

Transformation von Business-Software für Client-Server Architekturen *

Thomas Rauber
Angewandte Informatik II
Universität Bayreuth
rauber@uni-bayreuth.de

Gudula Rünger
Fakultät für Informatik
Technische Universität Chemnitz
ruenger@cs.tu-chemnitz.de

Kurzfassung

Business-Softwaresysteme enthalten über viele Jahre gewachsenes, geschäftsnotwendiges Expertenwissen, sind aber meistens nicht nach modernen Softwareentwicklungsmethoden entworfen und können daher oft nur schwer an sich wandelnde Hard- und Softwaretechnologien angepasst werden. Diese als Legacy-Problematik bezeichnete Situation ist für viele Unternehmen vorhanden und verursacht evtl. nicht unerhebliche Kosten für Wartungs- und Entwicklungsarbeiten. Eine häufige Fragestellung im Management- und Softwareentwicklungsbereich ist daher die kostengünstige und effektive Analyse, Dokumentation und Überführung solcher Softwaresysteme.

Speziell wird in diesem Beitrag die Überführung erprobter, monolithischer Unternehmenssoftware in moderne verteilte oder service-orientierte Systeme betrachtet. Dabei soll die interne modulare logische Struktur der Softwaresysteme für die Überführung durch ein Transformationswerkzeug genutzt werden. Zur flexiblen Softwareerzeugung für unterschiedliche verteilte oder heterogene Systeme wird ein inkrementeller Transformationsprozess für Legacy-Software vorgeschlagen, durch den schrittweise aus der gegebenen Grobstruktur eine speziellen Bedürfnissen entsprechende, verteilte Software entsteht. Hier spielt eine Separation der Aspekte der Geschäftslogik einerseits von Aspekten der Verteiltheit, Sicherheit oder Heterogenität andererseits eine wesentliche Rolle. Zur Durchführung des Transformationsprozesses soll ein funktions-, korrektkeits- und merkmals-erhaltendes Transformationswerkzeug entwickelt und realisiert werden. Die Separation von Transformation und Verteiltheit soll durch die Bereitstellung eines offenen, erweiterbaren Client-Server-Frameworks erreicht werden.

*Das Projekt *TransBS – Transformation monolithischer Business-Softwaresysteme in verteilte, workflowbasierte Client-Server-Architekturen* wird durch das Bundesministerium für Bildung und Forschung (BMBF) unter dem Förderkennzeichen 01 IS F10 gefördert. Weitere Informationen sind unter www.transbs.de zu finden. Das Projektkonsortium setzt sich aus der Berndt & Brungs Software GmbH, der Universität Bayreuth und der Technischen Universität Chemnitz zusammen.

1 Einleitung und Vorstellung des Themenkomplexes

1.1 Verteilte Business-Software

Softwaresysteme zur Realisierung von Geschäftsprozessen erfordern eine weitgehende Konfigurierbarkeit, Adaptionsmöglichkeit und Erweiterbarkeit gemäß kundenspezifischer Anforderungen, sollen aber möglichst kostengünstig realisiert werden. Ausgangspunkt ist dabei oft ein existierendes, über Jahre gewachsenes Softwaresystem, welches durch langjährige Arbeit und Expertenwissen über die jeweilige Unternehmensstruktur an die Erfordernisse des jeweiligen Kunden angepasst wurde, so dass nicht nur eine funktionierende, sondern auch für ein Unternehmen essentiell notwendige Softwareinfrastruktur im Laufe der Jahre gewachsen ist. Dem steht ein ständiger Wandel der Hard- und Softwarestrukturen und Technologien gegenüber, dem sich Unternehmen aus Gründen der Sicherheit, Leistungsfähigkeit, Kosten oder auch Verfügbarkeit von Plattformen anpassen müssen. Bisherigen Softwaresystemen fehlen häufig entscheidende Aspekte wie Verteiltheit, Anpassbarkeit an heterogene Plattformen oder Skalierbarkeit, die zu Entwicklungsbeginn nicht abzusehen waren, aber zunehmend an Wichtigkeit gewinnen und nachträglich nur schwierig oder mit erheblichem Aufwand zu integrieren sind, da sie eine vollständige Reorganisation der Systemarchitektur erfordern können und alle Bereiche eines Unternehmens betreffen können.

