎	BUC 1 / BUC 2 / BUC 3 …	System Use Case SUC1
Sicherheitssystem			◎	○	○	○	◎	△	Enterprise IT Modell / BUC1	System Use Case SUC2
Fahrzeugverwaltung		◎	○	◎	◎	◎	○		BUC 2 / User Verify BUC 3 / New CAR	System Use Case SUC3
Hardwareupdates		◎		◎			◎	◎	Enterprise IT Modell / BUC 1	Data / Analysis Model
Leichte Bedienbarkeit									Anforderungsdokument	
Komp mit Fuhrparksoftware		◎	△		◎	◎	◎	○	Anforderungsdokument	Prim. Deployment Model
On Line Help			◎					○	Anforderungsdokument	Data Analysis Model
Release Strategie							◎		Vertragliche Anforderung	Non funtional Requirements Model
Keine langen Ladezeiten		◎	◎	◎	◎	◎			Anforderungsdokument	Non funtional Requirements Model

Abbildung 37 Kundenanforderungen in Relation zur technischen Produktspezifikation

Würde man die Vorgehensweise aus der Geschäftsprozessanalyse beibehalten entsteht im zweiten Prozessschritt eine Tabelle wie in Abbildung 37 welche die Relationen zwischen Kundenanforderungen und Technischen Produktspezifikationen beschreibt. Neben diesen wird die modellbasierte Verfolgung von Anforderungen nur durch Referenzierung unter den Modellnamen und Typen gewährleistet.

Bei der Entwicklung dieses Ansatzes viel mir jedoch auf, dass eine Darstellung der Abhängigkeiten unter den Modellen im Sinne der Nachverfolgung wertvoll wäre, um diese ebenso wie die Umsetzung der Anforderungen besser nachvollziehen zu können. Damit ist es möglich die Einflussbereiche einer Anforderung nicht nur auf

das nächste Level der technischen Produktspezifikation zu mappen sondern ebenso die Anforderung und insbesondere, wenn man deren Veränderung in Betracht zieht, die möglichen Auswirkungen auf die nachfolgenden Modelle veranschaulichen zu können. Somit wäre es sinnvoll mehrere Relationstabellen zu verknüpfen und folgende Erweiterung einfließen zu lassen:

	Techn. Produktspezifikation	Kapazität Anzahl Fahrzeuge	Kapazität Anzahl User	DB Abfrage kleiner 5sek. Responsetime	Schnittstelle zu Userverwaltung	Schnittstelle zu Payment	SSL Verschlüsselung	Verwendung von XML Files	**Modelle des PIM**	System Use Case SUC1	System Use Case SUC2	System Use Case SUC3	Data / Analysis Model	System Behavior Model	Prim. Deployment Model	Non funtional Requirements Model
Kundenanforderungen									**Modelle des CIM**							
Userverwaltung		◯	⊙	⊙		⊙	⊙	⊙	BUC 1 / BUC 2 / BUC 3 …	⊙		⊙	◯			
Sicherheitssystem			⊙	◯	◯	◯	⊙	△	Enterprise IT Modell / BUC1	△	△	⊙	◯			
Fahrzeugverwaltung		⊙	◯	⊙	⊙	⊙	◯		BUC 2 / User Verify BUC 3 / New CAR		⊙	⊙	◯	◯		◯
Hardwareupdates		⊙		⊙			⊙	⊙	Enterprise IT Modell / BUC 1					⊙		
Leichte Bedienbarkeit				⊙					Anforderungsdokument				⊙	◯		⊙
Komp mit Fuhrparksoftware		⊙	△		⊙	⊙	⊙	◯	Anforderungsdokument					◯	◯	⊙
On Line Help			⊙					◯	Anforderungsdokument				⊙	◯		◯
Release Strategie							⊙		Vertragliche Anforderung						⊙	
Keine langen Ladezeiten		⊙	⊙	⊙	⊙	⊙			Anforderungsdokument	◯	◯	◯		⊙		⊙

Abbildung 38 Kundenanforderungen / Technische Produktspezifikation / Relationen der Modelle

Die Kundenanforderung der Userverwaltung und den damit verknüpften Rollen und Operationen spiegelt sich in den Modellen Business Use Case (BUC) 1/2/3 wieder. Diese werden auf der Ebene des PIM in die System Use Cases SUC 1 und SUC 3 übergeführt. Weiters fließen Informationen bezüglich der Daten in das Data Analysis Modell ein. Da jedoch nicht nur Daten aus der Userverwaltung in diesem Modell zu finden sind sonder auch Daten aus der Fahrzeugverwaltung darin abgebildet werden, wird hier eine mittelstarke Korrelation verwendet. Für die Fahrzeugverwaltung ist das BUC 2 und das BUC 3 von Relevanz da zum Beispiel das Anlegen eines neuen Fahrzeugs in der Fahrzeugverwaltung im System Use Case 3 weiter spezifiziert wird. Nachdem zum Anlegen eines neuen Fahrzeugs gewisse Userrechte gegeben sein

müssen, ist es notwendig auch eine Korrelation mit der Userverwaltung zu setzen. Das Enterprise IT Model in welchem die Hardware festgehalten wird ist korreliert mit dem System Behavior Modell da eine Aufwertung der Hardware sich positiv auf das System verhalten auswirkt. Ebenso gilt das für das hier beispielhaft angeführte BUC1 in welchem, wie oben bereits angesprochen, Teile der Userverwaltung modelliert sind. Die Performance der Userverwaltung wirkt sich natürlich ebenso auf das Systemverhalten aus weswegen auch hier eine Relation bestehen sollte. Dies gilt natürlich für alle Module.

Begleitend zu den einzelnen Schritten, mit denen diese Anforderungstabellen und die damit in Relation stehenden Modelltabellen entstehen, ist es sinnvoll Analysen dieser Tabellen durchzuführen:

3.10.3 Analyse der Tabellen:

Eine leere Zeile beispielsweise ohne jegliche Relationsbeziehungen weist eine Anforderung aus die nicht adressiert wurde. Ebenso können unkorrelierte Spalten entstehen welche unnötige Features eines Systems darstellen. Es treten auch einige Fragen auf, bei Zeilen oder Spalten in denen sich nur schwache Korrelationen befinden – Ist die Anforderung somit nicht wichtig fürs System oder wird die Anforderung durch die technischen Komponenten nicht adressiert? Ebenso könnten viele starke Korrelationen einer technischen Komponente zu mehreren Anforderungen implizieren, dass eine genauer detaillierte Spezifikation notwendig ist um mögliche Ressourcenengpässe auszuräumen oder die Komplexität zu verringern und diese Problematik somit in den Griff zu bekommen. Insofern ist diese Methode durchaus auch als Risikoidentifikationsmethode für ein Entwicklungsprojekt verwendbar. Viele Abhängigkeiten einer Komponenten zu Schnittstellen, die Verwendung von neuen Technologien mit welchen noch nicht gearbeitet wurde, oder Zeitdruck in der Implementierung auf Grund der Wichtigkeit einer Komponente für andere können als Risiken identifiziert werden und so in das Qualitätsmanagement des Softwareentwicklungsprozesses einfließen.

