Modellbasierte Softwareentwicklung:

Vom Requirements Engineering bis zum Testen

Dehla Sokenou, Erwin Tratar

GEBIT Solutions GmbH
Koenigsallee 75 b
14193 Berlin
www.gebit.de
[Dehla.Sokenou | Erwin.Tratar] (at) gebit.de

Abstract: Das Papier stellt einen Ansatz des modellbasierten Requirements Engineerings in der betrieblichen Anwendungsentwicklung vor, der eine durchgängige modellbasierte Softwareentwicklung von den Anforderungen bis hin zum Testen erlaubt. Dabei werden die Vorteile gegenüber der herkömmlichen Erfassung von Anforderungen beleuchtet und Erfahrungen mit dieser Technik vorgestellt.

1 Motivation

Modellbasierte Ansätze haben in einigen Bereichen der Softwareentwicklung zu einer deutlichen Verbesserung geführt, betrachtet man Produktivität, Kosten oder Zeit [Kra05]. Der Vorteil einer modellbasierten Entwicklung liegt in der durchgängigen Verwendung eines Modells, das von allen Beteiligten in einem Projekt verwendet wird. Dieses Modell dient oft als Grundlage für den Code (durch Generierung oder Ausführung) und bringt damit einen entscheidenden Zeitgewinn. Gleichzeitig ist das Modell Teil der Dokumentation der Software, die durch die Verbindung zum Code immer auf dem aktuellen Stand ist und damit Inkonsistenzen zwischen Dokumentation und erstelltem Softwareprodukt vermeidet.

Unterstützt wird die modellbasierte Entwicklung durch eine Reihe verfügbarer Werkzeuge (z.B. Eclipse), die durch Integration von (UML-) Modellierungs- und Implementierungsumgebung und Unterstützung durch Generierungswerkzeuge eine einheitliche Basis für die Softwareentwicklung schaffen. Ein Wechsel des Werkzeugs zwischen Entwurfs- und Implementierungsphase ist somit nicht mehr nötig.

Leider beschränkt sich – anders als z.B. in der Domäne der eingebetteten Systeme (siehe z.B. [Sol05,Ilic05,Sch05]) – der modellbasierte Ansatz in der Domäne der betrieblichen Anwendungssysteme meist auf die Phasen Design und Implementierung, in einigen Fällen wird auch der Test modellbasiert durchgeführt. Dabei besteht ein ebenso großes Nutzungspotential in der Anforderungsermittlung. Wir schlagen deshalb eine Erweiterung des modellbasierten Ansatzes auf das Requirements Engineering vor (*Model-Driven Requirements Engineering*, kurz *MDRE*, siehe auch [Tra05,Kra05,Kra06]).

Im folgenden Abschnitt geben wir einen Überblick über die Eckpfeiler der modellbasierten Entwicklung und ihre Übertragbarkeit auf das Requirements Engineering. In Abschnitt 3 zeigen wir die Vorteile des modellbasierten Requirements Engineering gegenüber dem Status Quo in der betrieblichen Anwendungsentwicklung und unsere Erfahrungen beim Einsatz der neuen Technik.

2 Modellbasierte Entwicklung

Der modellbasierten Entwicklung liegt die Idee zugrunde, möglichst viele Informationen über ein Softwareprojekt in einem – möglichst graphischen – Modell zu erfassen und anschließend benötigte Artefakte direkt aus dem Modell zu verwenden oder zu generieren. Dabei lassen sich unabhängig von der konkreten Ausprägung (z.B. MDA, MDD, MDSE) einige Eckfeiler der modelbasierten Entwicklung ausmachen:

- Ein Metamodell gibt die verfügbaren Artefakte, deren Eigenschaften und Beziehungen zu anderen Artefakten vor.

