Gesellschaft für Informatik e.V. (GI)

publishes this series in order to make available to a broad public recent findings in informatics (i.e. computer science and information systems), to document conferences that are organized in cooperation with GI and to publish the annual GI Award dissertation.

Broken down into
• seminars
• proceedings
• dissertations
• thematics
current topics are dealt with from the vantage point of research and development, teaching and further training in theory and practice. The Editorial Committee uses an intensive review process in order to ensure high quality contributions.

The volumes are published in German or English.

Information: http://www.gi-ev.de/service/publikationen/lni/

ISSN 1617-5468
ISBN 978-3-88579-272-7

This volume contains papers from the Software Management 2010 Conference on the transition of software projects to software products held in Aachen December 01 to 03, 2010. The topics covered in the papers range from key findings in research to descriptions of success-stories from industrial practice in the context of IT-Project- and IT-Product-Management.

Wolfram Pietsch, Benedikt Krams (Hrsg.): Vom Projekt zum Produkt

178

GI-Edition

Lecture Notes in Informatics

Wolfram Pietsch, Benedikt Krams (Hrsg.)

Vom Projekt zum Produkt

Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschaftsinformatik (WI-MAW), Aachen, 2010

Proceedings

Wolfram Pietsch, Benedikt Krams (Hrsg.)

Vom Projekt zum Produkt

Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschaftsinformatik (WI-MAW)

01. - 03. Dezember 2010
in Aachen

Gesellschaft für Informatik e.V. (GI)

Lecture Notes in Informatics (LNI) - Proceedings
Series of the Gesellschaft für Informatik (GI)

Volume P-178

ISBN 978-3-88579-272-7
ISSN 1617-5468

Volume Editors
Prof. Dr. Wolfram Pietsch
Betriebswirtschaftslehre, insbes. Internationales Vertriebs- und Servicemanagement
Fachhochschule Aachen
Eupener Str. 70, 52066 Aachen
E-Mail: pietsch@fh-aachen.de
Dipl.-Kfm. (FH) Benedikt Krams
Lehrstuhl für ABWL und Wirtschaftsinformatik II (Unternehmenssoftware)
Universität Stuttgart
Keplerstraße 17, 70174 Stuttgart
E-Mail: krams@wi.uni-stuttgart.de

printed by Köllen Druck+Verlag GmbH, Bonn

Vorwort

Viele erfolgreiche Softwaresysteme wurden nicht als Produkte konzipiert, sondern entstanden aus einer erfolgreichen Individualentwicklung. Viele Softwareanbieter haben so im Projektgeschäft begonnen und im Laufe der Zeit daraus ein Produktgeschäft entwickelt. Logischerweise ist das Projektmanagement, als Methode und Organisationsform konzipiert, ein klassischer Bestandteil des Instrumentariums der Anwendungsentwicklung. Auch die Entwicklung, die Anpassung und Einführung umfassender betrieblicher Standardsoftware-Produkte ist ein umfangreiches Projekt. Mit zunehmender Professionalisierung der Softwareindustrie steigt die Bedeutung des Produktmanagements im Methodenrepertoire und ist Gegenstand des aktuellen Diskurses in Praxis und Wissenschaft.

Nicht die Beherrschung des Projektmanagements und des Produktmanagements alleine, sondern der organisatorische und auch methodische Übergang vom Projektgeschäft zum Produktgeschäft stellt eine besondere Herausforderung dar: Individualsoftwareentwicklungen sind nicht ohne weiteres zu den Konditionen von Standardsoftwaresystemen möglich und historisch gewachsene Standardsoftwaresysteme sind nicht beliebig individualisierbar. Der Übergang vom Projekt zum Produkt ist kritisch für den Unternehmenserfolg; in der wissenschaftlichen Diskussion werden Projektmanagement und Produktmanagement allerdings unabhängig voneinander behandelt. Deshalb widmet sich die diesjährige Fachtagung Software-Management 2010 dem Übergang „Vom Projekt zum Produkt“ mit den folgenden Leitfragen:

- Warum sind unter welchen Voraussetzungen welche Ausprägungen von Projekten und Produkten vorteilhaft?
- Womit und wie ist die methodische und organisatorische Transition zwischen Projekten und Produkten erfolgreich zu bewältigen?

Der Tagungsband[1] beginnt in Teil I mit den Beiträgen zu den beiden Keynotes. Der Beitrag von Olaf Mackert, Tobias Hildenbrand und Almer Podbicanin, Senior Vice President der SAP AG, Vorstandsbereich Products & Solutions und Leiter des LEAN Center und Process Owner des Hauptprozesses „Product Design & Development“, zeigt die Evolution der Software-Entwicklung von der sequentiellen hin zur agilen Software-Entwicklung bei der SAP AG auf. Der zweite Beitrag von O. Univ. Prof. Dr. Dr. h.c. Heinrich C. Mayr, Rektor der Alpen-Adria-Universität Klagenfurt, widmet sich der Frage der Software-Integration im „Unternehmen“ Universität.

Dr. Kim Lauenroth von der Universität Duisburg-Essen (paluno – The Ruhr Institute for Software Technology) gibt auf unsere Einladung hin in Teil II einen Einblick in den Stand der Technik der Produktlinien und des Variantenmanagements.

[1] Die Beiträge in diesem Tagungsband sind nach ihrer Art (Keynote, eingeladener Vortrag, wissenschaftliches Programm, Tutorium) und alphabetischer Reihenfolge des Nachnamens des Erstautors aufgeführt.

Teil III des Tagungsbands beinhaltet das wissenschaftliche Programm mit elf ausgewählten Beiträgen aus Wissenschaft und Praxis. Im Rahmen eines Double-blind-Reviews[2] der Mitglieder des Programmkomitees wurden diese Beiträge, tlw. nach inhaltlichen Überarbeitungszyklen in Anlehnung an die Gutachten, zur Tagung zugelassen. Zu den Beiträgen im Einzelnen:

Einen empirisch begründeten, verbesserten Ansatz zur Bewertung von Projekten stellen Dirk Basten, Dominik Joosten und Werner Mellis der Universität zu Köln anhand eines deskriptiven Modells des IS-Entwicklungsprozesses vor.

Die Notwendigkeit von Campus-Management-Systemen verdeutlichen Markus Bick (Europäische Wirtschaftshochschule Berlin), Thomas Grechenig (TU Wien) und Thorsten Spitta (Universität Bielefeld). Der Beitrag bringt Klarheit in das Begriffsdickicht und stellt existierende Lösungen auf den Prüfstand.

Eine inkrementelle Entwicklung und Implementierung eines Transport Management Systems bei einem weltweit agierenden Logistikdienstleister beschreibt Jaroslav Blaha (Blaha Executive Consulting GmbH) in seinem Kurzbeitrag aus der Praxis.

Die Abgrenzung und der Vergleich projekt- und produktorientierter Geschäftsmodelle stehen im Mittelpunkt des Kurzbeitrages von Thomas Deelmann (T-Systems International GmbH).

Inwiefern Produktkernel im Spannungsfeld geforderter Standardisierung und Berücksichtigung individueller Kundenanforderungen hilfreich sind, schildern Ingo Elsen, Asma Hawari und Uwe Johnen (T-Systems GEI GmbH) anhand eines Flughafeninformationssystems.

Neue Möglichkeiten für das Marketing von IT-Produkten durch Anwendung der Erkenntnisse der New Lanchaster Theory erörtert Thomas Fehlmann (Euro Project Office AG), indem er Faktoren analysiert, die Marktanteile definieren, und mit den Geschäftszielen eines Unternehmens verknüpft.

Mit der wachsenden Bedeutung von Cloud Computing steigen auch die Anforderungen an die Betrachtung des ökonomischen Nutzens. Sabrina Lamberth und Anette Weisbecker (Fraunhofer IAO) erörtern die Wirtschaftlichkeit des Einsatzes dieser modernen IT-Dienstleistungen.

Erfolgs- und Misserfolgsfaktoren der Anwendungsentwicklung stellen Heiner Merz und Michael Stewen (Universität Stuttgart) auf Grundlage einer empirischen Untersuchung auf Basis einer Klassifikation nach ISO/IEC 12207 sowie ISO/IEC 15288 dar.

[2] Die Beiträge wurden von zwei Gutachtern ohne Kenntnis des Autors oder der betrachteten Unternehmung geprüft und bewertet.

Ein Reifemodell zur Kategorisierung und Abgrenzung von eServices schlagen Norman Pelzl und Georg Herzwurm (Universität Stuttgart) im Hinblick auf Service Marktplätze als Bestandteil des Cloud Computing vor.

Application Lifecycle Management setzen Stephan Salmann und Christian Horstman (Corporate Quality Consulting GmbH) für die strategische Steuerung der Transition von Projekten und Produkten ein. Dabei steht im Mittelpunkt ihres Beitrages die rationale Identifikation wertbeitragsteigender Projekte.

Das Konzept des value-based software engineering überträgt Harry Sneed (ANECON GmbH) auf die Softwarewartung: durch Bezugnahme auf verschiedener Theorien und Methoden wird eine Bewertungsmethodik unter Berücksichtigung von Kosten- und Nutzenaspekten vorgeschlagen und in einer Fallstudie dargestellt.

In Teil IV des Tagungsbandes finden Sie die zwei halbtägigen Tutorien:

Georg Herzwurm stellt in seinem Tutorium die Integration betriebswirtschaftlicher und technischer Methoden für ein erfolgreiches IT-Produktmanagement differenziert nach Geschäftsmodelltypen vor.

Stefan Jesse und Sixten Schockert erläutern die Zertifizierung zum „Certified Professional for Requirements Engineering“ aus Sicht eines einschlägigen Schulungsanbieters mit hoher Erfolgsquote. Sie messen dem Grundsatz der kundenorientierten Softwareentwicklung bei der Anforderungsanalyse und -management besondere Bedeutung für erfolgreiche Projekte bei.

Wir bedanken uns bei der Fachhochschule Aachen für die Bereitstellung der lokalen Infrastruktur, die von den Fachbereichen Elektrotechnik und Informationstechnik und Wirtschaftswissenschaften getragen wird, insbesondere Herrn Prof. Dr.-Ing. Michael Trautwein, dem Dekan des Fachbereichs Elektrotechnik und Informationstechnik als Mitglied des Organisationskomitees sowie Herrn Prof. Dr. Norbert Janz, als Dekan des Fachbereichs Wirtschaftswissenschaften. Weiter bedanken wir uns bei Prof. Dr. Georg Herzwurm; ohne die Unterstützung aus dessen Stuttgarter Team wäre die Tagung nicht zu stemmen gewesen. Nicht zuletzt bedanken wir uns bei Prof. Dr. Thorsten Spitta, Fachausschußsprecher, für die kontinuierliche, persönliche und tatkräftige Unterstützung bei der Vorbereitung und Durchführung der Tagung.

Aachen, im Dezember 2010

Wolfram Pietsch Benedikt Krams

Programmkomitee

Prof. Dr. Wolfram Pietsch, FH Aachen (Vorsitz)
Dr. Martin Bertram, Commerzbank
Dr. Andreas Birk, Software.Process.Management
Jens Borchers, SCHUFA Holding
Prof. Dr. Hans Brandt-Pook, FH Bielefeld
Prof. Dr. Peter Buxmann, TU Darmstadt
Dr. Thomas Deelmann, T-Systems Int. GmbH
Anton Dillinger, SAP
Prof. Dr. Stefan Eicker, Univ. Duisburg-Essen
Prof. Dr. Martin Engstler, HDM Stuttgart
Dr. Gerd Große, GFFT e.V.
Prof. Dr. Georg Herzwurm, Univ. Stuttgart
Prof. Dr. Thomas Hess, Univ. München
Prof. Dr. Wolfgang Hesse, Univ. Marburg
Reinhard Höhn, KMA Knowledge Mgmt.
Prof. Dr. Stefan Jacobs, FH Aachen
Gerrit Kerber, aragon INTERACTIVE
Dr. Ralf Kneuper, Dr. Kneuper Beratung
Dr. Christian Kop, Univ. Klagenfurt
Dr. Torsten Langner, Microsoft
Dr. Oliver Linssen, Liantis GmbH & Co. KG
Prof. Dr. Dr. Heinrich Mayr, Univ. Klagenfurt
Prof. Dr. Werner Mellis, Univ. zu Köln
Martin Mikusz, Univ. Stuttgart
Prof. Dr. Andreas Oberweis, Univ. Karlsruhe
Bernd Oestereich, oose.de
Prof. Dr. Roland Petrasch, TFH Berlin
Werner Simonsmeier, sd&m
Harry Sneed, AneCon
Prof. Dr. Thorsten Spitta, Univ. Bielefeld
Prof. Dr. Dirk Stelzer, TU Ilmenau
Prof. Dr.-Ing. Michael Trautwein, FH Aachen
Dr. Dirk Voelz, SAP Research
PD Dr. Anette Weisbecker, Fraunhofer IAO

Organisatorische Leitung

Prof. Dr. Wolfram Pietsch, FH Aachen (Vorsitz)
Prof. Dr.-Ing. Michael Trautwein, FH Aachen (stellv. Vorsitz)
Benedikt Krams, Univ. Stuttgart

Inhaltsverzeichnis

Teil I

Keynotes

Vom Projekt zum Produkt – SAP's Weg zum „Lean Software Product Development"

Olaf Mackert, Tobias Hildenbrand, Almer Podbicanin

SAP AG
Dietmar-Hopp-Allee 16
69190 Walldorf
Germany
{olaf.mackert, tobias.hildenbrand, almer.podbicanin}@sap.com

Abstract: Die Erstellung von Software, insbesondere von Unternehmenssoftware, ist ein komplexes Thema. Die Komplexität nimmt mit Anzahl der involvierten Mitarbeiter und der Auswahl an Produkten bzw. unterschiedlichen Kundensegmenten überproportional zu. Ziel dieser Abhandlung ist es, einen groben Überblick über die Geschichte eines großen Softwareanbieters zu geben und dabei insbesondere die Evolution des methodischen Ansatzes zur Softwareerstellung bzw. der Prozessmodelle darzulegen. Hierbei werden die Entwicklungsschritte hin zu aktuellen Lösungsansätzen aufgezeigt, mit welchen der Grad der Komplexität beherrschbar gemacht und eine höhere Skalierbarkeit der Entwicklungsorganisation erreicht werden soll.

1 Einleitung

Softwareerstellung erfolgt klassischerweise und vor allem in kleinen bis mittelgroßen Softwarehäusern intuitiv in enger Zusammenarbeit mit dem Kunden als Projektgeschäft. Wenn Softwarehäuser über die Zeit wachsen und neben reinen Kundenprojekten auch Standardsoftware anbieten, wird es sehr schwierig, die entsprechenden Skaleneffekte zu erzielen und gleichzeitig die Bedürfnisse des Marktes sowie der einzelnen Kunden zu treffen. Viele große Softwareanbieter führen daher Projektmanagement- und Prozessstandards ein.

Trotz bzw. gerade wegen des meist stark arbeitsteiligen Charakters dieser Standards kommt es häufig zu Problemen in der Skalierung – wie beispielsweise unzureichend durchgängige Verantwortlichkeiten, Nicht-Einhaltung gemeinsamer Prozessstandards sowie Multi-Tasking innerhalb von Teams bzw. bei den einzelnen Mitarbeitern. Zudem ist die Produktivität auf Unternehmensebene – z.B. gemessen in Umsatz pro Mitarbeiter – aufgrund dessen häufig rückläufig [OVoJ]. Eine verstärkte Ausrichtung möglichst aller Prozesse auf den Kunden mit Fokussierung auf den direkten Kundennutzen, wie beispielsweise in Lean Management-Ansätzen [Lik04], verspricht weitere Skaleneffekte durch höhere Kundenzufriedenheit – ähnlich wie im Projektgeschäft bei kleineren Unternehmen – und damit höhere Nachfrage und Umsätze.

Am Beispiel der SAP AG wird die Entwicklung vom anfangs rein kundenbezogenen Projektgeschäft hin zur heutigen Mischung aus Standardsoftwareentwicklung und speziellen Kundenprojekten (Geschäftsbereich „Custom Development“) beschriebenen. Ziel dieser Abhandlung ist es hierbei, die Evolution der Prozessstandards und deren unterschiedlichen Fokus auf Projekte, Produkte und Lösungen hin zum heutigen „Lean Software Product Development“-Ansatz aufzuzeigen. Im Lean-Ansatz tauchen Tugenden der Anfangszeit wieder auf, es wird versucht die Erfahrungen, die über Jahrzehnte gemacht wurden zu nutzen, um die Fehler der Vergangenheit nicht zu wiederholen.

Hierzu wird im folgenden Abschnitt zunächst ein Überblick über die historische Entwicklung des Entwicklungsansatzes, des Prozessmanagements bei SAP und eine Betrachtung der gewachsenen Probleme in der Standardsoftwareentwicklung gegeben. Abschnitt 3 stellt den aktuellen Ansatz „Lean Software Product Development“ vor und den Weg dorthin dar. Darüber hinaus gibt Abschnitt 4 eine Zusammenfassung der Ergebnisse. Abschnitt 5 lässt einen Ausblick auf mögliche Weiterentwicklungen und aktuelle Forschungsvorhaben zum Thema „Lean“ zu.

2 SAP’s Prozessmanagement-Evolution: Vom Projekt zur Lösung

Seit der Gründung 1972 bietet SAP ein Portfolio unterschiedlicher Unternehmenssoftware rund um das Kernprodukt, die Business Suite (früher R/2, R/3, etc.), an. Viele Bereiche dieser Software wurden ursprünglich entwickelt, um in großen Unternehmen eingesetzt zu werden. Seit einigen Jahren bietet SAP daher auch Softwarelösungen für kleine und mittelständige Betriebe an. Auch dieser Bereich wächst und wird ständig weiterentwickelt.

In den ersten zwei Jahrzehnten ihrer Unternehmensgeschichte entwickelte SAP sehr eng mit ihren meist lokalen Kunden zusammen Unternehmenssoftware. Diese dichte Zusammenarbeit half, den Fokus auf das Wesentliche nicht zu verlieren, sowie Geschäfts- und Technologietrends im Auge zu behalten. Ohne diese gute Zusammenarbeit wäre die Entwicklung eines vollständigen Client-Server-Systems wie SAP R/3, das alle kritischen Geschäftsprozesse unterstützt und in jedem Großindustriezweig einsetzbar ist, nicht so schnell von statten gegangen. Die Entwickler hatten ständigen Kontakt mit Kunden und Benutzern in verschiedensten Aufgabenbereichen. Nicht nur in der Rolle als Programmierer, sondern auch als Berater, Trainer und im Support.

Der Erfolg von R/3 gestaltete den Markt für kommerzielle Softwareanwendungen neu. Für neue Kunden war R/3 nicht mehr ein System, bei dem sie mithalfen, dies zu entwickeln und die Möglichkeit hatten, es mit zu formen. Es war zu einem *Produkt* geworden. Der Kunde wollte es schnell installieren sowie konfigurieren, danach sollte es sofort einsatzbereit sein und einwandfrei funktionieren, auch wenn die Geschäftsprozesse des einzelnen Kunden die eine oder andere Eigenheiten aufwiesen. Anforderungen wandelten sich von mehr Geschäftsfunktionalitäten zu einem höheren Grad der Konfigurierbarkeit. Mit ausgewählten strategischen Kunden wurde auch in der Standardproduktentwicklung kontinuierlich zusammengearbeitet.