Für diese Auswertung der Relationen kann auch ein Zahlensystem herangezogen werden mit welchem man dann nach mathematischen Regeln die Zusammenhänge berechnen könnte um somit eine eindeutige Gewichtung der einzelnen Anforderungen und der Relationen untereinander zu definieren. Ordnet man jeder starken Relation beispielsweise den Wert 9 zu, einer mittleren Korrelation den Wert 6, einer schwachen Korrelation den Wert 3 und gar keiner Relation den Wert 0, so würde sich für jede Zeile, also für jede Anforderung auf dieser Ebene durch Addition eine einzelne Zahl errechnen lassen. Mittels dieser kann man Aussagen über die Relation einer Anforderung zu den technischen Spezifikationen oder den anderen Ebenen im MDA Prozess machen. Je größer die Zahl desto stärker wird die Anforderung von den technischen Spezifikationen beeinflusst.

So würde zum Beispiel die Userverwaltung auf Grund von 5 starken Relationen und einer mittleren Relation zu den technischen Spezifikationen den Endwert von 51 erhalten. Die Fahrzeugverwaltung würde einen Endwert von 48 (4 x 9 + 2 x 6) erhalten. Der Unterschied liegt, und dies ist auch aus der Tabelle ersichtlich, an der Tatsache dass die SSL Verschlüsselung und die Verwendung von XML Files zur Speicherung der Fahrzeugdaten für die Fahrzeugverwaltung nicht so wichtig ist, wie für die Userverwaltung. Eine mögliche Konsequenz daraus wäre in der Entwicklung zu berücksichtigen dass die SSL Komponenten implementiert sein müssen bevor die Userverwaltung implementiert wird, da eine starke Korrelation besteht.

3.10.4 Die Einteilung in Subsysteme:

Auf der nächsten Ebene des Plattform Specific Models (PSM) werden wie bereits beschrieben die Subsystemanforderungen definiert und dem Prozess folgend mit der technischen Spezifikation des Systems in Verbindung gesetzt. In dem Beispiel welches wir hier durchlaufen, wurde als das Subsystem die Fahrzeugverwaltung gewählt.

Die das PSM beschreibenden Modelle wie das Service und Interface Modell oder das Architecture Model werden mit denen im vorhergehenden Prozessschritt definierten technischen Produktspezifikationen, oder auch technische

Produktanforderungen und den darin modellierten Anforderungen in Relation gesetzt. Dies ist notwendig um zu veranschaulichen welche Technische Spezifikation in welchen Komponenten bzw. welche Operationen, abhängig von welcher technischen Spezifikation und Technologie modelliert und später implementiert werden. Ebenso sind auch die Modelle des Plattformunabhängigen Modells in dieser Tabelle enthalten um die Anforderungsverfolgung zu gewährleisten.

		Fahrzeugverwaltung													
	Subsysteme	Fahrzeug buchen	Fahrzeug hinzufügen	Fahrzeug löschen	Fahrzeugdaten updaten	Userrechte überprüfen	Verfügbarkeit abfragen	Status verändern	**Modelle des PSM**	Service & Interface Model SI2	Service & Interface Model SI3	Architecure Model	Architecture Behavior Model	Conceptual Deployment Model	Quality Architecture Model
Tech. Produktspezifikation									**Modelle des PIM**						
Kapazität Anzahl Fahrzeuge		○	◎	◎	○		○		Analysis Model			◎			
Kapazität Anzahl User		◎	○	○		○	○	△	Analysis Model			◎			
DB Abfrage kleiner 5sek. Responsetime			◎	◎	◎		◎		Non functional Requirements Model				◎	△	◎
Schnittstelle zu Userverwaltung		◎	◎	◎	◎	◎			System Use Case SUC2	◎					
Schnittstelle Payment		◎				◎			System Behavior Model		◎	○		◎	
SSL Verschlüsselung		◎			◎	△			Prelim. Deployment Model	○			◎		
Status eines Fahrzeugs		◎	△	◎	○	◎	◎	◎	System Behavior Model		◎			◎	
Verwendung von XML Files		○	◎	◎	◎		◎		Fachkonzept 01			◎	○		
WEB Interface			◎	◎	◎				System Use Case SUC3	○	◎	○			

Abbildung 39 Technische Produktspezifikation in Relation mit einem Subsystem

Nachdem in diesen Bereich Modell – Modell Transformationen zum Einsatz kommen welche in der meisten Fällem über ein XMI File (XML Metadata Interchange , ein OMG Standard welcher für den Austausch von Metadaten Information über XML und insbesondere für UML und in Zusammenhang mit der Meta Object Family MOF verwendet wird) abgewickelt werden, könnte man als letzte Spalte in dieser Tabelle auch noch den Dateinamen des speziellen XMI Files welches für die Transformation der einzelnen Modelle in die nächste Ebene verantwortlich ist anführen. Somit wäre die Nachverfolgung der Operationen welche auf die einzelnen Modelle angewandt werden ebenso gewährleistet.

Bei einer automatisierten Transformation ist es in der Regel kaum möglich, gegen die im Metamodell festgelegten Architekturprinzipien zu verstoßen. Das bedeutet natürlich auch, dass für eine Anforderung, die vom Architekturmodell nicht berücksichtigt wird, die Architektur entsprechend erweitert werden muss. Dies ist genau im Sinne der MDA: Die Architektur des Systems steht im Mittelpunkt und wird ständig konsistent gehalten. Klassische "Workarounds", die die Architektur umgehen und häufig Ursache für Softwarekrankheiten sind, entfallen weitestgehend.

Einen besonderen Vorteil des MDA Prozesses stellt die Möglichkeit dar unterschiedliche Systemkomponenten auf unterschiedlichen Plattformen zu realisieren. Im Sinne der Anforderungsverfolgung bringt dies aber das Problem mit sich einzelne Anforderungen bis auf das Level der Plattform spezifischen Modelle welche für die Implementierung maßgeblich sind zu verfolgen. Auf Grund dessen soll hier eine weitere Tabelle veranschaulichen, wie dies mittels QFD Tabellen realisierte werden kann. Wir entfernen uns dafür von der Ebene des Model Frameworks mit welcher wir bis jetzt die Zusammenhänge unter den Modellen repräsentiert haben und erläutern an dem einfachen Beispiel des Autoverleihs wie man Anforderungen in die Module verfolgen kann. Es soll dargestellt werden, dass der Autoverleih mittels drei verschiedener Technologien realisiert wird: Zur Verwaltung der Daten wird eine SQL Datenbank verwendet, die Programmlogik des Autoverleihs wird mittels Java implementiert und für die Darstellung im Internet bzw. als User Interface kommen Java Server Pages (JSP) zum Einsatz.

