INSTITUT FÜR INFORMATIK

DynaMod: Dynamische Analyse für modellgetriebene Software-Modernisierung

André van Hoorn, Sören Frey, Wolfgang Goerigk,
Wilhelm Hasselbring, Holger Knoche, Sönke Köster,
Harald Krause, Marcus Porembski, Thomas Stahl,
Marcus Steinkamp, Norman Wittmüss

Bericht Nr. 1305
Juli 2013

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Schlussbericht des Projekts

DynaMod

Dynamische Analyse für modellgetriebene Software-Modernisierung

André van Hoorn[2], Sören Frey[2], Wolfgang Goerigk[1], Wilhelm Hasselbring[2], Holger Knoche[1], Sönke Köster[4], Harald Krause[3], Marcus Porembski[4], Thomas Stahl[1], Marcus Steinkamp[4], Norman Wittmüss[3]

[1] b+m Informatik AG,
Rotenhofer Weg 20, 24109 Melsdorf

[2] Universität Kiel, AG Software Engineering,
Christian-Albrechts-Platz 4, 24098 Kiel

[3] Dataport AöR,
Altenholzer Straße 10-14, 24161 Altenholz

[4] HSH Nordbank AG,
Schloßgarten 14, 24103 Kiel

Juli 2013

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS10051 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Inhaltsverzeichnis

1 Einleitung

Erfolgreiche Softwaresysteme leben lange. Gleichzeitig sind diese jedoch der enormen Geschwindigkeit der Fortentwicklung der technischen Komponenten und Plattformen unterworfen, so dass die Anwendungen technisch sehr schnell altern. Von dieser Alterung sind jedoch nicht nur Programmiertechniken betroffen, sondern auch die Softwarearchitekturen erodieren sehr schnell. Um dieser Alterung entgegenzuwirken, neue technologische Potentiale zu nutzen und auch auf zukünftige Anforderungen flexibel reagieren zu können, ist eine kontinuierliche Modernisierung von Softwaresystemen erforderlich.

Bei der Neuentwicklung von Softwaresystemen hat sich mit der Modellgetriebenen Softwareentwicklung (Model-Driven Software Development, MDSD) ein Konzept etabliert, das eine elegante Lösung dieser Problematik bietet: Anstatt das System vollständig in einer technischen Programmiersprache zu entwickeln, werden fachliche Aspekte mittels geeigneter, abstrakter Modellierungssprachen dargestellt. Hierbei handelt es sich oftmals um sogenannte domänenspezifische Sprachen (Domain Specific Languages, DSLs), die speziell auf die betreffende Anwendungsdomäne zugeschnitten sind und dadurch eine knappe und präzise Formulierung der relevanten Sachverhalte ermöglichen. Die Überführung dieser abstrakten Modelle in technische Artefakte, beispielsweise Quellcode in einer Programmiersprache, wird automatisiert durch Codegeneratoren vorgenommen. Auf diese Weise ist es möglich, durch Anpassung der Generatoren die Implementierung der Modelle zu verändern, ohne Modifikationen an den zugrundeliegenden Modellen vornehmen zu müssen.

Im Gegensatz zu Neuentwicklungen stehen bei vielen Bestandssystemen keine derartigen Modelle zur Verfügung. Klassische Ansätze der Modernisierung von Bestandssystemen versuchen stattdessen, die im Quellcode unmittelbar codierten Strukturen des bestehenden Systems automatisiert in Quellcode des Neusystems zu überführen. Da durch diesen Ansatz eine Transformation auf sehr elementarer Ebene stattfindet, kann dieser Ansatz der zuvor erwähnten Erosion der Anwendungsarchitektur nicht begegnen. Zudem ist auch die Übertragung elementarer Strukturen zwischen Programmiersprachen nicht trivial; häufig muss in der Zielsprache das originäre Konstrukt mit zusätzlichem Aufwand simuliert werden. Dadurch kommt es zu einer Aufblähung des Quellcodes, was der Wartbarkeit abträglich ist. Zuletzt bleiben technologische Potentiale der Zielplattform häufig ungenutzt, da das ursprüngliche System letztlich strukturuell unverändert übertragen wird.

Im DynaMod-Projekt wurde mit der modellgetriebenen Modernisierung (Model Driven Modernisation, MDM) ein neuer, innovativer Ansatz untersucht, Modelle aus bestehenden Softwaresystemen abzuleiten, die in einem MDSD-Prozess genutzt werden können und dem Bestandssystem auf diese Weise die zuvor beschriebene Flexibilität der Implementierung verleiht. Zur Ableitung dieser Modelle werden nicht nur die statischen Strukturen des Softwaresystems betrachtet; ein besonderer Schwerpunkt ist die Nutzung dynamischer Analyseverfahren, d.h. der Untersuchung des Verhaltens des Softwaresystems zur Laufzeit. Diese dynamischen Analysen erlauben Einblick in

die tatsächliche Nutzung des Systems durch die Nutzer und produziert somit Informationen, die zur Modernisierung eines Systems unabdingbar sind.

Von besonderem Interesse ist eine gleichzeitige Betrachtung statisch und dynamisch gewonnener Informationen, eine sogenannte hybride Analyse. Hierbei entfaltet die Nutzung abstrakter Modelle eine besondere Stärke, da die Modelle eine Plattform bieten, auf der die verschiedenen Daten zusammengeführt werden können. Auch Daten aus anderen Quellen, beispielsweise Expertenwissen, können den Modellen hinzugefügt werden und führen Wissen auf der Semantikebene hinzu, das automatisiert nicht erhoben werden kann. Auf diese Weise zeigen die Modelle ein strukturiertes und umfangreiches Bild der Anwendung, das als Grundlage für eine Modernisierung dienen kann.

Neben der eigentlichen Modernisierung lag ein weiterer Fokus auf der Nutzung der gewonnenen Analysedaten zum systematischen Testen der modernisierten Anwendung. Hier bestand das Ziel darin, Methoden zu entwickeln und zu erproben, die Tests zur Prüfung funktionaler und nicht-funktionaler Eigenschaften der Anwendung aus den Analysedaten generieren können.

2 Kurzdarstellung

2.1 Aufgabenstellung

Im Rahmen des Projekts DynaMod wurde untersucht, wie sich die Erfahrungen aus der modellgetriebenen Softwareentwicklung und die dynamische Analyse des Anwendungsverhaltens gewinnbringend zur Modernisierung von Software-Bestandssystemen einsetzen lassen. Zu diesem Zweck wurden wiederverwendbare Strategien, Methoden und Werkzeuge entwickelt und an Fallstudien erprobt. Neben der eigentlichen Modernisierung des Softwaresystems wurden dabei auch unterstützende Aspekte wie beispielsweise automatisiertes Testen betrachtet.