Der Problemkreis der Realisierung moderner, verteilter, effektiver Unternehmenssoftware wird in jüngster Zeit aus verschiedenen Gesichtspunkten und Wissenschaftsdisziplinen in praktischer und theoretischer Hinsicht aufgegriffen, darunter Management-Entscheidungsstrategien zur Softwareerneuerung, Aspekte der Geschäftsmodellierung, softwaretechnische Aspekte der Reengineering- oder Legacy-Problematik auf der Basis von Softwarearchitekturen oder ganz konkreter Fallstudien, sowie Softwaredesignprinzipien für verteilte Systeme. Letztere beinhalten Frameworkansätze [1], verteilte Objekte [2], verteilte Komponenten [3] und Middlewarelösungen [4].

Eine häufige Fragestellung im Management- oder Software-Engineeringbereich ist die kostengünstige und effektive Überführung erprobter monolithischer Unternehmenssoftware in moderne, verteilte Systeme, wobei auch gerade Softwareprodukte beachtet werden, die Ende der 80er oder Anfang der 90er Jahre im Hinblick auf den damaligen aktuellen Standard der Modularität und Flexibilität entworfen wurden, aber den technologischen Anforderungen der jüngsten Entwicklungen nicht mehr genügen. Die zahlreichen Ansätze zur Entwicklung verteilter Systeme berücksichtigen dieses Problem kaum in detaillierter Form, sondern beschäftigen sich entweder mit dem Design neuer Software oder mit der ganzheitlichen Ankopplung bestehender Softwarekomponenten über Wrapper- und Middlewarelösungen. Beides wird bestehender leistungsfähiger, modularer Unternehmenssoftware nicht gerecht. Die Verteiltheit hat daher kaum Eingang in populäre, weit verbreitete Unternehmenssoftware gefunden, obwohl dieser wichtige Aspekt durchaus bewusst ist und insbesondere zur Themenstellung dieses Projektes geführt hat.

Im Kontext der Verwendung verteilter Objekte wurde u.a. von der Object Management Group (OMG) die Common Object Request Broker Architecture (CORBA) definiert, die die Kommunikation zwischen Objekten in heterogenen verteilten Umgebungen un-

terstützt. Für die Entwicklung programmiersprachenunabhängiger Lösungen wird eine Interface Definition Language (IDL) bereitgestellt, die die Definition von Schnittstellen und Datenstrukturen erlaubt. Basierend auf CORBA und dem Remote Procedure Call (RPC) wurde der Mechanismus der Remote Method Invocation (RMI) von Java entwickelt. Die verteilte Ausführung des Component Object Model (COM) wird durch die Erweiterung DCOM ermöglicht. Diese Ansätze sind als Basis für die Realisierung des Client-Server-Frameworks relevant.

Für die Realisierung verteilter, interagierender Komponenten werden drei Systeme häufig verwendet: COM+ von Microsoft unterstützt die Ausführung von COM-Komponenten; Enterprise Java Beans (EJB) von Sun erlaubt die verteilte Ausführung von Java-Komponenten; das CORBA Component Model (CCM) erweitert EJB für andere Sprachen. Diese Systeme stellen einen allgemeinen Rahmen für verteilte Komponenten zur Verfügung, bieten aber keine Unterstützung für ein bestimmtes Anwendungsgebiet und sind nicht wie das im Rahmen des Projekts zu entwickelnde Client-Server-Framework auf die Verarbeitung von Workflows ausgerichtet.