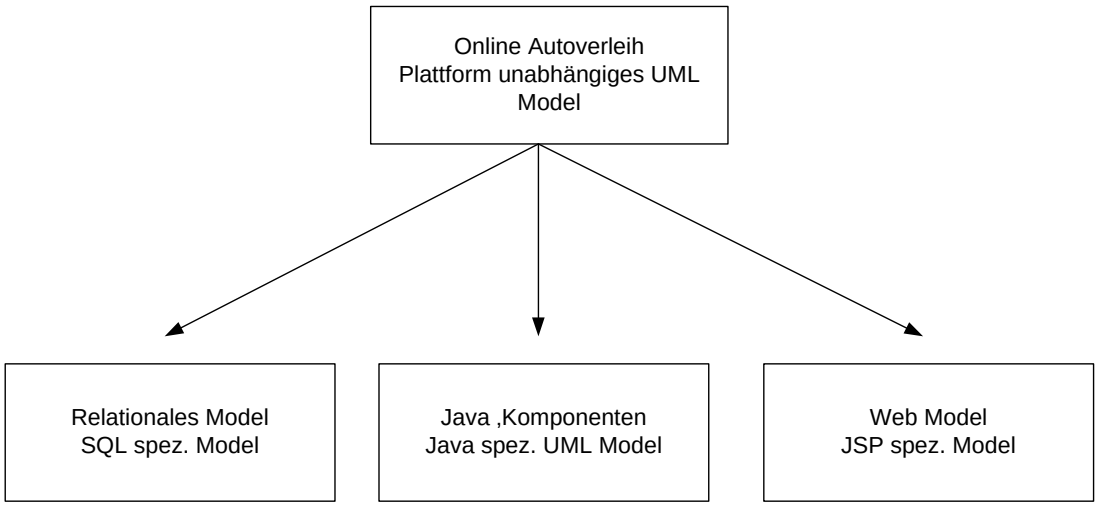


Abbildung 40 Aufteilung in Plattform Spezifische Modelle

Betrachten wir zum Beispiel das Buchen eines Fahrzeugs und einen User anlegen:

	Subsysteme	Web Komponente	Userverwaltung	Fahrzeugverwaltung	Datenbank	PSM MODELL	JSP – UML Modell	Java spez. UML Modell	SQL Modell
						Transformationsfile	UML -JSP Transformationsegel	UML -JAVA Transformationsregel	UML - SQL Transformationsregel
Anforderung						**PIM MODELL**			
AUTO BUCHEN		◎	◎	◎	◎	**Einzelnes UML MODELL**	◎	◎	◎
USER ANLEGEN		◎	◎	△	◎	**System USE Case**	◎	○	◎

Abbildung 41 Abhängikeiten PIM - PSM Modelle unter Transformationsberrücksichtigung

Ziel dieses Beispiel ist es zu Veranschaulichen dass ein einzelnes Modell in mehrere verschiedene Modelle aufspalten kann welche jedoch noch immer den gleichen Vorgang beschreiben, dies jedoch nur auf einem anderen Level der Abstraktion passiert. Aus diesem Grund haben wir hier für die erste Anforderung ausschließlich starke Relationen da die Anforderung des Auto buchen alle drei Systemkomponenten beansprucht. In der zweiten Anforderung welche das Anlegen eines Benutzers betrachtet besteht nur eine schwache Korrelation zur Fahrzeugverwaltung da diese nicht notwendig ist um diese Anforderung zu realisieren, jedoch der Benutzer auch an diese übergeben werden kann, um dort beispielsweise als Administrator Fahrzeuge zu verwalten. Dies wirkt sich auch auf die Korrelation unter den Modellen aus. Die Java spezifische UML Komponente hat daher einen nicht so großen Anteil in der Realisierung dieser Anforderung und wird auf Grund dessen mit einer mittel starken Korrelation repräsentiert.

Es soll mit Abb. 41 gezeigt werden dass man mittels QFD Tabellen über die verschiedenen Abstraktionsgrade auf den unterschiedlichen Ebenen Zusammenhänge veranschaulichen kann und diese Relationen sofern diese informationstechnologisch

gestützt sind, einerseits die Dokumentationsarbeit als wie auch die Übersicht in einem Softwareentwicklungsprozess besser gestalten können und somit zur Qualitätsverbesserung beitragen. Hinzu kommt, dass wenn eine Tabelle wie sie in Abb. 35 verwendet wird auch die Transformationsregeln berücksichtigt werden können (vgl. oben)

3.10.5 Umsetzung der Subsysteme:

Die nächste Tabelle soll die vorher definierten Subsysteme auf den Quellcode und somit die Implementierung abbilden.

Sourcecode / Subsysteme	Fahrzeugverwaltung.java	Userverwaltung.java	SQL Skript create tables	SQL Skript modify tables	Autoverleih.jsp	PayService.jsp	PSM Modell	Code Transformation
Fahrzeugverwaltung	◎	○	○	○	○		Java spez. UML Modell	Java spez Transformationsregel
Userverwaltung		◎	○	○	○		Java spez. UML Modell	Java spez Transformationsregel
Datenbank	○	○	◎	◎			SQL Modell	SQL Skript Generierung
WEB Komponente	○	○	△	△	◎	◎	JSP spez. Modell	JSP Transformationsregel

Abbildung 42 Vereinfachte Darstellung der Relationen von Subsystem zum Sourcecode

Die Abbildung zeigt was in diesem Schritt passiert stark vereinfacht. Für die einzelnen Subsysteme Fahrzeugverwaltung, Userverwaltung, Datenbank und Web Komponente werden die Quellcode Dateien erstellt. Die Basis dafür sind die oben Plattformspezifischen Modelle und die darauf angewandten spezifischen Regeln zur Codegenerierung. Gleichzeitig wird das Zusammenspiel der Komponenten bzw. der einzelnen Source Files mittels der Relationen dargestellt. In dieser Graphik bedeutet eine starke Korrelation dass diese Komponente in diesem Bereich des Quellcodes umgesetzt worden ist. Eine mittlere Korrelation bedeutet aber dass sehr wohl auch Überschneidungen zu den anderen Komponenten vorhanden sind und diese berücksichtigt werden müssen. So müssen beispielsweise über die Fahrzeugverwaltung oder die Userverwaltung SQL Skripts erstellt werden bzw. auf erstellte Skripts zugegriffen werden weswegen hier eine mittelstarke Korrelation

besteht. Ebenso bei der Web Komponente, welchen den Input für die Userverwaltung und die Fahrzeugverwaltung liefert, jedoch auf Grund dessen das die Userverwaltung oder Fahrzeugverwaltung die Aufgabe übernehmen die SQL Statements zu generieren ist eigentlich Relation zu den SQL Komponenten nur noch schwach. Nachdem bei der Implementierung auch teilweise auf Modell Code Transformationen zurückgegriffen wird, müssen auch diese hier berücksichtigt werden da in diesen die Regeln zur Transformation eines Modells in Quellcode festgehalten werden. Es macht hier keinen Sinn Relationen im Mapping der Modelle auf die Transformationen zu zeigen da diese auf dieser Ebene für jede Komponenten speziell definiert sein müssen und somit keine Relationen untereinander bestehen.