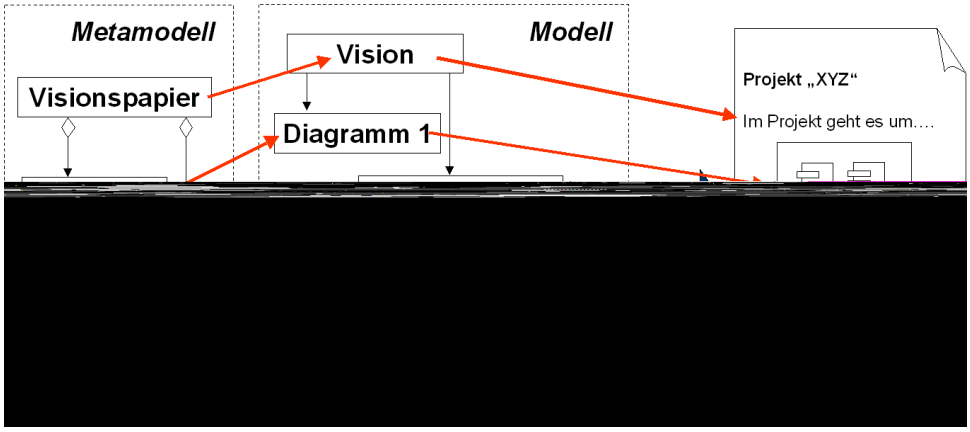

Abbildung 1: Requirements-Metamodell (links) und Dokumentenmodell (Mitte) und Dokument (rechts)

- Die Systembeschreibung stützt sich auf diese Artefakte und wird in einem Modell durchgeführt, dessen Struktur durch das Metamodell festgelegt ist.

- Es wird versucht, die Beschreibung möglichst frei von Redundanzen vorzunehmen. Eine Information, die an mehreren Stellen benötigt wird, wird entweder aus dem Modell generiert oder aus anderen Teilen des Modells entnommen.

- Änderungen am Modell haben direkte Auswirkung in der durch das Modell beschriebenen Anwendung. Dieser Bezug lässt sich auf unterschiedliche Weise realisieren, z.B. durch Modelltransformationen, Generierung von Quellcode aus dem Modell oder durch unmittelbare Verwendung von Modellinformationen zur Laufzeit (Repository-basierter Ansatz, executable UML).

Diese Eckpfeiler lassen sich ohne Weiteres auch auf das Requirements Engineering übertragen:

- Ein Metamodell gibt vor, welche Dokumente für die Anforderungen zu verwenden sind. Das können z.B. Glossar, Projektübersicht, Use-Case-Paket, Use-Case-Spezifikation, Testfall, Testplan, Change Request usw. sein. Auch Verweise auf andere Artefakte, z.B. UML-Diagramme, können definiert werden, z.B. dass eine Use-Case-Beschreibung einen Verweis auf ein Use-Case-Diagramm enthalten kann oder enthalten muss. Abbildung 1 zeigt als Beispiel auf der linken Seite ein Metamodell des TREND/Analyst[1], einem Werkzeug zur modellbasierten Erfassung von Anforderungen, das von GEBIT entwickelt wurde. Das Metamodell ist wie das UML-Metamodell in UML spezifiziert und kann für jedes Projekt individuell angepasst werden.

- Anforderungen werden strukturiert gemäß dem Metamodell erfasst. Für jedes Artefakt gibt es eine Vorlage oder eine Eingabemaske. Dadurch sind Aussehen, die Verknüpfung mit anderen Dokumenten und Diagrammen und weitere Eigenschaften fest vorgegeben und für alle Projektbeteiligten gleich. In Abbildung 1 ist dies in der Mitte und auf der rechten Seite dargestellt.

- Durch Verknüpfungen zu anderen Dokumenten kann das Anforderungsmodell frei von Redundanzen gehalten werden.

- Für den weiteren Softwareentwicklungsprozess lassen sich die Artefakte aus dem Anforderungsmodell weiterverwenden oder neue Artefakte (z.B. Templates für den Code oder für Testfälle) generieren.

