Qualitätsziel-orientierter Architekturentwurf und Traceability für weiterentwickelbare Software-Systeme

Zusammenfassung zur Dissertation von

Stephan Bode

Technische Universität Ilmenau
Stephan.Bode@arcor.de

Abstract: Die Evolution von Softwaresystemen erfordert häufige Anpassungen z. B. aufgrund sich ändernder Geschäftsprozesse oder Technologien. Bisherige Methoden unterstützen dies nur unzureichend aufgrund mangelnder Berücksichtigung von Qualitätszielen wie Weiterentwickelbarkeit sowie mangelnder Nachvollziehbarkeit von Architekturentwurfsentscheidungen. Das neue Konzept *Goal Solution Scheme*, das Qualitätsziele über Architekturprinzipien auf Lösungsinstrumente durch explizite Abhängigkeiten abbildet, hilft geeignete Architekturlösungen entsprechend ihrem Einfluss auf die Qualitätsziele wie Weiterentwickelbarkeit auszuwählen und Entwurfsentscheidungen nachzuvollziehen. Das Schema ist in ein zielorientiertes Architekturentwurfsvorgehen eingebettet, das etablierte Methoden und Konzepte des Requirements Engineering und Architekturentwurfs verbessert und integriert. Dies wird ergänzt durch ein Traceability-Konzept, welches eine (halb-)automatische Erstellung von Traceability Links mit hoher Genauigkeit und Trefferquote ermöglicht. Die Realisierbarkeit des Entwurfsansatzes wurde mit einer Fallstudie eines Softwaresystems für mobile Serviceroboter gezeigt. Ein prototypisches Werkzeug names EMFTrace zeigt die Anwendbarkeit der Konzepte.

1 Einleitung

Softwaresysteme werden heute mit häufigen Forderungen nach Veränderungen konfrontiert, z. B. aufgrund sich ändernder Geschäftsprozesse oder Technologien. Die Software und speziell ihre Architektur muss mit diesen häufigen Änderungen zurecht kommen, um dauerhaft nutzbar zu bleiben, da eine Ablösung existierender Softwaresysteme durch Neuentwicklungen wegen der damit verbundenen Risiken gewöhnlich keine akzeptable Lösung darstellt.

Während der Software-Evolution können Änderungen zu einer Strukturverschlechterung der Architektur führen, der Architekturerosion. Dies erschwert weitere Änderungen aufgrund von Inkonsistenzen oder fehlendem Programmverstehen oder verhindert sie gar. Um Änderungen zu unterstützen und die Erosion zu vermeiden, müssen speziell Qualitätsziele wie Weiterentwickelbarkeit, Performanz oder Gebrauchstauglichkeit sowie die Nachvollziehbarkeit von Entwurfsentscheidungen beim Architekturentwurf berücksichtigt werden. Dies wird jedoch oft vernachlässigt.

Existierende Entwurfsmethoden unterstützen den Übergang von den Qualitätzielen zu geeigneten Architekturlösungen nur unzureichend, da immer noch eine Lücke zwischen Methoden des Requirements Engineering und Architekturentwurfs existiert. Insbesondere fehlt Unterstützung für die Weiterentwickelbarkeit und die Nachvollziehbarkeit von Entwurfsentscheidungen durch explizite Modellabhängigkeiten.

Diese Arbeit präsentiert drei neue miteinander verzahnte Konzepte. Zuerst wird das *Goal Solution Scheme* vorgestellt, das Qualitätsziele über Architekturprinzipien auf Lösungsinstrumente durch explizite Abhängigkeiten abbildet. Das Schema hilft geeignete Architekturlösungen entsprechend ihrem Einfluss auf die Qualitätsziele auszuwählen. Es wird besonders hinsichtlich Weiterentwickelbarkeit diskutiert. Zum zweiten wird das Schema in ein *zielorientiertes Architekturentwurfsvorgehen* eingebettet, das etablierte Methoden und Konzepte des Requirements Engineering und Architekturentwurfs verbessert und integriert. Drittens wird dies ergänzt durch ein *Traceability-Konzept, welches einen regelbasierten Ansatz mit Techniken des Information Retrieval verbindet*. Dies ermöglicht eine (halb-)automatische Erstellung von Traceability Links mit spezifischen Typen und Attributen für eine definierte Semantik sowie mit hoher Genauigkeit und Trefferquote.