Wenn man nun jedoch die einzelnen Anforderungen an das System betrachtet so ist eine allgemeine Tabelle wie ich sie eben beschrieben haben nicht sinnvoll. Man weiß zwar welches Subsystem auf welcher technischen Plattform und in welchem Source File definiert ist, eine genaue Information welche Funktionalität und somit welche Anforderung, das einzelne Source File in sich birgt und welche Funktionen auch in anderen Files zur Verfügung stehen muss ist mit Übersicht wie dieser nicht gegeben. Auf Grund dessen ist es sinnvoll im Sinne der Anforderungsverfolgung die Subsysteme und die darin realisierte Funktionalität in einer Tabelle zu veranschaulichen

Source Code Datein		Fahrzeugverwaltung Source Code	Userverwaltung Source Code	SQL Source Code	Java Server Pages Jsp.Source Code	Payment Component Source Code	
Subsysteme & Funktionalität							
Fahrzeugverwaltung	Fahrzeug hinzufügen	◎	○	○	○		
	Fahrzeug löschen	◎	○	○	○		
	Fahrzeugdaten updaten	◎	○	○	○		
	Userrechte überprüfen	○	◎	○			
	Verfügbarkeit abfragen	◎		○			
	Status Fahrzeug verändern	◎					
Userverwaltung	User anlegen		◎	○	○		
	User löschen		◎	○	○		
	Userrechte definieren	○	◎	○	○		
	Userstatus abfragen		◎	○			
	Userstatus verändern						
Web Komponente	Fahrzeugliste anzeigen			○	◎		
	Fahrzeug buchen	○	○	○	◎		
	Bestätigung anzeigen				◎		
	Fahrzeug bezahlen				○	◎	

Abbildung 43 Detailierte Funktionalität in Relation zum Source Code

Auch in Abbildung 34 gilt die gleiche Definition der Korrelationen wie in Abbildung 33. Eine starke Korrelation bedeutet die Umsetzung in diesem Teil der Implementierung, eine mittelstarke Korrelation bedeutet dass eine Relation zu einem anderen Source File besteht.

3.10.6 Validation der Anforderungen:

Im letzten Schritt des hier definierten SoftwareQFD Frameworks steht die Validation der Implementierung im Vordergrund. In einer Tabelle werden die Anforderungen, der Quellcode und die dazugehörigen vor der Implementierung erstellten Testfälle in Relation gesetzt. Dies dient einerseits dazu um zu überprüfen ob die Anforderungen so umgesetzt wurden wie sie modelliert wurde, und um das System auf seine Funktionalität zu überprüfen. Normalerweise wird zu diesem Zweck eine so genannte Tracability Matrix herangezogen. Diese setzt normalerweise Anforderungsdokumente wie ein Busines Use Case mit einem oder mehreren

Testfällen in Verbindung um die Anforderung zu überprüfen: Nachfolgende Graphik zeigt eine Tracability Matrix:

Anforderungen	Anforderungen getestet	Req 1 BUC 1	Req 1 BUC 1	Req 2 BUC 1	Req 2 BUC 2	Non funct Req 1 Seq. Dia 2
Testfälle	6	1	2	2	2	2
durchgeführt	50%		1	1	1	
1.1	2	x	x			
1.2	1		x			
2.1	3			x	x	x
2.2	1			x		
2.3	1				x	
3.1	1					x

Abbildung 44 Tracability Matrix [IQ_1]

Auf Basis dieser Traceability Matrix kann man nun mittels der QFD Notation eine ähnliche Matrix etablieren welche neben den Anforderungen und Testfällen auch die relevanten Source Code Dateien enthält und die Relationen auf Grund der Notation genauer definiert. Mittels der Relationen können Abhängigkeiten einzelner Testfälle untereinander mit der oben definierten Notation gezogen werden. Eine starke Korrelation validiert die Anforderung mittels Zugriff auf ein spezifisches Source File welches im Rahmen eines Subsystems definiert ist und eine bestimmt Funktionalität beinhaltet. Eine mittelstarke Korrelation bedeutet dass diese Anforderung einen Einfluss auf den Testfall und auf die zu validierende Komponente hat.

Zu Abb 45: Auf der vertikalen Achse werden zunächst die Subsysteme und die darin gewünschte Funktionalität beschrieben. Weiters befinden sich in der Spalte rechts davon der Name der Quellcode Datei in welcher die gewünschte Funktionalität umgesetzt wurde. Hier könnte man zusätzlich auch die spezifische Lines of Code angeben in welchen die Operation durchgeführt wird. Auf der horizontalen Achse

werden die Anforderungen gelistet. In diesem Fall führen wir beispielsweise die Anforderungsidentifikationsnummer an. Es könnte hier aber auch auf Dokumente, oder wie oben in der Requirementstracabilitymatrix gezeigt Use Cases stehen in welchen die Anforderungen beschrieben sind.

Ebenso sollte auch die Teststrategie in diese Matrix einfließen, denn es können unterschiedliche Tests in unterschiedlichen Phasen der Entwicklung durchgeführt werden und demnach auch für die Anforderungsverfolgung unterschiedliche Ergebnisse zum Vorschein bringen.

		Testfall zu Anforderung	Anfoderungsidentifikationsnummer						Auswertung der Testfälle	Modultest Testfall OK	Modul Testfall Not OK	Systemintegrationstest OK	Systemintegrationstest Not OK
			1.1.1	1.1.2	1.1.2	1.1.3	2.1.1	2.1.2					
Subsysteme & Funktionalität		**Source Code Datein**											
Fahrzeugverwaltung	Fahrzeug hinzufügen	Fahrzeugverwaltung.java											
	Fahrzeug löschen	Fahrzeugverwaltung.java										X	
	Fahrzeugdaten updaten	Fahrzeug.java			◎		○				X		
	Userrechte überprüfen	User.interface				◎				X			
	Verfügbarkeit abfragen	Fahrzeugverwaltung.java					◎			X			X
	Status verändern	Fahrzeug.java											
Userverwaltung	User anlegen	Userverwaltung.java											
	User löschen	Userverwaltung.java					◎				X		
	Userrechte definieren	User.java											
	Userstatus abfragen	User.java											
	Userstatus verändern	User.java											
Web Komponente	Fahrzeugliste anzeigen	Jsp.Source Code											
	Fahrzeug buchen	Jsp.Source Code											
	Bestätigung anzeigen	Java Server Pages Jsp.Source Code											
	Fahrzeug bezahlen	Payment Component Source Code											

RELATIONEN

Abbildung 45 Validierung der Anforderungen

3.10.7 Conclusio über die Verwendung des Software QFD zur Anforderungsverfolgung:

In diesem Kapitel wurde gezeigt wie die Tabellen des QFD Frameworks zur Anforderungsverfolgung eingesetzt werden können. Die hier beschriebene

Verwendung von Tabellen um Abhängigkeiten von einzelnen Anforderungen den, beschreibenden Modellen und den auf die Modelle angewandten Transformationen bilden die unter dem Punkt 3.9 definierten Prozessschritte zur Verfolgung von Anforderungen ab und geben Antworten auf die dort definierten Fragen. Die Vor- und Nach – teile die eine Anwendung solcher Tabellen mit sich bringt sollen nun zusammengefasst und erläutert werden.

Vorteile & Nachteile in der Verwendung von SQFD Tabellen:

Ein Vorteil ist mit Sicherheit die Analyse des Geschäftsprozess und den daraus resultierenden Anforderungen mit Hilfe von Korrelationsmatrizen. Der Prozess beschreibt ihn als Definition von Abhängigkeiten zwischen Geschäftsprozess und Kundenanforderungen. Dies dient zur Qualitätssicherung der Anforderungen zu einem frühen Zeitpunkt der Softwareentwicklung.