Die Frage stellt sich natürlich, ob das auch zu Vorteilen im Softwareentwicklungszyklus führt. Diese Frage beleuchten wir im folgenden Abschnitt, indem wir das modellbasierte Requirements Enginering mit dem Status Quo der Anforderungsermittlung in der betrieblichen Anwendungsentwicklung vergleichen.

---

[1] TREND/Analyst ist in einer kostenfreien Community Edition verfügbar: www.gebit.de.

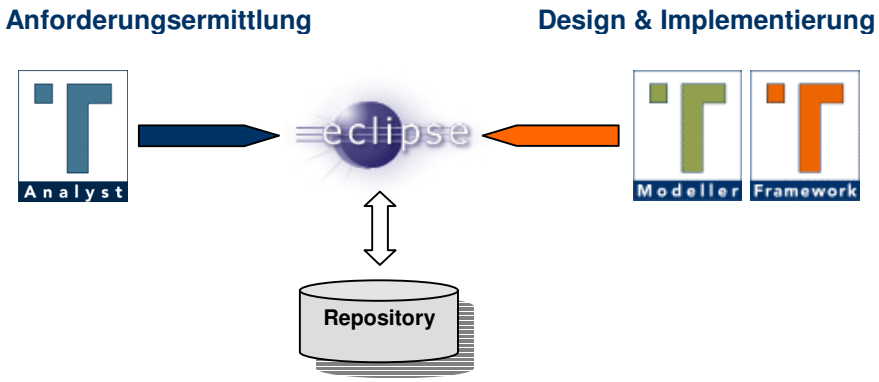

Abbildung 2: Integration durch modellbasierte Entwicklung[2]

3 Modellbasiertes Requirements Engineering

In diesem Abschnitt sollen die Vorteile einer modellbasierten Vorgehensweise in der Anforderungsermittlung aufgezeigt werden. Zunächst geben wir jedoch einen Überblick über den Status Quo im Requirements Engineering.

3.1 Status Quo im Requirements Engineering

Gängige Softwaresysteme zur Erfassung von Anforderungen (ein Überblick findet sich in [Ver05]) werden überwiegend in internationalen Großunternehmen eingesetzt, bei denen die Erfassung von Anforderungen eng mit gesetzlichen Regelungen verknüpft ist oder es eine enge Zusammenarbeit zwischen Herstellern und Zuliefern gibt (z.B. Pharma-, Verkehrs- oder militärische Industrie).

Anders sieht es dagegen im Bereich der klassischen betrieblichen Softwareentwicklung aus. Hier sind diese Systeme eher selten anzutreffen. Stattdessen werden Anforderungen häufig mit mehr oder weniger strukturell vorgegebenen Word-Dokumenten erfasst, wobei die vorgegebene Struktur ohne Probleme vom Anwender verändert werden kann. Dies führt häufig dazu, dass die so entstandenen Dokumente nicht mehr einheitlich sind und von Außenstehenden nur schwer verstanden werden können. Auch andere Aspekte sprechen gegen diese Praxis, z.B. dass eine Auswertung oder gar die automatische Ableitung von neuen Artefakten aus diesen Textdokumenten nur sehr schwer bis unmöglich sind.

Ein Fortschritt auf diesem Gebiet ist die Verwendung von graphischen Modellierungssprache wie z.B. der UML. Typischerweise werden hier Use-Cases verwendet, an denen die Anforderungen in Form von textuellen Beschreibungen hinterlegt sind. Dadurch wird der Medienbruch zur Anwendungsentwicklung vermieden und alle Projektbeteiligten sprechen die gleiche Sprache. Auch Verknüpfungen zu anderen Diagrammen wie Aktivitäten sind leicht möglich. Doch ein UML-Werkzeug ist nicht primär auf die Erfassung von Anforderungen ausgerichtet. So wird einerseits der Anwender auf fachlicher Seite nicht ausreichend geführt, andererseits können sich vermeintliche Kleinigkeiten wie die Sortierung der Ausgabedokumente zu einem großen Problem ausweiten. So gibt es z.B. bei den Use-Cases in einem Diagramm keine ersichtliche Sortierung, die aber bei der Ausgabe in einem Pflichtenheft meist gewünscht ist. Eine Nummerierung der Use-Cases als Abhilfe wird spätestens dann zum Problem, wenn ein neuer Use-Case zwischen zwei bestehenden eingefügt werden muss.