Bereits beginnend mit SAP R/2 und letztendlich mit SAP R/3 gelang, zusammen mit weltweit agierenden Kunden, der Aufstieg zum globalen Softwareanbieter. Innerhalb von wenigen Jahren wuchs die SAP Entwicklungsorganisation von ca. 2.000 auf über 14.000 Entwickler an. In der Zeit von R/2 wurden den Entwicklern generalistische Fähigkeiten abverlangt (vgl. oben). Innerhalb der starken Wachstumsphase der Entwicklungsorganisation wurde der Entwickler immer mehr zum Spezialisten einer bestimmten Technologie, eines spezifischen Entwicklungszweiges oder einer bestimmten Anwendungsdomäne. Dies zeigte sich u.a. darin, dass für R/3 bereits eine eigene Beratungsorganisation aufgebaut wurde. Der Anstieg der Spezialisierung korreliert unter anderem mit dem Grad der Arbeitsteilung positiv. Inzwischen war es üblich, dass eine Expertengruppe die Anforderungen erhob und in die Entwicklungsbereiche trug, an anderer Stelle wurden ausführliche Spezifikationen und Architekturentwürfe erstellt, andere Gruppen programmierten die Basisfunktionalitäten und Algorithmen, wieder andere testeten usw. [Sch10].

Im Laufe der 90er Jahre wurde klar, dass der Kommunikationsfluss zwischen den hochspezialisierten Teams, Abteilungen und Organisationen zum Problem geworden war und dies die Hauptursache für Probleme mit der Produktqualität auf verschiedenen Integrationsebenen darstellte. Um diesen Problemen Herr zu werden und die Entwicklungsaufwände über alle beteiligten Organisationseinheiten hinweg koordinieren zu können sowie aufgrund von externen Anforderungen durch bspw. ISO 9000 und der *Food and Drug Administration* (FDA), führte SAP mit Beginn der 90er Jahre ein erstes Entwicklungs-Prozess-Framework ein. Entsprechend ihren Hauptaktivitäten und Qualifikationen wurden Abteilungen Verantwortungen zugewiesen. Es existieren feste Übergabepunkte und Vorschriften zwischen den beteiligten Abteilungen, wodurch das Prozessmodell eher als wasserfallartig zu beschreiben ist. Dieses Prozessmodell wurde von den meisten Entwicklungsabteilungen ein Mal pro Jahr durchlaufen und wurde über die Jahre weiter entwickelt (siehe Abbildung 1).

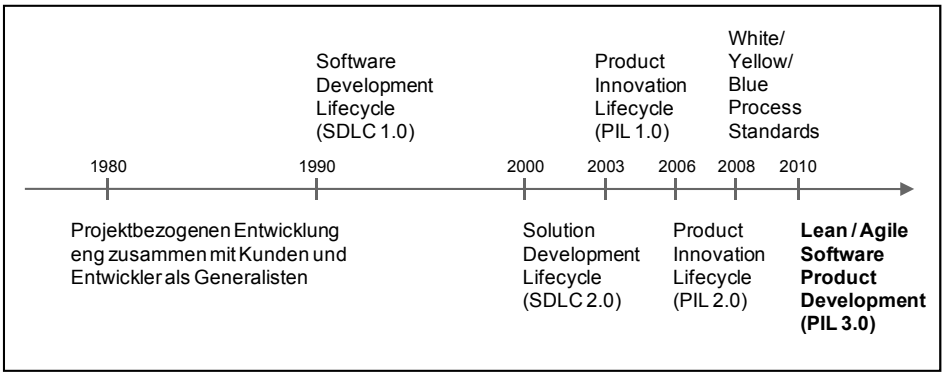

Abbildung 1: Evolution der SAP-Prozessstandards

Als zwischen 2001 und 2004 agile Methoden einen breiteren Einsatz in der Industrie fanden [Boe04; Dyb08; Din10], fing auch SAP an, diese zu pilotieren. Einige Teams begannen bspw. mit Extrem-Programming [Bec99] und Scrum [Sch01] zu experimentieren. Dieser Ansatz war für reguläre Produktentwicklung jedoch nicht einsetzbar, da Entwicklerteams in der Regel keinen direkten Zugriff mehr auf Kunden oder Benutzer hatten, sondern nur über eine Produkt- bzw. Solution- Management-Organisation. Daher wurde agiles Projektmanagement zunächst nur bei Prototypentwicklung mit starker Kundenbeteiligung verwendet und als eigener „Track“ im *Product Innovation Lifecycle* (PIL) aufgenommen (siehe Abbildung 1). Aufgrund unterschiedlicher Produkteigenschaften und daraus resultierenden Projektanforderungen wurden neben den Standard-PIL-Tracks zusätzlich die sogenannten „White/Yellow/Blue-Prozessstandards“ definiert [Sch10].

White-Prozessstandard: für innovative Lösungen, Prototyp- und Forschungsprojekte. Diese Projekte können ihre Entwicklungsmethoden, wie bspw. Scrum oder traditionelles Projektmanagement, und Tools frei wählen.

Yellow-Prozessstandard: für neue Produkte und große Projekterweiterungen. Als Projektmanagement-Methode wird Scrum eingesetzt (agiler PIL-Track, s.o. und Abbildung 1), Anwender und Kunden müssen bei der Entwicklung miteinbezogen werden. Mit den eigentlichen Entwicklungszyklen wird erst begonnen, wenn ein durchkalkulierter Business Case existiert.

Blue-Prozessstandard: kontinuierliche Weiterentwicklung von existierenden Standardprodukten, wie beispielsweise Anpassungen auf Grund gesetzlicher Bestimmungen, kleinere Anpassungen, um neue Geschäftsprozesse zu implementieren etc. Allerdings ist es in solch einem Projekt nicht erlaubt, dem Produkt neue Technologien hinzuzufügen und umfassende Änderungen an der Benutzeroberfläche vorzunehmen. Als Projektmanagement-Methode kommt hier ein eher traditioneller Ansatz zum Einsatz.

Trotz aller bisherigen Prozessstandards und deren Fokus auf Produkte und Lösungen, gelang es jedoch seit 1997 nicht mehr, Skaleneffekte bzw. Umsatzwachstum mit der wachsenden Anzahl an Mitarbeitern und Produkten zu erzielen [OVoJ]. Basierend auf dieser Erkenntnis wurde nach den möglichen Ursachen hierfür gesucht.

Als *Symptome* stellten sich zunächst folgende Punkte heraus: zu dicht aufeinander folgende Veränderungen, Ad-Hoc-Gegenmaßnahmen bei unterschiedlichsten Problemen, Einmischung und Nichtbeachtung von Managemententscheidungen, lange Durchlaufzeiten in der Entwicklung, Produkte, die nicht den Kundenanforderungen entsprechen bzw. bei denen "Over-Engineering" betrieben wird.

Als *Gründe* wurden u.a. fehlende Gesamtverantwortlichkeit für Kundenzufriedenheit, nicht gelebte Prozessstandards sowie ständiges Wechseln zwischen verschiedenen Tätigkeiten identifiziert. Die Mitarbeiter werden durch dieses Multitasking und die daraus resultierenden fortlaufenden Unterbrechungen unverschuldet weniger produktiv.

Als weitere wesentliche *Ursachen* für die Performance-Probleme in der Entwicklung wurden unklare Produktdefinition und –verantwortung, mangelnde Teamfähigkeit („Silo-Denken") und die deutlich schnellere Entwicklung des Markts für betriebliche Standardsoftware erarbeitet.

Diese Analyse stellt die Grundlage für das SAP „Road 2 Lean" Programm dar.

3 Das "Road 2 Lean"-Programm

Lean Management [Lik04] und agile Entwicklungsmethoden [Agi01] bieten für viele der aktuellen Probleme in der Softwarebranche Lösungsansätze (vgl. Abschnitt 2). Wie bei personenorientierten Prozessen üblich, ist auch bei diesen Ansätzen sowohl Pragmatismus als auch Prozessdisziplin von Nöten. Fehlt einerseits der *Pragmatismus*, versinken die Beteiligten schnell im Chaos oder arbeiten sich an erhöhter Bürokratie ab; fehlt andererseits ein gewisses Maß an *Disziplin*, kann im besten Fall ein „kreatives Chaos" erreicht werden (Abbildung 2). Lean Management in Verbindung mit agilen Entwicklungsmethoden für die Softwareerstellung bietet ein gesundes Mittelmaß zwischen Pragmatismus und Disziplin [Sha07; Lar08].

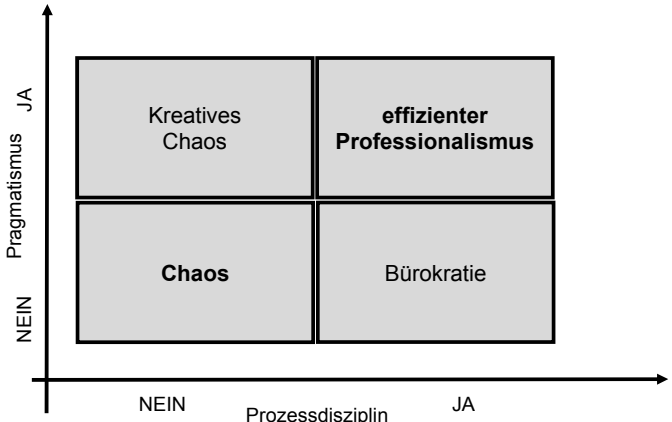

Abbildung 2: Systematik personenorientierter Prozesse

Da Lean nicht gleich agil ist und vice versa, wurden zunächst die Gemeinsamkeiten von Lean und Agilen Methoden untersucht und aufgezeigt (siehe Abbildung 3). Die drei nennenswertesten Gemeinsamkeiten sind das starke Miteinbeziehen des Kunden, der teambasierte Ansatz sowie das Ziel, möglichst hohe Qualität bereits in der ersten Iteration zu erlangen. Darüber hinaus soll die Menge an Anforderungen stets eindeutig in Form eines sog. „Backlogs" priorisiert sein [Sha07].

Aus dem Agilen Manifest [Agi01] geht zudem hervor, dass funktionierende Software als primäres Artefakt aus Kundensicht und zentrales Fortschrittsmaß wichtiger ist als eine umfassende Dokumentation. Darüber hinaus ist eine schnelle Reaktion auf Veränderungen, bspw. von Kundenanforderungen im Projektverlauf, wichtiger als einem festgeschriebenen Projektplan bzw. einer zu Anfang detaillierten Spezifikation zu folgen.

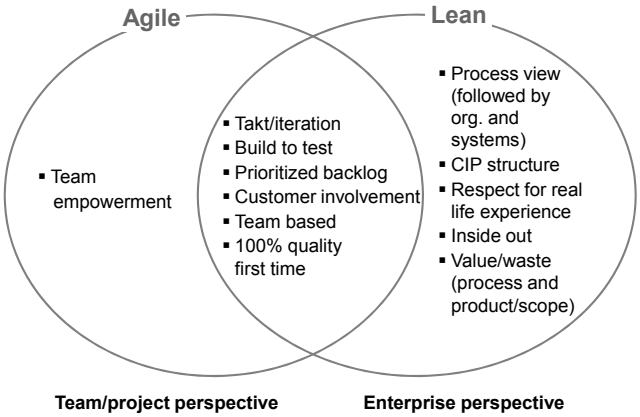

Abbildung 3: Vergleich zwischen Agilen und Lean Prinzipien [Sha07]

Im Unterschied zu traditionellen (plangetriebenen) Ansätzen ist der agile bzw. der Lean-Ansatz nutzengetrieben [Hil06]. Dies zeigt sich u.a. darin, dass sich die Verteilung der fest definierten und variablen Projektparameter des „eisernen Dreiecks“ (Auslieferungstermin, Ressourcenverbrauch und die Menge an Kundenanforderung) grundsätzlich unterscheiden. Während es in der traditionellen Softwareerstellung üblich ist, die Anforderungen, einmal aufgeschrieben, als unveränderlich zu betrachten, Ressourcen und zugesagte Auslieferungsdaten zu ändern, gilt im Lean-Umfeld: Ressourcen und zugesagte Termine sind fix, wohingegen nur die Menge der Anforderungen variabel ist. Allerdings sind alle Anforderungen zueinander eindeutig priorisiert [Sch09], sodass weniger gewichtige in Absprache mit den Kunden gestrichen oder in ein nachfolgendes Release verschoben werden können. Dies erfordert bei allen Beteiligten, sowohl Softwareanbieter als auch Kunden, ein entsprechendes Umdenken aber auch viel Disziplin (siehe Abbildung 2). So sind Kunden beispielsweise im Projektumfeld zunächst ggf. skeptisch und befürchten, nichts oder nur sehr wenig geliefert zu bekommen. Folglich ist ein gegenseitiges Vertrauensverhältnis notwendig, um ein solches Modell in der Praxis umzusetzen und erfolgreich anzuwenden.

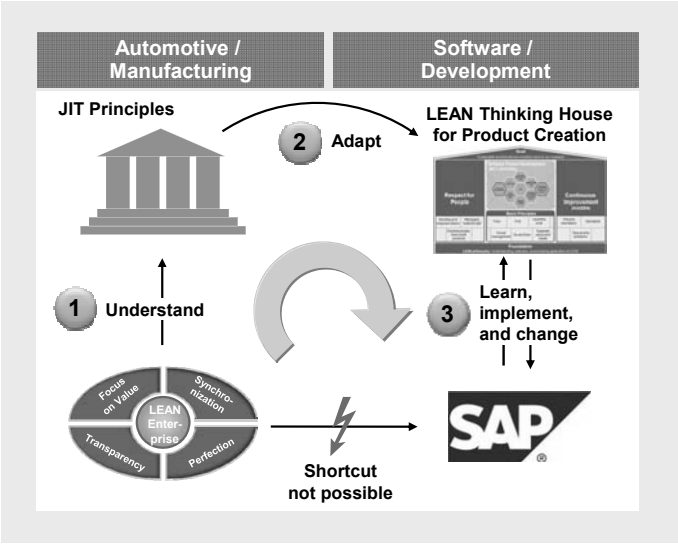

Abbildung 4: Von Lean Manufacturing zu Lean Software Development [SAP09]

Abbildung 4 illustriert den grundlegenden Ansatz, wie „Lean Production“ bzw. die Werte eines „Lean Enterprise“ [Lik04] auf die Softwareindustrie übertragen werden kann. Zunächst war es notwendig, ein einheitliches Verständnis eines „Lean Enterprise“ zu schaffen. Die Hauptbestandteile sind dabei: (a) Konzentration auf nutzenstiftende Tätigkeiten („focus on value“), (b) Synchronisation, (c) Transparenz und (d) Perfektion.

Nach der Identifikation der fundamentalen Probleme und deren Ursachen (siehe auch Abschnitt 2) wurden die etwas konkreteren *Just in Time*-Prinzipien (JIT) aus dem Toyota Produktionssystem (TPS) [Lik04] miteinbezogen. Die JIT-Prinzipien beinhalten „(One-Piece) Flow“ (durchgängige Bearbeitung eines oder weniger Artefakte), „Takt“ (Arbeiten in festen Zeitintervallen), „Pull“ (Entwicklung und Produktion werden primär durch Kundennachfrage gesteuert) und „Zero Defects“ (Fehlervermeidung in Prozess- und Produktqualität).

Aufgrund der Erkenntnis, dass eine direkte Übertragung der Werte und Prinzipien einer „Lean Production“ (wie bei Toyota und Porsche) auf die Softwareindustrie nicht möglich ist (siehe Abbildung 4), wurde als Framework für „Lean Software Product Development“ bei SAP das „Lean Thinking House for Product Creation“ in Anlehnung an Larman und Vodde [Lar08] entwickelt (Abbildung 5). Dies stellt das für SAP adaptierte Verständnis von Lean dar. Auf dieser Grundlage wurden die ersten Lean-Entwicklungsprozesse bei SAP implementiert. Diese Prozesse werden über einen *Countinous Improvement*-Prozess (CIP) ständig weiterentwickelt und optimiert [Rot09].

Das „Lean Thinking House for Product Creation“ vereint Lean- und JIT Prinzipien, erweitert diese um die von SAP definierten Kernelemente, welche zur Umsetzung der Agilen- und Lean- Prinzipien dienen (Abbildung 5). Diese basieren ebenfalls auf der Verknüpfung von Prinzipien aus Lean Management und agiler Softwareentwicklung (siehe Abbildung 3 sowie [Sha07]). Die Kernelemente beinhalten u.a. eine komplette und durchgängige Produktverantwortung (bez. Qualität, Kosten und Auslieferung) in einer Hand, interdisziplinäre Teams zur möglichst eigenständigen Entwicklung einzelner Produkte und Produkteigenschaften („Feature Teams“) [Lar08], ein stets priorisierter Backlog bspw. mit Anforderungen aus Kundensicht (z.B. „User Stories“) [Coh10; Coh04; Gei07] sowie Arbeiten in Takten [Red07] ein potenziell auslieferbares Produktinkrement pro Entwicklungszyklus (bspw. pro Sprint bei Scrum [Sch01; Sch07]).

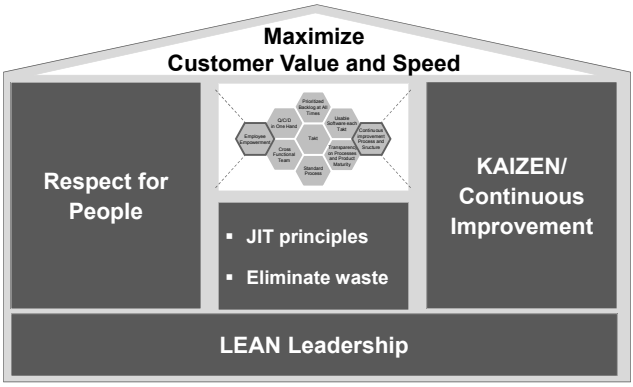

Abbildung 5: LEAN Thinking House (basierend auf [Lar08])

Basierend auf den positiven Erfahrungen mit der Implementierung von Lean und agilen Prinzipien aus mehreren parallelen Pilotprojekten in allen Hauptentwicklungsbereichen wurde Ende 2009 (vgl. Abbildung 6) die Entscheidung getroffen, diese Prinzipien sowie die neun Kernelemente in der gesamten Entwicklungsorganisation umzusetzen, auch über mehrere Vorstandsbereiche hinweg. Abbildung 6 zeigt die Entwicklung der vom „Road2Lean"-Programm betroffenen Mitarbeiter in der Entwicklungsorganisation. Bei dieser laufenden Veränderung liegt derzeit der Fokus auf dem Aspekt „Qualität", mit der Frage wie nach der Umstellung auf kurze Entwicklungszyklen (Takte, Sprints) weiterhin sehr gute Qualität geliefert werden kann. Hierzu wird aus der agilen Welt *Test-Driven Development* (TDD) eingesetzt [Bec03; Fre09].