Die Entwicklung komponentenbasierter Softwaresysteme wird im Rahmen des Component-Based Software Engineering (CBSE) betrachtet [5]. CBSE-Verfahren basieren meist auf einem der oben genannten Komponentenansätze (COM+, EJB, CCM) und stellen Techniken zur Entwicklung neuer modularisierter Softwaresysteme bereit, beinhalten aber auch Verfahren für die Weiterentwicklung von Softwaresystemen [6] und für das automatisierte Zusammensetzen existierender Komponenten entsprechend vorgegebener Anforderungen [7, 8]. Dabei spielen auch Techniken des Generative Programming [9] und des Aspect-Oriented Programming (AOP) [10] eine Rolle. Für langfristig angelegte Projekte sollten Strategien zur Wiederverwendung von Softwarekomponenten berücksichtigt werden [11]. Diese Techniken sind für das vorliegende Projekt wichtig, liefern aber wieder nur einen allgemeinen Rahmen und sind nicht auf die Verarbeitung von Workflows ausgerichtet.

1.2 Transformation von Business-Software

Die Legacy-Problematik und die Transformation von Altsoftware wird seit vielen Jahren von verschiedenen Forschergruppen behandelt. Hierbei handelt es sich meist um sprach- und paradigmenpezifische Einzelansätze zur Modularisierung, Überführung in Objektorientierung oder Erkennen der Geschäftslogik. Neue Ansätze umfassen auch Aspekte der Verteiltheit [12], indem sie z.B. Middleware-Lösungen zur Datenintegration zur Verfügung stellen [13]. Dabei spielen Performance-Aspekte eine wichtige Rolle [14], da z.B. bei der Überführung von Altsoftware in Web-Services durch die Verwendung der zugehörigen Protokolle wie SOAP (Simple Object Access Protocol) [15] zusätzliche Latenz- und Datenübertragungszeiten entstehen. Für die Transformation von Softwaresystemen werden Werkzeuge wie DMS entwickelt [16] und auf große Softwaresysteme angewendet [12]. Dabei können graphbasierte Visualisierungsverfahren vorteilhaft eingesetzt werden [17]. Für die Identifikation von Modulen der Altsoftware können Clustering-Verfahren eingesetzt werden [18, 19, 20]. Trotz der Vielzahl an Einzelansätzen wurden ganzheitliche Methoden zur inkrementellen Softwaretransformation bisher nicht in geeignetem Umfang genutzt. Dies liegt insbesondere auch an der Vielzahl verschiedener, durchaus inkompatibler

verteilter Plattformen (z.B. CORBA oder EJB), was bewirkt, dass jede neue verteilte Realisierung derselben Geschäftslogik letztendlich neu implementiert werden muss.

Die Model Driven Architecture (MDA) [21, 22] greift diese Problematik auf und nutzt einen modellbasierten Ansatz zur schrittweisen Erzeugung verteilter komponentenbasierter Software. Diese schrittweise Erzeugung startet mit einer UML-Spezifikation der statischen Schnittstellen und des dynamischen Verhaltens der Komponenten in einem plattformunabhängigen Modell (PIM). Zur Erzeugung eines plattformspezifischen Modells (PSM) werden eine Reihe von Abbildungen in Form von UML-Erweiterungen vorgegeben, etwa CORBA-Profiles oder Enterprise Applications Integration (EAI) Profiles. Im dritten Schritt erzeugen MDA-Werkzeuge den Programmcode für die spezifische Plattform. Die Behandlung von Legacy-Software wird bei diesen Transformationsschritten nicht berücksichtigt. Hier liegt ein wesentlicher neuer Beitrag des vorliegenden Projektes.

Weitere wichtige Aspekte verteilter Unternehmenssoftware ist die Transparenz der Verteiltheit sowie die Möglichkeit expliziter Workflowdefinition. Die in verteilten Plattformen übliche Transparenz der Verteiltheit ist zur Gestaltung von effizienter Unternehmenssoftware ungeeignet, da ohne explizites Wissen kostenintensive Remote-Anfragen entstehen können. Diese Problematik soll im vorliegenden Projekt in den Transformationsprozess einbezogen werden.