2.2 Voraussetzungen

Die wesentlichen technischen Voraussetzungen zur Durchführung des Projekts wurden durch den umfangreichen Fundus an quelloffener Software im Java- und Eclipse-Umfeld erfüllt. Hier sind besonders das an der Universität Kiel entwickelte *Kieker*-Framework zur dynamischen Analyse von Softwaresystemen, der Parsergenerator *ANTLR* und das *Eclipse Modeling Framework (EMF)* zu nennen, auf denen die dynamischen bzw. statischen Analysekomponenten aufgebaut werden konnten [14]. Auch boten die von der Object Management Group (OMG) entwickelten Standards, insbesondere das Knowledge Discovery Metamodel (KDM) und das Structured Metrics Metamodel (SMM) wichtige konzeptionelle Grundlagen für die Gestaltung der Modelle [3].

Die wichtigste organisatorische Voraussetzung war die Verfügbarkeit von Fallstudien, an denen die Strategien, Techniken und Werkzeuge erprobt werden konnten. Diese Fallstudien wurden von den zwei assoziierten Partnern (Dataport und HSH Nordbank) bereitgestellt, die das Projekt begleiteten.

2.3 Planung und Ablauf

Die Laufzeit des Projekts DynaMod erstreckte sich vom Januar 2011 bis zum Dezember 2012. Was die inhaltliche Schwerpunktsetzung anging, war dieser Zeitraum auf zwei in etwa gleich große Blöcke aufgeteilt; bis Ende 2011 lag der Fokus zunächst auf der Entwicklung der benötigten Analysewerkzeuge, während im Jahr 2012 vorrangig erforderliche Modelltransformationen sowie die Generierung von Programmcode und anderen Artefakten in den Vordergrund gerückt wurde. Dabei war jedoch stets vorgesehen, die zuvor geschaffenen Analysemethoden und -werkzeuge auf Grundlage der Anforderungen aus den Folgetätigkeiten weiterzuentwickeln.

Zwischen dem Industriepartner b+m und der Universität Kiel bestand über den gesamten Projektzeitraum eine intensive Kommunikation, so dass viele Fragen unmittelbar beantwortet werden konnten. Zudem fanden im Monatsrhythmus Koordinierungs-

treffen mit allen am Projekt beteiligten Partnern statt, um den assoziierten Partnern die Projektfortschritte zu präsentieren und das weitere Vorgehen abzustimmen. Zum Ende jedes Quartals wurde ein Meilensteintreffen abgehalten. Besonders erwähnenswert sind drei Termine, an denen Mitgliedern des Top-Managements der assoziierten Partner der aktuelle Stand des Projekts präsentiert wurde. Insgesamt fanden folgende Termine über und im unmittelbaren Anschluss an die Projektlaufzeit statt:

Datum	*Projekttermine*	*Ort*	*Teiln.*
14.01.2011	Vorbereitungstreffen zum Kick-Off	b+m, Melsdorf	12
08.02.2011	Kick-Off-Treffen	b+m, Melsdorf	17
07.03.2011	Koordinierungstreffen	CAU, Kiel	10
11.04.2011	Koordinierungstreffen	HSH, Kiel	10
09.05.2011	Koordinierungstreffen	Dataport, Altenholz	10
30.06.2011	Meilensteintreffen	b+m, Melsdorf	15
18.07.2011	1. Koordinierungstreffen	CAU, Kiel	7
29.08.2011	Koordinierungstreffen	HSH, Kiel	8
31.08.2011	Projektpräsentation vor Mitgliedern des Top-Managements der beteiligten Partner	b+m, Melsdorf	15
10.10.2011	Koordinierungstreffen	Dataport, Altenholz	6
10.11.2011	Koordinierungstreffen	b+m, Melsdorf	7
14.12.2011	2. Meilensteintreffen	b+m, Melsdorf	17
19.01.2012	Koordinierungstreffen	CAU, Kiel	9
22.02.2012	Koordinierungstreffen	Dataport, Altenholz	8
16.03.2012	Koordinierungstreffen	HSH, Kiel	9
20.03.2012	Projektpräsentation vor Mitgliedern des Top-Managements der beteiligten Partner	b+m, Melsdorf	18
30.05.2012	Koordinierungstreffen	HSH, Kiel	7
12.06.2012	3. Meilensteintreffen	b+m, Melsdorf	11
08.08.2012	Koordinierungstreffen	Dataport, Altenholz	8
11.09.2012	Koordinierungstreffen	CAU, Kiel	8
23.10.2012	Koordinierungstreffen	b+m, Melsdorf	8
15.01.2013	Projektabschlusstreffen mit Mitgliedern des Top-Managements der beteiligten Partner	b+m, Melsdorf	18

Die Inhalte des Projekts wurden auf verschiedenen Veranstaltungen vorgestellt:

Datum	*Veranstaltung*	*Art*	*Ort*
21.–25.02.2011	GI-Fachtagung „Software Engineering (SE 2011)“	Stand	Karlsruhe
01.03.2011	„International Workshop Model-Driven Software Migration (MDSM 2011)“ im Rahmen der „European Conference on Software Maintenance and Reengineering (CSMR 2011)“	Vortrag	Oldenburg
02.–04.05.2011	13. Workshop Software Reengineering (WSR 2011)	Vortrag	Bad Honnef
08.06.2011	KoSSE-Tag	Vortrag	Lübeck
20.06.2011	Treffen des GI-Arbeitskreises Modellgetriebene Software-Architektur (AK MDA)	Vortrag	Hamburg
27.–29.06.2011	SEACON	Vortrag	Hamburg
03.09.2011	Workshop „Traceability, Dependencies and Software Architecture“ im Rahmen der 5. European Conference on Software Architecture (ECSA 2011)	Vortrag	Essen
09.09.2011	HSH Nordbank	Vortrag	Kiel
21.10.2011	HSH Nordbank	Vortrag	Hamburg
17.11.2011	„Kundenforum Architektur“ von Capgemini	Vortrag	Hamburg
23.11.2011	KoSSE-Workshop	Vortrag	Kiel
15.12.2011	Vorlesung Software Reengineering (Prof. Dr. R. Koschke), Universität Bremen	Gast-VL	Bremen
23.–24.04.2012	ACM/SPEC Int. Conf. on Performance Engineering (ICPE 2012)	Poster-/Werkzeug-Demo	Boston , MA, USA
02.–04.05.2012	14. Workshop Software Reengineering (WSR 2012)	2 Vorträge	Bad Honnef
11.05.2012	HSH Nordbank	Vortrag	Kiel
17.–18.09.2012	Tagung der Projekte des Programms „KMU innovativ“	Präsentation	Dresden
21.11.2012	ObjektSpektrum Information Days	Vortrag	Frankfurt
29.–30.11.2012	Kieker Days 2012	Vortrag	Kiel
05.06.2013	KoSSE-Tag 2013	Vortrag	Lübeck

Die Veröffentlichungen sind in Abschnitt 3.6 aufgeführt.