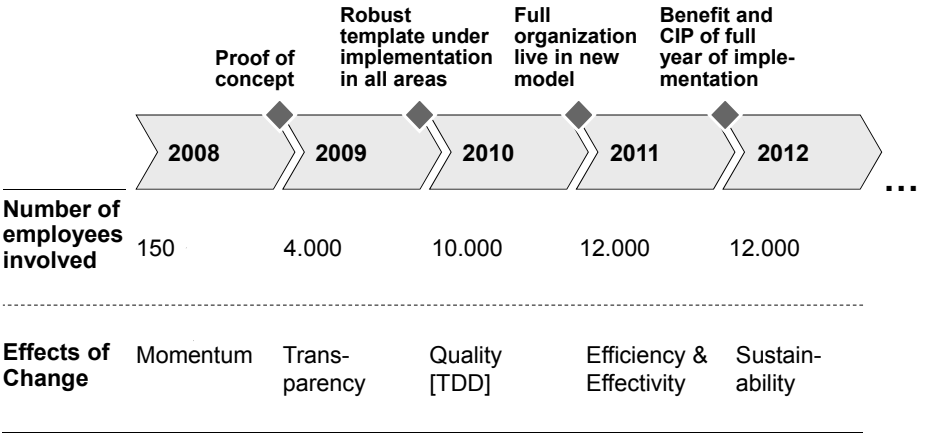

Abbildung 6: Entwicklung des Road2Lean-Pilotprogramms

Ähnlich wie in den Anfangszeiten von SAP (siehe 80er und 90er Jahre in Abbildung 1) wird neben der Standardproduktentwicklung im Bereich „Custom Development" wie bisher sehr kundennah und projektbezogen gearbeitet. Auch im Custom Development ist eine wachsende Anzahl von agilen Projekten, beispielsweise mit Scrum als Projektmanagement-Methode, zu verzeichnen. In der kundenbezogenen Entwicklung kommt es ebenso auf das gegenseitige Vertrauen der beteiligten Partner und zusätzlich auf die gesetzlich vorgeschriebenen Anforderungen des Kunden bezüglich der Vollständigkeit der Spezifikation (Lasten-/Pflichtenheft) zu Projektbeginn an.

4 Fazit

Die Erfahrungen der Vergangenheit, die Parallelen in vergleichbaren Unternehmen und das Feedback aus den heutigen Entwicklungsbereichen deuten darauf hin, dass mit Lean Management und agilen Prinzipien einige der grundlegenden Probleme, wie beispielsweise die direkte Orientierung an Kundenanforderungen, adressiert werden konnten (vgl. Ende Abschnitt 2). Der heutige Prozessmanagement-Ansatz, basierend auf den Lean-Prinzipien, beinhaltet auf der einen Seite Scrum als „Projektmanagement-Methode", andererseits wird hierdurch nur ein grober, aber fester Rahmen für die eigenständige Entwicklung von Produkten durch Team-Strukturen vorgegeben (siehe Abbildung 2). Entsprechend der agilen Prinzipien (u.a. die Betonung auf lauffähiger Software und die Möglichkeit, flexibel auf sich ändernde Kundenanforderungen zu reagieren, siehe Abschnitt 3 und Abbildung 3) liegt der Fokus klar auf dem *Produkt*, das am Kunden ausgerichtet ist. Grundsätzlich wird versucht, die Tugenden der Vergangenheit wieder zu beleben und gleichzeitig die Fehler von damals im aktuellen Kontext nicht zu replizieren.

Die Herausforderung dabei besteht darin, agile Ansätze wie Scrum, über die Teamebene hinaus auf verschiedene Entwicklungsbereiche oder gar auf ein komplettes Softwareunternehmen zu skalieren [Lef07; Lar08; Sch10]. Hierbei helfen die Erkenntnisse aus dem Lean Management-Umfeld sowie neuere Arbeiten zum Thema „Lean Product Development" [Rei09].

5 Ausblick

Parallel zur aktuellen Implementierung der neun Kernelemente (siehe Abschnitt 3) wurde ein Forschungsprojekt im Rahmen der BMBF[1]-Initiative „Softwarecluster bzw. Spitzencluster für Prozessinnovation in der Softwareentwicklung" mit einem Zeitrahmen von insgesamt drei Jahren bis 2013 ins Leben gerufen. Dieser Forschungsverbund setzt sich aus Unternehmen der Softwareindustrie sowie einschlägigen Forschungseinrichtungen und Universitäten zusammen.

Ziel dieses Forschungsvorhabens ist parallel zur operativen Einführung von Lean Management und agilen Methoden in der Softwareindustrie die kausalen Wirkungszusammenhänge innerhalb und zwischen Teams [Lee10] sowie den Prozessfluss in einem „*Lean (Software) Product Development*"-System [Rei09] besser zu verstehen, zu erklären und letztendlich auch Erkenntnisse zu dessen Verbesserung zu erlangen, von der wiederum die Implementierung in der Entwicklung profitieren kann.

[1] BMBF = Bundesministerium" für Bildung und Forschung

Ein Unter-Cluster mit Beteiligung unterschiedlicher SAP-Entwicklungsbereiche, SAP Research und der Universität Mannheim hat basierend auf aktuellen Forschungsfragen drei Blöcke von Fokusthemen identifiziert: (1) Erklärung der Wertschöpfung von Implementierung- und Produktteams, (2) Analyse und Optimierung des Informationsflusses zwischen den Teams, d.h. im Gesamtentwicklungssystem entlang der Wertschöpfungskette sowie (3) Erklärung und Verbesserung des Lern- und Innovationsverhaltens von Teams und der Organisation als Ganzes.

Fokusthema 1 (Team-Ebene) soll Einflussfaktoren wie bspw. die Autonomie und Diversität von Teams auf Output-Größen wie Lieferung von Funktionalität innerhalb von Zeit- und Budgetrestriktionen erheben und Rückschlüsse auf Erfolgsfaktoren ermöglichen [Lee10]. Komplementär dazu will Fokusthema 2 (teamübergreifende Ebene) den Informationsfluss von Portfolio-Entscheidungen hin zu Backlog Items unterschiedlicher Granularität analysieren, Abhängigkeiten [Hil08] und Warteschlangenverhalten [Lar08] in diesem Prozess aufdecken und damit Aussagen zur Optimierung der Durchlaufzeit einzelner Produkt-Features herleiten [Rei09]. Methodisch analog zu Thema 1 will Fokusthema 3 (Lernverhalten) das Lern- und Innovationsverhalten von Teams und der Entwicklungsorganisation erklären [Ham08; Edm07; Sav09].

Neben diesen Fokusthemen werden im Kontext des BMBF-Forschungsprojekts weitere Forschungsfragen wie beispielsweise die sozialwissenschaftlichen Hintergründe agiler Methoden sowie die Auswirkungen der Veränderungen durch Lean und agiler Entwicklung auf die Mitarbeiterzufriedenheit und die wahrgenommene Arbeitsbelastung betrachtet.

Außerhalb des Clusters, jedoch auch im Lean-Kontext und von SAP unterstützt, entsteht in Zusammenarbeit mit der Universität Stuttgart eine Forschungsarbeit, die sich mit der Fragestellung beschäftigt, wie ein flexibler und schlanker Prozess für die Entwicklung von Softwareprodukten, beispielsweise die Skalierung von Scrum, bei großen Softwareherstellern aussehen sollte [Sch10].

Danksagungen

Wir danken den Kollegen Anton Dillinger, Martin Fassunge, Rainer Schmidtke und Joachim Schnitter für ihre Kommentare und Anmerkungen.

Literaturverzeichnis

[Agi01] Agile Alliance, 2001: Agile Manifesto. http://agilemanifesto.org/

[Bec99] Beck, K. 1999. Embracing change with extreme programming. Computer, Jg. 32, H. 10, S. 70–77. http://dx.doi.org/10.1109/2.796139

[Bec03] Beck, K. 2003. Test-Driven Development: By Example. Addison-Wesley

[Boe04] Boehm, B. W., Turner, R. 2003. Balancing agility and discipline: a guide for the perplexed, Addison-Wesley

[Coh04] Cohn, M. 2004. User Stories Applied: For Agile Software Development. Addison-Wesley Professional

[Coh10] Cohn, M. 2010. User stories: für die agile Software-Entwicklung mit Scrum, XP u.a. mitp

[Din10] Dingsoyr, T., Dybå, T., Brede M., N. 2010. Agile Software Development: Current Research and Future Directions. Springer

[Dyb08] Dyba, T., Dingsoyr, T. 2008. Empirical studies of agile software development: A systematic review. Information and Software Technology, Volume 50, Issues 9-10, August, pp. 833-859

[Edm07] Edmondson, A. C., Dillon, J. R., Roloff, K. 2007. Three perspectives on team learning: Outcome improvement, task mastery, and group process. Walsh, J. P., Brief, A.P. (Eds.), The Academy of Management annals, Hillsdale, NJ: Psychology Press, pp. 269-314

[Fre09] Freeman, S., Pryce, Nat, 2009. Growing Object-Oriented Software, Guided by Tests. Addison-Wesley Professional

[Gei07] Geisser, M., Heinzl, A., Hildenbrand, T., Rothlauf, F. 2007. Verteiltes, internetbasiertes Requirements-Engineering. WIRTSCHAFTISINFORMATIK, Volume 49 (3), pp 199-207

[Ham08] Hamilton, P. 2008. Wege aus der Softwarekrise: Verbesserungen bei der Softwareentwicklung. Springer

[Hil06] Hildenbrand, T., Geisser, M., Nospers, M. 2006. Die Übertragbarkeit der Open Source-Entwicklungsmethodik in die Unternehmenspraxis. Softwaretechnik-Trends, Volume 26 (1), pp 37-42

[Hil08] Hildenbrand, T. 2008. Improving Traceability in Distributed Collaborative Software Development – A Design Science Approach. Peter Lang Publishing Group

[Lar08] Larman, C., Vodde, B. 2009. Scaling lean & agile development: thinking and organizational tools for large-scale Scrum. Addison-Wesley

[Lee10] Lee, G. and Xia, W. 2010. Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data. MIS Quarterly, (34: 1), pp.87-114.

[Lef07] Leffingwell, D. 2007. Scaling software agility: best practices for large enterprises. Addison-Wesley

[Lik04] Liker, J. K. 2004. The Toyota Way. McGraw-Hill Professional

[Red07] Redmiles, D., van der Hoek, A., Al-Ani, B., Hildenbrand, T., Quirk, S., Sarma, A., Silveira, S., Filho, R., de Souza, C., Trainer, E. 2007). Continuous Coordination: A New Paradigm to Support Globally Distributed Software Development Projects. WIRTSCHAFTSINFORMATIK, Volume 49, pp. S28-S38

[Rei09] Reinertsen, D. G. 2009. The Principles of Product Development Flow: Second Generation Lean Product Development. Celeritas Publishing

[Rot09] Rother, M. 2009. Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results. McGraw Hill Professional

[Sav09] Savelsbergh, C. M. J. H., van der Heijden, B. I. J. M., Poell, R. F. 2009. The Development and Empirical Validation of a Multidimensional Measurement Instrument for Team Learning Behaviors. Small Group Research: 40, pp. 578-607

[Sch10] Schnitter, J., Mackert, O. 2010. Introducing Agile Software Development at SAP AG – Change Procedures and Observations in a Global Software Company. Proc. ENASE 2010, Athen, pp.132-138
[Sch01] Schwaber, K., Beedle, M. 2001. Agile Software Development with Scrum. Prentice Hall
[Sch07] Schwaber, K. 2007. The enterprise and Scrum. Microsoft Press
[Sch09] Scheibmayr, S., Hildenbrand, T., Geisser, M. 2009. An Experimental, Tool-Based Evaluation of Requirements Prioritization Techniques in Distributed Settings. Proceedings of the Fifteenth Americas Conference on Information Systems (AMCIS), San Francisco, California
[Sha07] Shalloway, A. 2007. Jack be Agile, Jack be Lean. BETTER SOFTWARE June 2007. P.32
[SAP09] SAP AG, 2009. interne Schulungsunterlagen

Software-Integration im „Unternehmen“ Universität: von den Kernprozessen bis zum Controlling

Heinrich C. Mayr

Alpen-Adria-Universität Klagenfurt
Universitätsstrasse 65-67, 9020 Klagenfurt, Austria
mayr@ifit.uni-klu.ac.at

Abstract: Die wachsende Autonomie der Universitäten, d.h. ihrem Wandel von Dienststellen zu selbstständigen Körperschaften, verlangt vom Universitäts-management naturgemäß zunehmend unternehmerisches Denken und Handeln. Dementsprechend wächst auch die strategische Bedeutung einer integrierten IT-Führung aller Universitätsprozesse, also der universitären Geschäftsprozesse. Typischerweise ist dabei die Landschaft der eingesetzten Informations-, Software- und Data Warehouse-Systeme aber nicht homogen, sondern sie ist meist ein Mix aus Eigenentwicklungen und mehr oder weniger gut aufeinander abgestimmten Standardprodukten. Der Weg zu einem Management-Informationssystem, das allen Bedürfnissen einer flexibel agierenden Universitätsleitung gerecht werden könnte, ist noch weit. Dies wohl auch deshalb, weil die mit der Autonomie verbundene Profilentwicklung zu einer stärkeren Spezialisierung, also dem Gegenteil einer Standardisierung universitärer Abläufe geführt hat. Davon unberührt gelten aber Forschung, Lehre und Weiterbildung nach wie vor als die Kernprozesse einer Universität, auf deren bestmögliche Unterstützung die Universitäts-Organisation und alle anderen Universitätsprozesse auszurichten sind.

Die Alpen-Adria-Universität (AAU) setzt dabei auf Selbstorganisation und Selbstbedienung der Lehrenden, Forschenden und Lernenden. D.h. der klassische „Schalter“ ist out, alle Serviceprozesse werden auf Studierende bzw. MitarbeiterInnen und deren jeweilige Kontexte zugeschnitten. Dazu gehört die Gewährleistung der Barrierefreiheit für Menschen mit Beeinträchtigungen in gleicher Weise wie die Gestaltung ortsabhängiger Services bzw. die Integration spezifischer Infrastrukturen.

Der Beitrag gibt einen Überblick, welche dieser Services wie realisiert und integriert werden, und wo Ansätze zum Übergang von der Projekt- zur Produktentwicklung bestehen. Etwas detaillierter wird dabei auf das derzeit in Ausrollung befindliche Vollkostenrechnungssystem der AAU eingegangen, welches dem Universitätsmanagement künftig weitestgehende Kostentransparenz und damit bessere Entscheidungsgrundlagen bieten wird.

Teil II

Eingeladener Vortrag

Vom Projekt zum Produkt durch Produktlinien und Variantenmanagement

Kim Lauenroth

paluno – the Ruhr Institute for Software Technology
Universität Duisburg-Essen
Gerlingstraße 16
45127 Essen
kim.lauenroth@paluno.uni-due.de

Abstract: Die Produktlinienentwicklung ist ein etabliertes Paradigma zur parallelen Entwicklung gleichartiger Softwaresysteme mit hoher Qualität und in kurzer Zeit. Dieser Beitrag gibt einen Überblick über die wesentlichen Konzepte und Bestandteile der Produktionsentwicklung und diskutiert, wie Konzepte der Produktlinienentwicklung einen erfolgreichen Übergang vom Projekt- zum Produktgeschäft unterstützen können.

1 Einleitung

Die Produktlinienentwicklung ist ein etabliertes Paradigma zur parallelen Entwicklung gleichartiger Softwaresysteme mit hoher Qualität und in kurzer Zeit [PBL05]. Die zentrale Idee der Produktlinienentwicklung ist die proaktive Wiederverwendung von Entwicklungsartefakten.

Die Wiederverwendung beschränkt sich hierbei nicht nur auf Realisierungsartefakte wie beispielweise Komponenten oder Codefragmente. Die Proaktivität der Wiederverwendung zeichnet sich dadurch aus, dass bereits das Requirements Engineering und das Design in der Produktlinienentwicklung auf die Wiederverwendung ausgerichtet werden. Eine zentrale Rolle bei dieser Ausrichtung spielt das Varianten- bzw. Variabilitätsmanagement, das darauf abzielt, die notwendige Anpassbarkeit in den Entwicklungsartefakten (d.h. Anforderungen, Architektur, Code, etc.) zu identifizieren, zu dokumentieren und in den einzelnen Entwicklungsphasen zu verwalten.

Dieser Beitrag gibt einen Überblick über die wesentlichen Entwicklungsprozesse der Produktlinienentwicklung (Abschnitt 2) und über das Variantenmanagement (Abschnitt 3). Anschließend wird diskutiert, wie Konzepte der Produktlinienentwicklung einen erfolgreichen Übergang vom Projekt- zum Produktgeschäft unterstützen können (Abschnitt 4).

2 Entwicklungsprozesse der Produktlinienentwicklung

Die Produktlinienentwicklung besteht im Unterschied zur Einzelsystementwicklung aus zwei Entwicklungsprozessen (vgl. [WL99]). Abbildung 1 zeigt das Rahmenwerk der Produktlinienentwicklung nach [PBL05]. Das Rahmenwerk unterscheidet die Entwicklungsprozesse Domänen-Engineering und Applikations-Engineering, welche im Folgenden vorgestellt werden.

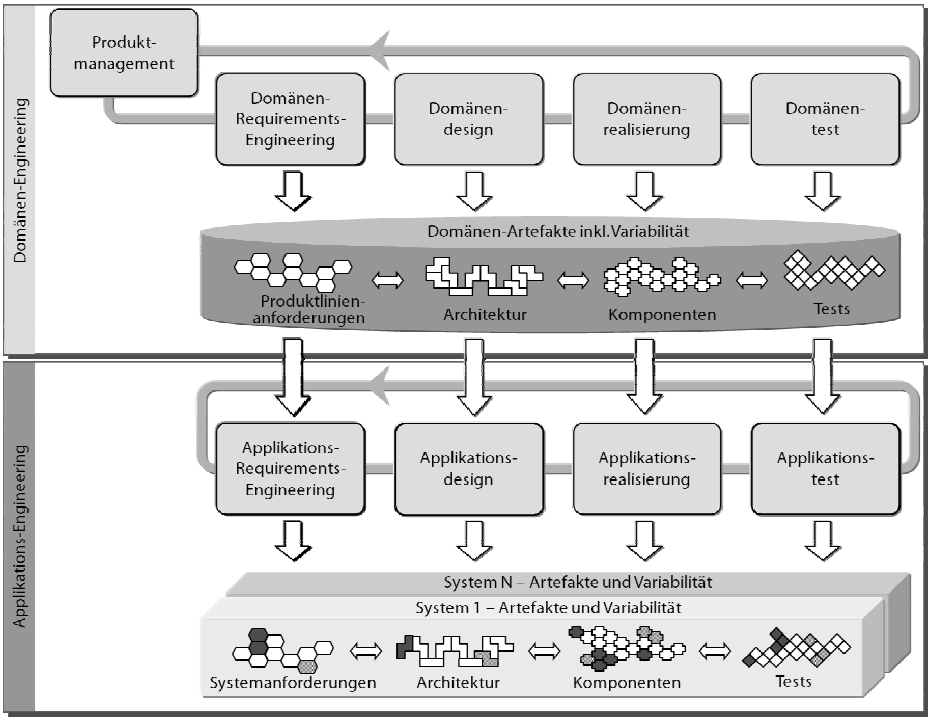

Abbildung 1: Rahmenwerk der Produktlinienentwicklung [PBL05]

2.1 Domänen-Engineering

Das Ziel des *Domänen-Engineering* besteht darin, Domänen-Artefakte der Produktlinie zu planen und zu entwickeln. Unter Domänen-Artefakten werden alle Entwicklungsartefakte verstanden, die für die Entwicklung von Produkten der Produktlinie bereitgestellt werden. Es werden des Weiteren zwei Arten von Domänen-Artefakten unterschieden:

- *Gemeinsame Domänen-Artefakte* sind Bestandteil aller Produkte der Produktlinie. Dies sind zum Beispiel gemeinsame Komponenten, die in allen Produkten vorhanden sind oder Anforderungen, die von allen Produkten erfüllt werden müssen.

- *Variable Domänen-Artefakte* sind wählbar bzw. anpassbar, um kunden- oder marktspezifische Produkte der Produktlinie zu entwickeln. Dies sind zum Beispiel Komponenten, die für spezifische Anwendungsfälle konfiguriert werden können oder auch als Ganzes gewählt werden können.