Was man im Kontext der Thematik bezüglich der Geschäftsprozessanalyse nicht außer Acht lassen sollte ist der Wiederverwendungswert einer solchen Analyse. Bei vielen Softwareherstellern ist die Erfahrung für die Erstellung Domänenspezifische Software nicht unwesentlich. Mit der Dokumentation der Tabellen respektive der Geschäftsprozesse, optimiert man im Falle einer Neuimplementierung die Analysezeit, da man auf bestehende Tabellen zurückgreifen kann und mit diesen die Auswirkungen von Änderungen im Geschäftsprozess auf die Kundenanforderungen bis hin zur Implementierung dargestellt werden können.

Das Nachvollziehen von Änderungen wird durch die den Prozess beschreibenden Tabellen gesichert. Dadurch dass der obere Teil einer Tabelle direkt in die nächste Tabelle einfließt ist eine Nachverfolgung gewährleistet. Somit gilt, dass unabhängig davon auf welcher Ebene des Prozess eine Veränderung geschieht sie sowohl nach vorwärts in den nächsten Prozessschritt als wie auch nach Rückwärts in den vorhergehenden Prozessschritt verfolgt werden kann. Dies entspricht der Definition von Anforderungsverfolgung.

Dies ist somit auch im Sinn der Frage welche wir zu Beginn dieser Arbeit aufgebracht haben, nämlich welche Möglichkeiten die Tabellen des QFD zur Anforderungsverfolgung bieten können, relevant. Die hier gezeigte Verwendung von Tabellen entfernt sich nicht wesentlich vom Ursprung des QFD nämlich Kundenanforderungen in Qualitätsprodukte überzuführen und zeigt vor allem welche Möglichkeiten gegeben sind Relationen, in unserem Fall auch unter Modellen aufzuzeigen und durch die Darstellung von Abhängigkeiten, darin einzelne Anforderungen zu verfolgen.
In Zusammenhang mit der Nachverfolgung durch verschiedene Modelle, auf verschiedenen Ebenen des Entwicklungsprozess, sei insbesondere auch die Möglichkeit der expliziten Gewichtung von Abhängigkeiten durch eine mathematische Auswertung der Relationen wie sie unter Punkt 3.10.2 beschrieben wird, erwähnt. Die Wichtigkeit einzelner Anforderungen und somit auch die Wichtigkeit der diese Anforderungen abbildenden Modelle, kann gemessen werden. Somit werden Dringlichkeiten einzelner Kundenwünsche dargestellt, die in der Implementierung berücksichtigt werden können.

Eine wichtige Komponente in der Softwareentwicklung, insbesondere was die Qualität einer Software betrifft, sind Fehler die Aufgrund einer falschen Spezifikation oder durch Änderungen welche am System vorgenommen werden, zu vermeiden. Insbesondere in der modellbasierten Entwicklung, wo durch den Versuch Anforderungen durch automatisierte Transformationen über die verschiedenen Ebenen der Abstraktionsgrade und den Entwicklungsprozess zu implementieren, bietet eine Darstellung wie sie hier in diesem theoretischen Ansatz gezeigt wird, die Möglichkeit, Fehlerquellen zu minimieren und auch eine Kontrolle der Veränderungen bis auf das Level der Implementierung, als wie auch deren Validation zu gewährleisten. Insbesondere wird gezeigt, dass eine Einbindung der Transformationen in die Tabellen welche die Veränderungen veranschaulichen, möglich und notwendig ist, um die Verfolgung von Anforderungen zu gewährleisten. Zusätzliche Tabellen können ebenso zur Entwicklung von Transformationsregeln in den Prozess miteinbezogen werden und würden quasi einen eigenen Transformationsentwicklungsprozess beschreiben. Der hier dargestellte Ansatz sollte sich auf Grund dieser Erkenntnisse positiv auf die Qualität als wie auch positiv auf

die Kosten auswirkten, denn einerseits können letztere schon durch eine bessere Anforderungsanalyse oder einer bessere Transformationsgestaltung und somit weniger Fehler und weniger Change Requests, während der Entwicklung gesenkt werden.

Ein Nachteil welcher mit der Verwendung von QFD Tabellen mit Sicherheit entsteht ist die Komplexität der Tabellen. Die Erstellung der oben gezeigten theoretischen Beispiele, welches vom Umfang her recht klein angenommen wurde stellte sich als äußerst komplex und langwierig dar. Eine Verwendung dieses Ansatzes müsste auf jeden Fall auf der Unterstützung eines vordefinierten Programms oder vordefinierten Excel Tabellen mit Makros aufbauen um den Aufwand für die Verfolgung von Anforderungen möglichst gering zu halten. Alles andere würde in diesem Zusammenhang keinen Sinn machen da der durch die Wartung der Tabellen entstehende Mehraufwand durchaus für das manuelle Nachvollziehen einzelner Anforderungen aufgewendet werden könnte.

Ein zusätzkicher Punkt der im Zusammenhang mit der Verwendung von QFD Tabellen als positiv zu sehen ist, ist die Darstellung von Zielkonflikten. Neben der Durchführung des Ansatzes im Entwicklungsprozess zur Verfolgung von Anforderderungen, kann man ihn auch rein zur Planung durchführen. Durch diese Art der Simulation und aus der damit verbundene Auswertung der Relationen können Zielkonflikte unter einzelnen Anforderungen, unter Subsystemen, unter technischen Möglichkeiten oder auch phasenübergreifend entstehen und somit frühzeitig erkannt und abgewendet werden können. Weiters könnte mittels so genannter Informationspfade ein einheitlicher Prozess definiert werden. Diese würden die Verwendung von Tabellen in unterschiedlichen Entwicklungsprozessen anhand enthaltenen Information abbilden und so ein entwicklungsprozessspezifisches Mapping zur Verfügung stellen. Der Informationspfad würde also die Verwendung der Tabellen, zu den unterschiedlichen Zeitpunkten der Entwicklung, mit den vorhandenen Informationen, welche in Korrelation gesetzt werden, abbilden.

Eine weitere Verbesserung wäre explizit in einer Tabelle die Kosten in Relation zu den Anforderungen zu gewichten, um beispielsweise diese einerseits abschätzen zu

können und somit für die Preisgestaltung heranziehen zu können oder einen Zahlungsplan zu definieren, welcher in Korrelation mit Meilensteinen, die die Fertigstellung wichtiger Subsysteme definieren wieder Geld in das Entwicklungsprojekt kommt um laufende Kosten zu decken. Hierbei stellt sich mir auch die Frage, wenn es möglich sein kann Kosten in dieses Schema einfließen zu lassen so müsste es auch möglich sein eine weitere oft kritische Komponente im Softwareentwicklungsprozess einfließen zu lassen: Ressourcen. Welches Team setzt welche Anforderung um und wie wirkt sich beispielsweise ein Abgang einer Schlüsselressource in einem dieser Teams auf das Projekt und die Qualität des Produkts aus. Es ist also durchaus eine interessante Überlegung, die wichtigsten Punkte aus dem Rahmen des Projektmanagements in diesen Ansatz einfließen zu lassen und somit auch die organisatorische Komponente eines Softwareentwicklungsprozess in die Qualitätssicherung oder wie wir es hier beschrieben haben, in die qualitätsbezogene Entwicklung von Software einfließen zu lassen.