2.4 Wissenschaftlicher und technischer Stand vor Projektdurchführung

Der wissenschaftliche und technische Stand vor Projektbeginn wird im Folgenden anhand der drei wesentlichen Themenfelder, die vom Projekt DynaMod berührt wurden, dargestellt.

Modellgetriebene Softwareentwicklung Modellgetriebene Softwareentwicklung ist ein Oberbegriff für Techniken, die aus formalen Modellen automatisiert lauffähige Software erzeugen [12]. Im Umfeld dieser grundlegenden Tätigkeit, die von entsprechenden Generatoren vorgenommen wird, haben sich insbesondere auf der Modellebene zahlreiche Themenfelder herausgebildet, die sowohl in der Wissenschaft als auch in der Praxis große Aufmerksamkeit genießen. Hier sind Modell-zu-Modell-Transformationen und domänenspezifische Sprachen besonders hervorzuheben. Erstere erlauben es, aus bestehenden Modellen weitere Modelle abzuleiten, was im Kontext des DynaMod-Projekts insbesondere zur Überführung des Modells der Altanwendung in ein Modell für die Zielanwendung von Interesse war. Durch domänenspezifische Sprachen wird es ermöglicht, Begebenheiten einer Fachdomäne knapp und präzise in einer speziell darauf zugeschnittenen Sprache zu formulieren. Obwohl in beiden Themenfeldern auch heute noch viel aktive Forschung und Entwicklung stattfindet, konnte bereits zum Projektbeginn auf solide Konzepte und ausgereifte Werkzeuge zurückgegriffen werden.

Reverse Engineering und Softwareanalyse Reverse Engineering im Kontext von Softwareentwicklung hat das Ziel, ein existierendes Softwaresystem methodisch zu analysieren, um dessen Bestandteile und deren Interaktion zu identifizieren [1]. Derartige Analysen können sowohl statisch als auch dynamisch erfolgen; während statische Analysen auf einer Betrachtung der Programme basieren, ohne dass diese ausgeführt werden, betrachten dynamische Analysen diese Programme, während sie tatsächlich ausgeführt werden.

Insbesondere statische Analysetechniken sind bereits seit vielen Jahren Gegenstand der Forschung, da diese eine elementare Grundlage zur Entwicklung effizienter Compiler bilden. Statische Analysewerkzeuge stehen für viele verbreitete Programmiersprachen zur Verfügung.

Dynamische Analysen bieten gegenüber statischen den besonderen Vorteil, dass sich durch diese Informationen über das tatsächliche Verhalten der Anwendung gewinnen lassen. Damit werden auch statisch nicht analysierbare Aspekte wie das Nutzungsverhalten durch den Anwender der Beobachtung zugänglich gemacht. Auch dynamische Analysetechniken werden bereits seit Jahren erfolgreich eingesetzt, im besonderen Fokus steht hierbei oftmals die Betrachtung von Performanceaspekten [18].

Software-Modernisierung Die Überführung eines Altsystems auf eine neue technologische Plattform wird als Software-Migration bezeichnet, wenn die bereitgestellte Funktionalität nach dem Transformationsvorgang unverändert bleibt. Die Software-Modernisierung ermöglicht bei der Überführung von Altsystemen im Vergleich zur reinen Software-Migration noch Anpassungen und Erweiterungen der umgesetzten Funk-

tionalität. Da diese Abgrenzung bis dato noch relativ unscharf getroffen wurde, findet sich in der Literatur häufig eine synonyme Verwendung der Begriffe.

Eine Vorarbeit von besonderer Bedeutung für das DynaMod-Projekt ist das Hufeisenmodell des Software Engineering Institute. Es definiert drei Abstraktionsebenen, auf denen die Transformation vom Quell- zum Zielsystem durchgeführt werden kann. Die unterste Ebene dieses Modells ist codezentrisch, operiert also unmittelbar auf der Implementierung. Die nächsthöhere Ebene abstrahiert zu Funktionen, so dass beispielsweise eine Überführung aus dem prozeduralen ins objektorientierte Paradigma ermöglicht wird. Die oberste Ebene befindet sich auf der Abstraktionsebene der Softwarearchitektur. Das Hufeisenmodell empfiehlt, die Transformation auf dieser obersten Ebene durchzuführen und das abstrakte Architekturmodell aus den konkreteren Repräsentationen (Quellcode und funktionale Ebene) zu extrahieren. Nach der Transformation auf Architekturebene erfolgt die Verfeinerung dann analog in umgekehrter Richtung.

Ebenfalls von besonderer Bedeutung waren die Metamodelle der Initiative Architecture-Driven Modernization (ADM) der Object Management Group (OMG).

2.5 Zusammenarbeit mit anderen Stellen

Während der Projektlaufzeit wurde intensiv mit den assoziierten Partnern Dataport und der HSH Nordbank zusammengearbeitet. Zudem wurde mit den Projekten MENGES und CloudMIG kooperiert, die ebenfalls an der Christian-Albrechts-Universität angesiedelt waren. DynaMod ist Teil des Kompetenzverbundes Software Systems Engineering KoSSE (`http://www.kosse-sh.de`). Eine Zusammenarbeit mit weiteren Stellen erfolgte nicht.