Die Realisierbarkeit des Ansatzes wird an einer Fallstudie eines Softwaresystems für mobile Serviceroboter gezeigt. Ein prototypisches Werkzeug names EMFTrace wurde als eine erweiterbare Plattform basierend auf Eclipse-Technologie implementiert, um die Anwendbarkeit der Konzepte zu zeigen. Es integriert Entwurfsmodelle von externen CASE-Werkzeugen mittels XML-Technologie in einem gemeinsamen Modell-Repository, wendet Regeln zur Linkerstellung an und bietet Validierungsfunktionen für Regeln und Links.

2 Bewertung des Standes der Technik

Im folgenden wird ein kurzer Abriss zur Bewertung von Methoden und Konzepten des Standes der Technik gegeben, die die Entwicklung der drei neuen Konzepte dieser Arbeit maßgeblich beeinflusst haben.

Beschreibung funktionaler und qualitativer Anforderungen Für die Analyse, Verfeinerung und grafische Repräsentation von Qualitätszielen und ihren Beziehungen untereinander können z. B. das NFR Framework [CNYM00], das i* Framework [Yu95] oder die standardisierte User Requirements Notation (URN) [ITU08] genutzt werden. Obwohl diese Ansätze des Anforderungsmanagements gut zur Darstellung der Beziehungen zwischen Qualitätszielen geeignet sind, haben sie aus Architektursicht einige Nachteile und Beschränkungen. Aufgrund ihres Ursprungs zielen sie eindeutig auf die Anforderungsanalysephase. Daher berücksichtigen sie nur unzureichend Architekturprinzipien und technische Randbedingungen und helfen wenig beim zielgerichteten Architekturentwurf mittels Vorschlag geeigneter Lösungsinstrumente. Aber die Ansätze können gut in Architekturentwurfsmethoden eingebettet werden, um Nutzen aus ihnen zu ziehen.

Evolutionsunterstützung durch Architekturentwurfsmethoden Für den Softwarearchitekturentwurf existieren einige etablierte Methoden wie QASAR [Bos00], ADD [BKB02] oder Globale Analyse [HNS00], welche in ein allgemeines Modell mit den Phasen Analyse, Synthese und Evaluierung [HKN+07] passen. Die Methoden versuchen gewöhnlich Qualitätsziele und funktionale Anforderungen mittels Architekturmustern [HA07] oder sogenannten Taktiken [BKB02] auszubalancieren. Jedoch fehlt ein systematischer, zielgerichteter Ansatz mit Anleitung zur Auswahl geeigneter Architekturentwurfselemente insbesondere für die Synthesephase und ausgerichtet auf Weiterentwickelbarkeit. Darüber hinaus fehlt die Integration mit den zielorientierten Ansätzen des Requirements Engineering (RE). Dies resultiert in einer immer noch vorhandenen konzeptionellen Lücke zwischen den Ansätzen der RE-Gemeinschaft sowie der Architekturentwurfsmethoden und schließlich in den wohl bekannten Evolutionsproblemen.

Entwurfs-Traceabiltiy Traceability, Rückverfolgbarkeit, ist ein Konzept, das das Nachvollziehen von Entwurfsentscheidungen ermöglichen kann. Sogenannte Traceability Links können verschiedene Softwareartefakte in Beziehung zueinander setzen und ermöglichen so die Verfolgung von Entwurfsentscheidungen sowie die Durchführung einer Änderungsauswirkungsanalyse. Ansätze der Requirements Traceability, z. B. [GF94, RJ01], erlauben es die Herkunft von Anforderungen und ihre Änderungen zurückzuverfolgen. Leider gibt es nur wenige Traceability-Ansätze für Entwurfsartefakte. Information Retrieval-basierte Ansätze, z. B. [ACC+02, CHSB+05], nutzen Ähnlichkeitsmaße auf Bezeichnern und können automatisiert werden. Allerdings ist die Genauigkeit und Trefferquote ihrer nachgelagerten Linkerstellung stark beschränkt. Konsequenterweise sollten Traceability Links bereits während des Entwurfs und der Änderung von Architekturmodellen erstellt werden. Daher müssen die Schritte zur Linkerstellung in die Entwurfsmethoden eingebettet werden. Regelbasierte Ansätze, z. B. [SZPMK04, MGP08], können Links halb-automatisch während der Durchführung von Entwurfsaktivitäten erzeugen, sodass der Entwickler nur bei Konflikten und Mehrdeutigkeit eingreifen muss. Die Korrektheit und Vollständigkeit der Links hängt hier hauptsächlich von der Qualität der Regeln ab. Leider gibt es bisher keine Ansätze, die in den Entwurfsprozess integriert sind und Links zwischen verschiedenartigen Artefakten, insbesondere Qualitätszielen und Entwurfselementen, erstellen können.