Eine Problematik die mir während der Entwicklung dieses Ansatzes aufgefallen ist, ist dass es nicht immer einfach ist die Korrelationsnotation oder besser die Bedeutung für die einzelne Korrelation im Zusammenhang mit den in Korrelation gesetzten Bereichen zu definieren. In manchen Tabellen sind die 4 Relationen (stark, mittel, schwach und keine) äußerst sinnvoll, vor allem zu Beginn des Entwicklungsprozess wo viele unterschiedliche Abhängigkeiten existieren. Im weiteren Verlauf würden aber 2 bzw. 3 Relationen stark, mittel und gar nicht auch reichen da beispielsweise der Zusammenhang einer Anforderung mit einer technischen Spezifikation schwer über diese Definition auszuweisen ist. Hierbei ist es also sehr wichtig der Tabelle eine genaue Definition der Relation hinzuzufügen um die Komplexität der Tabelle nachvollziehen zu können.

4 Kapitel: Zusammenfassung und Ausblick:

Dieses Kapitel soll die in den vorhergehenden Kapitel angeführten Basics und Theorien zusammenfassen und einen Ausblick auf die Zukunft der MDA und Anforderungsverfolgung geben:

Ich möchte zu Beginn der Conclusio uns noch einmal die ersten beiden Thesen die wir zu Beginn aufgestellt haben und deren Verifikation oder Falsifikation uns die Motivation für diese Arbeit gegeben haben, erneut einzeln anführen und diskutieren Die 3. These wurde bereits in einer eigenen Conclusio unter Punkt 3.10.6 behandelt:

1. These

Die modellgetriebene Entwicklung unterstützt Anforderungsverfolgung auf Grund von umfangreichen Modellen in verschiedenen Abstraktionsgraden, bei denen auf Grund automatisierter Modelltransformationen besonders auf Konsistenz geachtet werden muss. Kann man diese Aussage verifizieren bzw. durch das Zusammenführen der beiden Ansätze Anforderungsverfolgung und Modellbasierte Entwicklung beantworten.

Ad 1:

Als ich mich das erste Mal mit dem Ansatz der modellbasierten Softwareentwicklung auseinander gesetzt habe war mir zunächst nicht bewusst, wie detailliert eigentlich schon auf Grund des Konzepts der MDA, Möglichkeiten für Anforderungsverfolgung geboten werden. Ich musste erst die TOP Down Struktur der Abbildung 5 entwickeln und adaptieren, um eine Basis dafür zu erstellen die Anforderungsverfolgungselemente darin zu erkennen und dann auch in weiterer Folge diese Ansätze zu definieren und graphisch in einem Modell umzusetzen.

Die Basis für Anforderungsverfolgung bietet die genaue Spezifikation der Anforderungen und diese ist meiner Ansicht nach im Rahmen einer sorgfältigen Analyse der Geschäftsprozesse zu realisieren. Wie wir im Kapitel Modellbasierte

Softwareentwicklung unter Die Geschäftsprozesse und das Computer Independent Model (CIM) gesehen haben passiert dies so wie es sein sollte zu Beginn eines MDA Prozesses. Auch durch die Integration von Quality Function Deployment in Kapitel 3 wurde eine Methodik gezeigt welche den MDA Prozess und Anforderungsverfolgung darin unterstützt. Hierbei wurde der Ansatz des einmaligen QFD wie es zum Teil in der Softewarentwicklung üblich ist, um eine genaue Definition von Kundenanforderungen und das Mapping auf die Produktanforderungen zu erhalten für den MDA Prozess adaptiert. Diese Adaption bedeutet das QFD über den gesamten Entwicklungsprozess zu integrieren. Daraus resultieren bessere Möglichkeiten die Qualität auch über verschiedene domänespezifische Softwareentwicklungsprojekte hinaus im Unternehmen zu etablieren.

Die Anwendung von Transformationsregeln um die Überführung eines CIMs ins PIM und weiterführende Modelle zu gewährleisten müssen mindestens genauso detailliert definiert werden, wie die Anforderungen um den Graubereich der Fehlerquellen im MDA Prozess zu minimieren. Denn die einerseits bei manchen Projekten automatisiert vorgenommenen Transformationen können genauso zu Inkonsistenz führen wie Fehler in den Analysen. Auch hier bietet der entwickelte Ansatz des QFD detaillierte Möglichkeiten die Anforderungen auf die Modelle und die darauf angewandten Transformationen zu mappen und somit zu dokumentieren.

Fraglich bleibt jedoch ob die Tools, welche zur Anforderungsverfolgung herangezogen werden diese Transformationsregeln ebenso auf Konsistenz und Nachvollziehbarkeit prüfen können, wie es manuell passiert, oder wie die einzelnen Modelle überprüft werden. Wir haben zwar ein Tool recht ausführlich beleuchtet jedoch konnte ich darin keine konkreten Mechanismen oder explizite Regeln zur Überprüfung solcher Transformationsregeln erkennen. Und meiner Meinung nach liegen genau dort die Probleme der MDA Entwicklung da die einzelnen Transformationen und die Regeln wohl ebensoviel Aufmerksamkeit benötigen wie deren Überprüfung.

Rein prinzipielle denke ich dass sich der Ansatz der modellbasierten Softwareentwicklung sehr gut mit Anforderungsverfolgung verknüpfen lässt und alleine auf Grund seiner Struktur und den darin angewandten Mechanismen und vor allem durch den durchgängigen Toolsupport im Prozess sehr gute Grundvoraussetzungen bietet.

Nachdem mit UML Tools und den damit verbundenen Möglichkeiten der Modellierung wirklich ein konsistenter Prozess durchlaufen werden kann, in welchem dank Erweiterungen wie SysML auch die Anforderungsanalyse in die Modellierung einfließt, denke ich dass diese These durchaus als erfüllt anzusehen ist. Neben Vorteilen wie diesen besteht auch die Möglichkeit einerseits durch die Modelle die auf den verschiedenen Abstraktionsebenen entstehen eine Art High Level Dokumentation zu erstellen welch den Reuse Wert für ein Produkt massiv steigern kann. Andererseits kann durch die Verwendung eines Anforderungsmanagementtools gemeinsam mit den äußerst mächtigen Modellierungstools automatisiert Dokumentationen erstellt werden, worauf ich besonders im Kapitel 3 eingegangen bin. Somit wird den Entwicklern allein durch den Prozess eigentlich die Arbeit erleichtert und die mühsame Erstellung von Dokumentation nach der eigentlichen Durchführung eines Projekts kommt nicht zu Stande.

Ich möchte an dieser Stelle einen kurzen Exkurs einschieben, in dem auf die Zukunft der MDA eingegangen wird, da die zweite These eher die Thematik der Anforderungsverfolgung behandelt und ich es als wichtig erachte im Rahmen dieser Arbeit einen Ausblick zu geben.