3 Eingehende Darstellung

3.1 Vorstellung der erzielten Ergebnisse

Im Folgenden werden die im Projekt DynaMod erzielten Ergebnisse vorgestellt. Da viele Ergebnisse durch die Fallstudien geprägt sind, werden diese zunächst kurz dargestellt. Die Darstellung der Ergebnisse orientiert sich an der Arbeitspaketplanung, die in Abbildung 3.1 dargestellt ist. Im Anschluss werden weitere wichtige Ergebnisse vorgestellt, deren hervorstehende Bedeutung erst im Projektverlauf offenbar wurde und die daher in der originären Arbeitspaketplanung nicht gesondert berücksichtigt wurden. Die Arbeitspakete 6 bis 8 werden hierbei nicht gesondert aufgeführt: Die Ergebnisse der Evaluation anhand der Fallstudien (AP 6) werden im jeweiligen Kontext genannt, durch die Projektleitungstätigkeit (AP 7) sind keine Artefakte entstanden. Die Publikation der Ergebnisse (AP 8) wird in Kapitel 3.6 aufgegriffen.

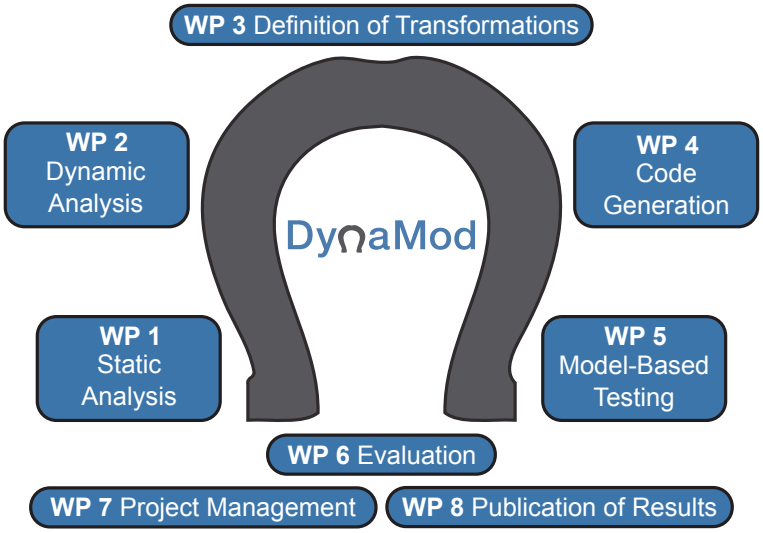

Abbildung 3.1: Arbeitspakete im Projekt DynaMod

DynaMod-Modernisierungsprozess

Die Arbeitspakete skizzieren auch bereits den Modernisierungsprozess, der im Projekt DynaMod entwickelt und erprobt wurde. Zunächst wird durch statische (AP 1) und dynamische Analyse (AP 2) ein Architekturmodell der bestehenden Anwendung erzeugt.

Dieser Prozess ist iterativ; falls zur Beantwortung späterer Fragen weitere Analyseschritte notwendig sind, werden diese durchgeführt und die gewonnenen Informationen mit den bestehenden zusammengeführt. Auf Basis des Architekturmodells der bestehenden Anwendung kann im Anschluss zunächst eine geeignete Zielplattform gewählt werden. Steht diese fest, kann mit der Definition von Transformationen (AP 3) begonnen werden, die aus dem Modell in der Ist-Architektur ein Modell in der Zielarchitektur ableiten. Diese bilden die Grundlage für die Codegenerierung (AP 4) für das Zielsystem.

Während der Modernisierung wird zudem eine große Menge Wissen über die Fachdomäne der Anwendung gewonnen, das ebenfalls in geeignete Modelle gefasst wird. Dieses Wissen bildet nicht nur das Fundament für die Modelltransformationen, indem es einen gemeinsamen Bezugspunkt für Ist- und Zielanwendung darstellt. Es ist zudem eine wichtige Grundlage zur Generierung von Tests für das Zielsystem im Rahmen von modellgetriebenem Testen (AP 5). Abbildung 3.2 stellt den beschriebenen Prozess und die dabei entstehenden Artefakte übersichtlich dar.

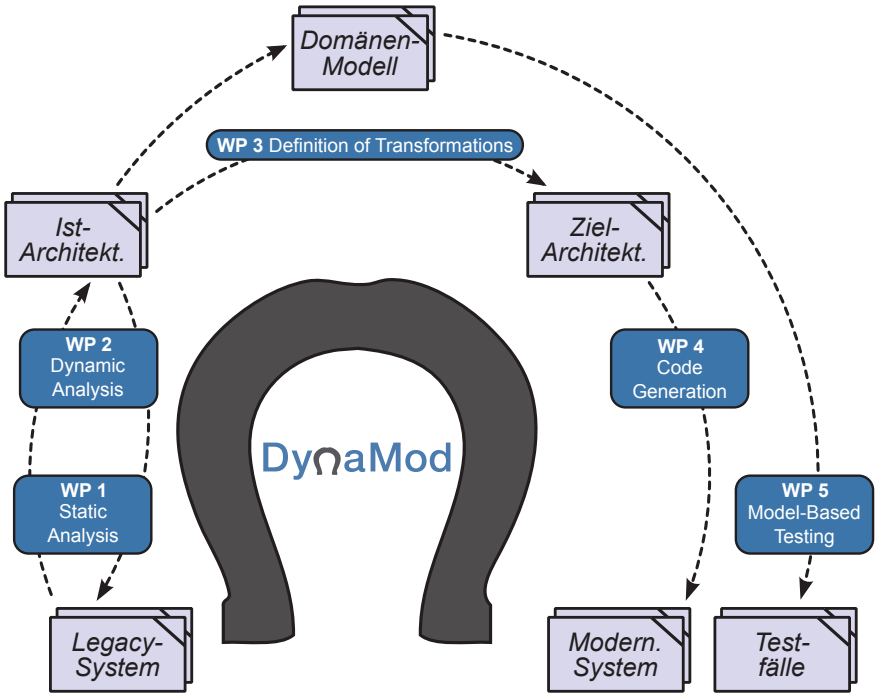

Abbildung 3.2: Darstellung des DynaMod-Modernisierungsprozesses anhand der Arbeitspaketplanung

3.1.1 Fallstudien(systeme)

Als Fallstudien dienten zwei Softwaresysteme, die von den assoziierten Partnern bereitgestellt wurden und konkreten Modernisierungsbedarf aufwiesen. Diese sollen im Folgenden kurz charakterisiert werden.

AIDA-SH, Dataport

Bei AIDA-SH handelt es sich um ein Bestandsverwaltungssystem für öffentliche Archive. Mittels der Software können die Archivare neu ins Archiv aufgenommenes Material verzeichnen und im Bestand recherchieren. Darüber hinaus erlaubt die Software die Erstellung sogenannter Findbücher, eines klassischen Recherchewerkzeugs für Archivare.