Die Workflow-Technologie zur formalen Beschreibung der automatischen Abarbeitung von Geschäftsprozessen ist ein wesentlicher Beitrag zur Entwicklung von Unternehmenssoftware. Eine Vielzahl von Workflow-Management-Systemen zur Abarbeitung von Workflows wurde realisiert, behandelt jedoch meist keine verteilten Workflows. Die Workflow Management Coalition (WFMC) [23] definiert neben Standards für die Workflow-Beschreibung bzw. Architekturen von Workflow-Management-Systemen auch Verteiltheitsaspekte. Eine direkte Einbeziehung von Workflows in Technologien verteilter Komponenten ist bisher weniger durchgeführt worden [3]. Dies gilt ebenso für die Behandlung von Legacy-Software und eine Überführung in workflowbasierte Software.

Das vorliegende Projekt wird die Gebiete der Modernisierung von Legacy-Software, verteilter Plattformen, expliziter Verteiltheit und Workflow-Technologie durch die Bereitstellung eines inkrementellen Transformationswerkzeuges und eines speziellen Client-Server-Rahmens verbinden und zur flexiblen Transformation von Legacy-Software nutzen.

1.3 Ziele und innovativer Gehalt

Ziel des Projektes ist die Entwicklung einer Methodik zur Überführung existierender, sich im Einsatz befindender, monolithischer Unternehmens-Softwaresysteme in eine moderne, komponentenbasierte, verteilte, skalierbare, auf Open-Source-Produkten basierende Client-Server-Architektur mit konfigurierbaren Workflows für heterogene Plattformen. Dabei soll die interne modulare und logische Struktur der gegebenen Software berücksichtigt werden und als Grundlage der Komponentenbildung und Auswahl der Verteiltheit dienen. Wesentlicher Schwerpunkt ist zum einen die Entwicklung und prototypische Umsetzung eines funktions-, korrektkeits- und merkmalerhaltenden Transformationswerkzeuges zur Überführung der Module von Business-Softwaresystemen in Komponenten, die in einem Client-Server-System verteilt ausgeführt werden können.

Zur Transformation von Business-Software sollen neue Methoden und Verfahren entwickelt und eingesetzt werden, die in einem Transformationswerkzeug prototypisch realisiert werden sollen. Ein Hauptbeitrag liegt damit in der Entwicklung einer Transformationsmethodik, die aus einer stufenweisen Vorgehensweise bestehen soll und in mehreren automatisierten Schritten aus einer gegebenen monolithischen Software eine auf einer Client-Server-Architektur basierende verteilte Software erstellt. Bei der Anwendung dieser neuartigen Werkzeuge sollen interaktive Schritte die Transformationsrichtung der gegebenen Unternehmenssoftware steuern können. So können etwa bei der exemplarischen Transformation von GBware andere Ziele verfolgt werden als bei anderen zu transformierenden Softwaresystemen, so dass auch hier eine Form der Endnutzerkonfigurierbarkeit entsteht. Hier soll ein Beitrag durch Gestaltung der interaktiven Schritte im Transformationswerkzeug TransBS geleistet werden.

Ein offenes, erweiterbares Client-Server-Framework CBFram soll dem Werkzeug TransBS zur Erzeugung der endgültigen Unternehmenssoftware im Client-Server-Stil dienen. Durch diesen Ansatz ist eine weitgehende Entkopplung der Transformationssoftware von gegebener Software (beispielsweise GBware) und Zielsoftware (beispielsweise GBwareD mit CBFram) realisierbar. Dies ermöglicht einerseits die Entwicklung eines flexiblen und schlanken Werkzeugs TransBS, da alle wesentlichen Aspekte der Verteiltheit und Sicherheit im Rahmen CBFram erfaßt werden sollen. Andererseits ist durch die Entkopplung der Teilmodule GBware, TransBS und CBFram auch eine Entkopplung der prototypischen Realisierung möglich, so dass die architekturmäßige Planung und Realisierung des Gesamtzieles dieses Projekts auch als Beitrag zur räumlich verteilten Erstellung von Software durch drei verteilte Teams auf Basis einer Gesamtarchitektur angesehen werden kann.