3 Begriff Weiterentwickelbarkeit

Obwohl es bereits einige Definitionen für Weiterentwickelbarkeit (engl. evolvability) gibt (z. B. [RLL98, BCE08]), existiert noch kein standardisiertes Qualitätsmodell und kein einheitliches Verständnis der Weiterentwickelbarkeit und insbesondere der Differenzierung zu Wartbarkeit. Breivold et al. [BCE08] diskutieren Weiterentwickelbarkeit im Detail, aber ihre Verfeinerung in Untermerkmale ist unvollständig und sie bieten keine Anleitung für eine zielgerichtete Berücksichtigung von Weiterentwickelbarkeit beim Architekturentwurf. Da der Begriff von zentraler Bedeutung ist für diese Arbeit und insbesondere die Diskussion im Rahmen des Goal Solutions Schemes, erfolgt hier eine Definition des

Qualitätsziels und eine Abgrenzung insbesondere zu Wartbarkeit, Wartung sowie Evolution (siehe auch [RB09]). Demnach ist Weiterentwickelbarkeit die Fähigkeit eines Softwaresystems sich während seiner Lebenszeit an Änderungen und Verbesserungen in Anforderungen und Technologien anzupassen, die die architekturelle Struktur des Systems beeinflussen, mit den geringsten Kosten unter Beibehaltung der Architekturintegrität.

4 Das Goal Solution Scheme

Wie bereits oben erwähnt ist die systematische Unterstützung für die Behandlung von Qualitätszielen wie Weiterentwickelbarkeit, Performanz oder Gebrauchstauglichkeit durch existierende Konzepte mangelhaft. Zur Unterstützung einer qualitätszielorientierten Entwicklung ist eine Abbildung zwischen Problemraum und Lösungsraum notwendig. Das Goal Solution Scheme (GSS) wurde entwickelt, um eine Abbildung zwischen Elementen beider Räume explizit modellieren und somit systematisch geeignete Architekturlösungen aus Qualitätszielen ableiten zu können. Aufgrund der verschiedenen Konzepte des Problem- und Lösungsraums besteht dazwischen eine konzeptuelle Lücke, welche zu Schwierigkeiten bei der Erstellung und Pflege der Beziehungen zwischen beiden führt. Um die Lücke zu schließen, wurden zwei zusätzliche Ebenen zwischen den Zielen und den Lösungsinstrumenten eingeführt, wie in Abbildung 1 zu sehen.

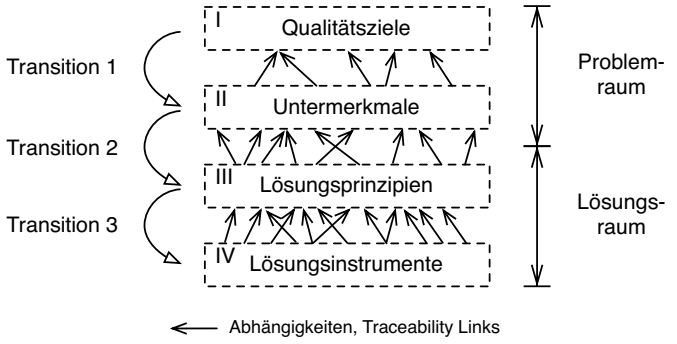

Abbildung 1: Struktur des Goal Solution Schemes.

Die Ebenen des Schemas entsprechen den Phasen während des Entwicklungsprozesses und enthalten die Elemente dieser Phasen. Ebene I und II repräsentieren den Problemraum, während die Ebenen III und IV den Lösungsraum darstellen. Von oben nach unten zeigt das Schema mögliche Verfeinerungen und Entscheidungen während des Architekturentwurfs und der Implementierung von Qualitätszielen. Die Beziehungen zwischen den Elementen des Schemas bilden einen Graph mit einer baumähnlichen Struktur. Jede Beziehung drückt eine Abhängigkeit aus, z. B. eine Verfeinerung von Zielen. Die Änderung eines Elementes erfordert die Änderung der abhängigen Elemente. Außerdem repräsentieren die Beziehungen zwischen den Ebenen einen positiven oder negativen Einfluss der Elemente

des Lösungsraums auf die Elemente des Problemraums. Gewichte auf den Beziehungen drücken den Einfluss quantitativ aus und werden für die Entscheidungsfindung verwendet. Die Abhängigkeiten können außerdem als Traceability Links repräsentiert werden, welche während des Entwurfs aufgezeichnet werden und die Entscheidungen reflektieren.