Die Software wurde mit der Programmiersprache Visual Basic 6 und der zugehörigen Laufzeitumgebung entwickelt und ist bereits seit über 10 Jahren im produktiven Betrieb. Die Fallstudie verfügt über zwei wesentliche Modernisierungstreiber:

1. Die Programmiersprache Visual Basic 6 und die zugehörige Laufzeitumgebung, die zum Ausführen der damit entwickelten Programme benötigt wird, werden vom Hersteller Microsoft seit 2008 nicht mehr gewartet. Es besteht somit dringender Bedarf, auf eine neue technische Plattform zu wechseln, da eine Lauffähigkeit unter zukünftigen Betriebssystemversionen nicht mehr gewährleistet ist.

2. Seitens der Kunden liegen Erweiterungswünsche vor, die eine umfangreiche Restrukturierung der bestehenden Anwendung sowie die Integration neuer Technologien erfordern. Dies ist auf Basis der bestehenden Plattform nicht möglich.

Nordic Analytics, HSH Nordbank

Nordic Analytics ist eine finanzmathematische Rechenbibliothek, die die Bewertung bzw. Bepreisung von Finanzprodukten erlaubt. Die Bibliothek wird sowohl von den Händlern zur Bewertung einzelner Geschäfte als auch zur Bewertung des Gesamtbestands der Bank in Batchläufen verwendet.

Die Bibliothek ist in der Programmiersprache C# auf Basis des .NET-Frameworks von Microsoft seit 2004 entwickelt. Der wesentliche Modernisierungstreiber dieser Anwendung ist der Wunsch, zur Durchführung der komplexen Berechnungen auf die Fähigkeiten moderner Multikernprozessoren zurückzugreifen. Die dazu notwendigen Anpassungen lassen sich jedoch in der bestehenden Architektur der Bibliothek nicht realisieren, so dass eine Überarbeitung der Softwarearchitektur erforderlich ist.

3.1.2 Arbeitspaket 1: Statische Analyse

Inhalt des Arbeitspakets 1 war die Entwicklung von Werkzeugen, um Softwaresysteme statisch, d.h. ohne Ausführung (z.B. auf Basis des Quellcodes), analysieren zu können und auf Basis der gewonnenen Daten Modelle abzuleiten. Des Weiteren waren die gewonnenen Informationen von großer Bedeutung, um geeignete Ansatzpunkte für dynamische Analysen (siehe AP 2) in der Anwendung zu identifizieren.

Ein besonderes Augenmerk für die statische Analyse lag auf der Fallstudie AIDA-SH, da für deren Modernisierungsziel umfangreiche und detaillierte Informationen über die derzeitige Implementierung notwendig waren. Da für die Programmiersprache Visual Basic 6 keine Sprachspezifikation verfügbar ist, wurde diese auf Basis der existierenden Fallstudie im notwendigen Umfang rekonstruiert. Während der Projektlaufzeit wurden ein Parser auf Basis des Parsergenerators *ANTLR* sowie eine Analysekomponente entwickelt, die statisch-semantische Aspekte wie Bezeichnerauflösung verarbeiten und auf Basis der gewonnenen Informationen ein vollständiges Sprachmodell

der Anwendung erzeugen konnte. Zudem war diese Komponente in der Lage, die Nutzung besonderer Fähigkeiten der Sprachumgebung, insbesondere Datenbankzugriffe und Oberflächenereignisse, detailliert zu analysieren, was zur Vorbereitung der dynamischen Analyse von besonderer Bedeutung war.

Auf Grundlage der Erfahrungen aus der Implementierung der statischen Analysekomponente für Visual Basic 6 wurden zudem Werkzeuge entwickelt, die in Programmen in der Programmiersprache COBOL automatisiert Instrumentierungspunkte lokalisieren und mit Instrumentierungscode versehen konnten. Da insbesondere im Finanzdienstleistungsbereich viele bestehende Anwendungen in COBOL implementiert sind, ist diese Sprache von großem Interesse für Modernisierungsprojekte. Im Rahmen einer Diplomarbeit an der CAU wurde das Vorgehen anhand einer externen Fallstudie evaluiert.

3.1.3 Arbeitspaket 2: Dynamische Analyse

Ziel von Arbeitspaket 2 war die Vorbereitung und Durchführung von dynamischen Analysen des Bestandssystems, d.h. der Beobachtung des Systems während des Betriebs ohne dessen Analyse.

Eine große technische Herausforderung während des Projektverlaufs war die Adaption der bestehenden Analysekomponente *Kieker* für die verschiedenen Fallstudiensysteme [18, 17, 11]. Da es sich bei Kieker um eine Java-Anwendung handelt, jedoch keine der Fallstudien Java nutzt, mussten Möglichkeiten zur plattformübergreifenden Interaktion gefunden werden. Für die Laufzeitumgebungen COM und .NET, die von Visual Basic 6 bzw. C# genutzt werden, wurden mittels sogenannter Bridges die Komponenten *Kieker.COM* und *Kieker.NET* entwickelt. Diese erlauben die unmittelbare Nutzung des Kieker-Kerns innerhalb der betreffenden Plattformen und sind auf den DynaMod- und Kieker-Webseiten veröffentlicht. Für den Anschluss an die Programmiersprache COBOL kam aus technischen Gründen der direkte Anschluss nicht in Betracht; hier wurde statt dessen auf ein standardisiertes Datenaustauschformat zurückgegriffen.

Für Altsprachen existieren i.d.R. keine Werkzeuge zur Aspekt-orientierten Programmierung (AOP), welche u.a. hilfreich sind, um Programme zu instrumentieren. Aus diesem Grund wurde das Werkzeug AOP-Werkzeug AspectLegacy entwickelt, das es erlaubt, Instrumentierungscode in Visual-Basic-6-Programme einzuweben. Wie in AP 1 bereits erwähnt, wurden ähnliche Techniken ebenfalls verwendet, um COBOL-Programme zu instrumentieren.