3.2 Vorteile des modellbasierten Ansatzes

Betrachtet man die im vorherigen Abschnitt beschriebenen Probleme, so liegt es nahe, einerseits die Vorteile des Ansatzes der UML-basierten Erfassung von Anforderungen beizubehalten, ohne die Nachteile in Kauf zu

[2] Eclipse und das Eclipse-Logo sind Markenzeichen der Eclipse Foundation (www.eclipse.org).

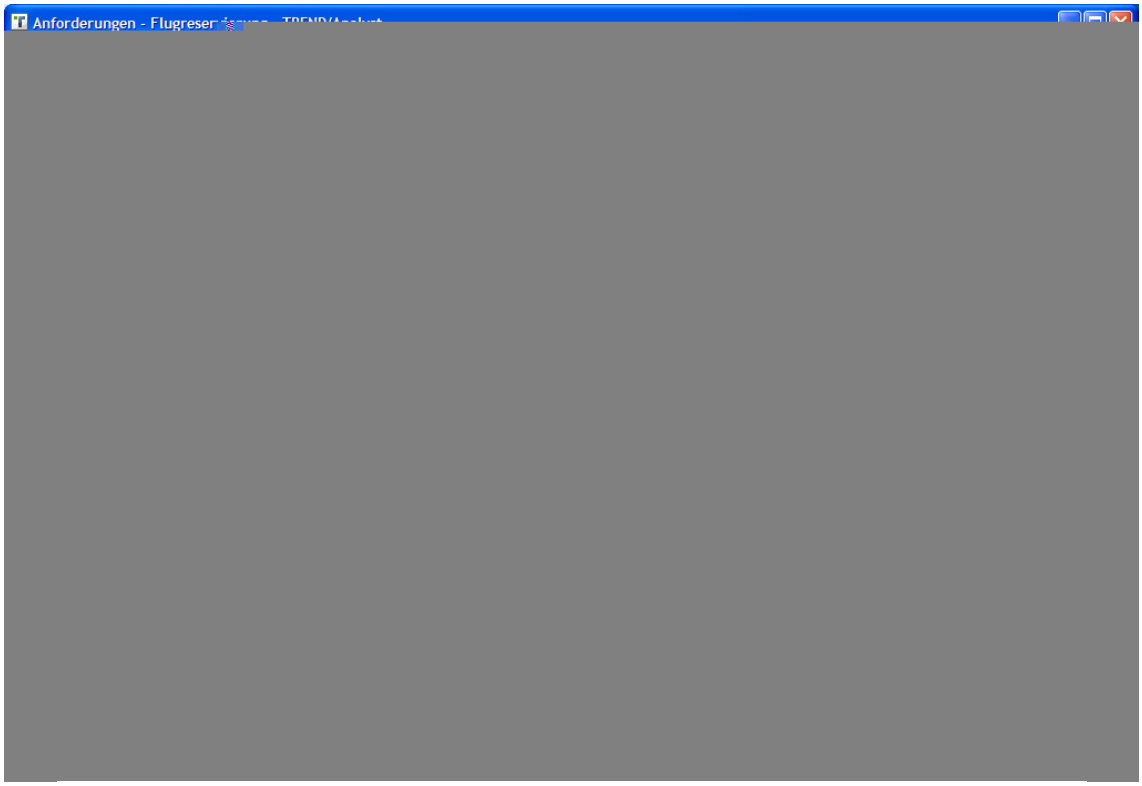

Abbildung 3: Beispiel für modellbasierte Anforderungserfassung

nehmen. Unser Ansatz ist es deshalb, sowohl die Erfassung von Anforderungen als auch die eigentliche Softwareentwicklung auf eine gemeinsame Basis zu stellen. Dabei sollten alle verwendeten Werkzeuge Hand in Hand arbeiten, was zum Beispiel durch die Verwendung eines gemeinsamen Repositories realisiert sein kann. Im Folgenden gehen wir auf die Vorteile des modellbasierten Requirements Engineering im Einzelnen ein.