Zur Entwicklung von Domänen-Artefakten werden im Rahmen des Domänen-Engineering alle aus der Einzelsystementwicklung bekannten Aktivitäten ausgeführt [PBL05]:

- Das *Produktmanagement* befasst sich mit den ökonomischen Aspekten der Produktlinie und ist für marktstrategische Fragestellungen der Produktlinie verantwortlich.
- Das *Domänen-Requirements-Engineering* hat das Ziel, Requirements Engineering für die Produktlinie zu betreiben, um Anforderungen an die Produktlinie zu gewinnen. Im Unterschied zur Einzelsystementwicklung besteht die Herausforderung im Domänen-Requirements-Engineering insbesondere darin, gemeinsame und variable Anforderungen der Produktlinie zu identifizieren und für die nachgelagerten Aktivitäten verfügbar zu machen.
- Das *Domänendesign* entwickelt die Architektur der Produktlinie basierend auf den Produktlinienanforderungen. Die besondere Herausforderung für das Domänendesign besteht darin, die gemeinsamen und variablen Produktlinienanforderungen möglichst effizient umzusetzen und eine Architektur zu entwickeln, die möglichst robust gegenüber Änderungen der Produktlinienanforderungen ist.
- Die *Domänenrealisierung* implementiert die wiederverwendbaren Bestandteile der Produktlinie. Die besondere Herausforderung der Domänenrealisierung besteht darin, die gemeinsamen und variablen Anforderungen möglichst effizient umzusetzen.
- Der *Domänentest* hat die Aufgabe, eine möglichst hohe Qualität der Domänen-Artefakte zu gewährleisten. Die besondere Herausforderung des Domänentests besteht zum einen darin, die Variabilität der Domänen-Artefakte bei der Qualitätssicherung zu berücksichtigen. Dies beinhaltet zum einen, die noch nicht implementierten Bestandteile der Domänen-Artefakte zu berücksichtigen, und zum anderen, die Qualität aller möglichen Variantenausprägungen sicherzustellen.

Ein zentraler Unterschied zwischen der Einzelsystementwicklung und dem Domänen-Engineering ist die explizite Betrachtung und Behandlung von Variabilität in allen Aktivitäten des Domänen-Engineering. Diese zu allen Aktivitäten des Domänen-Engineering querschneidende Aktivität ist Teil des Variantenmanagements und wird im Detail in Abschnitt 3.2 beschrieben.

2.2 Applikations-Engineering

Im *Applikations-Engineering* werden spezifische Produkte der Produktlinie basierend auf den gemeinsamen und variablen Domänen-Artefakten der Produktlinie abgeleitet. Hierzu werden im Applikations-Engineering die folgenden Aktivitäten ausgeführt [PBL05]:

- Im *Applikations-Requirements-Engineering* werden Anforderungen an das abzuleitende Produkt definiert. Die besondere Herausforderung besteht darin, möglichst viele Produktlinienanforderungen wiederzuverwenden und die Abweichungen zwischen kunden- bzw. marktspezifischen Anforderungen und Produktlinienanforderungen zu erkennen.
- Im *Applikationsdesign* wird die Architektur für das abzuleitende Produkt durch Wiederverwendung der Produktlinienarchitektur definiert. Die besondere Herausforderung besteht darin, einen möglichst hohen Grad an Wiederverwendung zu erzielen.
- In der *Applikationsrealisierung* wird das Produkt durch Wiederverwendung der Produktlinienkomponenten realisiert. Die besondere Herausforderung besteht darin, kunden- oder marktspezifische Anpassungen der jeweiligen Produkte mit möglichst geringem Aufwand zu realisieren.
- Im *Applikationstest* wird die Qualität des abgeleiteten Produktes gesichert. Die besondere Herausforderung im Applikationstest besteht darin, die Ergebnisse des Domäne-Test wiederzuverwenden und möglichst keine zum Domänentest redundanten Tests vorzunehmen, d.h. es sollten nur solche Qualitätssicherungsmaßnahmen durchgeführt werden, die kunden- oder marktspezifische Teile des Produktes betreffen.

Ein zentraler Unterschied zwischen der Einzelsystementwicklung und dem Applikations-Engineering besteht darin, dass das alle Aktivitäten des Applikations-Engineering auf den Domänen-Artefakten der Produktlinie aufsetzen und damit auch die Varianteninformation berücksichtigen und nutzen können. Diese Berücksichtigung und Ausnutzung der Varianteninformation ist ebenfalls Bestandteil des Variantenmanagements und wird in Abschnitt 3.3 beschrieben.

3 Variantenmanagement

Die explizite Betrachtung und Behandlung der Variabilität ist, neben der Unterscheidung der zwei Entwicklungsprozesse Domänen- und Applikations-Engineering, ein weiteres wesentliches Unterscheidungsmerkmal zwischen Produktlinien- und Einzelsystementwicklung (vgl. [Kr02]). Variabilität wird in der Literatur definiert als die Fähigkeit eines Domänenartefakts verändert bzw. angepasst zu werden (vgl. [GB02]).

Aufgrund der zentralen Rolle der Variabilität in der Produktlinienentwicklung hat es sich in Forschung und Praxis durchgesetzt, die Variabilität einer Produktlinie in einem eigenständigen Modell, dem sogenannten Variabilitätsmodell, zu definieren. Das Variabilitätsmodell einer Produktlinie hat die Aufgabe zu dokumentieren, in welcher Weise variable Domänenartefakte einer Produktlinie variieren können bzw. anpassbar sind (vgl. [PBL05]), z.B. kann das Variabilitätsmodell definieren, dass sich bestimmte variable Artefakte ausschließen. Dieses eigenständige Variabilitätsmodell bildet die Arbeitsgrundlage für alle Aktivitäten in der Produktlinienentwicklung, die Bezug zur Variabilität der Produktlinie haben.

Im Folgenden wird daher zunächst das orthogonale Variabilitätsmodell als eine mögliche Ausprägung für die Dokumentation von Variabilität im Variantenmanagement vorgestellt. Im Anschluss daran wird in Abschnitt 3.2 bzw. 3.3 das Variantenmanagement im Domänen- bzw. Applikations-Engineering beschrieben.

3.1 Dokumentation von Variabilität mit dem orthogonalen Variabilitätsmodell

Das orthogonale Variabilitätsmodell (OVM, vgl. [PBL05]) unterscheidet zur Dokumentation von Variabilität die folgenden Konzepte:

- *Variationspunkte* beschreiben, was oder welcher Aspekt in einer Produktlinie variiert. Es werden obligatorische und optionale Variationspunkte unterschieden. Obligatorische Variationspunkte müssen bei der Ableitung eines Produktes betrachtet werden, optionale Variationspunkte können betrachtet werden. Darüber hinaus können interne und externe Variationspunkte unterschieden werden. Externe Variationspunkte sind allgemein sichtbar, interne Variationspunkte sind nur für Entwickler und nicht für Kunden sichtbar.
- *Varianten* beschreiben, wie etwas in einer Produktlinie variiert, d.h. Varianten beschreiben mögliche Ausprägungen einen Variationspunktes.
- *Variabilitätsabhängigkeiten* zwischen Variationspunkt und Variante beschreiben, welcher Weise eine Variante an einem Variationspunkt gewählt werden darf. Es werden obligatorische, optionale und alternative Variabilitätsabhängigkeiten unterschieden.
- *Bedingungsabhängigkeiten* beschreiben Einschränkungen in der Auswahl von Varianten und Variationspunkten. Es werden Erfordert- und Ausschlussabhängigkeiten unterschieden.

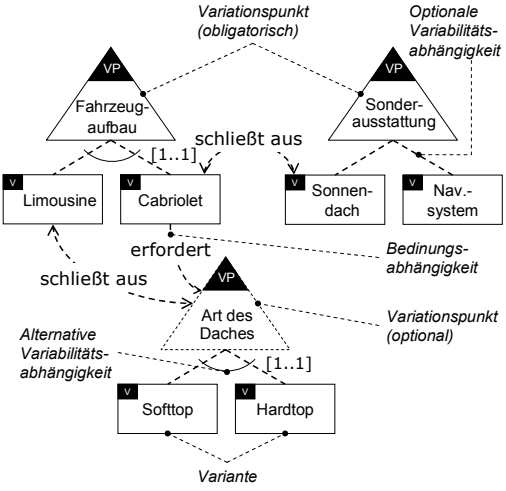

Abbildung 2: Beispiel für ein OVM [LP09]

Abbildung 2 zeigt ein einfaches orthogonales Variabilitätsmodell für PKWs als Beispiel. Es spezifiziert, dass ein Kunde über den Fahrzeugaufbau und die Sonderausstattung entscheiden muss (beides obligatorische Variationspunkte). Der Kunde kann entweder Limousine oder Cabriolet wählen (alternative Abhängigkeit[1]). Wenn der Kunde ein Cabriolet wählt, muss er über die Art des Daches entscheiden (optionaler Variationspunkt, verbunden mit der Variante Cabriolet durch eine Erfordert-Bedingungsabhängigkeit). Gleichzeitig darf er kein Sonnendach als Sonderausstattung wählen (Ausschluss-Bedingungsabhängigkeit). Wenn der Kunde die Variante Limousine wählt, darf er die Art des Daches nicht entscheiden (Ausschluss-Bedingungsabhängigkeit).

Neben der eigentlichen Variabilitätsinformation müssen auch die Auswirkungen der Variabilität auf die verschiedenen Domänen-Artefakte dokumentiert werden. Hierzu definiert das orthogonale Variabilitätsmodell die sogenannten *Artefaktabhängigkeiten*. Eine Artefaktabhängigkeit dokumentiert die Auswirkungen der Variabilität auf die Domänen-Artefakte durch eine Beziehung zwischen Varianten und (Teilen von) Domänen-Artefakten. Wenn eine Variante z.B. mit einer Anforderung in Beziehung steht, ist diese Anforderung nur dann für ein Produkt relevant, wenn die entsprechende Variante gewählt ist. Wenn ein Artefakt(teil) mit keiner Variante in Beziehung steht, handelt es sich um ein gemeinsames Artefakt (oder einen gemeinsamen Artefaktteil), der Bestandteil aller Produkte der Produktlinie ist.

Abbildung 3 zeigt ein Beispiel für Artefaktabhängigkeiten. Zum Beispiel ist die Variante Sonnendach über eine Artefaktabhängigkeit mit der Anforderung R-2 verbunden. Damit ist diese Anforderung nur dann für ein Produkt zu erfüllen, wenn die Variante Sonnendach gewählt ist. Die Anforderung R-3 steht mit keiner Variante in Beziehung. Damit muss diese Anforderung für alle Produkte der Produktlinie erfüllt werden (und damit hat jedes Fahrzeug eine Klimaanlage als Sonderausstattung).

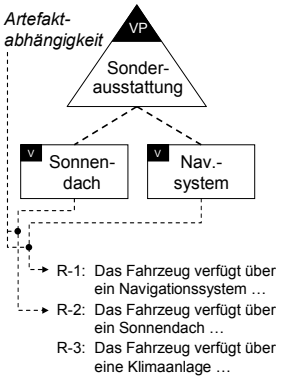

Abbildung 3: Beispiel für Artefaktabhängigkeiten [LP09]

[1] Die Kardinalität in Form von „[1..1]“ gibt an, dass bei dieser Alternative mindestens eine und höchstens eine Variante gewählt werden darf. Andere Kardinalitäten sind ebenfalls möglich, z.B.: beschreibt „[0..2]“, dass keine oder höchstens zwei Varianten gewählt werden dürfen.

3.2 Variantenmanagement im Domänen-Engineering

Das Variantenmanagement im Domänen-Engineering ist eine zu den regulären Entwicklungsaktivitäten querschneidende Aktivität und erfüllt die folgenden Aufgaben [LP09]:

- *Identifikation von Variabilität*: Die Aufgabe der Identifikationsaktivität besteht darin, während der Durchführung der Aktivitäten des Domänen-Engineering Variabilität in den Domänen-Artefakten zu identifizieren. Im Sinne des orthogonalen Variabilitätsmodells besteht die Identifikation von Variabilität darin, mögliche Variationspunkte und Varianten zu identifizieren.
- *Auswirkungsanalyse*: Die Aufgabe der Auswirkungsanalyse besteht darin, die Auswirkung der identifizierten Variabilität auf die Domänen-Artefakte und die übrige Variabilität der Produktlinie zu untersuchen. Im Sinne des orthogonalen Variabilitätsmodells bedeutet die Auswirkungsanalyse zum einen die Analyse der neu identifizierten Variabilität auf neue Bedingungsabhängigkeiten zu anderen Elementen des Variabilitätsmodells und zum anderen die Analyse der Variabilität in Bezug auf Artefaktabhängigkeiten zu Domänen-Artefakten der Produktlinie.
- *Dokumentation von Variabilität*: Die Aufgabe der Dokumentationsaktivität besteht darin, die identifizierte Variabilität zu dokumentieren. Zur Dokumentation der identifizierten Variabilität kann das orthogonale Variabilitätsmodell inklusive der Artefaktabhängigkeiten verwendet werden. (siehe Abschnitt 3.1).

3.3 Variantenmanagement im Applikations-Engineering

Das Variantenmanagement im Applikations-Engineering ist analog zum Domänen-Engineering eine querschneidende Aktivität und erfüllt die folgenden Aufgaben [BHL06]:

- *Kommunikation der Variabilität*: Bei der Ableitung eines Produktes von einer Produktlinie muss die Variabilität der Produktlinie in geeigneter Form an diejenigen Stakeholder kommuniziert werden, die für die jeweilige Entwicklungsaktivität die Entscheidungen treffen sollen. Dies können beispielsweise Kunden, Nutzer des Produktes oder auch Entwickler der Produktlinie sein. Die adäquate Kommunikation der Variabilität an die Stakeholder ist ein essenzieller Erfolgsfaktor für ein Produktlinienprojekt, damit die Stakeholder eine umfassende Grundlage für eine fundierte Entscheidung treffen können.
- *Bindung der Variabilität*: Nachdem eine Entscheidung in Bezug auf die Variabilität getroffen wurde, müssen die Auswirkungen der Variabilität auf die Domänen-Artefakte realisiert werden. Dieser Prozess wird im Allgemeinen auch als Bindung der Variabilität bezeichnet. Zu diesem Prozess gehört beispielsweise, dass nicht gewählte Anforderungen verworfen oder nicht gewählte Komponenten entfernt werden.
- *Auswahldokumentation*: Die Aufgabe der Auswahldokumentation besteht darin, die getroffenen Auswahlentscheidungen der Stakeholder zu dokumentieren.

- *Dokumentation applikationsspezifischer Anpassungen*: Typischerweise können nicht alle Kundenwünsche und Anforderungen durch eine Produktlinie erfüllt werden. Kundenindividuelle Anpassungen an ein Produkt einer Produktlinie werden in diesem Zusammenhang auch als applikationsspezifische Anpassungen bezeichnet und können für die Weiterentwicklung einer Produktlinie von großer Relevanz sein.

4 Unterstützung des Übergangs vom Projekt zum Produkt

Ein Projekt wird im Folgenden dahingehend verstanden, dass ein Projekt ein auf einen spezifischen Kunden hin zugeschnittenes System erstellt. Ein Produkt hingegen ist ein standardisiertes System, das mit dem Ziel erstellt wurde, die Anforderungen eines Marktes zu erfüllen. In der weiteren Diskussion wird insbesondere der Fall betrachtet, dass ein laufendes Projekt unter der Prämisse durchgeführt wird, dass das Ergebnis des Projektes in einem nachgelagerten Schritt zu einem Produkt ausgebaut wird.

4.1 Implikationen aus der Trennung von Domänen- und Applikations Engineering

Im Projektgeschäft werden typischerweise alle Projektaktivitäten auf die Erfüllung der Kundenwünsche hin ausgerichtet, d.h. die konzeptuellen Aktivitäten fokussieren ausschließlich die Anforderungen und Bedürfnisse des Projektes, die Realisierungsaktivitäten setzen diese um und die Qualitätssicherungsmaßnahmen prüfen die korrekte Umsetzung. Die Trennung von Domänen- und Applikations-Engineering in der Produktlinien-entwicklung motiviert sich unter anderem aus der Erkenntnis, dass die erfolgreiche Entwicklung von wiederverwendbaren Domänen-Artefakte eine gezielte Planung und strategische Realisierung erfordert.

Für den erfolgreichen Übergang vom Projekt zum Produkt impliziert dies, dass neben den Anforderungen des konkreten Projektes auch mögliche Anforderungen aus einem späteren potenziellen Produktkontext erfasst, analysiert und dokumentiert werden müssen. Das Requirements Engineering eines Projektes kann parallel zur Erhebung der Anforderungen für das Projekt eine Analyse in Bezug auf mögliche gemeinsame Anforderungen vornehmen, die auch vom späteren Produkt erfüllt werden müssen. In dem Fall, dass Anforderungen nur für den spezifischen Projektkunden relevant sind, kann eine zusätzliche Erhebung möglicher Anforderungen an das geplante Produkt erfolgen. Mögliche Ergebnisse dieses Schrittes sind:

- *Gemeinsame Anforderungen für Projekt und Produkt*: Analog zu gemeinsamen Anforderungen in der Produktlinienentwicklung sind dies Anforderungen, die sowohl für das Projekt als auch für das geplante Produkt gelten.
- *Bekannte variable Anforderungen*: Analog zur Produktlinienentwicklung sind dies Anforderungen, deren Ausprägungen sowohl für das Projekt als auch für das geplante Produkt bekannt sind.
- *Potenziell variable Anforderungen*: Dies sind Anforderungen des Projektes, für die jedoch vermutet wird, dass diese für das geplante Produkt anders ausgeprägt sein können.

- *Spezifische Anforderungen für das Projekt:* Dies sind Anforderungen, die ausschließlich für das Projekt definiert wurden und keine Rolle für das Produkt spielen.

Die zuvor beschriebenen Erkenntnisse über Gemeinsamkeiten und Unterschiede in den Anforderungen von Projekt und Produkt stellen für genommen bereits einen großen Mehrwert dar. Nach Abschluss der Projekttätigkeiten können diese Informationen genutzt werden, um das Projektergebnis zu einem Produkt weiterzuentwickeln. Die Produktlinienentwicklung bietet allerdings noch darüberhinausgehende Möglichkeiten. Der zweite Schritt der strategischen Vorgehensweise in der Produktlinienentwicklung ist ein gezieltes Design und eine gezielte Realisierung von Domänen-Artefakten basierend auf den Ergebnissen aus dem Domänen-Requirements-Engineering.

Für den Übergang vom Projekt zum Produkt impliziert dies, dass die Design- und Realisierungsaktivitäten für das aktuelle Projekt nicht ausschließlich auf den Anforderungen des Projektes basierend sollten, sondern nach Möglichkeit auch bereits die Ergebnisse in Bezug auf das geplante Produkt berücksichtigen sollten. Das zusätzliche Wissen um das geplante Produkt (bspw. in Form von gemeinsamen oder variablen Anforderungen) erlaubt einen wesentlich gezielteren Entwurf der Architektur und eine gezieltere Realisierung in Bezug auf eine schnelle und leichte Anpassung des Projektes hin zum Produkt. Mögliche Ergebnisse dieses Schrittes sind:

- *Gemeinsame Realisierungsartefakte für Projekt und Produkt*: Analog zu gemeinsamen Realisierungsartefakten in der Produktlinienentwicklung sind dies Artefakte, die unverändert im Projekt und im Produkt verwendet werden können.
- *Spezifische Realisierungsartefakte für das Projekt*: Dies sind Realisierungsartefakte, die ausschließlich für das Projekt erstellt werden.
- *Potenziell variable Realisierungsartefakte*: Dies sind Realisierungsartefakte, die für das Projekt entwickelt wurden, wobei Anpassungsmöglichkeiten vorgesehen sind.
- *Variable Realisierungsartefakte*: Dies sind Realisierungsartefakte, die durch definierte Anpassung sowohl im Projekt als auch im Produkt verwendet werden können.