Besonderes Augenmerk dieses Arbeitspakets lag auf der sogenannten generativen Instrumentierung: Hierbei wird die Instrumentierung auf Modellen unter Verwendung geeigneter Abstraktionen spezifiziert; der notwendige Instrumentierungscode wird auf Basis dieser Spezifikation automatisiert erzeugt und an den betreffenen Lokationen in die Software eingefügt. Neben der Instrumentierung werden auch Werkzeuge generiert, die die gewonnenen Daten aus der dynamischen Analyse gemäß der Spezifikation aggregieren, so dass die Analyseergebnisse auf das Abstraktionsniveau der Spezifikation gehoben werden. Zur Durchführung der generativen Instrumentierung wurden geeignete Metamodelle definiert und Werkzeuge entwickelt, so dass das Konzept erfolgreich erprobt werden konnte.

Im weiteren Projektverlauf wurde dieses Konzept auf das Structured Metrics Metamodel (SMM) der Object Management Group übertragen. Da dieses nicht alle erforderlichen Möglichkeiten aufwies, waren Erweiterungen dieses Metamodells notwendig.

Die Ergebnisse dieser Arbeiten mündeten in das Werkzeug MAMBA (Measurement Architecture for Model-Based Analysis).

Im Rahmen einer Diplomarbeit an der CAU wurde eine umfangreiche dynamische Analyse der Fallstudie Nordic Analytics durchgeführt, um die exakte Funktionsweise der Bibliothek bei der Berechnung aufzuzeigen. Besonderer Fokus dieser Analyse lag auf der Identifikation von Strukturen, die der Parallelisierung entgegenstanden und der Quantifikation von deren Nutzung.

Die Fallstudie AIDA-SH konnte auf zwei Arbeitsplätzen über einen Zeitraum von 6 Monaten im produktiven Betrieb dynamisch analysiert werden. Der Fokus lag hier auf der Erhebung der von den Anwendern tatsächlich genutzten Bestandteilen der Anwendung und der Art und Weise, wie diese die Anwendung genau bedienten. Diese Informationen flossen im Anschluss in die Gestaltung des Zielbildes der Anwendung ein.

3.1.4 Arbeitspakete 3 und 4: Transformation und Code-Generierung

Wie bereits bei der Beschreibung des Modernisierungsprozesses in Abschnitt 3.1 erläutert, stehen Transformationsdefinition und Codegenerierung in einem engen Zusammenhang, da beide unmittelbar durch die Wahl der Zielplattform beeinflusst werden. Als technische Zielplattform wurde im Projekt die bestehende generative Architektur *b+m gear* von b+m gewählt, da diese ein repräsentatives Beispiel für eine industriell genutzte, generative Plattform darstellt.

Auf Basis der Fallstudie AIDA-SH wurden zwei Transformationen in die Zielplattform definiert und realisiert. Die erste Transformation überführte das aus der Altanwendung extrahierte Datenbankmodell, das die fachlichen Entitäten der Anwendung repräsentierte, in ein Entitätsmodell der gear-Plattform. Mittels der zweiten Transformation konnten Maskenflussmodelle, die aus dem während der dynamischen Analyse erhobenen Nutzungsverhalten rekonstruiert wurden, in Oberflächenmodelle der gear-Plattform übertragen werden. Auf diese Weise konnte erfolgreich in kurzer Zeit ein lauffähiger Maskenprototyp vollständig automatisiert aus dem Modell der bestehenden Anwendung generiert werden.

Obwohl die gear-Plattform bereits über Fähigkeiten zur Codegenerierung verfügte, waren Erweiterungen notwendig, um die gewünschte Funktionalität bereitzustellen. Die gear-Plattform wurde im Rahmen des Projekts dahingehend erweitert, dass grundsätzliche Persistenzservices, die sogenannten CRUD-Operationen (Create, Retrieve, Update, Delete), vollständig aus den Entitätsmodellen generiert werden konnten.

3.1.5 Arbeitspaket 5: Modellbasiertes Testen

Ziel von Arbeitspaket 5 war die Ableitung ausführbarer Testfälle aus den Daten über die Nutzung der Anwendung. Grundlage für dieses Arbeitspaket bildeten insbesondere die im Rahmen der dynamischen Analyse von AIDA-SH gewonnenen Daten über die Bedienung der Anwendung durch die Nutzer.

Zu diesem Zweck wurde zunächst ein Vorgehensmodell entwickelt, um die gesammelten Rohdaten auf die im Rahmen der Analysen gewonnenen fachlichen Informationen über die Anwendung zu beziehen. Dazu wurden in Zusammenarbeit mit den Ent-

wicklern fachliche Anwendungsfälle spezifiziert. Diese wurden anschließend in Form von Anwendungsszenarien in der Anwendung operationalisiert, d.h. es wurde erhoben, wie der betreffende Anwendungsfall in der Anwendung durchgeführt werden kann. Auf Grundlage dieser Szenarien war es anschließend möglich, die Anwendungsfälle in den während der dynamischen Analyse erhobenen Daten zu identifizieren und Daten über deren Nutzungshäufigkeit zu ermitteln.

Unter Verwendung der Szenarien und der Nutzungsdaten wurden im Anschluss Testpläne für das Testwerkzeug *JMeter* (inkl. Erweiterung *Markov4JMeter* [16]) generiert. Die Nutzungsdaten wurden insbesondere dazu genutzt, die verschiedenen Testfälle realitätsnah zu gewichten. So konnten bei Lasttests der aus der Anwendung AIDA-SH generierte Prototyp mit einer produktionsnahen Lastverteilung getestet werden.

3.1.6 Modell-Repositories

Während der Projektlaufzeit stellte sich immer mehr heraus, dass der propagierte Ansatz sehr hohe Ansprüche an die Speicherung der verschiedenen Modelle stellt. Neben der eigentlichen Speicherung müssen Aspekte wie die effiziente Recherche, die Evolution der Modelle und auch Vernetzungen zwischen den einzelnen Modellen unterstützt werden.

Aus diesem Grund wurden während der Projektlaufzeit verschiedene Techniken und Technologien zur geeigneten Speicherung dieser Modelle in sogenannten Modell-Repositories untersucht. Große Probleme bereitete dabei oft die schiere Größe der Modelle, da insbesondere die Ergebnisse der dynamischen Analyse oftmals ein großes Datenvolumen umfassten. Während der Projektlaufzeit konnte keine bestehende Technologie gefunden werden, die eine zufriedenstellende Speicherung ermöglicht hätte, weswegen die diesbezüglichen Arbeiten auch über das Ende der Projektlaufzeit in Form einer Masterarbeit fortgesetzt wurden.