Ebene I ist Teil des Problemraums und beinhaltet die Qualitätsziele. Da es gewöhnlich schwieriger ist, die Qualitätsziele eines Softwaresystems zu realisieren als seine Funktionalität, sollten diese Qualitätsziele explizit modelliert werden, wie dies auch von den zielorientierten RE Ansätzen vorgeschlagen wird. Ebene I enthält die hochrangigen Qualitätsziele wie Weiterentwickelbarkeit, Performanz und Gebrauchstauglichkeit.

Ebene II ist auch noch Teil des Problemraums und enthält die Untermerkmale der hochrangigen Qualitätsziele, wie z. B. Änderbarkeit, Wiederverwendbarkeit und Testbarkeit. Diese Ebene wurde eingeführt, um die erwähnte Lücke zwischen Problem- und Lösungsraum mittels Zielverfeinerung zu verringern. Die Beziehungen der Transition 1 zwischen Ebene I und II repräsentiert eine Abbildung von hochrangigen Qualitätszielen zu Unterzielen ähnlich einem Qualitätsmodell. Zur Modellierung dieser Beziehungen kann die URN verwendet und auf Qualitätsmodelle wie ISO 9126 zurückgegriffen werden. Später können auch Präferenzen des Kunden für die Ziele auf den beiden Ebenen vergeben werden. Somit dient Ebene II zur Ausbalancierung der Ziele und zur Konfliktlösung.

Ebene III mit den Lösungsprinzipien wurde mit der Intention eingeführt, die Lücke zwischen Problem- und Lösungsraum zu schließen. Es ist viel einfacher den Einfluss von Prinzipien auf Qualitätsziele zu bestimmen als direkt den Einfluss von Lösungen, denn die meisten Prinzipien wurden entwickelt mit der klaren Absicht zur Unterstützung von Qualitätszielen. Dementsprechend enthält diese Ebene Lösungsprinzipien mit bekanntem Einfluss auf Qualitätseigenschaften einer Software. Lösungsprinzipien existieren in verschiedenen Forschungsgebieten. Typischerweise sind sie in Fachbüchern zu finden. Beispiele für Lösungsprinzipien der Softwaretechnik sind lose Kopplung oder Trennung von Zuständigkeiten. Der Übergang von Problem- zu Lösungsraum manifestiert sich in Transition 2 von Ebene II zu III. Hier erfolgt die Abbildung von Qualitätszielen oder Unterzielen zu Lösungsprinzipien. Die Beziehungen drücken den Einfluss der Prinzipien auf die Ziele aus. Da dieser Übergang zum Kern jeder Entwurfsmethode gehört, können entsprechende Konzepte auch dort gefunden werden, wie z. B. die Taktiken von ADD. Die Methode QASAR erwähnt die Notwendigkeit Qualitätsziele durch funktionale (technische) Lösungen umzusetzen, bietet aber kein Konzept zur Modellierung dessen. Die Einführung von Ebene III stellt eine wesentliche Leistung des GSS dar, weil Prinzipien eine bedeutende Rolle beim Entwurf spielen. Verglichen mit bisherigen Arbeiten wie den zielorientierten RE Ansätzen bedeutet die Prinzipienebene eine signifikante Verbesserung.

Ebene IV deckt die Lösungsinstrumente verschiedener Abstraktionsstufen ab, wie z. B. Architekturmuster, -stile, aber auch Frameworks und fertige Komponenten. Die Lösungsinstrumente sind beschrieben mit Vorbedingungen für ihre Anwendbarkeit und einer Menge von Einflusswerten bezüglich der Lösungsprinzipien und Qualitätsziele. Die Einflusswerte der Lösungsinstrumente werden mit den Beziehungen der Transition 3 zwischen Ebene III und IV abgebildet.