Zusammenfassend kann festgehalten werden, dass das Wissen um die Gemeinsamkeiten und Unterschiede (bzw. Varianten) zwischen dem Projekt und dem geplanten Produkt einen entscheidenden Beitrag für eine erfolgreiche Produktentwicklung darstellt.

4.2 Implikationen aus dem Variantenmanagement

Für die Produktlinienentwicklung ist das Variantenmanagement ein essenzieller Erfolgsfaktor, um die Komplexität der Gemeinsamkeiten und Varianten der Produktlinie zu beherrschen. Im vorherigen Abschnitt wurde gezeigt, dass das Wissen um Gemeinsamkeiten und Varianten zwischen Projekt und Produkt einen großen Wert besitzt. Durch ein angemessenes Variantenmanagement im Projekt kann dieses Wissen für den Entwicklungsprozess und das geplante Produkt verfügbar und damit nutzbar gemacht werden. Prinzipiell kann der Reifegrad des Variantenmanagement wie folgt klassifiziert werden:

- *Zufällig*: Gemeinsamkeiten und Varianten werden zufällig identifiziert, aber nicht dokumentiert. Das Wissen über mögliche Gemeinsamkeiten und Varianten befindet sich ausschließlich im Kopf der Entwickler.
- *Unstrukturiert*: Gemeinsamkeiten und Varianten werden zufällig identifiziert und unstrukturiert dokumentiert, beispielsweise durch textuelle Annotationen in Entwicklungsartefakten.
- *Strukturiert*: Gemeinsamkeiten und Varianten werden bewusst identifiziert, sowie eigenständig und strukturiert dokumentiert, beispielsweise in gesonderten Dokumenten oder Modellen.
- *Umfassend*: Zusätzlich zu den Gemeinsamkeiten und Varianten werden die Auswirkungen der Gemeinsamkeiten und Varianten auf die jeweiligen Entwicklungsartefakte analysiert und strukturiert durch geeignete Nachvollziehbarkeitsinformationen dokumentiert.

In Abhängigkeit davon, in welcher Ausprägung das Variantenmanagement für ein Projekt betrieben wird, ergeben sich verschiedene Vorteile für den Übergang von Projekt zum Produkt. Im Requirements Engineering für das Projekt kann bereits ein erster Mehrwert erzielt werden, wenn die Gemeinsamkeiten und Varianten von Projekt und Produkt unstrukturiert gemanagt werden. Durch das Bewusstsein über die Existenz von Varianten können Gemeinsamkeiten und Unterschiede identifiziert und für weitere Entwicklungsaktivitäten verfügbar gemacht werden. Ein strukturiertes Variantenmanagement macht dieses Wissen strukturiert für weitere Entwicklungsaktivitäten im Projekt verfügbar. Das umfassende Variantenmanagement im Requirements Engineering bietet den größten Mehrwert für den Übergang vom Projekt zum Produkt, da nicht nur die Gemeinsamkeiten und Varianten, sondern auch die Auswirkungen in Bezug auf die Anforderungen verstanden und dokumentiert werden. In dieser Ausprägung würde das Requirements Engineering für das Projekt in wesentlichen Aspekten dem Requirements Engineering für eine Produktlinie entsprechen.

Die Design- und Realisierungsaktivitäten profitieren aufgrund der typischen Komplexität von Design und Realisierungsartefakten nur in geringer Weise von einem unstrukturierten Variantenmanagement. Ein strukturiertes Variantenmanagement in Bezug auf Design- und Realisierungsaktivitäten erlaubt es hingegen, dass die Gemeinsamkeiten und Unterschiede zwischen Projekt und Produkt explizit erfasst werden und ermöglicht dadurch einen wesentlich effektiveren Übergang vom Projekt zum Produkt. Zum einen sind die gemeinsamen Aspekte bekannt, d.h. die Design- und Realisierungsartefakte können gezielt entwickelt und auf Bestandteile untersucht werden, die unverändert in das Produkt übernommen werden können. Des Weiteren sind die variablen Aspekte bekannt. Dies erlaubt wiederum eine gezielte Entwicklung und Suche nach Anpassungen der vorhandenen Design- und Realisierungsartefakte.

Ein umfassendes Variantenmanagement der Design- und Realisierungsartefakte bietet den größten Mehrwert für den Übergang vom Projekt zum Produkt, da neben den explizit dokumentierten Gemeinsamkeiten und Varianten auch die Auswirkungen auf die Design- und Realisierungsartefakte bekannt und dokumentiert sind. Dies ermöglicht zum einen den unmittelbaren Zugriff auf diejenigen Bestandteile der Design- und Realisierungsartefakte, die unverändert in das Produkt übernommen werden können und zum anderen werden explizit diejenigen Stellen dokumentiert, die einer Anpassung im geplanten Produkt bedürfen.

5 Zusammenfassung und Fazit

In diesem Beitrag wurden die wesentlichen Konzepte der Produktlinienentwicklung vorgestellt: Trennung von Domänen- und Applikations-Engineering sowie die explizite Betrachtung von Gemeinsamkeiten und Varianten durch das Variantenmanagement. Es wurde gezeigt, dass diese Konzepte einen erfolgreichen Übergang vom Projekt zum Produkt unterstützen können.

Eine wesentliche Erkenntnis für den Übergang vom Projekt zum Produkt aus der Produktlinienentwicklung ist das Wissen um die Gemeinsamkeiten und Unterschiede von Projektergebnis und Produkt. Das Vorhandensein dieses Wissens erlaubt eine wesentlich strukturiertere Ausrichtung der Entwicklungsaktivitäten des Projektes in Bezug auf das geplante Produkt. Beispielsweise können frühzeitig gemeinsame Bestandteile realisiert werden, die unverändert vom Projekt in das Produkt übertragen werden können. Verschiedene Ausprägungen des Variantenmanagements können dieses Wissen in mehr oder minder stark strukturierter Weise den jeweiligen Entwicklungsaktivitäten zuführen.

Literaturverzeichnis

[BHL06] Bühne, S., Halmans, G., Lauenroth, K., Pohl, K.: Scenario-based Application Requirements Engineering. In: Software Product Lines - Research Issues in Engineering and Management. Springer, 2006.

[GB02] Geyer, L.; Becker, M.: On the Influence of Variabilities on the Application Engineering Process of a Product Family. Proceedings of the 2nd the Second Software Product Line Conference 2002.

[Kr02] Krueger, C.: Variation Management for Software Product Lines – A Case Study. Proceedings of the 2nd the Second Software Product Line Conference 2002.

[LP09] Lauenroth, K.; Pohl, K.: Variabilität als eigenständige Sicht auf Software-Produktlinien. In: Produkt-Variabilität im gesamten Lebenszyklus, Workshop im Rahmen der Tagung Software Engineering 2009.

[PBL05] Pohl, K.; Böckle, G.; van der Linden, F.: Software Product Line Engineering – Foundations, Principles, and Techniques. Springer, Heidelberg, 2005.

[WL99] Weiss, D.; Lai, C.: Software Product-Line Engineering – A Family-Based Software Development Process. Addison-Wesley, Reading, 1999.

Teil III

Wissenschaftliches Programm

Prozess- und Produkterfolg in IS-Entwicklungsprojekten – Die Perspektive der Auftragnehmer

Dirk Basten, Dominik Joosten, Werner Mellis

Seminar für Wirtschaftsinformatik und Systementwicklung
Universität zu Köln
Pohligstraße 1
50969 Köln
{dirk.basten, dominik.joosten, werner.mellis}@uni-koeln.de

Abstract: Zielsetzung: Das Ziel des vorliegenden Beitrags besteht darin, empirisch gestützt ein deskriptives Modell des IS-Entwicklungsprozesses zu entwickeln, das eine valide Messung des Gesamterfolgs von IS-Projekten gestattet. **Ansatz:** Die Modellentwicklung geht von einem idealtypischen IS-Entwicklungsprozess aus, der die beiden Komponenten Prozess und Produkt beinhaltet. Unter Verwendung der Strukturgleichungsmodellierung werden Daten einer quantitativen Fragebogenstudie mit Projektbeteiligten auf Seiten der Auftragnehmer hinsichtlich der zu berücksichtigenden Erfolgskriterien analysiert. **Ergebnisse:** Subjektiv wahrgenommener Erfolg kann anhand von Effizienz und individueller Zufriedenheit der Mitarbeiter (Prozess) sowie Kundenzufriedenheit (Produkt) gemessen werden. **Konklusionen:** Die bisherige alleinige Konzentration auf die Planungstreue in der Erfolgsbeurteilung von Projekten ist unzureichend und muss erweitert werden. **Originalität:** Die vorliegende Studie ist ein neuer Ansatz zur verbesserten Bewertung von Projekten.

1 Einleitung

Die umfassende, d. h. über eine reine Beurteilung der Planeinhaltung hinausgehende Bewertung des Erfolgs von Informationssystem- bzw. Softwareprojekten (im Folgenden: IS-Projekt, IS-Projekterfolg) ist ein bis heute nicht befriedigend gelöstes Problem [TF08]. Sowohl in der Praxis als auch der empirischen IS-Forschung existiert eine Vielzahl unterschiedlicher Konzepte und Konstrukte zur Definition und Messung des IS-Projekterfolgs. Diese stimmen darin überein, dass sie den IS-Projekterfolg als ein mehrdimensionales Konstrukt auffassen [Al02a], [Sa96], das neben der Planungstreue weitere Erfolgskomponenten aufweist. Diese so genannten Dimensionen bilden für die verschiedenen Stakeholder eines Projekts relevante Kriterien des Erfolgs ab [At99], [Ke06], [Wa98]. In Bezug auf die Beantwortung der Frage, welche weiteren Dimensionen in eine adäquate Erfolgsbeurteilung einzubeziehen sind, kann allerdings keine Übereinstimmung festgestellt werden [Al02b], [BH01], [De96], [HL92]. D. h., ein einheitliches, weil allgemein anerkanntes, Konstrukt des IS-Projekterfolgs existiert nicht.

Ferner muss konstatiert werden, dass ein großer Teil der verschiedenen verwendeten Konzeptualisierungen und Operationalisierungen eine systematische und theoretisch fundierte Herleitung vermissen lässt.

Sowohl für die unternehmerische Praxis als auch die empirische IS-Forschung wäre ein solches einheitliches und anerkanntes Konstrukt des IS-Projekterfolgs, das eine valide Messung des Erfolgs von Projekten gestatten würde, von großem Wert. Ergebnisse von empirischen Studien, die den IS-Projekterfolg als Zielgröße verwenden, etwa um die Zweckmäßigkeit von Gestaltungsempfehlungen des IS-Entwicklungsprozesses zu untersuchen, sind nicht vergleichbar, sofern sie unterschiedliche Konstrukte zur Messung verwenden. Ohne eine korrekte Prognose des Projekterfolgs während des Projektes und ohne eine korrekte Messung nach Abschluss des Projektes besteht ein kritisches Risiko, dass Projektverantwortliche falsche Entscheidungen für das aktuelle und für zukünftige Projekte treffen.

Ziel des vorliegenden Beitrages ist es, empirisch gestützt, ein deskriptives ökonomisches Modell des Softwareentwicklungsprozesses zu entwickeln, das valide Aussagen über den Erfolg von IS-Projekten abzuleiten gestattet. Für die Modellentwicklung sind zunächst zwei Fragen zu beantworten:

(1) Was ist ein erfolgreiches Projekt, d. h. formaler ausgedrückt, welche Eigenschaften charakterisieren ein erfolgreiches Projekt oder was sind die Dimensionen des Konstrukts IS-Projekterfolg?

(2) Wie kann der Erfolg eines IS-Projekts über seine Dimensionen valide gemessen werden?

2 Theoretische Herleitung

Traditionell wird der Erfolg von Projekten im Allgemeinen genauso wie von IS-Projekten anhand der Planungstreue, also der Einhaltung des Budgets und des Zeitplans sowie der Erfüllung der spezifizierten funktionalen und nicht-funktionalen Anforderungen, gemessen [At99], [Pi04]. Dies ist zum einen darauf zurückzuführen, dass die Ziele von Projekten anhand dieser Kriterien definiert werden [Ke06]. Zum anderen stellt die Planungstreue eine objektive Bewertung dar, die einfach zu messen ist [PS88].

Die Einhaltung von Plänen entspricht in vielen Fällen jedoch nicht der subjektiven Wahrnehmung von Projektbeteiligten hinsichtlich des Gesamterfolgs [FM07]. Daher werden trotz der Betonung der Wichtigkeit der Planeinhaltung weitere Erfolgskriterien, wie bspw. Kundenzufriedenheit und Prozesseffizienz, gefordert [RR00], [TF08], [Wa98]. Obwohl Einigkeit darüber herrscht, dass der IS-Projekterfolg ein mehrdimensionales Konstrukt ist, kann anhand der Vielzahl der unterschiedlichen Messansätze (vgl. dazu bspw. die unterschiedlichen Ansätze in [Al02b], [BH01], [BK92], [HL92]) festgestellt werden, dass immer noch keine Einigkeit darüber besteht, welche Dimensionen bei der Bewertung des Gesamterfolgs eines IS-Projekts zu berücksichtigen sind.

Abhängig von der Vertragsgestaltung (Festpreis, Entlohnung des Aufwands, Vertragsstrafen) legt der Auftragnehmer im Allgemeinen großen Wert auf die Einhaltung von Zeit- und Budgetplänen. Für den Auftraggeber dagegen ist in der Regel die Qualität des entwickelten Produkts von größter Bedeutung. Es erscheint fraglich, ob die Herleitung einer Erfolgsbeurteilung, die den Interessen aller Projektbeteiligten genügt, zielführend oder überhaupt möglich ist, da sich die Erfolgswahrnehmung der verschiedenen Gruppen unterscheiden [FB92], [FW99], [Pi04], [St86]. In dieser Arbeit wird der IS-Projekterfolg aus Sicht des Auftragnehmers und dort speziell aus Sicht von Projektleitern und -mitarbeitern untersucht. Der Anspruch von Objektivität und das Ziel der Vereinheitlichung muss jedoch nicht aufgegeben werden, da der Auftragnehmer selbst ein vitales Interesse daran hat, die Sicht des Auftraggebers ebenfalls zu berücksichtigen.

Losgelöst davon stellt sich die Frage, wie ein methodisch und sachlich korrektes und angemessenes Vorgehen zur Entwicklung eines Modells des IS-Projekterfolgs aussehen kann. In einer idealtypischen Betrachtung besteht ein Projekt aus einem Prozess, dem IS-Entwicklungsprozess, und einem Produkt, dem IS, das durch diesen Prozess erzeugt wird. Für eine Erfolgsbeurteilung eines solchen Projektes sind ebenfalls diese beiden Komponenten heranzuziehen, da

(a) das erstellte Produkt den Nutzen für den Auftragnehmer manifestiert, welcher durch das Projekt erzielt werden soll und

(b) entsprechend einer Annahme, wie sie Ansätzen wie etwa dem PSQM [Me04] oder dem CMMI [CKS08] zugrunde liegt, die Qualität bzw. der Erfolg des Entwicklungsprozesses die Qualität bzw. den Erfolg des Produktes determiniert.

Aus betriebswirtschaftlicher Sicht sind zur Bewertung eines Projektes die Effizienz und die Effektivität als geeignete Konzepte heranzuziehen (vgl. ähnliche Ansätze [Ba99], [HL92], [Li10]). Ein Prozess kann betriebswirtschaftlich als erfolgreich bewertet werden, wenn er dem ökonomischen Prinzip folgt, d. h. wenn er in Bezug auf Ressourcenverzehr und Leistungserstellung optimal, also effizient ist. Ein Produkt als Ergebnis eines Prozesses ist hinsichtlich der Zielerreichung der mit dem Zweck seiner Erstellung verbundenen Erwartungen, d. h. also in Bezug auf seine Effektivität zu bewerten. Somit ergibt sich folgendes konzeptuelles Modell (vgl. Abb. 1) des IS-Projekterfolgs.

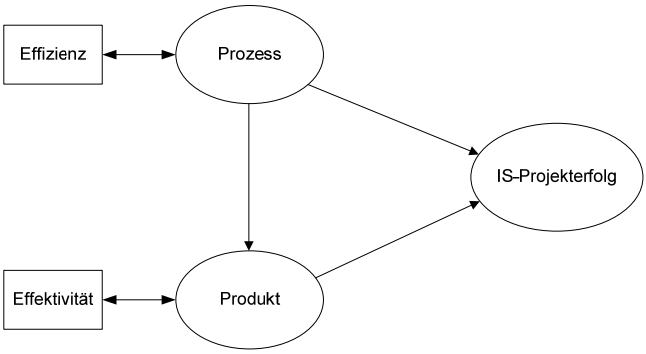

Abbildung 1: Konzeptuelles Modell des IS-Projekterfolgs

Im Folgenden wird nun überprüft, inwieweit das theoretisch entwickelte Modell durch empirische Ergebnisse gestützt werden kann. Dazu wird zunächst die empirische Studie beschrieben, die dazu herangezogen wird. In dieser werden die Einflussfaktoren des IS-Projekterfolgs auf den durch Entscheidungsträger auf Auftragnehmerseite subjektiv wahrgenommenen Gesamterfolg erhoben. Dem weiteren Vorgehen liegt die Annahme zugrunde, dass jene Faktoren, die einen besonders hohen Einfluss auf den durch die Studienteilnehmer subjektiv wahrgenommenen Gesamterfolg des IS-Projekterfolgs haben, geeignet sind, diesen objektiv zu messen. Diese Annahme erlaubt die Verwendung der Strukturgleichungsmodellierung als Analyseinstrument, da so das Problem kausalanalytisch formuliert werden kann.

3 Beschreibung der Studie

In dieser Arbeit werden erste Ergebnisse einer laufenden Studie vorgestellt, die in Kooperation mit der Deutschen Gesellschaft für Projektmanagement e.V. (GPM) durchgeführt wird. Ziel dieser Studie ist die Erhebung der Erfolgswahrnehmung von Projektbeteiligten auf Seiten von Auftragnehmern von IS-Projekten. Dazu werden Projektleiter und weitere Entscheidungsträger in IS-Projekten mittels eines Fragebogens hinsichtlich ihrer Wahrnehmung verschiedener potenzieller Dimensionen des IS-Projekterfolgs in Bezug auf ein konkretes zugrunde zu legendes Projekt befragt. Ferner werden die potenziellen Dimensionen des IS-Projekterfolgs hinsichtlich ihrer beigemessenen Bedeutung losgelöst vom konkreten Projekt durch die Teilnehmer bewertet.

3.1 Herleitung potenzieller Erfolgsdimensionen

Die Studie geht von einer als vollständig erachteten Menge potenzieller Dimensionen des IS-Projekterfolgs aus. Zur Identifizierung der möglichen Dimensionen wurden von uns zwei unabhängige Forschungsprojekte durchgeführt, die im Folgenden kurz erläutert werden.