Gemeinsame Sprache

Einer der wichtigsten Vorteile ist sicher die gemeinsame Sprache zwischen Analyse- und Entwicklungsteam, da neben textuellen Anforderungsdokumenten auch standardisierte Diagramme wie UML und BPMN-Diagramme verwendet werden können. Dies wird zudem gefördert, wenn das Werkzeug für die Anforderungserfassung in die normale Softwareentwicklungsumgebung integriert wird, ähnlich wie die Integration der Modellierungswerkzeuge die modellbasierte Entwicklung gefördert hat. Wird im Unternehmen bereits modellbasiert entwickelt, so kann nun der gesamte Softwareprozess von den Anforderungen bis hin zum Test in gleicher Weise erfolgen.

Die Integration in eine gemeinsame Werkzeugumgebung bietet zudem den Vorteil, dass der Entwickler unmittelbaren Zugriff auf die Anforderungsdokumente besitzt. So kann der Entwickler durch eigene Verknüpfungen von z.B. Aktivitäten zu einem Use-Case das Modell ergänzen.

Die Benutzung eines gemeinsamen Repositories hat mehrere Vorteile. Neben der Versionskontrolle, die ja für beide Seiten selbstverständlich sein sollte, können auch Aspekte wie der projektbezogene Zugriff auf einzelne Dokumente berücksichtigt werden. Zudem arbeiten alle Seiten immer auf dem gleichen Stand, so dass Änderungen im Modell schnell auf Entwicklerseite verfügbar sind.

Auch eine Verknüpfung von Dokumenten mit einem Issue-Tracking-System ist möglich. Dabei werden initial erzeugte Artefakte aus der Anforderungsdefinition als Issues angelegt und können anschließend vom Entwickler weiter bearbeitet werden. Dies ist auch nur möglich, da die strukturierten Informationen aus den Anforderungsdokumenten einfach auf die Felder im Issue-Tracking-System gemappt werden können.

Führung des Anwenders

Durch die Definition eines Metamodells zur Beschreibung der Anforderungsdokumente unterliegen alle Dokumente einer festgelegten Struktur. Zu jedem Dokument sind die Beziehungen zu anderen Dokumenten klar vorgegeben.

Beides führt dazu, dass der Anwender effektiv geführt wird, vom Anlegen eines initialen Projektdokuments bis zur Pflege von Change Requests. In den Dokumenten werden typischerweise strukturierte Daten erfasst, deren Eingabe durch Formularfelder unterstützt werden (siehe Abbildung 3). Die Eingabefelder leiten sich aus den Beschreibungen im Metamodell ab und können je nach Definition strukturierten Text oder Verweise auf andere Dokumente, aber auch z.B. auf UML-Diagramme. Ein Beispiel dafür ist z.B. eine Use-Case-Spezifikation, die strukturierten Text sowie einen Verweis auf das Use-Case-Diagramm enthält, in dem der referenzierte Use-Case spezifiziert wurde.

Im Modell selbst werden nur die eingegebenen Werte abgelegt (z.B. Texte, Zahlen, Verweise auf andere Dokumente), so dass Struktur (Metamodell) und Daten (Modell) streng voneinander getrennt sind.

Ableitung anderer Artefakte

Der wohl größte Vorteil der modellbasierten Entwicklung ist die Ableitung anderer Artefakte aus den vorhandenen. Typischerweise wird aus einer graphischen Beschreibung, z.B. einem UML-Modell, Code erzeugt. Doch auch aus Anforderungsbeschreibungen lassen sich andere Artefakte ableiten.