Als ein Ergebnis bietet das GSS die Ausrichtung von Entwurfsprinzipien und funktiona-

len Lösungen an Qualitätszielen. Mit den explizit modellierten Beziehungen klassifiziert es Lösungsinstrumente entsprechend ihrem Einfluss auf die Ziele. Das Schema erleichtert die Identifikation von Zielkonflikten sowie deren Lösung durch Priorisierung und Verfeinerung sowie durch die Abbildung auf Lösungsprinzipien und -instrumente. Die Lösungsinstrumente dienen als Quelle für Vorschläge von Entwurfsalternativen während des Entscheidungsprozesses basierend auf einer quantitativen Bewertung der Instrumente entsprechend ihres Einflusses. Das GSS fördert darüber hinaus die Erstellung von Traceability Links zwischen Zielen und Entwurfsartefakten. Das Goal Solution Scheme repräsentiert die signifikanteste Neuerung und Leistung der Arbeit. Es basiert auf Prinzipien des modellbasierten Entwurfs und kombiniert Ideen des zielorientierten RE mit Architekturentwurfsprinzipien und -aktivitäten. Es wurde in mehreren Fallstudien entwickelt und angewendet und im Detail für Weiterentwickelbarkeit beschrieben. Das GSS für Weiterentwickelbarkeit stellt eine konsolidierte Sicht auf das Qualitätsziel und seine Untermerkmale dar und bestimmt den Einfluss fundamentaler Entwurfsprinzipien auf das Ziel. Außerdem wird eine Menge von Entwurfsmustern bezüglich ihres Einflusses auf Weiterentwickelbarkeit bewertet. Somit wird das abstrakte Ziel Weiterentwickelbarkeit greifbarer und kann während des Entwurfs leichter umgesetzt werden.

5 Zielorientierter Architekturentwurf

Im Rahmen der Dissertation wurde das Goal Solution Scheme in eine zielorientierte Architekturentwurfsmethode (GOAD – Goal-oriented Architectural Design) eingebettet. Die GOAD-Methode fokussiert auf eine vorwärts gerichtete, iterativ inkrementelle Entwicklung mit Backlog-Konzept und folgt der allgemeinen Aufteilung in die drei Phasen Architekturanalyse, Architektursynthese und Architekturevaluierung (vgl. [HKN+07]). Die Methode überwindet Beschränkungen existierender Methoden und Konzepte indem es die besten Teile weiterentwickelt und zusammen mit dem Goals Solution Scheme integriert.

GOAD beginnt nach der Anforderungserhebung mit einer Zielmodellierung, welche von den zielorientieren Requirements Engineering Ansätzen adaptiert wurde, und nutzt die Zielpriorisierung zur Fokussierung des Entwurfs. Auf diese Weise wird Transition 1 des GSS unterstützt. Für die Architekturanalyse wurde die Methode Globale Analyse von Hofmeister et al. weiterentwickelt und integriert. Globale Analyse nutzt sogenannte Faktorentabellen und Themenkarten zur Analyse des Einflusses von Qualitätszielen und funktionalen Anforderungen sowie zur Ermittlung geeigneter Architekturstrategien anhand von Lösungsprinzipien. Für die Architekturstrukturierung während der Architektursynthese bietet GOAD Aktivitäten, die von der Attribute Driven Design (ADD) Methode adaptiert wurden. Dabei wird auf die Ausnutzung des GSS im Detail eingegangen.

Die qualitätszielorientierte Architekturentwurfsmethode und das Goal Solution Scheme gemeinsam ermöglichen einen leichteren Übergang vom Problemraum zum Lösungsraum (Tansition 2 von Ebene II zu III und weiter zu IV des GSS). Die Methode bietet Unterstützung für die Auswahl von Entwurfslösungen unter zu Hilfenahme der Themenkarten der Globalen Analyse sowie mittels der Einflusswerte der explizit modellierten Abhängigkeiten im GSS. Dieses Vorgehen berücksichtigt explizit Architekturprinzipien

und Entwurfsbeschränkungen und es unterstützt die Entscheidungsfindung mit einem Vorrat an Lösungsinstrumenten, welche mittels Fallstudien quantitativ hinsichtlich ihres Einflusses auf das Qualitätsziel Weiterentwickelbarkeit bewertet wurden. Für die Integration von ausgewählten Lösungsinstrumenten in die Gesamtarchitektur finden die von der QASAR Methode vorgeschlagenen Architekturtransformationsschritte Anwendung. Außerdem werden in GOAD zur Detaillierung der Architektur die Softwarekategorien der Quasar-Methode [Sie04] für eine verbesserte Trennung von Zuständigkeiten durch explizite Abhängigkeiten integriert. Weiterhin erfolgt in der Arbeit eine Diskussion über die bottom-up vs. top-down Strukturierung der Architektur bezüglich der Unterstützung der Weiterentwickelbarkeit der resultierenden Software. Die Anwendbarkeit der zielorientierten Architekturentwurfsmethode wird mittels einem Fallstudienprojekt nachgewiesen.