Zur Identifizierung bereits in der Literatur vorgeschlagener Erfolgsdimensionen wurde ein systematisches Literaturreview von zehn, bis auf eine Ausnahme, englischsprachigen Zeitschriften durchgeführt, die aufgrund ihrer Bedeutung für die IS-Forschung ausgewählt wurden. Die Zeitschriften des Senior Scholars' Basket of Journals (http://home.aisnet.org/displaycommon.cfm?an=1&subarticlenbr=346) sind in der Auswahl enthalten. Die Suche nach relevanten Beiträgen erfolgte dabei in einer vollständigen manuellen Durchsicht der Artikel, die in der Zeit von Januar 1980 bis Juni 2007 in diesen Zeitschriften veröffentlicht wurden.

In einem zweiten Forschungsprojekt wurden semi-strukturierte Interviews basierend auf der Repertory-Grid-Technik (für nähere Informationen zu dieser Methode vgl. [Cu08], [TH02]) und erweitert um das so genannten Laddering (für nähere Informationen zu dieser Methode vgl. [RG88], [Ru02]) mit 11 leitenden Managern von IS-Projekten in Deutschland geführt.

Potentielle Erfolgsdimensionen
Anforderungserfüllung (Erfüllung der funktionalen Anforderungen)
Budgettreue/Termintreue[1]
Effizienz des Prozesses
Flexibilität/Wartbarkeit des Produkts
Grad der Fehlerfreiheit des Produkts
Individuelle Benutzerzufriedenheit
Individuelle Mitarbeiterzufriedenheit
Qualität der Planung
Qualität der Schätzung
Qualität des Produkts (Erfüllung der nicht-funktionalen Anforderungen)
Technische Innovativität des Produkts/Lerneffekte
Transparenz im Projekt
Unterstützung der strategischen Unternehmensziele
Wirtschaftlicher Erfolg des Produkts
Wirtschaftlicher Erfolg des Projekts
Zufriedenheit des Auftraggebers (als Organisation)/Kundenzufriedenheit
Zufriedenheit des Auftragnehmers (als Organisation)

Tabelle 1: Übersicht der identifizierten Erfolgsdimensionen

[1] Budget- und Termintreue sind hoch korreliert und bilden in der statistischen Auswertung einen gemeinsamen Faktor, auch wenn argumentiert werden kann, dass sie konzeptuell unterschiedlich sind.

Die Interviews zielten darauf ab, durch den paarweisen Vergleich von jeweils zwei aus insgesamt vier Projekten der Befragten Faktoren zu erheben, die in dem erfolgreicheren Projekt zum Erfolg führten. Die Verdichtung dieser Faktoren lieferte anschließend mögliche Erfolgsdimensionen.

Die Ergebnisse der beiden Studien wurden konsolidiert. Eine Übersicht aller identifizierten Dimensionen in alphabetischer Reihenfolge kann Tabelle 1 entnommen werden. Die Definitionen dieser Dimensionen wurden entweder aus der entsprechenden Literaturquelle oder den zugehörigen Interviews abgeleitet.

3.2 Entwicklung des Fragebogens

Die Teilnehmer bewerten im Fragebogen sowohl ein konkretes Projekt hinsichtlich der potenziellen Erfolgsdimensionen jeweils anhand von 3 bis 6 Items als auch davon losgelöst die Bedeutung der Dimensionen für den IS-Projekterfolg. Zusätzlich wurde ebenfalls die Wahrnehmung des Gesamterfolgs und des Prozess- bzw. Produkterfolgs erfasst. Bis auf Ausnahmen (objektivierbare Maße der Planung) erfolgt die Beurteilung jeweils auf einer 7-Punkt-Likert-Skala (1 „trifft ganz und gar nicht zu“ - 7 „trifft voll und ganz zu“). Die Items wurden ausgehend von den aus der Literatur und den Interviews gewonnen Definitionen initial durch Mitarbeiter unseres Seminars entwickelt. Der Fragebogen wurde im Folgenden sowohl von projektexternen Forschern evaluiert als auch einem Pretest durch mehrere Softwareexperten unterzogen. Dies führte sowohl zu einer Präzisierung der Definitionen als auch der Items.

3.3 Erhebung der Daten

Die Teilnehmergewinnung für die Studie erfolgt über zwei Wege. Zum einen wird durch die GPM in ihren Newslettern für die Teilnahme geworben. Zum anderen wird auf Unternehmenskontakte des Seminars zurückgegriffen. Kontaktpersonen werden per Email um eine Teilnahme gebeten. Für die vorliegende Analyse konnten insgesamt 39 auswertbare Fragebögen verwendet werden.

3.4 Charakteristika der Teilnehmer und untersuchten Projekte

Die Mehrheit der Teilnehmer (72 %) hat in den betrachteten Projekten die Rolle des Projektleiters übernommen. Die übrigen Teilnehmer sind überwiegend Entwickler und Analysten. Die Berufserfahrung im Bereich IT liegt bei einem Mittelwert von 17 Jahren. Es handelt sich daher um sehr erfahre Softwareexperten, die im Durchschnitt bereits an mehr als 30 IT-Projekten insgesamt und mehr als 17 Projekten in ihrer jeweils angegeben Rolle teilgenommen haben. Zwei Drittel der Teilnehmer verfügen über einen Hochschulabschluss.

61 % der Projekte liegen intern (unternehmens- und/oder konzernintern) vergebene Aufträge zugrunde. Der Anteil der intern vergebenen Projekte sowie Anzahl der Studienteilnehmer schwächt möglicherweise die Verallgemeinerbarkeit der Ergebnisse.

Eine Übersicht über die Branchen, in denen die Projekte durchgeführt werden, liefert Tabelle 2. Tabelle 3 zeigt weitere Charakteristiken der untersuchten Projekte. Gemessen an den Projektkosten sind die Projekte verglichen mit anderen Studien umfangreicher (vgl. z. B. [Th03]).

Branche	**Anteil**
Banken / Versicherungen / Finanzdienstleistungen	51 %
Logistik / Verkehr	15 %
Telekommunikation / Software	10 %
Chemie / Pharmaindustrie	5 %
Verwaltung / Öffentlicher Dienst	5 %
Sonstiges	5 %
Versorgung	3 %
Fertigungsindustrie	3 %
Luft- und Raumfahrt	3 %

Tabelle 2: Branchen der Projekte

	Median	Mittelwert
Projektdauer in Monaten	11	18
Projektkosten in €	500.000	3.533.000
Mitarbeiterzahl	10	21

Tabelle 3: Projektdauer, -kosten, und -mitarbeiterzahl

4 Auswertung und Ergebnisse

Die Analyse des Einflusses der einzelnen Dimensionen auf den Gesamterfolg wird mit Hilfe von reflektiven Konstrukten und der Strukturgleichungsmodellierung auf Basis des Partial Least Squares-Ansatzes (PLS) durchgeführt. Dazu wurde die Software SmartPLS [RWW05] verwendet. PLS ist bei dieser Auswertung aufgrund des explorativen Charakters der Untersuchung, der Anzahl von Konstrukten und des Stichprobenumfangs gegenüber den kovarianz-basierten Verfahren zu bevorzugen [CN99], [Fo87].

Zur Überprüfung der Eindimensionalität der einzelnen Dimensionen wurden zunächst Faktorenanalysen mit PASW Statistics 18 (Programm zur statistischen Auswertung, ehemals SPSS) durchgeführt. Items, die nicht einem Faktor eindeutig zugeordnet werden konnten, wurden entfernt.

Die Ergebnisse der anschließenden Pfadanalyse (Path Weighting Scheme, Mean 0, Var 1, Maximum Iterations 300) zeigen, dass nur wenige der potenziellen 17 Dimensionen für den wahrgenommenen Gesamterfolg entscheidend sind, d. h., dass Sie einen bedeutsamen Einfluss auf diesen aufweisen. Größten Einfluss auf den durch die Auftragnehmervertreter wahrgenommen Gesamterfolg haben die effiziente Nutzung der Ressourcen und die Zufriedenheit der Projektmitarbeiter. Für das entwickelte Produkt ist in erster Linie die Kundenzufriedenheit wichtig. Der Planungstreue und der Erfüllung der funktionalen Anforderungen kommt für den Gesamterfolg keine entscheidende Rolle zu. Obwohl sie vergleichbar hohe Pfadgewichte aufweisen, ist ihre Effektstärke nicht bedeutend, d. h., sie leisten keinen relevanten Beitrag zur Erklärung der Varianz des Gesamterfolgs. Die nachfolgenden Analysen beziehen sich daher nur auf die ausgewählten Dimensionen. Eine Liste der Definitionen der Dimensionen und eine Übersicht der zugehörigen Items finden sich in den Anhängen A und B.

4.1 Messmodell

Um Aussagen über das Strukturmodell treffen zu können, muss für das Messmodell ein ausreichendes Maß an Reliabilität und Validität nachgewiesen werden [FL81]. Für die Bewertung der Reliabilität und Validität eines Messmodells mit ausschließlich reflektiven Indikatoren stehen die vier folgenden Kriterien zur Verfügung.

Die *interne Konsistenz* aller Konstrukte kann entsprechend Cronbach's Alpha (für jedes Konstrukt größer als 0,84) [Nu78] und der Konstruktreliabilität (für jedes Konstrukt größer als 0,89) nachgewiesen werden [WLJ74].

Indikatorreliabilität ist gegeben, wenn mindestens 50 % der Indikatorvarianz durch das zugehörige Konstrukt erklärt wird [HRS09]. Aufgrund der Standardisierung der Daten kann dies anhand der quadrierten Faktorladungen überprüft werden. Für den Indikator mit der geringsten Faktorladung beträgt die Indikatorreliabilität 0,59, so dass dieses Kriterium erfüllt wird.

Für *Konvergenzvalidität* muss die durchschnittliche extrahierte Varianz (DEV) je Dimension mindestens 0,5 betragen [FL81]. Dieses Kriterium wird erfüllt, wie in Tabelle 4 ersichtlich ist.

	1	2	3	4
1. Individuelle Mitarbeiterzufriedenheit	**0,82**			
2. Prozesseffizienz	0,60	**0,89**		
3. IS-Projekterfolg	0,80	0,77	**0,85**	
4. Kundenzufriedenheit	0,50	0,54	0,74	**0,87**

Tabelle 4: Diskrimante Validität (Hinweis: Die fettgedruckten Werte auf der Diagonalen sind Wurzeln der DEV-Werte. Die übrigen Zahlen sind die Konstruktkorrelationen.)

Die *Diskriminanzvalidität* entsprechend des Fornell-Larcker-Kriteriums besagt, dass eine latente Variable die Varianz der eigenen Indikatoren besser erklären sollte als die Varianz der anderen latenten Variablen [FL81].

Daher muss die jeweilige Wurzel des DEV-Wertes eines Konstruktes größer sein als die Korrelation mit anderen Konstrukten. Dies wird durch die Crossloadings bestätigt. Die Ladungen aller Indikatoren sind für das zugehörige Konstrukt höher als für die übrigen latenten Konstrukte [Ch98].

4.2 Strukturmodell

Die erklärte Varianz (R^2) des Gesamterfolgs liegt bei 84,7 %. Dieser Wert entspricht einem substantiellen Ergebnis [Ch98]. Abb. 2 zeigt eine Übersicht über die wichtigsten Bestandteile des IS-Projekterfolgs. Angegeben sind jeweils das Pfadgewicht und die Stärke des Effekts [Co88]. Effektstärken, die größer als 0,35 sind weisen auf einen starken Einfluss hin.

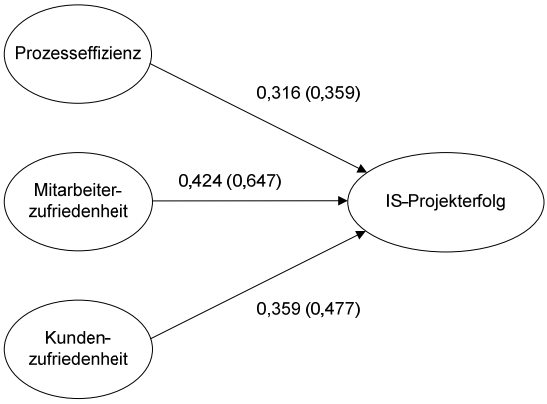

Abbildung 2: Modell des IS-Projekterfolgs

5 Diskussion

Die Auswertung zeigt zunächst, dass Prozesseffizienz, Mitarbeiterzufriedenheit und Kundenzufriedenheit den größten Einfluss auf den durch Projektbeteiligte auf Seiten von Auftragnehmern von IS-Projekten wahrgenommenen Gesamterfolg von IS-Projekten haben. Von der oben getätigten Annahme ausgehend, dass das Ausmaß des Einflusses auf die subjektive Wahrnehmung als Kriterium der Eignung für eine objektive Messung des IS-Projekterfolgs verwendet werden kann und soll, sind diese drei Dimensionen für ein Messmodell des IS-Projekterfolgs heranzuziehen. Dabei kann festegestellt werden, dass Prozesseffizienz und Mitarbeiterzufriedenheit mit dem Prozess assoziierte Dimensionen sind, Kundenzufriedenheit bezieht sich im Wesentlichen auf das erstellte Produkt, obgleich sicher auch der Projektprozess Einfluss auf die Kundenzufriedenheit haben kann.

Kundenzufriedenheit kann darüber hinaus im Wesentlichen als ein Maß für Effektivität hinsichtlich der Erreichung der Projektziele verstanden werden.

Es ist nicht schlüssig argumentierbar, dass ein Kunde zufrieden ist, wenn die Projektziele nicht in hohem Maß erreicht wurden. Die Rolle der Mitarbeiterzufriedenheit kann als Einflussfaktor auf die Prozessqualität interpretiert werden – zufriedene Mitarbeiter sind motiviert und arbeiten engagiert.

Somit ergibt sich ein Gesamtmodell zur Messung des Erfolgs von IS-Projekten (vgl. Abb. 3), das den konzeptuellen Überlegungen entspricht und durch empirische Ergebnisse ergänzt und gestützt wird. Sowohl der Projektprozess als auch das erstellte Produkt werden zur Bewertung des Gesamterfolgs herangezogen. Das Produkt wird mit der Kundenzufriedenheit als einem Maß der Effektivität bewertet. Die Effizienz wird zur Bewertung des Prozesses herangezogen. Neben Effizienz und Kundenzufriedenheit wurde die Mitarbeiterzufriedenheit empirisch als von großer Bedeutung für die Projekterfolgsbeurteilung bestimmt. Diese kann zwar dem Prozess zugeordnet werden aber nicht direkt als Maß der Effizienz.

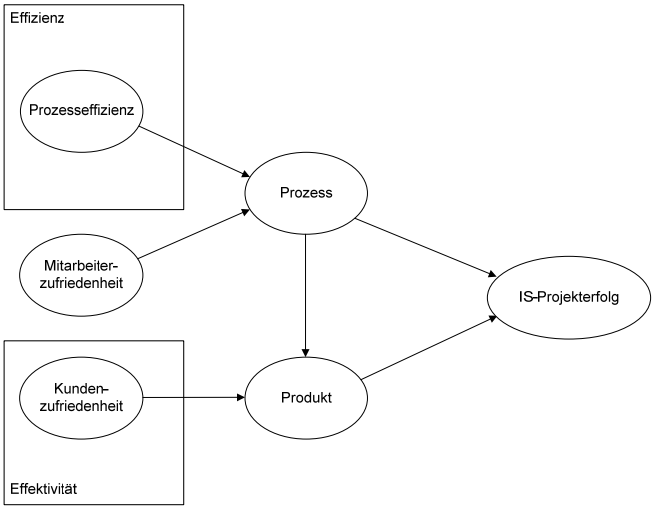

Abbildung 3: Modell des IS-Projekterfolgs

Für einen Projektleiter implizieren diese Ergebnisse, dass er in erster Linie auf eine effiziente Durchführung des Projektes sowie ein hohes Maß an Mitarbeiterzufriedenheit hinzuwirken hat. Anders als erwartet zeigt die Planungstreue nur einen sehr geringen Einfluss auf den wahrgenommenen Gesamterfolg. Dies ist in sofern erstaunlich, als Planeinhaltung das wesentliche Instrument zur Fortschrittskontrolle und Steuerung von Projekten darstellt. Häufig werden Planungstreue und Effizienz gleich gesetzt (vlg. bspw. die Studien [Al02a], [SLD97], [FB92]). Dahinter steht vermutlich die Annahme, dass Projekte so geplant werden, dass nur so viele Ressourcen bereitgestellt werden, wie zur sachgemäßen Durchführung des Projekts benötigt werden. In einem solchen Fall wäre die Einhaltung der Pläne als effizient zu betrachten. Dies kann allerdings ex ante nicht vorausgesetzt werden. Die Korrektheit der Planung ist aus vielen Gründen in Frage zu stellen, unter anderem, da die den Plänen zugrunde liegenden Schätzungen häufig ungenau sind [HK91], [Ke87], [Mo04] und politischen Einflussnahmen unterliegen [GRM08], [Le90].

Selbst wenn Planungstreue und Effizienz häufig miteinander einhergehen, können sie unter keinen Umständen gleichgesetzt werden. Der Unterschied zwischen diesen beiden Konzepten wird durch die vorliegende Studie bestätigt. Neben der geringen Bedeutung der Planungstreue fällt auf, dass auch die Erfüllung funktionaler und nicht-funktionaler Anforderungen nur eine geringe Bedeutung für die Erfolgswahrnehmung hat. Dies kann aber durch die hier eingenommene Perspektive der Auftragnehmer erklärt werden können.

Insgesamt erklären die drei als am bedeutsamsten identifizierten Dimensionen den individuell wahrgenommenen Projekterfolg der Projektbeteiligten auf Auftragnehmerseite zu einem sehr großen Teil ($R^2 = 84{,}7\ \%$). Ein konkretes Messmodell des IS-Projekterfolgs kann die drei identifizierten Dimensionen verwenden. Diese sind entsprechend ihrer Pfadgewichte zu gewichteten und mittels der verwendeten Items zu operationalisieren.

Mit Hilfe des gewählten Ansatzes ist es darüber hinaus möglich, Aussagen über die beiden zusätzlichen, im Folgenden betrachteten Fragen zu tätigen:

(3) Wie kann ein Projekt durch gezielte Maßnahmen so gesteuert werden, dass sein Erfolg maximal wird?

Die zur Messung des Erfolgs verwendeten Dimensionen sind gleichzeitig Einflussfaktoren des IS-Projekterfolgs. Als solche wurden sie mittels der Strukturgleichungsmodellierung empirisch bestimmt. Es kann somit argumentiert werden, dass Prozesseffizienz, Kundenzufriedenheit und Mitarbeiterzufriedenheit nicht nur als Messgrößen von Interesse sind, sondern genau jene Faktoren sind, die durch das Projektmanagement anzustreben sind. Konkret heißt das, dass etwa bei einer Abwägung, ob die Planeinhaltung auf Kosten der Mitarbeiterzufriedenheit (Überstunden, Leistungsdruck) unter allen Umständen durchzusetzen ist, möglicherweise eine Entscheidung zugunsten der Mitarbeiterzufriedenheit die richtige wäre. Grundsätzlich muss der Projektleiter über das entwickelte Produkt die Kundenzufriedenheit sicherstellen. Allerdings sind die drei Dimensionen Maßgrößen auf einem vergleichsweise hohen Aggregationsniveau. Es ist allgemein anerkannt und durch unsere Untersuchung bestätigt, dass Kundenzufriedenheit von größter Bedeutung für den Projekterfolg ist und dies entsprechend zu messen ist. Zur Steuerung kann eine solche Größe allerdings nicht geeignet sein. Stattdessen sind die Einflussfaktoren der Kundenzufriedenheit zu bestimmen und diese sind durch das Projektmanagement anzustreben.