2 Projektstatus

2.1 Anforderungen an Transformationen für Business-Software

Die Forschungs- und Entwicklungsarbeiten des vorgeschlagenen Projektes umfassen drei Themenkomplexe: ein Transformationswerkzeug TransBS auf Basis compilerbasierter Methoden, eine verteilte Laufzeitumgebung in Form eines Client-Server-Frameworks CBFram zur Einbindung der erzeugten Softwarekomponenten und die prototypische Anwendung des Transformationssystems TransBS zur Erzeugung der workflowbasierten verteilten Referenzsoftware. Anforderungen an die durch den Transformationsprozess zu erzeugende Software sind durch die Anforderung der vollständigen Transparenz auf Kunden- und expliziter Realisierung und Sichtbarkeit aller erwarteten Eigenschaften für die Softwareentwickler getragen und beinhalten im Einzelnen:

- Beibehaltung der Funktionalität des bisherigen Softwaresystems bei gleichzeitiger Möglichkeit der Definition von Teilsystemen, die als unterschiedliche Varianten mit evtl. reduzierter Funktionalität kleineren Unternehmen kostengünstig angeboten werden können;
- flexible verteilte Abarbeitungsmöglichkeit auf modernen, heterogenen Plattformen

unter Beibehaltung der Lauffähigkeit auf den bisher eingesetzten, zentralisierten Plattformen;

- explizite Integration von Verteiltheit und Crosscutting-Komponenten (Zugriffsstrukturen, Synchronisation, Koordination) mit der Möglichkeit der effizienten Realisierung;
- Beibehaltung wichtiger Softwaremerkmale wie Flexibilität, Modularität und Erweiterbarkeit bei gleichzeitiger Sicherstellung einer effizienten Realisierung in verteilten, heterogenen Umgebungen;
- Skalierbarkeit mit dynamischer Erweiterbarkeit und Anpassung an kundenspezifische Anforderungen;
- Ausfallsicherheit durch die Möglichkeit redundanter Datenhaltung mit Datenkonsistenz bei verteilter Manipulation.

Zur Entkopplung der Funktionalität des resultierenden Systems von den Anforderungen der jeweiligen Kunden wird eine workflowbasierte Verarbeitung zugrundegelegt. Der Einsatz konfigurierbarer Workflows soll eine weitgehende Parametrisierung, Konfigurierbarkeit, Adaption und Erweiterung der zu verarbeitenden Geschäftsprozesse ohne Änderung der Systemarchitektur und möglichst ohne weitgehende Änderung der die Workflows verarbeitenden Module erlauben. Das Ergebnis ist eine statische, aber erweiterbare Architektur, die dynamische Workflows verarbeitet.

Als Referenzsoftware soll GBware dienen [24]. Die Unternehmenssoftware GBware wurde von der Berndt & Brungs Software GmbH entwickelt und wird von einer Vielzahl von Handels- und Dienstleistungsunternehmen eingesetzt, bietet aber noch keine verteilte Abarbeitung. GBware verfügt über diverse Standardmodule sowie viele Sondermodule, die speziell gemäß der Wünsche einzelner Kunden entwickelt wurden. Zusätzlich gibt es Internetanwendungen. GBware wird als Referenzanwendung des Transformationsansatzes dienen und soll als verteiltes Softwareprodukt (GBwareD) bei Kunden zum Einsatz kommen.

2.2 Ansatz des Transformationswerkzeuges

Zur Erreichung der genannten Anforderungen soll das Transformationswerkzeug TransBS einen mehrstufigen Transformationsprozess unterstützen, so dass die genannten Eigenschaften nacheinander integriert werden. Dies erlaubt eine Separation der verschiedenen zu transformierenden Eigenschaften und damit den Einsatz von compilerbasierten Methoden zur Durchführung der Transformation. Grundlage soll eine Koordinationsebene sein, die das Zusammenspiel von Modulen und späteren Komponenten formal beschreibt, ohne die gesamte innere Realisierung der Module kennen zu müssen. Die Koordinationsebene kann auch die Vorstufe für die Verteiltheit darstellen, auf der die gewünschte Verteiltheit definiert werden kann. Grundprinzipien des Transformationswerkzeuges sollen sein:

- sprachorientierte Spezifikation der Interaktion von Modulen bzw. Komponenten bestehender und zu erzeugender Softwaresysteme (Spezifikationskomponenten für Softwarearchitekturen und deren Feinstrukturierung);
- automatisierter Transformationsprozess der Koordinationsebene als mehrstufiger Prozess bestehend aus Grob- und Feinstrukturabbildung der Softwarearchitekturen sowie Verwendung eines flexiblen Filtermechanismus zur Konkretisierung der Komponentenverteilung;
- Spezifikationskonzept zur Beschreibung von konkreten Transformationsszenarien;
- Nutzung von Methoden des Übersetzerbaus zur konkreten Durchführung der spezifizierten Transformationsanforderungen.

Erfahrungen aus der Entwicklung des mehrstufigen, auf Koordinationssprachen basierenden Werkzeugs TwoL sollen zur Entwicklung des Transformationswerkzeuges TransBS genutzt werden. Dieses compilerbasierte Transformationswerkzeug erzeugt aus einem Spezifikationsprogramm ein task-basiertes, paralleles Programm (TwoL-Compiler) [25, 26]. Für das hier zu erstellende Transformationswerkzeug TransBS zur Entkopplung von Komponentenerzeugung und Erzeugung der Verteiltheit soll das Transformationssystem in zwei Grobschritten organisiert werden.

- **Komponentenerzeugungs-Transformation:** Der erste Transformationsschritt dient der Erzeugung einer Koordinations-Spezifikation für ein verteiltes Softwaresystem. Diese Spezifikation wird aus der Spezifikation der Modulstruktur des gegebenen (monolithischen) Softwaresystems, der Spezifikation evtl. vorhandener Ausführungsbeschränkungen und der Spezifikation der Hardwarestruktur der Ausführungsplattform erzeugt und erlaubt für eine spezielle Ausführungsplattform die Auswahl einer effizienten Realisierung.
- **Filter-Transformation:** Der zweite Transformationsschritt erzeugt aus der Koordinations-Spezifikation des verteilten Softwaresystems ausführbare Koordinations- und Softwarekomponenten für eine gegebene Ausführungsplattform, die in das Client-Server-Framework integriert werden können. Die generierten Softwarekomponenten enthalten u.a. Stubs zur Integration der Komponenten des monolithischen Softwaresystems mit den zugehörigen Datentransferoperationen, die bei einer verteilten Abarbeitung verwendet werden.

Aus einem auf Open-Source-Produkten basierenden, verteilten Workflowsteuersystem, das innerhalb des Projektes RAfEG [27] prototypisch realisiert wurde, können Erfahrungen hinsichtlich verteilter Workflowsteuersysteme für hierarchische, verteilte Workflows zur Realisierung konfigurierbarer verteilter Workflows in Business-Softwaresystemen genutzt werden [28].

Die Anforderungen an zu transformierende Alt-Software erfordern hingegen eine andere Vorgehensweise als bei neu zu erstellender Software. Zur Erzeugung der Verteiltheit durch das Transformationswerkzeug TransBS sollen hier Verteiltheitsaspekte als Rahmen vorgegeben werden. Dies sollen u.a. sein:

- Entwicklung eines komponentenbasierten Client-Server-Frameworks (CBFrame) mit Kommunikationsprotokoll und Schnittstellen zur Integration existierender Module;
- Konzeption zur Beschreibungssprachenmodellierung konfigurierbarer Workflows zur Realisierung kundenspezifischer Geschäftsprozesse;
- Realisierung von Schnittstellen unter Einbeziehung verteilter Komponenten (verteilte Workflow-Engine, verteilte Datenbanken).

Zur Sicherstellung der korrekten Überführung der gegebenen Unternehmenssoftware sind Korrektheitsmechanismen einzubringen. Für die Erzeugung effizienter Software soll weiter eine Performancebewertung stattfinden.