3.1.7 Planung und Steuerung von Modernisierungsprojekten

Ein weiterer wichtiger Aspekt, dessen Bedeutung im Projektverlauf hervortrat, ist die Planung und Steuerung von Modernisierungsprojekten. Denn trotz des Zeitgewinns durch die Modernisierung sind insbesondere in heterogenen Softwarelandschaften Projektlaufzeiten von mehreren Jahren zu erwarten.

Aus diesem Grund müssen derartige Projekte sehr gewissenhaft geplant und gesteuert werden. So sind beispielsweise Interimslösungen, die nur während der Modernisierung genutzt werden, potentiell mehrere Jahre im produktiven Einsatz und müssen trotz ihres temporären Charakters entsprechenden Ansprüchen genügen. Im Rahmen des Projekts wurde daher auch untersucht, ob und inwieweit die durch die verschiedenen Analysen erhobenen und in den Modellen gespeicherten Informationen zur Planung und Steuerung eines Modernisierungsprojekts genutzt werden können. Zudem wurden sogenannte Migrationsarchitekturen entwickelt, die die technische Plattform für die Modernisierung eines einzelnen Softwaresystems innerhalb einer bestehenden Softwarelandschaft ermöglichen.

3.1.8 Domänenanalyse

Ein Aspekt, der insbesondere bei der Fallstudie Nordic Analytics von besonderer Bedeutung war, war die systematische Analyse der Anwendungsdomäne. Hierzu wurden in gemeinsamen Workshops die fachlichen Zusammenhänge, die der Anwendung zugrundeliegen, im Detail durchleuchtet und in Form von (Meta-)Modellen aufbereitet. Auf Grundlage dieser Domänenmodelle war es möglich, die geeigneten architekturellen Veränderungen zu entwerfen und deren Umsetzbarkeit anhand eines technischen Prototyps zu belegen.

3.2 Wichtigste Positionen des zahlenmäßigen Nachweises

Die Kosten bzw. Ausgaben entfielen zum weitaus größten Teil auf die Position 0837 (Personalkosten) und zu einem sehr kleinen Teil auf die Position 0838 (Reisekosten). Auf die weiteren Positionen des Verwendungsnachweises entfielen keine Kosten bzw. Ausgaben.

3.3 Notwendigkeit und Angemessenheit der geleisteten Arbeit

Die Kombination von statischer und dynamischer Analyse verbunden mit der modellgetriebenen und generativen Softwaremodernisierung stellen ein für die wirtschaftliche und wissenschaftliche Verwertung und den Anschluss herausragendes Alleinstellungsmerkmal im Kontext von Reengineering und Modernisierung dar. Gleichzeitig sind die notwendigen Projektarbeiten für ein KMU eine Forschungs-Investition mit nicht unerheblichem Risiko. Das Verbundprojekt DynaMod hat die Grundlage für die wirtschaftliche Umsetzung konkreter Modernisierungsprojekte geschaffen und wesentliche wissenschaftliche Beiträge geleistet. Das Geschäftsfeld Engineeering der b+m Informatik AG ist nachhaltig gestärkt, es sind wissenschaftliche Folgeprojekte entstanden, und das Projektkonsortium plant weitere gemeinsame Aktivitäten. Alle durchgeführten Projektarbeiten und Ergebnisse waren dazu notwendig und angemessen.

3.4 Darstellung des voraussichtlichen Nutzens

Wie im vorangegangenen Abschnitt erläutert, konnte während der Projektlaufzeit der gesamte geplante Modernisierungsansatz von der Analyse bis zur Codegenerierung und zum Testen exemplarisch anhand einer Fallstudie realisiert werden. Bei Präsentationen des Ansatzes vor Vertretern der Industrie fand dieser stets großes Interesse, so dass ein großes Marktpotential unterstellt werden kann.

3.5 Fortschritte bei anderen Stellen

Während der Projektlaufzeit wurde bekannt, dass an der Universität Paderborn (s-lab) an einem zu DynaMod vergleichbaren Ansatz gearbeitet wird. Genauere Informationen zu diesem Projekt finden sich in [5].

3.6 Erfolgte und geplante Veröffentlichungen

Im Zusammenhang mit dem DynaMod-Projekt sind die im Folgenden aufgelisteten Publikationen erfolgt. In dieser Aufstellung werden lediglich Titel und Autoren genannt, für Details wird per Referenz auf das Literaturverzeichnis verwiesen.

Publikation	*Ref.*
A. van Hoorn, S. Frey, W. Goerigk, W. Hasselbring, H. Knoche, S. Köster, H. Krause, M. Porembski, T. Stahl, M. Steinkamp, and N. Wittmüss: *DynaMod project: Dynamic analysis for model-driven software modernization*	[13]
A. van Hoorn, H. Knoche, W. Goerigk, and W. Hasselbring: *Model-Driven Instrumentation for Dynamic Analysis of Legacy Software Systems*	[14, 15]
S. Frey, A. van Hoorn, R. Jung, W. Hasselbring, and B. Kiel: *MAMBA: A Measurement Architecture for Model-Based Analysis*	[2]
H. Knoche, A. van Hoorn, W. Goerigk, and W. Hasselbring: *Automated Source-Level Instrumentation for Dynamic Dependency Analysis of COBOL Systems*	[7, 8]
S. Frey, A. van Hoorn, R. Jung, B. Kiel, and W. Hasselbring: *MAMBA: Model-Based Software Analysis Utilizing OMG's SMM*	[3, 4]
A. van Hoorn, J. Waller, and W. Hasselbring: Kieker: *A Framework for Application Performance Monitoring and Dynamic Software Analysis*	[18]

Darüber hinaus wurden folgende Abschlussarbeiten im Kontext des Projekts verfasst:

Abschlussarbeit	*Betreuung*	*Ref.*
Felix Magedanz: *Dynamic analysis of .NET applications for architecture-based model extraction and test generation*	CAU, HSH	[9]
Christian Wulf: *Automatic conformance checking of C#-based software systems for cloud migration*	CAU, HSH	[19]
Bettual Richter: *Dynamische Analyse von COBOL-Systemarchitekturen zum modellbasierten Testen*	CAU, b+m	[10]
Benjamin Kiel: *Investigating the Use of Graph Databases for Large Model Repositories*	CAU, b+m	[6]

Die folgenden Open-Source-Werkzeuge wurden im Kontext des Projekts veröffentlicht:

Open-Source-Werkzeug	*URL*
Die im Rahmen des DynaMod-Projekts entwickelten Erweiterungen des Kieker-Frameworks wurde als Open-Source-Projekt verfügbar gemacht. Insbesondere wurden Kieker-Adapter für C#, Visual Basic 6 und COBOL entwickelt.	`http://kieker-monitoring.net/`
AspectLegacy — AOP für Altsprachen, die von sich aus kein AOP unterstützen.	`http://dynamod.sf.net/`
Visual Basic 6 Project File Parser and Generator — Bietet Zugriff auf Visual Basic 6 Projektdateien auf Basis des Ecore-Metamodells.	`http://dynamod.sf.net/`
MAMBA — Eine Mess-Architektur für modellbasierte Analyse, basierend auf dem SMM-Meta-Modell der OMG	`http://mamba-framework.sf.net/`

Besonders hervorzuheben ist, dass die SPEC Research Group das Kieker Framework nach einer umfangreichen Begutachtung zur Nutzung empfiehlt, siehe auch `http://research.spec.org/tools.html`.