Die Zufriedenheit der Projektmitarbeiter und des Kunden können überprüft werden, indem kontinuierlich Zufriedenheitsbefragungen durchgeführt werden. Im Falle des Kunden kann in Bezug auf das Produkt bspw. auf den Einsatz von Prototypen zurückgegriffen werden. Entsprechend der Ergebnisse können Maßnahmen zur Verbesserung eingesetzt werden. Das Verhältnis von erzieltem Ergebnis und eingesetzten Ressourcen erfasst die Effizienz. Die eingesetzten Ressourcen werden in der Regel im Projektmanagement erfasst. Die Bewertung des erzielten Ergebnisses während des laufenden Prozesses erscheint dagegen schwierig (vgl. bspw. das 90 %-Syndrom [Ab88]). Es muss untersucht werden, ob dies anhand des Umsetzungsgrades von funktionalen und nicht-funktionalen Anforderungen möglich ist. Ein weiterer möglicher Ansatz wäre die projektübergreifende Operationalisierung dieser Kennzahl, um somit die Vergleichbarkeit von Projekten sicherzustellen.

(4) Wie erfolgreich sind Projekte tatsächlich in Deutschland?

Zur Beantwortung dieser Frage kann die Erfolgsrate der untersuchten Projekte betrachtet werden. Diese unterscheidet sich von den Erfolgsraten, die bisher in Studien veröffentlicht wurden. Der Median des Gesamterfolgs für die Projekte beträgt 6. Die Verteilung in Tabelle 5 zeigt deutlich, dass der wahrgenommene Erfolg positiver ausfällt als es bspw. aufgrund der Chaos Summary 2009 [Th09] erwartet werden könnte. Demzufolge sind nur 32% aller Projekte erfolgreich.

„Das betrachtete Projekt war insgesamt ein sehr großer Erfolg".	**Anteil**
„Trifft voll und ganz zu."	34 %
„Trifft zu."	39 %
„Trifft eher zu."	8 %
Indifferent	5 %
„Trifft eher nicht zu."	3 %
„Trifft nicht zu."	8 %
„Trifft ganz und gar nicht zu."	3 %

Tabelle 5: Wahrgenommener Gesamterfolg aus Sicht der Befragten

6 Fazit

Projekte werden nicht als erfolgreich wahrgenommen, wenn hohe Planungstreue vorliegt, sondern wenn zufriedene Mitarbeiter einen effizienten Prozess durchführen und der Kunden am Ende mit dem Ergebnis des Projektes zufrieden ist. Die Einhaltung von evtl. unrealistisch vereinbarten Plänen sollte daher möglichst vermieden werden. Wichtig erscheint dagegen eine kontinuierliche Überprüfung der Zufriedenheit und des Verhältnis von eingesetzten Ressourcen und erzieltem Ergebnis. Entsprechend dieser Kriterien liegt die Erfolgsrate von IS-Projekten bei über 75 % (vgl. Tabelle 5).

Es muss geklärt werden, auf welche Weise erzielte Ergebnisse während des Entwicklungsprozesses gemessen werden können, um die Effizienz zu überprüfen. Für weitere Forschungsbemühungen muss vor allem auch die Auftraggeberperspektive eingehend betrachtet werden. Zudem wäre es sowohl auf Auftragnehmer- als auch Auftraggeberseite interessant, ob verschiedene Stakeholdergruppen unterschiedliche Wahrnehmungen haben. Auf Auftraggeberseite könnte dies auch Aufschlüsse darüber geben, welche Kriterien für die Zufriedenheit verantwortlich sind. Aufbauend auf einer größeren Anzahl von Datensätzen im Verlauf der Studie könnte zudem eine detaillierte Analyse der Beziehungen zwischen den einzelnen Dimensionen weitere Erkenntnisse darüber liefern, wie sich bspw. die Zufriedenheit der Mitarbeiter bildet.

Literaturverzeichnis

[Ab88] Abdel-Hamid, T.K.: Understanding the «90% syndrome» in software project management: a simulation-based case study. Journal of Systems and Software, 8(4):319–330, 1988.

[Al02a] Aladwani, A.M.: An Integrated Performance Model of Information Systems Projects. Journal of Management Information Systems, 19(1):185-210, 2002.

[Al02b] Aladwani, A.M.: An empirical examination of the role of social integration in system development projects. Information Systems Journal, 12(4):339-353, 2002.

[At99] Atkinson, R.: Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. International Journal of Project Management, 17(6):337-342, 1999.

[Ba99] Baccarini, D.: The Logical Framework Method for Defining Project Success. Project Management Journal, 30(4):25-32, 1999.

[BK92] Banker, R.D.; Kemerer, C.F.: Performance Evaluation Metrics for Information Systems Development: A Principal-Agent Model. Information Systems Research, 3(4):379-400, 1992.

[BH01] Barki, H.; Hartwick, J.: Interpersonal conflict and its management in information system development. MIS Quarterly, 25(2):195-228, 2001.

[Ch98] Chin, W.W.: The partial least squares approach to structural equation modeling. In (Marcoulides, G.A., Hrsg.): Modern methods for business research. Erlbaum, Mahwah, 1998; S. 295-358.

[CKS08] Chrissis, M.B.; Konrad, M.; Shrum, S.: CMMI. Guidelines for process integration and product improvement. Addison-Wesley, Upper Saddle River, 2008.

[CN99] Chin, W.W.; Newsted, P.R.: Structural Equation Modeling Analysis With Small Samples Using Partial Least Squares. In (Hoyle, R.H., Hrsg.): Statistical Strategies for Small Sample Research. Sage, Thousand Oaks, 1999; S. 307-341.

[Co88] Cohen, J.W.: Statistical power analysis for the behavioral sciences. Erlbaum, Hillsdale, 1988.

[Cu08] Curtis, A.M. et al.: An Overview and Tutorial of the Repertory Grid Technique in Information Systems Research. Communications of the AIS, 23(3):37-62, 2008.

[De96] Deephouse, C. et al.: Software processes and project performance. Journal of Management Information Systems, 12(3):187-205, 1996.

[FB92] Freeman, M.; Beale, P.: Measuring project success. Project Management Journal, 23(1):8-17, 1992.

[FL81] Fornell, C.; Larcker, D.F.: Evaluating Structural Equation Models with Unobservable Variables and Measurement Error. Journal of Marketing Research, 18(1):39-50, 1981.

[FM07] Furulund, K.M.; Moløkken-Østvold, K.: The role of effort and schedule in assessing software project success - An Empirical Study, Unpublished Paper, 2007.

[Fo87] Fornell, C.: A Second Generation of Multivariate Analysis: Classification of Methods and Implications for Marketing Research. In (Houston, M.J., Hrsg.): Review of Marketing. American Marketing Association, Chicago, 1987; S. 407-450.

[FW99] Fowler, A.; Walsh, M.: Conflicting perceptions of success in an information systems project. International Journal of Project Management, 17(1):1-10, 1999.

[GRM08] Glass, R.L.; Rost, J.; Matook, M.S.: Lying on software projects. IEEE Software, 25(6):90-95, 2008.

[HK91] Heemstra, F.J.; Kusters, R.J.: Function point analysis: evaluation of a software cost estimation model. European Journal of Information Systems, 1(4):229-237, 1991.

[HL92] Henderson, J.C.; Lee, S.: Managing I/S design teams: a control theories perspective. Management Science, 38(6):757-777, 1992.

[HRS09] Henseler, J.; Ringle, C.M.; Sinkovics, R.R.: The Use of Partial Least Squares Path Modeling in International Marketing. Advances in International Marketing, 20(IV):277–319, 2009.
[Ke06] Kerzner, H.: Project Management. A Systems Approach to Planning, Scheduling, and Controlling. Wiley, New York, 2006.
[Ke87] Kemerer, C.F.: An empirical validation of software cost estimation models. Communications of the ACM, 30(5):416-429, 1987.
[Le90] Lederer, A.L. et al.: Information system cost estimating: A management perspective. MIS Quarterly, 14(2):159-176, 1990.
[Li10] Liu, J.Y.-C. et al.: Relationships among interpersonal conflict, requirements uncertainty, and software project performance. International Journal of Project Management, doi:10.1016/j.ijproman.2010.04.007, 2010.
[Me04] Mellis, W.: Projektmanagement der SW-Entwicklung. Vieweg, Wiesbaden, 2004.
[Mo04] Moløkken-Østvold, K. et al.: A survey on software estimation in the norwegian industry. In (IEEE Computer Society, Hrsg.): Proceedings of the 10th International Symposium on Software Metrics, Chicago, 2004. IEEE Computer Society Press, New York, 2004; S. 208-219.
[Nu78] Nunnally, J.C.: Psychometric theory. McGraw-Hill, New York, 1978.
[Pi04] Pinto, J.K.: The Elements of Project Success. In (Cleland, D.I., Hrsg.): Field Guide To Project Management. Wiley, Hoboken, 2004; S. 14-27.
[PS88] Pinto, J.K.; Slevin, D.: Critical success factors across the project life cycle. Project Management Journal, 19(3):67-75, 1988.
[RG88] Reynolds, T.J.; Gutman, J.: Laddering Theory, Method, Analysis, and Interpretation. Journal of Advertising Research, 28(1):11-31, 1988.
[RR00] Ravichandran, T.; Rai, A.: Quality management in systems development: an organizational system perspective. MIS Quarterly, 24(3):381-415, 2000.
[Ru02] Rugg, G. et al.: Eliciting information about organizational culture via laddering. Information Systems Journal, 12(3):215-229, 2002.
[RWW05] Ringle, C.M.; Wende, S.; Will, S.: SmartPLS 2.0 (M3) Beta, Hamburg 2005, http://www.smartpls.de.
[Sa96] Saarinen, T.: An expanded instrument for evaluating information system success. Information & Management, 31(2):103-118, 1996.
[SLD97] Shenhar, A.J.; Levy, O.; Dvir, D.: Mapping the Dimensions of Project Success. Project Management Journal, 28(2):5-13, 1997.
[St86] Stuckenbruck, L.C.: Who determines project success? In (Project Management Institute, Hrsg.): Proceedings of the 18th Annual Seminar/Symposium. Project Management Institute, Upper Darby, 1986; S. 85-93.
[TF08] Thomas, G.; Fernández, W.: Success in IT projects: a matter of definition. International Journal of Project Management, 26(7):733-742, 2008.
[TH02] Tan, F.B.; Hunter, M.G.: The Repertory Grid Technique: A Method for the Study of Cognition in Information Systems. MIS Quarterly, 26(1):39-57, 2002.
[Th03] The Standish Group International (ed.): Chaos Chronicles Version 3.0, 2003.
[Th09] The Standish Group International (ed.): CHAOS Summary 2009. The 10 laws of CHAOS, 2009.
[Wa98] Wateridge, J.: How can IS/IT projects be measured for success? International Journal of Project Management, 16(1):59-63, 1998.
[WLJ74] Werts, C.E.; Linn, R.L.; Jöreskog, K.G.: Intraclass Reliability Estimates: Testing Structural Assumptions. Educational and Psychological Measurement, 34(1):25–33, 1974.

Anhang A – Definitionen der Dimensionen

Dimension	Definition
IS-Projekterfolg	IS-Projekterfolg soll den durch einen Projektbeteiligten subjektiv wahrgenommenen Gesamterfolg eines IS-Entwicklungsprojekts bezeichnen.
Prozesseffizienz	Effizienz ist allgemein das Verhältnis von Mitteleinsatz und Ausmaß der Zielerreichung bzw. Ausprägung einer Zielgröße in Bezug auf ein Vorhaben, beispielsweise ein Projekt oder aber auch einen Produktionsprozess. In Bezug auf ein IS-Entwicklungsprojekt bezeichnet Effizienz das Verhältnis von aufgewendeten Mitteln, d. h. Budget und darin insbesondere Personalaufwand, und dem Ausmaß der Erreichung der Projektziele.
Mitarbeiter-zufriedenheit	Die individuelle Mitarbeiterzufriedenheit bezeichnet die allgemeine und subjektiv wahrgenommene Zufriedenheit der Projektmitarbeiter mit ihrer Tätigkeit im Rahmen des Projekts.
Kundenzufriedenheit	Kundenzufriedenheit bezeichnet die allgemeine und subjektiv wahrgenommene Zufriedenheit des Auftraggebers, d. h. der Organisation des Kunden insgesamt oder eines relevanten Kundenvertreters, mit dem Projekt, d. h. mit dem Projektverlauf und dem Produkt als Projektergebnis.

Anhang B – Verwendete Items

Dimension	Code	Item
IS-Projekterfolg	1.1	Das betrachtete Projekt war insgesamt ein sehr großer Erfolg.
	1.2	In meiner Wahrnehmung waren alle Projektbeteiligten insgesamt mit dem Projektverlauf und dem Projektergebnis zufrieden.
	1.3	Sollte ich das gleiche Projekt erneut durchführen, würde ich alles genauso machen.
	1.4r	Das betrachtete Projekt wies eine Reihe gravierender Probleme auf, die nicht gelöst werden konnten.
Prozesseffizienz	8.1	Das Projekt wurde insgesamt sehr effizient durchgeführt.
	8.2r	In das Projekt wurde mehr Aufwand investiert, als nötig gewesen wäre.
	8.3	Mit den verwendeten Mitteln wäre kein besseres Projektergebnis zu erreichen gewesen.
	8.5r	Das erstellte Projektergebnis wäre auch mit geringerem Aufwand in gleichem Maße erreichbar gewesen.
Kundenzufriedenheit	11.1	Der Kunde war aus meiner Sicht sowohl mit dem Produkt als Projektergebnis als auch mit dem Projektverlauf insgesamt sehr zufrieden.
	11.2	Der Kunde hat sich in aller Regel positiv zu dem Gesamtprojekt geäußert.
	11.3r	Der Kunde hat häufig seine Unzufriedenheit mit dem Gesamtprojekt zum Ausdruck gebracht.
Individuelle Mitarbeiter-zufriedenheit	14.1	Die Projektmitarbeiter sind meiner Kenntnis nach mit Ihrer Arbeit im Verlauf des Projekts durchgehend sehr zufrieden gewesen.
	14.2r	Meine Arbeit im Rahmen des Projekts hat mich nicht zufrieden gestellt.
	14.5	Ich habe meine Arbeitssituation als positiv wahrgenommen.
	14.6r	Die Mitarbeiter haben sich im Projektverlauf häufig über ihre Arbeit beklagt.

(Invers spezifizierte Items sind mit r gekennzeichnet)

Campus-Management-Systeme
– Vom Projekt zum Produkt –

Markus Bick[1)], Thomas Grechenig[2)], Thorsten Spitta[3)]

[1)] ESCP Europe Wirtschaftshochschule Berlin,
Wirtschaftsinformatik

[2)] TU Wien, Softwaretechnik und Interaktive Informatik

[3)] Universität Bielefeld, Angewandte Informatik/Wirtschaftsinformatik

Markus.Bick@escpeurope.de
thomas.Grechenig@inso.tuwien.ac.at
thSpitta@wiwi.uni-bielefeld.de

Abstract: Immer mehr Hochschulen führen Campus-Management-Systeme ein, um der vorwiegend aufgrund des Bologna-Prozesses komplexer werdenden Prozesse gerecht werden zu können. Vor diesem Hintergrund befasst sich der vorliegende Beitrag mit zwei Fragen. Erstens: *Was ist überhaupt ein Campus-Management-System?* Gehören alle Anwendungssysteme einer Hochschule dazu? Zweitens: *Wie können aus den in vielen Hochschulen laufenden Pilotprojekten Standardsysteme werden?* Nach einer Bestandsaufnahme und Bewertung der sichtbaren Ansätze bzw. Projekte werden mögliche Wege zu Standardprodukten aufgezeigt. Dabei wird den Hochschulen eine Checkliste zur Prüfung bereits existierender Lösungen an die Hand gegeben, um deren Bewertung im Hinblick auf eine mögliche Auswahl zu erleichtern.

1 Das Problem

Die im Zuge des Bologna-Prozesses zunehmende Umstellung deutschsprachiger Studiengänge auf sog. konsekutive Abschlüsse macht auch für diejenigen Organisationen „betriebliche“ Informationssysteme notwendig, die bisher glaubten, darauf weitestgehend verzichten zu können: die Hochschulen. Für eine effektive Steuerung und Planung der Lehre und der damit verbundenen Prüfungen werden jetzt dringend Systeme gebraucht, die die relevanten Vorgangsdaten der korrespondierenden Prozesse innerhalb der Hochschulen einheitlich festhalten bzw. „buchen“ und dabei zu Führungsinformationen verdichten. Da mit der mehr oder minder amtlich erzwungenen Umstrukturierung auch eine Öffnung der Studiengänge und Fakultäten einher geht, ist es auch nicht mehr möglich, dass Fachbereiche weiter mit eigenen Insellösungen arbeiten.

Für Organisationen wie Unternehmen oder Behörden ist es seit langem selbstverständlich und gehört auch zum Lehrbuchwissen (z. B. [Me04]), dass man für eine erfolgreiche Unternehmensführung u. a. auch Systeme mit einer einheitlichen Datenbasis betreibt oder anstrebt. Auch für Hochschulen ist diese Erkenntnis nicht neu ([Kü97, S.422ff.], [SK95], [Sp97], [We96]), die Umsetzung wurde aber lange ignoriert. So kam es, dass erst in jüngster Zeit Hochschulen wie z. B. die Universität Hamburg [UHH10] oder die Technische Universität München [TUM08] millionenschwere Ausschreibungen starteten, um ein *Campus-Management-System* (CM-System) zu implementieren. Insbesondere die Prozesse in der Lehre sind ohne einheitliche Buchungssysteme vor allem für große Fakultäten nicht mehr zu beherrschen. Diese Prozesse sind in den Hochschulen von heute geprägt durch

- eine extreme Erhöhung der Anzahl zu buchender Vorgänge (studienbegleitendes Prüfen),
- den Zwang zur Verkürzung der Prozesslaufzeiten mit verbindlichen Endterminen, insb. beim Übergang vom Bachelor zum Master,
- den Wunsch der Hochschul- und Fakultätsleitungen nach hochwertiger, zeitnaher Information,
- die bessere Nutzung der durch die neuen Studienmodelle knapperen räumlichen Ressourcen (stark gestiegene Anwesenheitsquote),
- hohe Anforderungen an den Schutz sensibler, personenbezogener Daten (elektronische Prüfungsakten, Auskünfte und Bescheinigungen über Internet-Technologien).

Seit mehr als 30 Jahren bietet in Deutschland die HIS Hochschul-Informations-System GmbH – deren Gesellschafter alle Bundesländer und zu 25% der Bund sind – sogenannte „Standardsysteme“ im Hochschulbereich an. Die HIS-Software wird dabei nicht als releasefähige Standardsoftware betrieben und dies wohl auch in Zukunft nicht werden können. Eine gewisse amtlich verordnete Benutzung hat dazu geführt, dass sich in diesem wichtigen Bereich keine privatwirtschaftliche Konkurrenz entwickeln konnte. Neben softwaretechnischen Defiziten wird vor allem die institutionelle Unzuverlässigkeit von HIS bemängelt [Sp97], die seit Jahren mehr verspricht[1] als die vorhandenen Softwareentwicklungs- und Wartungskapazitäten jemals zu leisten im Stande sein können (s. bspw. [TUM10]).