3 Erfahrungen, Bewertungen

Die Überführung existierender, historisch gewachsener Softwaresysteme in modulare und skalierbare Softwareprodukte, die auf heterogenen Plattformen verteilt ablaufen, ist eine der großen Herausforderungen im Softwarebereich. Das geplante Projekt entwickelt anhand eines konkreten Beispiels eine Strategie zur systematischen Überführung in ein komponentenbasiertes Client-Server-System. Das entstehende verteilte Softwaresystem soll die Funktionalität des bisherigen Produkts umfassen, sichert aber auch eine einfache Adaption und Erweiterbarkeit.

Das Innovationspotential des Transformationswerkzeuges TransBS basiert auf dem schrittweisen Transformationsansatz. Dieser Ansatz ermöglicht ein inkrementelles Vorgehen bei der Komponentenbildung bzw. der Erzeugung der Verteiltheit für die zu transformierende Alt-Software. Hierdurch ist das Transformationswerkzeug auf viele unterschiedliche Unternehmenssoftwaresysteme anwendbar. Insbesondere die Filter-Transformationsphase kann gezielt Funktionalitäts- und Performance-Anforderungen an die zu erzeugende verteilte Unternehmenssoftware berücksichtigen, was u.a. durch die Erzeugung expliziter Verteiltheit erreicht wird. Angesichts der weiten Verbreitung von Legacy-Software und anstehender Software-Managemententscheidungen werden für das Transformationwerkzeug TransBS große Einsatzmöglichkeiten gesehen.

4 Ausblick

Im Rahmen des Projektes soll die Realisierung eines Systemprototyps durchgeführt werden, dessen komponentenorientierter Aufbau in Form eines Baukastensystems eine leichte Erweiterbarkeit durch Hinzufügen neuer Komponenten ermöglicht. Die Verwendung konfigurierbarer Workflows stellt einen Einsatz des Systems für eine Vielzahl von Unternehmens-Softwaresystemen sicher, auch wenn diese mit unterschiedlichen Datensätzen und -formaten arbeiten. Das in den Arbeitsschritten beschriebene Vorgehen der wissenschaftlich fundierten Entwicklung eines lauffähigen Prototypen gewährleistet die Relevanz der Projektergebnisse für die Praxis.

Die entwickelte Architektur trägt dazu bei, dass die Ergebnisse langfristig und in verschiedenen Unternehmensfeldern nachhaltig nutzbar sind. Der direkte Anwendungsnutzen wird

durch die Transformation der Referenzanwendung und die Anwendung auf ein relevantes Testscenario mit verschiedenen Kundenworkflows sichergestellt, so dass eine frühe Verwertung ermöglicht wird. Darüber hinaus werden in der prototypischen Implementierung sowohl Praktikabilität als auch Relevanz der Methoden evaluiert und zugleich über das Feedback der Pilotkunden eine Markteignung sichergestellt.