So lassen sich der Anforderungsbeschreibung verschiedene Arten von neuen Dokumenten ableiten. Ein Pflichtenheft lässt sich z.B. aus einem Artefakt vom Typ „Visionspapier“, gefolgt von den einzelnen Anforderungen aus den Artefakten vom Typ „Use-Case-Spezifikation“ erzeugen. Genauso leicht lassen sich weitere Dokumente erzeugen, z.B. die ausführlichen Spezifikationen einzelner Use-Cases, eine Übersicht aller Use-Cases mit ihrem aktuellen Status einer Projektphase oder auch Testfallbeschreibungen. Nicht in jedem Dokument müssen alle Daten aus den jeweiligen Artefakten enthalten sein.

Darüber hinaus können Anforderungsdokumente zur Erzeugung einer Projektstruktur und erster Designdokumente und Code-Artefakte für die nachfolgenden Entwicklungsphasen dienen. Dabei gelten die gleichen Prinzipien wie im modellbasierten Entwicklungsansatz.

Verfolgung im Modell

Auch für den Bereich des Requirements- und Projektmanagement ergeben sich aus dem modellbasierten Ansatz Vorteile. Metriken und Auswertungen sind auf der Basis von strukturiert erfassten Anforderungen leicht zu berechnen. So lässt sich für ein Artefakt die Eigenschaft „Status“ oder „Fertigstellungsgrad“ definieren, die zur automatischen Auswertung des aktuellen Projektstatus dient. Durch die Anbindung eines Issue-Tracking-Systems kann der Status einer Anforderung ins Modell zurückverfolgt werden, indem z.B. der Entwickler den Status eines Issues auf „bearbeitet“ oder „geschlossen“ setzt. Dabei kann der Status im Modell vom Status im Issue-Tracking-System abweichen und durch ein Mapping zugeordnet werden. Auf Basis dieser Informationen lassen sich nun leicht Projektfortschrittsdokumente extrahieren oder Abfragen nach noch nicht umgesetzten Anforderungen definieren.

Dadurch, dass der Entwickler textuelle Anforderungsdokumente mit erstellten UML-Diagrammen wie Aktivitäten verknüpfen kann, ist eine leichte Verfolgbarkeit von den Anforderungen zu Design und Code und zurück möglich. So lässt sich leicht feststellen, welche Auswirkungen eine Änderung von Anforderungen auf das bereits modellierten und implementierte System hat.

3.3 Erfahrungen mit modellbasiertem Requirements Engineering

Unsere Erfahrungen mit dem modellbasierten Requirements Engineering zeigen, dass die Integration auch des Requirements Engineering in den modellbasierten Softwareentwicklungsprozess möglich ist und damit eine Durchgängigkeit von der Anforderungsermittlung bis zur Implementierung bietet. Werden zudem modellbasierte Techniken im Test eingesetzt, wird auch diese Phase in den durchgängigen Prozess integriert.

Der vorgestellte Ansatz ist bereits in Projekten verwendet worden. Dabei reicht die Spannbreite von Projekten klassischen Softwareprojekten über Projekte mit verteilter Entwicklung, wo besonders die problemlose Zusammenarbeit über die repository-basierten Speicherung der Dokumente eine Rolle spielt, bis hin zu Projekten, die mit agilen Prozessen durchgeführt werden.

Gerade bei agilen Prozessen, bei denen eine konstante Änderung und Weiterentwicklung des Programms im Mittelpunkt steht, und damit auch eine ständige Änderung und Weiterentwicklung der Anforderungen, hat ein modellbasierter Ansatz viele Vorteile. Zu Projektbeginn liegen typischerweise noch nicht alle Anforderungen vor, stattdessen werden diese nach und nach ergänzt und verfeinert, während gleichzeitig bereits entworfen und implementiert wird. Der einfache und schnelle Zugriff auf die jeweils aktuellen Anforderungsdokumente sowie die leichte Verfolgbarkeit von Anforderungen im weiteren Prozess sind Änderungen für den Entwickler leicht nachzuvollziehen.