4.1 Exkurs: Die Zukunft der MDA:

Der MDA Prozess unterscheidet sich meiner Meinung nach nur gering von vielen anderen Entwicklungsprozessen bzw. lässt er sich mit dem Agile Prozess , RUP oder dem Extreme Programming Ansatz verbinden. Modelle haben auch bei vergleichbaren Prozessen massiv an Einfluss gewonnen und sind aus der modernen Softwareentwicklung nicht mehr wegzudenken.

Wenn man sich die vergangenen Entwicklungen ansieht so war es die Objektorientierung welche seit 1985 die prozeduralen Sprachen abgelöst, hat und seit dieser wurden keine massiv neuen Konzepte ins Leben gerufen. Vor den prozeduralen Sprachen wurde mit Assambler Sprachen programmiert welche wiederum eine Verbesserung zum vorhergehenden reinen Maschinencode waren. Was aber die eigentliche Quintessenz dieser Herleitung veranschaulichen soll, ist dass sich der Fokus im Rahmen der Programmierung verschoben hat. Während früher noch die low level Konstrukte relevant für die Programmierung waren hat sich der Fokus auf die high level Konstrukte verschoben. Dass diese jedoch auch nur mittels der low level Konstrukte funktionierten ist nicht mehr ganz im Bewusstsein vieler Personen die im Softwareentwicklungsprozess beschäftigt sind. Die Programmierung selbst beschäftigt sich nicht mehr mit der Erstellung reinen Maschinen Codes sondern in diesem Bereich hat sich durch den Fortschritt Automation breitgeschlagen und Compiler übernehmen hier wertvolle Arbeit.

Genauso denke ich wird es mit der MDA und den Softwareentwicklungsprozessen unserer Zeit geschehen. Der Fokus wird sich vom Code Level weg verschieben auf die PIM und PSM Ebene auf welcher die Modellierung komplettiert und adaptiert wird um dann automatisiert Code daraus zu generieren. Die Chancen bestehen sogar darin dass man im Laufe der Zeit die Anzahl der Ebenen minimieren kann und z.B. die Erstellung des PSM vernachlässigt und Code gleich aus dem PIM erzeugt wird.

Dieser Vision werden immer Programmierer gegenüberstehen welche meinen dass die Programmerstellung aus einem Modell nicht zu 100 Prozent realisierbar ist. Ich denke jedoch, dass besonders auf Grund dessen das die MDA noch in den Kinderschuhen steckt und der Prozess jetzt schon relativ erfolgreich ist, hier noch ein massiver Potential vorhanden ist, welches ausgelotet werden muss. In Angelegenheiten von Interoperabilität und Portabilität und Wartung werden sich viele Türen mittels MDA auftun. Und auch wenn zur Zeit noch Probleme in der Codegenerierung vorliegen und immer noch manuelle Erweiterungen programmiert werden, so bin ich einerseits zuversichtlich dass sich das bald ändert und die Programmierer immer mehr Zeit haben sich komplexeren Aufgaben als der Erstellung von maschinell generierbarem Code widmen können.

Somit wird es auch zu einer Neuverteilung der Aufgaben im SE Prozess kommen. Gesetztenfalls die MDA wird in Zukunft vermehrt eingesetzt, so kann man davon ausgehen, dass die Rollen wie wir sie eingangs in unserer Graphik angegeben haben, nicht mehr existieren werden, sondern durch Rollen wie den PSM Entwickler, oder den Transfomationsdefinitions Enwickler und den PIM Analysten ersetzt werden.

Auch im Bereich der Tools bemerkt man bereits auch jetzt dass es kaum noch reine Entwicklungstools gibt. In den meisten ist bereits eine Modellierungskomponenten integriert und auch in der oben angeführten Toolübersicht bez. MDA kann man erkennen dass große Hersteller wie IBM mit der Rational Suite oder Borland ebenso auf den Zug der MDA aufspringen und deren Entwicklungsumgebungen entweder in diese Richtung anpassen oder neue erstellen.
Insbesondere auch durch die Unterstützung großer Hersteller in der Softwareindustrie wird es dazu kommen, dass sich Modelle und deren Sprachen weiterentwickeln und somit auch die oben bereits erwähnten Tools. Die für
MDA Entwicklungen zugeschnittenen Sprachen werden was den Status angeht mit heutigen Programmiersprachen vergleichbar sein und werden sowohl mit dynamischen als auch statischen Aspekten umgehen können.

Worin ich persönlich durchaus noch ein Potential von Verwendung von MDA gemeinsam mit dem Konzept der Service Orientierten Architektur (SOA).

Exkurs SOA: Ziele

Primärziel ist, die historisch gewachsene, heterogene Systemlandschaft effizient an Änderungen im Geschäftsprozess anpassen zu können. Im Einzelnen soll dadurch Software erstellt werden, die

- *einfach an neue Bedürfnisse angepasst werden kann (Flexibilität)*
- *wieder verwendbar ist*
- *verteilt installiert werden kann*
- *an Geschäftsprozesse angepasst ist.*

Sekundärziele sind:

- *Kostenvorteile durch schnelle Optimierung*
- *schnelle Reaktion auf Herausforderungen möglich*
- *mittelfristig Einsparungen*
- *schrittweise Restrukturierung komplexer Anwendungssysteme*

[Aus Wikipdia, die freie Enzyklopädie Stand 06. Feb. 2006]

Gerade was die Thematik der Wiederverwendbarkeit und Wartung angeht steht der MDA ja ein ziemliches Potential zur Verfügung und wenn man sich die aktuellen Geschäftsdaten großer Anbieter im IT Markt ansieht, so ist ein eindeutiger Trend aus Verknüpfung von Services gemeinsam mit Hardware und Software zu identifizieren. Dieser Markt kann wesentlich besser durch die Konzepte der MDA bedient werden und gilt als großes Potential.

Ein einzelnes Telefon auf der Erde wäre sinnlos – Tausende hingegen ergeben einen Sinn – und heutzutage hat schon fast jeder ein Mobiltelefon bzw. man ist schon verwundert wenn jemand keines hat. Was ich mit diesen Aussagen zeigen will ist die Notwendigkeit von Standards in der heutigen Informationstechnologie. Die Möglichkeit Daten aus verschiedenen Systemen auszutauschen und Verständnis unter einzelnen Applikationen zu schaffen ist die Herausforderung von Standards. Die Verwendung von UML, SysML, die Standardisierung von Metamodellen neben der neuen Definition von Standards werden notwendig sein, je älter die MDA wird

um deren Entwicklung voranzutreiben. Hier liegen sicher neben der Entwicklung von standardisierten Tools die Herausforderungen für die MDA.

2. These

Können Anforderungen durchgängig verfolgt werden? Existiert eine Technologie um eine komplette durchgängige Anforderungsverfolgung zu erhalten. Wie werden Abhängigkeiten zwischen Anforderungen veranschaulicht und nachgezogen.