Literatur

- [1] S.M. Lewandowski. Frameworks for component-based client/server computing. *ACM Comput. Surv.*, 30(1):3–27, 1998.
- [2] W. Emmerich. *Engineering Distributed Objects*. John Wiley & Sons, 2000.
- [3] W. Emmerich. Distributed Component Technologies and their Software Engineering Implications. In *Proc. of Int. Conf. Software Engineering*, pages 537–546. ACM Press, 2002.
- [4] Ph.A. Bernstein. Middleware - A Model for Distributed System Services. *Commun. of the ACM*, 39(2), 1996.
- [5] I. Crnkovic and M. Larsson. Challenges of component-based development. *Journal of Systems and Software*, 61(3):201–212, 2002.
- [6] A. Mehta and G.T. Heineman. Evolving Legacy System Features into Fine-Grained Components. In *Proc. of the Int. Conf. on Software Engineering*, pages 417–427, 2002.
- [7] F. Cao, B.R. Bryant, C.C. Burt, R.R. Rajee, A.M. Olson, and M. Auguston. A Component Assembly Approach Based On Aspect-Oriented Generative Domain Modeling. *Electronic Notes in Theoretical Computer Science*, 114:119–136, 2005.
- [8] W. Zhao, B.R. Bryant, J. Gray, C. C. Burt, R. R. Rajee, M. Auguston, and A.M. Olson. A Generative and Model Driven Framework for Automated Software Product Generation. In *Proc. 6th ICSE Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction*, pages 103–108, May 2003.
- [9] K. Czarnecki and U.W. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison Wesley, 2000.
- [10] G. Kiczales, J. Lamping, A.Mendhekar, C. Maeda, C.V. Lopes, J.-M. Loingtier, and J. Iwin. Aspect-oriented programming. In *Proc. Conf on Object-Oriented Programming (ECOOP'01), LNCS 1241*, pages 327–353, 1997.
- [11] T. Ravichandran and M.A. Rothenberger. Software Reuse Strategies and Component Markets. *Communication of the ACM*, 46(8):109–114, 2003.
- [12] R.L. Akers, I.D. Baxter, and M. Mehlich. Re-Engineering C++ Components Via Automatic Program Transformation. In *Proc. of ACM Symposium on Partial Evaluation and Program Manipulation*, pages 51–55. ACM Press, 2004.
- [13] G. Menkhaus and U. Frei. Legacy System Integration using a Grammar-based Transformation System. *CIT - Journal of Computing and Information Technology*, 12(2):95 – 102, 2004.

- [14] M. Litoiu. Migrating to Web Services: a performance engineering approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 16:51–70, 2004.
- [15] R. Brunner and J. Weber. *Java Web Services*. Prentice Hall, 2002.
- [16] I.D. Baxter, C. Pidgeon, and M. Mehlich. DMS: Program Transformations for Practical Scalable Software Evolution. In *Proc. of the 26th Int. Conf. on Software Engineering*, pages 625–634. IEEE Press, 2004.
- [17] F. Balmas. Displaying dependence graphs: a hierarchical approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 16(3):151–185, 2004.
- [18] R. Al-Ekram and K. Kontogiannis. Source Code Modularization Using Lattice of Concept Slices. In *Proc. of Euromicro Working Conference on Software Maintenance and Reengineering (CSMR'04)*, pages 195–203, 2004.
- [19] C. Lung and M. Zaman. Applying Clustering Techniques to Software Architecture Partitioning, Recovery and Restructuring. *Journal of Systems and Software*, 73(2):227–244, 2004.
- [20] P. Andritsos and V. Tzerpos. Software Clustering based on Information Loss Minimization. In *Proc. of Working Conference on Reverse Engineering (WCRE2003)*, pages 334–344, 2003.
- [21] MDA Model Driven Architecture, <http://www.omg.org/mda>.
- [22] J. Siegel. Why use the Model Driven Architecture to Design and Build Distributed Applications. In *Proc. of Int. Conf. Software Engineering*, page 37. ACM Press, 2005.
- [23] WFMC Workflow Management Coalition, <http://www.wfmc.org>.
- [24] GBware, <http://www.bbs-online.de/dokumente/gbware.pdf>.
- [25] T. Rauber and G. Rünger. A Transformation Approach to Derive Efficient Parallel Implementations. *IEEE Transactions on Software Engineering*, 26(4):315–339, 2000.
- [26] T. Rauber and G. Rünger. The Compiler TwoL for the Design of Parallel Implementations. In *Proc. 4th Int. Conf. on Parallel Architecture and Compilations Techniques, PACT96*, pages 292–301, Boston, USA, 1996. IEEE Computer Society Press.
- [27] RAFEG, www.rafeg.de, BMBF Projekt Förderkennzeichen 01 ISC 07.
- [28] D. Beer, S. Höhne, H. Petersohn, T. Pöhnitzsch, G. Rünger, and M. Voigt. Designing a Distributed Workflow System for E-Government. In *Proc. 24th IASTED International Conference on Modelling, Identification, and Control*, CD-ROM, pages 583–588, Innsbruck, Austria, 2005. ACTA Press.