6 Traceability-Konzept und Toolunterstützung

Inspiriert durch die existierenden Ansätze von Cleland-Huang et al. [CHSB+05] und Spanoudakis et al. [SZPMK04], wurde ein Konzept zur (halb-)automatischen und regelbasierten Erstellung von Traceability Links entwickelt, was zusätzlich die Information Retrieval Technik n-Gram-Matching anwendet. Abbildung 2 zeigt das Konzept im Überblick.

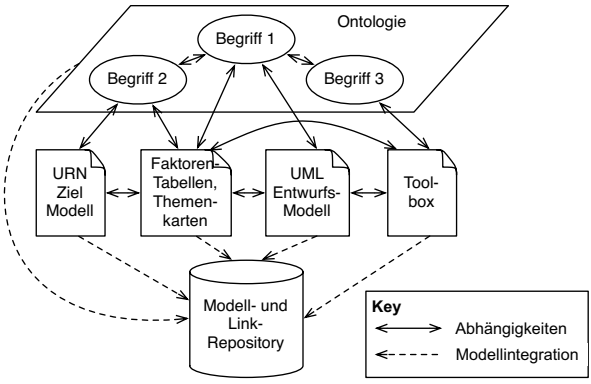

Abbildung 2: Überblick über das Traceability-Konzept

Artefakte die durch das Konzept verlinkt werden, sind a) Zielmodelle, die in URN notiert sind, b) Faktorentabellen und Themenkarten der Globalen Analyse, c) UML Modelle für die Entwurfsspezifikation, sowie d) Ontologien modelliert mit der Web Ontology Language (OWL) als semantisches Netz wichtiger Begriffe aus den verschiedenen Entwurfsphasen. Die Abhängigkeiten zwischen den Entitäten dieser Artefakte werden als Links mit einem spezifischen Typ zur Verbesserung der semantischen Ausdruckskraft der Links repräsentiert. Zu diesem Zweck wurden ein Traceability Metamodell entwickelt und Linktypen analysiert, geclustert und in einem Katalog gesammelt. Die Links werden zusammen mit den Entwurfsmodellen in einem gemeinsamen Modell-Repository abgelegt.

Das Traceability-Konzept trägt zum zerklüfteten Forschungsgebiet der Traceability-Ansätze bei, indem es verschiedene bereits existierende Ideen zu einem umfassenden Ansatz integriert und mehrere Entwurfsartefakte, die durch die GOAD-Methode verwendet werden, verknüpft, anstatt sich nur auf einen kleinen Ausschnitt zu beschränken. Insbesondere erfolgt die Verlinkung von Qualitätszielen bis hin zu Entwurfselementen wie UML-Komponenten über die konzeptuelle Lücke zwischen Problem- und Lösungsraum hinweg. Ziel des Konzeptes ist es die Entwurfsschritte zu verfolgen, die zu den Entwurfsartefakten geführt haben, um eine Auswirkungsanalyse für zukünftige evolutionäre Änderungen zu ermöglichen. Daher werden alle Modelle in einem Repository integriert und als eine Neuheit des Konzeptes mit Ontologiebegriffen verknüpft, um mehr explizite Abhängigkeiten nicht nur innerhalb sondern auch zwischen den Modellen zu ermöglichen.

Das erstellte Traceability Metamodell ist die Basis für Toolunterstützung. Ein Katalog konsolidierter Linktypen wurde ebenfalls modelliert, um ihn im Repository abzulegen. Als eine Neuheit des Konzeptes wird ein regelbasierter Ansatz für die (halb-)automatische Identifikation und Aufzeichnung von Links kombiniert mit der n-Gram-Matching Technik. Dies führt auf der einen Seite zu einem vertretbaren Aufwand für die Regeldefinition und resultiert auf der anderen Seite in einer hohen Korrektheit und Vollständigkeit der identifizierten Links. Die Regeln werden mittels XML Schema Definition (XSD) definiert und ebenfalls als Modell im Repository vorgehalten. N-Gram-Matching wird verwendet als Information Retrieval Technik zur Ermittlung von Ähnlichkeiten zwischen Modellelementbezeichnern und somit zur Erhöhung der Trefferwahrscheinlichkeit der Regeln.