Ad 2:

Nach über 100 Seiten die ich über MDA und Anforderungsverfolgung verfasst habe bezeichne ich diese These als erfüllt. Die dazu verwendete Technologie heisst SysML und bietet die Möglichkeiten in Modelle eigene Tags als Requirements einfließen zu lassen und somit Zusammenhänge durch den MDA Entwicklungsprozess zu verfolgen. Auch eine einheitliche Struktur zur Modellierung von Anforderungen und deren Abhängigkeiten untereinander wird mittels dieser Technologie zur Verfügung gestellt. Es wäre jedoch falsch SysML rein auf Anforderungsverfolgung zu reduzieren da es ebenso wie UML eine sehr mächtige Modellierungsumgebung zur detaillierten Entwicklung von Systemen bietet.

Die Probleme die im Rahmen der Anforderungsverfolgung entstehen beziehen sich auf die automatisierten Transformationen im MDA. In diesem Bereich muss oft noch manuell nachkorrigiert werden da insbesondere bei der Transformation von PIM auf PSM noch Probleme mit der Weiterverarbeitung durch Codegeneratoren entstehen.
Ebenso muss auf Konsistenz einzelner Ebenen im Entwicklungsprozess und ebenenübergreifend gearbeitet werden.

Mit Reqtify wurde ein Requirementsmanagementtool vorgestellt welches nicht komplett den Anforderungen einer total automatisierten Anforderungsverfolgung gerecht werden kann, jedoch sehr wohl mit SysML Elementen umgehen kann und somit eine sehr gute Möglichkeit zur Anforderungsverfolgung bietet. Dieses Tool ist auf Grund der einfachen Handhabung und den sehr brauchbaren Funktionen besonders für kleine Projekte empfehlenswert wo der Aufwand für manuelle

Anforderungsverfolgung oder besser rein manuelle Anforderungsverfolgung zu hoch wäre. Außerdem ist die graphische Darstellung der Abhängigkeiten in Reqtify gut umgesetzt.

Literaturverzeichnis:

[Alioth, 80, S.156f] Alioth, A.: Entwicklung und Einführung alternativer Arbeitsformen. Schriften zur Arbeitspsychologie, Band 27, Bern, Stuttgart,Toronto: Huber, 1980, zgl. Diss. Hochschule St. Gallen.

[Birchmaier 04] Model Transformations, Birchmaier Peter , Uni Zürich

[Blah 03] M.Blaha, Models of Models , JOOP 5 , 92/93

[Cockburn 2002] Cockburn Alistair, Agile Software Development. Boston: Addison – Wesely 2002

[Crow,96] Crow, K. (1996): Costumer-Focused Development with QFD

[Davis 93] Davis, Alan. Software Requirements: Objects, Functions, & States. Englewood Cliffs, N.J: Prentice-Hall Inc., 1993.

[EWMT 05] Bézivin, J., Jouault, F., Paliès, J.: Towards Model Transformation Design Patterns, In Proceedings of the First European Workshop on Model Transformation,Rennes (2005)

[Gottel/ Finkelstein 1994] O. C. Z. Gotel, A. C. W. Finkelstein, „An analysis of the requirements traceability problem“, 1st International Conference on Requirements Engineering, pp. 94-101, 1994

[Hofmann 00] Hofmann, H. F.: Requirements Engineering - A Situated Discovery Process. Gabler Edition Wissenschaft, Deutscher Universitäts-Verlag 2000

[IEEE 610.12 1990] IEEE Std 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology –D8ioescription

[IEEE 830 1998] IEEE Std 830 1998 IEEE recommended practice for software requirements specifications

[IQ_1] Wikipedia die freie Enzyklopedie Stand 12. Juni 2007
http://en.wikipedia.org/wiki/Traceability_matrix

[Kleppe, 03] MDA Explained , The Model driven Architecture, Practice and Promise Anneke Kleppe, Jos Warmer, Wim Blast Addison - Wesley 2003

[Kontio, 98] J. Kontio and 0. Pitkinen, Commitment Management in Software Projects and Contracts 1998. Technical Report B- 138/98, ISBN 95 1-22-3959-O. Helsinki University of Technology. Bspoo.

[Kotonya / Sommerville 98] Requirements Engineering: Processes and Techniques. Chichester, John Wiley & Sons 1998

[Liu. X.F 01] Software Quality Function Deployment, Liu X.F. Jan 2001, published in Potentials, IEEE Volume 19 Issue 5 on Pages 14-16

[M.C. Paulk, B. Curtis, M.B. Chrissis, C.V. Weber, 93] “Capability Maturity Model for Software”, Version 1.1, Technical Report, CMU-SEI-93-TR-024, February 1993

[Maciasek 01] L.A Maciasek “Requirements Analysis and System Design – Developing Information Systems with UML” Addison Wesley 2001

[MDA Tools 05] Designermodelle , Bernhard Merkle, Heise ix5 2005 Magazin http://www.heise.de/ix/artikel/2005/05/102/ix0505106.tabelle.pdf Stand 13.02.07

[OMG 01] MDA Guide Version 1.0.1 Joaquin Miller and Jishnu Mukerji OMG Document Number omg/2003-06-01 Released: 12th June 2003 http://www.omg.org/mda/presentations.htm Stand 13.02.06

[OMG 02] Roles in MDA, Stephen J Mellor Andrew Watson, http://www.omg.org/mda/presentations.htm Stand 13.02.07

[OMG 03] Developing in OMG’s Model-Driven Architecture Jon Siegel and the OMG Staff Strategy Group Object Management Group White Paper November, 2001 http://www.omg.org/mda/presentations.htm Stand 13.02.07

[Petratsch, Meinberg 06] Roland Petrasch Oliver Meimberg Model-Driven Architecture. Eine praxisorientierte Einführung in die MDA Dpunkt Verlag Juni 2006

[QFD Yoji Akao 88] Yoji Akao, QFD Quality Function Deployment – Wie die Japaner Kundenwünsche in Qualittätsprodukte umsetzen, Japans Standard Association

[Ramesh / Jarke 01] Toward reference models for requirements traceability/ Ramesh, B.; Jarke, M.; IEEE Transactions on Volume 27, Issue 1, Jan. 2001 Page(s):58 - 93 Digital Object Identifier 10.1109/32.895989

[Ramesh et al. 95] B. Ramesh, T. Towers, C. Stubbs, M. Edwards, Implementing Requirements Tracability: A Case Study, IEEE, 1995

[Reqtify 04] Pohl, Scheel 2004 , Toolvorstellung Reqtify, Technische Universität Berlin

[Shaham-Gafni, Hartman 05] Shaham-Gafni, Y. and A. Hartman, ModelWare Traceability: A Survey. 2005, IBM Haifa Research Lab.

[SysML 05] Systems Modeling Language (SysML) Specification version 1.0 alpha SysML Partners (www.sysml.org)

[TNI 03] Reqtify V2 User Manual www.tni-software.com Stand 13.02.07

[TNI Software] Fraser Chatburn , Artisan Softwaretools , 2006 www.tni-software.com 13.02.07

[Watkins/ Neal 94] Watkins, R.; Neal, M.; Why and how of requirements tracing Software, IEEE Volume 11, Issue 4, July 1994 Page(s):104 – 106

[Wieringa 96] Requirements Engineering; Frameworks for Understanding. Chichester, John Wiley & Sons 1996