Ergänzt wird das modellbasierte Requirements Engineerung durch modellbasiertes Design und Implementierung. Verwendet wird eine Single-Source-Strategie, so dass Modell und Implementierung immer konsistent zueinander sind. Dadurch lassen sich die Requirements bis zum Code verfolgen.

Allerdings erfordert der modellbasierte Ansatz eine andere Arbeitsweise als das übliche Vorgehen bei der Anforderungserfassung. Durch das Metamodell sind die Strukturen klar vorgegeben, was eine strukturierte Arbeitsweise voraussetzt. Ein einfaches Verschieben von Textblöcken ist schwieriger als in einem monolithischen Textdokument. Gleichzeitig ist durch die Anpassung des Metamodells eine Flexibilität gegeben, die eine Anpassung an die eigene Arbeitsweise im Grenzen möglich macht.

Zudem muss ein Werkzeug für die modellbasierte Anforderungserfassung von allen am Projekt Beteiligten, die Anforderungen erfassen oder Anforderungsdokumente verändern, akzeptiert und benutzt werden.

4 Zusammenfassung

Mit dem modellbasierten Requirements Engineering werden Prinzipien der modellbasierten Entwicklung konsequent auf die Anforderungsermittlung ausgedehnt. Wichtig ist eine homogene Werkzeuglandschaft, die von Anforderungsermittlern und Entwicklern gemeinsam benutzt wird und so die Verfolgbarkeit von Anforderungen im gesamten Prozess ermöglicht. Durch die Generierung von Dokumenten wie Pflichtenheft oder Projektfortschrittsdokument sind diese immer auf dem aktuellen Stand. Bei Änderung der Struktur der Anforderungen oder auch des Formats dieser generierten Dokumente müssen diese nicht mehr manuell angepasst werden, sondern die Anpassung erfolgt nur im Metamodell bzw. in der Beschreibung der Transformation. Ein solcher Ansatz kann die Qualität der Anforderungen gerade in Bereichen verbessern, in denen Anforderungen bisher informell erfasst wurden.

Literaturverzeichnis

[Ilic05] Ilic, D.; Troubitsyna, E.: A Formal Model-Driven Approach to Requirements Engineering. Technischer Report, Nr. 667, Turku Centre for Computer Science (TUCS), 2005.

[Kra05] Krauss, T: Durchgängige Modellorientierung macht Anwendungsentwicklung produktiver. OBJEKTspektrum, Nr. 2/2005, S.1-4.

[Kra06] Krauß, T; Versteegen, G.; Mühlbauer, S..: Model-Driven Requirements Engineering. Informatik Spektrum, Nr. 6/2006, S. 460-464.

[Sch05] Schätz, B.; Fleischmann, A.; Geisberger, E.; Pister, M.: Modellbasierte Anforderungsentwicklung mit AutoRAID, Object-Orientied Software-Engineering (OOSE), Net.ObjectDays-Conference, 2005.

[Sol05] Solheim, H.; Lillehagen, F.; Petersen, S.A.; Jorgensen, H.; Anastasiou, M.: Model-driven visual requirements engineering. 13th IEEE International Conference on Requirements Engineering, 2005.

[Tra06] Tratar, E.: Anforderungen erfassen, aber mit System – Modellbasiertes Requirements Engineering. OBJEKTspektrum, Nr. 6/2006, S.29-31.

[Ver05] Versteegen, G (Hrgs.); Hood, C.; Kreß, A.; Stevenson, R.; Wiebel, R.: iX-Studie Anforderungsmanagement – Methoden und Techniken, Einführungsszenarien und Werkzeuge im Vergleich. Heise Verlag, 2005.