Um die Anwendbarkeit des Traceabiilty Konzeptes zu zeigen, wurde ein Tool namens EMFTrace entworfen und implementiert. EMFTrace ist eine erweiterbare Plattform basierend auf Eclipse Technology. Das Tool setzt das Traceability-Konzept vollständig um und verknüpft existierende CASE-Tools, die der Erstellung der in der GOAD Methode verwendeten Artefakten dienen, wie jUCMNav für Zielmodelle, Visual Paradigm für UML-Modelle und Protegé für Ontologien. Zur Integration der Artefakte greift EMFTrace auf das Modell-Repository EMFStore zurück und bietet EMF-basierte Metamodelle für jedes Artefakt. Als wichtiger Vorteil gegenüber anderen Ansätzen ist hier die Verwendung von standardisierten Modellierungssprachen wie URN und UML zu nennen, denn die Standards sind eher stabil und selten Gegenstand von Änderungen. Die Traceability Links und Regeln werden ebenfalls mit EMF-basierten Metamodellen verwaltet. EMFTrace verwendet einen Regelinterpreter zur Anwendung der Regeln, einen Linkmanager zur Erzeugung von Traceability Links und bietet die Fähigkeit Ketten von Traceability Links zu identifizieren und zu verwalten sowie Konsistenzprüfungen zur Pflege der Links durchzuführen.

7 Evaluierung des Ansatzes

Zur Evaluierung der vorgestellten Konzepte wurde eine Fallstudie durchgeführt am Fachgebiet Neuroinformatik und Kognitive Robotik der Technischen Universität Ilmenau, Dort wurde in einem evolutionären Szenario über mehrere Jahre eine Softwareplattform für mobile Serviceroboter entwickelt, die einem Reegineering unterzogen wurde. Ein Teil der Softwareplattform, das Framework für die Kommunikation der einzelnen Softwarekompo-

nenten des Roboters, das neu entwickelt wurde, war Objekt der Fallstudie. Das Goal Solution Scheme und die GOAD-Methode half den Entwicklern in der Fallstudie die relevanten Qualitätsziele zu identifizieren, diese zu priorisieren und geeignete Lösungsinstrumente für einen weiterentwickelbaren Entwurf auszuwählen. Darüber hinaus wurden zur Evaluierung des Traceability-Konzeptes 1716 Modellelemente mit CASE-Tools erfasst. Die im Anschluss mit EMFTrace und 76 definierten Regeln erstellten Traceability Links wiesen eine Genauigkeit von 86,4% und eine Trefferquote von 84,6% auf.

8 Fazit und Ausblick

Abschließend kann gesagt werden, dass die vorgestellten Konzepte der Dissertation, das Goal Solution Scheme, die zielorientierte Architekturentwurfsmethode GOAD und das Traceability-Konzept samt Toolunterstützung durch EMFTrace, einen umfassenden und praktikablen Ansatz zur Entwicklung weiterentwickelbarer Software und zur Nachvollziehbarkeit von Entwufsentscheidungen darstellen. Der Ansatz fokussiert systematisch auf die Erfüllung der Qualitätsziele beim Entwurf, insbesondere auf Weiterentwickelbarkeit. Mittels der Traceability Links können häufige Softwareänderungen besser analysiert werden, was eine der größten Herausforderungen heutiger Softwareentwicklungsprojekte darstellt. Wo immer möglich, wurden etablierte Methoden und Konzepte aufgegriffen und kombiniert. Außerdem schlägt der integrierende Ansatz eine Brücke vom Requirements Engineering zum Architekturentwurf mit Konzepten aus beiden Forschungsgebieten. In zukünftigen Arbeiten sollte das Goal Solution Scheme für weitere Qualitätsziele und Lösungsprinzipien sowie -instrumente untersucht und der gesamte Ansatz für weitere evolutionäre Projekte hinsichtlich Reengineering angewendet werden. Das Traceability-Konzept könnte zur weiteren Verbesserung durch zusätzliche Ansätze ergänzt sowie um eine Änderungsauswirkungsanalyse erweitert werden.