Über die DynaMod-Webseite unter `http://kosse-sh.de/dynamod/publikationen/` sind die Veröffentlichungen, Präsentationsfolien, Abschlussarbeiten und Open-Source-Werkzeuge verfügbar. Die sechs Veröffentlichungen werden als Anlage zu diesem Schlussbericht eingebunden.

Literaturverzeichnis

[1] A. Abran, P. Bourque, R. Dupuis, and J.W. Moore, editors. *Guide to the Software Engineering Body of Knowledge – SWEBOK*. IEEE, 2004.

[2] Sören Frey, André van Hoorn, Reiner Jung, Wilhelm Hasselbring, and Benjamin Kiel. MAMBA: A measurement architecture for model-based analysis. Technical Report TR-1112, Department of Computer Science, Kiel University, Germany, December 2011.

[3] Sören Frey, André van Hoorn, Reiner Jung, Benjamin Kiel, and Wilhelm Hasselbring. MAMBA: Model-based analysis utilizing OMG's SMM. In *Proceedings of the 14. Workshop Software-Reengineering (WSR '12)*, pages 37–38, May 2012.

[4] Sören Frey, André van Hoorn, Reiner Jung, Benjamin Kiel, and Wilhelm Hasselbring. MAMBA: Model-based analysis utilizing OMG's SMM. *Softwaretechnik-Trends*, 32(2):49–50, May 2012.

[5] Marvin Grieger, Baris Güldali, and Stefan Sauer. Sichern der Zukunftsfähigkeit bei der Migration von Legacy-Systemen durch modellgetriebene Softwareentwicklung. In *Proceedings of the 14. Workshop Software-Reengineering (WSR '12)*, pages 25–26, May 2012.

[6] Benjamin Kiel. Investigating the use of graph databases for large model repositories, June 2013. Master's Thesis, Kiel University.

[7] Holger Knoche, André van Hoorn, Wolfgang Goerigk, and Wilhelm Hasselbring. Automated source-level instrumentation for dynamic dependency analysis of COBOL systems. In *Proceedings of the 14. Workshop Software-Reengineering (WSR '12)*, pages 33–34, May 2012.

[8] Holger Knoche, André van Hoorn, Wolfgang Goerigk, and Wilhelm Hasselbring. Automated source-level instrumentation for dynamic dependency analysis of COBOL systems. *Softwaretechnik-Trends*, 32(2):45–46, May 2012.

[9] Felix Magedanz. Dynamic analysis of .NET applications for architecture-based model extraction and test generation, October 2011. Diploma Thesis, Kiel University.

[10] Bettual Richter. Dynamische Analyse von COBOL-Systemarchitekturen zum modellbasierten Testen ("Dynamic analysis of COBOL system architectures for model-based testing", in German), August 2012. Diploma Thesis, Kiel University.

[11] Matthias Rohr, André van Hoorn, Jasminka Matevska, Nils Sommer, Lena Stoever, Simon Giesecke, and Wilhelm Hasselbring. Kieker: Continuous monitoring and on demand visualization of Java software behavior. In *Proceedings of the*

IASTED International Conference on Software Engineering 2008 (SE'08), pages 80–85, February 2008.

[12] Thomas Stahl, Markus Völter, Sven Efftinge, and Arno Haase. *Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management*. dpunkt.verlag, 2 edition, 2007.

[13] André van Hoorn, Sören Frey, Wolfgang Goerigk, Wilhelm Hasselbring, Holger Knoche, Sönke Köster, Harald Krause, Marcus Porembski, Thomas Stahl, Marcus Steinkamp, and Norman Wittmüss. DynaMod project: Dynamic analysis for model-driven software modernization. In *Joint Proceedings of the 1st International Workshop on Model-Driven Software Migration (MDSM 2011) and the 5th International Workshop on Software Quality and Maintainability (SQM 2011)*, volume 708 of *CEUR Workshop Proceedings*, pages 12–13, March 2011. Invited paper.

[14] André van Hoorn, Holger Knoche, Wolfgang Goerigk, and Wilhelm Hasselbring. Model-driven instrumentation for dynamic analysis of legacy software systems. In *Proceedings of the 13. Workshop Software-Reengineering (WSR '11)*, pages 26–27, May 2011.

[15] André van Hoorn, Holger Knoche, Wolfgang Goerigk, and Wilhelm Hasselbring. Model-driven instrumentation for dynamic analysis of legacy software systems. *Softwaretechnik-Trends*, 31(2):18–19, May 2011.

[16] André van Hoorn, Matthias Rohr, and Wilhelm Hasselbring. Generating probabilistic and intensity-varying workload for Web-based software systems. In *Performance Evaluation — Metrics, Models and Benchmarks: Proceedings of the SPEC International Performance Evaluation Workshop 2008 (SIPEW '08)*, volume 5119 of *Lecture Notes in Computer Science*, pages 124–143. Springer, June 2008.

[17] André van Hoorn, Matthias Rohr, Wilhelm Hasselbring, Jan Waller, Jens Ehlers, Sören Frey, and Dennis Kieselhorst. Continuous monitoring of software services: Design and application of the Kieker framework. Technical Report TR-0921, Department of Computer Science, University of Kiel, Germany, November 2009.

[18] André van Hoorn, Jan Waller, and Wilhelm Hasselbring. Kieker: A framework for application performance monitoring and dynamic software analysis. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE 2012)*, pages 247–248. ACM, April 2012.

[19] Christian Wulf. Automatic conformance checking of C#-based software systems for cloud migration, March 2012. Master's Thesis, Kiel University.