Literatur

[ACC+02] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia und E. Merlo. Recovering Traceability Links between Code and Documentation. *IEEE TSE*, 28(10):970–983, 2002.

[BCE08] H. Breivold, I. Crnkovic und P. Eriksson. Analyzing Software Evolvability. In *Proc. COMPSAC 2008*, Seiten 327–330. IEEE, July 2008.

[BKB02] L. J. Bass, M. Klein und F. Bachmann. Quality Attribute Design Primitives and the Attribute Driven Design Method. In *Revised Papers from the 4th International Workshop on Software Product-Family Engineering*, Seiten 169–186. Springer, 2002.

[Bos00] J. Bosch. *Design and use of software architectures: Adopting and evolving a product-line approach*. ACM Press/Addison-Wesley, 2000.

[CHSB+05] J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya und S. Christina. Goal-Centric Traceability for Managing Non-Functional Requirements. In *Proc. ICSE '05*, Seiten 362–371. IEEE, May 2005.

[CNYM00] L. Chung, B. A. Nixon, E. Yu und J. Mylopoulos. *Non-functional Requirements in Software Engineering*. Kluwer, 2000.

[GF94] O. C. Z. Gotel und A. C. W. Finkelstein. An Analysis of the Requirements Traceability Problem. In *Proceedings of the First International Conference on Requirements Engineering, Colorado Springs, CO, USA*, Seiten 94–101. IEEE, April 1994.

[HA07] N. Harrison und P. Avgeriou. Pattern-Driven Architectural Partitioning: Balancing Functional and Non-functional Requirements. In *Second International Conference on Digital Telecommunications, 2007 (ICDT '07)*, Seiten 21–26. IEEE, July 2007.

[HKN+07] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran und P. America. A general model of software architecture design derived from five industrial approaches. *Journal of Systems and Software*, 80(1):106–126, January 2007.

[HNS00] C. Hofmeister, R. Nord und D. Soni. *Applied Software Architecture*. Addison-Wesley Longman, Boston, MA, USA, 2000.

[ITU08] ITU-T. Recommendation ITU-T Z.151 User requirements notation (URN) – Language definition, Nov 2008.

[MGP08] P. Mäder, O. Gotel und I. Philippow. Rule-Based Maintenance of Post-Requirements Traceability Relations. In *Proc. RE '08*, Seiten 23–32, USA, 2008. IEEE.

[RB09] M. Riebisch und S. Bode. Software-Evolvability. *Informatik-Spektrum*, 32(4):339–343, August 2009.

[RJ01] B. Ramesh und M. Jarke. Toward Reference Models for Requirements Traceability. *IEEE Trans. Softw. Eng.*, 27(1):58–93, 2001.

[RLL98] D. Rowe, J. Leaney und D. Lowe. Defining systems architecture evolvability - a taxonomy of change. In *Proc. ECBS'98*, Seiten 45–52, Jerusalem, Israel, 1998. IEEE.

[Sie04] Johannes Siedersleben. *Moderne Software-Architektur: Umsichtig planen, robust bauen mit Quasar*. dpunkt.verlag, Heidelberg, Germany, 2004.

[SZPMK04] G. Spanoudakis, A. Zisman, E. Perez-Minana und P. Krause. Rule-Based Generation of Requirements Traceability Relations. *JSS*, 72(2):105–127, 2004.

[Yu95] E. Siu-Kwong Yu. *Modelling Strategic Relationships for Process Reengineering*. Dissertation, University of Toronto, Toronto, Ontario, Canada, 1995.

Stephan Bode wurde geboren am 5. Februar 1983 in Mühlhausen/Thüringen. Von der Technischen Universität Ilmenau erhielt er 2008 einen Abschluss als Diplom-Informatiker. Danach begann er seine Dissertation an der TU Ilmenau, gefördert durch ein Graduiertenstipendium des Landes Thüringen. Außerdem war er als wissenschaftlicher Mitarbeiter an der Universität tätig. Seine Promotion zum Doktor-Ingenieur im September 2011 erfolgte mit dem Prädikat summa cum laude. Seine Forschungsinteressen betreffen Softwarearchitekturentwurfsmethoden, Software-Evolution, Softwarequalität sowie Traceability. Er ist Autor mehrerer Veröffentlichungen auf nationalen und internationalen Workshops und Konferenzen. Seit Oktober 2011 arbeitet er als Technical Consultant bei Senacor Technologies AG.