Vor diesem Hintergrund sind die großen Anstrengungen zu bewerten, die fast alle Hochschulen im deutschsprachigen Raum unternehmen, um die neu modularisierten Studiengänge weitestgehend störungsfrei abzuwickeln. Dabei überwiegen Eigenentwicklungen, wie in Bielefeld [Br+09] oder Göttingen [Ra09].

Dies führt zu zwei Kernfragen, die dieser Beitrag beantworten will:

- Was ist ein Campus-Management-System?
- Wie könnten aus den Pilotprojekten in den Hochschulen Produkte entstehen?

[1] Das gut zu lesende Papier [De+09] gehört in die Liste solcher Versprechungen. Sie wurden bereits auf der zitierten GI-Jahrestagung 1997 in Aachen gemacht [Sp97].

Ziel ist es, diese Fragen in den beiden zentralen Abschnitten des vorliegenden Beitrags zu klären. Da die Frage der Wirtschaftlichkeit solcher Systeme unseren Rahmen und Fokus überschreiten würde, sei hierzu auf [Be09] oder [Sp+10] verwiesen.

2 Was ist ein Campus-Management-System?

Es ist allgemeiner Konsens und sogar in einem wohl weltweit erfolgreichen IT-Produkt deutschen Ursprungs manifestiert, was ein sog. „Betriebliches Informationssystem" bzw. „Enterprise Resource Planning (ERP)-System" ist (vgl. [Fi99] oder [Me04]). Die Bezeichnungen der „Module" etwa innerhalb des R/3- Systems der SAP AG folgen im Wesentlichen den üblichen Begriffen betriebswirtschaftlicher Funktionen [AR99, S. 36ff.]. Das Modell dieser Funktionsbezeichnungen ist der Industriebetrieb.[2]

Im Folgenden werden die generalisierbaren „betrieblichen" Funktionen einer Hochschule daraufhin untersucht, ob sie sich durch vorhandene Standardsoftware abdecken lassen oder ob sie hochschulspezifisch sind und damit auch spezifische Softwarekomponenten benötigen, die man *Campus-Management-System* nennen könnte.

2.1 Grundfunktionen und Prozesse in Hochschulen

Eine Hochschule bietet als Dienstleister einer Gesellschaft mit dem Prozess *Lehre* Qualifikationen von Absolventen und durch den Prozess *Forschung* Erkenntnis. Dies erscheint einfach, ist es aber aufgrund der hier vorgenommenen Abstraktion nur auf den ersten Blick. In der Mikrostruktur geben die verschiedenen Disziplinen dem jeweiligen Prozess ein vielfältiges Gesicht. Unterstützend gibt es einen weiteren Prozess, der ebenfalls stark disziplinabhängig ist und den wir dem üblichen Sprachgebrauch folgend *Bibliothek* nennen.

Nun ist es nicht Ziel dieses Beitrags, die genannten drei Grundfunktionen einer Hochschule in ihren Details zu diskutieren. Der Fokus muss vielmehr auf den Funktionen liegen, in denen sich standardisierbare Prozesse mit hohen Wiederholungsraten finden, die nach einer Effizienz- und Sicherheitserhöhung verlangen. Ein Beispiel für *dringenden* Unterstützungsbedarf wäre das schnelle und sichere Eintragen der Noten einer Veranstaltung mit 800 Teilnehmern.

Ein Beispiel für *nicht dringenden* Bedarf wäre die tägliche Ausleihe und Rückgabe von 5000 Büchern. Die Softwareunterstützung für Bibliotheken hat eine lange Tradition und gilt im Wesentlichen als zufrieden stellend.

[2] Siehe für die Systeme Navision und R/3 bspw. [SB08, S. 131].

Obwohl die Unterstützung der Forschung zunehmend auf Antragsverfahren und Drittmittel abzielt, die sich z. T. durch IT unterstützen lassen, werden wir diesen Prozess hier nur am Rande streifen. Es hat sich nichts Grundsätzliches geändert, zumal die Fallzahlen sehr viel geringer sind als in der Lehre. Zudem beruhen die Unterstützungsmöglichkeiten weitestgehend auf Office-Systemen oder sind derart fachspezifisch, dass sich eine Einbindung in ein zentrales System verbietet. Als Service eines Campus-Management-Systems wünschenswert sind sicherlich eine für die Hochschule einheitliche Projekt- und Publikationsdatenbank.

Als spezifischer Bedarf bleibt somit eine wirksame, für jede Hochschule einheitliche, nachhaltig zur Verfügung stehende IT-Unterstützung des Prozesses *Lehre* (Abschnitt 2.2). Dieser ist umso schwergewichtiger, als Bachelor (BA) und Master (MA) äußerst repetitiv und von hoher Benutzer-Komplexität gekennzeichnet sind: Alle Studierenden und Lehrenden sind Stakeholder. Damit handelt es sich bei Entwicklung und/oder Einführung eines Campus-Management-Systems um ein sehr ernst zu nehmendes IT-Projekt [Gr+10, S. 91].

Ergänzend kommen die Unterstützungsprozesse der *Verwaltung* hinzu, wie *Personalwesen*, *Rechnungswesen/Controlling* oder *Beschaffung*. Diese Prozesse lassen sich gut durch Standardsoftware abdecken.

Dies führt zu der in Abbildung 1 dargestellten Sicht zentral zu betreibender Anwendungssysteme einer Hochschule, in der das Campus-Management-System zunächst nur grob angedeutet ist (s. auch [Gr+10, S. 133]).

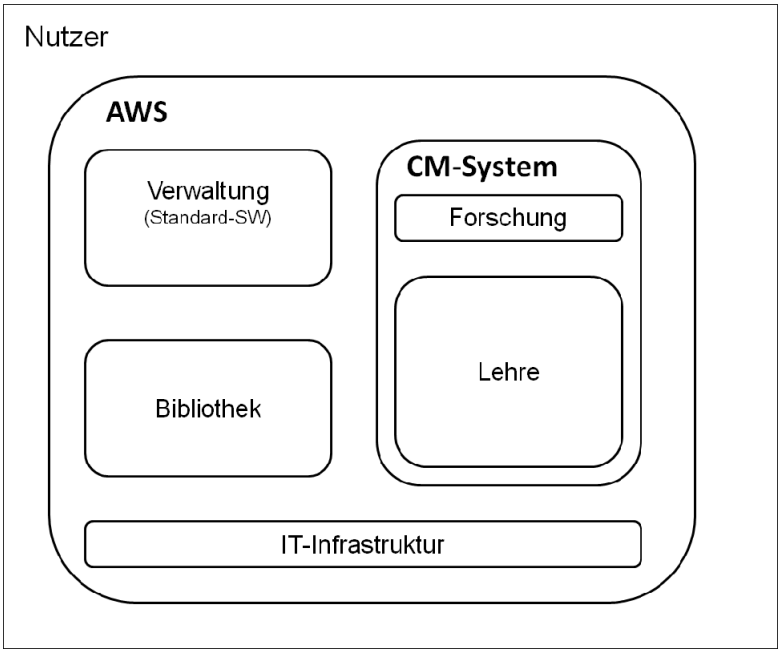

Abbildung 1: Zentrale Anwendungssysteme (AWS) einer Hochschule

In dieser Sicht zählen Anwendungssysteme, die den einzelnen Arbeitsplatz oder Arbeitsgruppen unterstützen, zur *IT-Infrastruktur*. Nicht explizit eingezeichnet ist die Funktionalität *E-Learning*. Sie wird erst bei einer genaueren Betrachtung des Prozesses Lehre sichtbar.

2.2 Der Prozess Lehre

Der Lehrprozess zerfällt in zwei grundsätzlich verschiedene Teilprozesse, die *Administration* und die *Durchführung*[3]. Die *Administration* ist gut standardisierbar und muss für alle Lehrveranstaltungen aller Disziplinen strukturell gleich sein. Dies gilt für alle Lehr- und Prüfungsformen, von der betreuten Hausarbeit bis zur Massenvorlesung. Die Prozessschritte sind, selbst bei sukzessiver Bewertung in kleinen Seminaren ohne explizite Prüfung:

Organisation → Durchführung → [Prüfung] → Bewertung.

Abbildung 2 zeigt daraus den Teilprozess einer expliziten Prüfung.

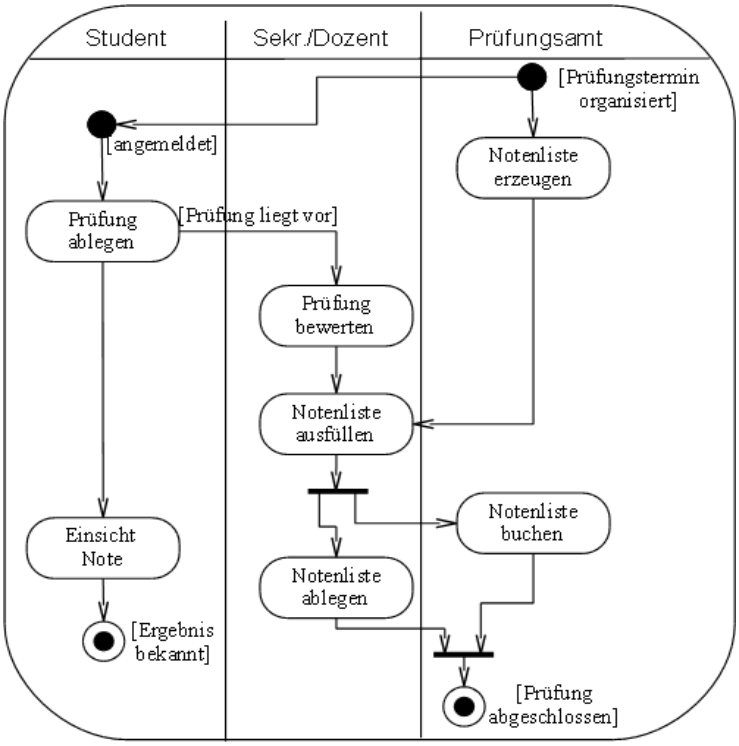

Abbildung 2: Der Teilprozess *Prüfung* aus dem Administrationsprozess der Lehre

[3] Die inhaltliche Vorbereitung durch den Dozenten behandeln wir nicht zuletzt aus Gründen der Selbstbeschränkung hier nicht.

In einer üblichen deutschsprachigen Universität läuft dieser Prozess zum Semesterende rund zweitausend Mal ab, was zu rund 30 Mio. Buchungen führt.[4] Solche Mengengerüste und die damit verbundene Datensicherheit und -konsistenz kann nur noch durch ein entsprechendes Anwendungssystem geleistet werden, das auf einer zentralen, transaktionssicheren und gut skalierenden Datenbank aufsetzt.

Der Teilprozess *Durchführung* kennt nur noch die Akteure Dozent(en) und Studierende. Er ist wenig standardisierbar und hängt von der Anzahl der Studierenden, dem Aufwand des Lehrenden und auch von der Fachdisziplin und dem Lehrinhalt ab. In diesem Teilprozess kann E-Learning-Software hilfreich sein. Wenn sie eingesetzt werden soll, ist zu fragen, wo es Überschneidungen mit administrativen Teilen des Lehrprozesses gibt. Wir kommen auf sie zurück, wenn die strukturbildenden Bestandteile administrativer Software im folgenden Abschnitt an Hand des Beispiels *Person* besprochen sind.

2.3 Teilsysteme administrativer Software und ihre Schnittstellen

Die Funktionen administrativer Anwendungssysteme werden durch eine einheitliche Datenbasis integriert [AR99, S. 5], [Gr+10, S. 161]. Die Struktur der Datenbasis bestimmt maßgeblich die Aufteilung von Systemen in Teilsysteme, die über möglichst wenige Schnittstellen kommunizieren sollen. Dies ist eine alte Erkenntnis, die schon in einer 1988 abgeschlossenen Habilitationsschrift als Kernaussage enthalten war [Sp89, Kap. 6].

Was heißt „wenige Schnittstellen“? Man will ein Campus-Management-System als selbständigen Kern definieren, der nicht mit unnötiger Kommunikation oder Spezialfunktionen belastet ist. Das Konstruktionsprinzip hierfür ist, dass originäre Daten immer nur in genau einem Teilsystem erzeugt bzw. geändert werden dürfen: Nur *ein* System kann der Owner eines Datentyps sein. Lange bevor dieses Prinzip unter dem Namen *Objektorientierung* bekannt wurde, war es bereits intuitiv in den 80er Jahren im System R/2 der SAP berücksichtigt worden.

Damit dies für ein Campus-Management-System konkreter wird, zeigt Abbildung 3 das grundlegende Datenmodell eines solchen Systems. Die vielfältig möglichen abgeleiteten Daten sind nur beispielhaft dargestellt. Auch die Komplexität der hier nur grob skizzierten Klassen, insb. die komplexe Klasse *Studiengang*, kann hier nur angedeutet werden.

Am Datenmodell der Abbildung 3 kann man auch die Schnittstellenproblematik der Anwendungssysteme aus Abbildung 1 zeigen, die wir im Folgenden exemplarisch am Beispiel der Personaldaten diskutieren.

[4] Rechenbeispiel: (250 Dozenten * 4 + 1000 wissenschaftliche Mitarbeiter *1) * 15000 Studierende.

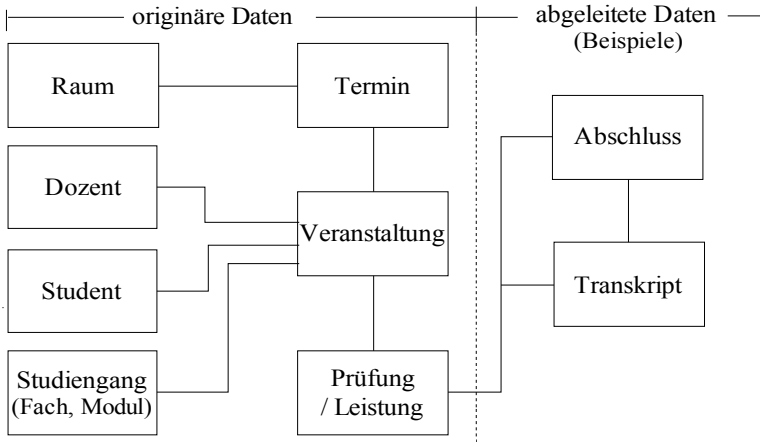

Abbildung 3: Grobes Datenmodell eines Campus-Management-Systems

In Lehrbüchern findet man oft eine Klasse *Person*, spezialisiert in die erbenden Klassen *Dozent* und *Student*. Damit hätte man bereits den ersten schweren Entwurfsfehler gemacht, der zwei verschiedene Systeme unnötig verkompliziert. Owner von *Dozent* darf nämlich *nicht* das Campus-Management-System sein, dies ist hier beispielsweise SAP/HR (Human Resources). Das Campus-Management-System braucht eine Importschnittstelle aus HR für diejenigen Attribute, die es für seine Klasse *Dozent* benötigt. Ohne eine solche Klasse wäre kein Campus-Management-System arbeitsfähig. Das HR-Modul wiederum würde geradezu „missbraucht" und müsste auch unnötigen Schnittstellenverkehr[5] verkraften, würde es mit Studierenden-Daten belastet.

Auch alle anderen Teilsysteme (Bibliothek, Forschung, IT-Infrastruktur, E-Learning) dürfen keine Funktionen enthalten, die Personaldaten verändern. Dem gegenüber macht es sehr viel Sinn, die Klasse *Student* als wichtigsten Stakeholder des Systems ausschließlich im Campus-Management-System anzusiedeln. Wenn nicht erkannt wird, dass *Bewerber* und *Alumni* nur Rollen eines komplexen Datentyps *Student* sind, erzeugt das weitere Entwurfsfehler, die unnötige Schnittstellen provozieren. Dass eine funktionsorientierte Modularisierung (isolierte Systeme für Bewerber, Aktive und Alumnis) sehr negativ ist, wissen wir schon seit dem Bahn brechenden Artikel von Parnas [Pa72].

Aus dieser beispielhaften Diskussion über die Minimierung von Schnittstellen lässt sich eine grundlegende technische Anforderung an jedes Campus-Management-System ableiten: Ein CM-System muss offenen Schnittstellen-Standards genügen. Dies gilt als „fundamentale Best Practice" [Gr+10, S. 216]; Bick/Börgmann bezeichnen *Platform Independence* sogar als 'Knockout Criterion' [BB09, S. 111] für die Entwicklung und/oder Auswahl derartiger Systeme. Die große Bedeutung dieses Prinzips zeigt im folgenden Abschnitt ein Blick auf die Funktionen eines Campus-Management-Systems.

[5] So war beispielsweise die Schnittstelle zwischen den HIS-Systemen SOS (Studentendaten) und POS (Prüfungsverwaltung) in der Fakultät Wirtschaftswissenschaften der Universität Bielefeld lange problembehaftet. Wegen ihrer hohen Störungsanfälligkeit wurde sie möglichst nur zweimal im Semester benutzt.

2.4 Funktionen eines Campus-Management-Systems

Ein Campus-Management-System muss die Ressourcen seiner Stakeholder für den kombinatorisch hoch komplexen Prozess Lehre effizient und problemorientiert verwalten. Problemorientierung heißt vor allem, dass Pläne praktisch ausführbar sein müssen, insb. Studienpläne. *Studierbarkeit* wirksam zu unterstützen, muss eines der wichtigsten Ziele eines Campus-Management-System sein [St+07, S. 20]. Die Ressourcen, die hierfür miteinander in Einklang gebracht werden müssen, sind *Zeit*, *Raum* und *Menschen*. Dies führt neben der Grunddatenverwaltung von Studierenden, Raum und Studiengang (s. o.) auf die Grundfunktionen *Raumplanung*, (individuelle) *Stundenpläne* und *Prüfungsverwaltung*. Diese Grundfunktionen bedingen natürlich eine einheitliche Darstellung des Lehrangebots. Alle diese Funktionen basieren auf Vorgangsdaten[6], die von Benutzern in Prozessen gebucht werden. Aus den originären Grund- und Vorgangsdaten werden für eine Vielzahl von Stakeholdern abgeleitete Daten erzeugt, umgangssprachlich oft „Informationen“ genannt. Jetzt kommen die Führungskräfte aus Fachbereichen und Universitätsleitung hinzu, meist unterstützt durch eine Controllingfunktion. Grechenig et al. zeigen die sehr umfangreiche Tabelle der Stakeholder eines Campus-Management-Systems [Gr+10, S. 196f.], Bick/Börgmann eine detaillierte Liste der 'Areas of Responsibility' , z. B. Grunddatenverwaltung, Studentendaten-Pflege und Veranstaltungsverwaltung [BB09, S. 108].

In den Funktionen können sich Überschneidungen mit E-Learning-Systemen ergeben, bspw. bei der Bereitstellung von Lerninhalten oder der Betreuung von Gruppen im Rahmen großer Veranstaltungen. Hier sollten zwei Zurechnungskriterien für das Campus-Management-System oder für ein entsprechendes Spezialsystem gelten.

1. *Generalisierbarkeit* dürfte bei der Verteilung von Lernmaterialien gegeben sein, denn sie ist für fast alle Veranstaltungstypen aller Disziplinen erforderlich. Dies ergibt sich schon aus der notwendigen Nachbarschaft zu den inhaltlich/organisatorischen Veranstaltungsbeschreibungen.

2. *Bezug zu den Grunddaten*: Eine Gruppeneinteilung für Tutorien bzw. Aufgabengruppen oder eine Zuteilung zu kapazitätsbeschränkten Lehrveranstaltungen umfasst die Studierenden einer Veranstaltung und ggf. Tutoren als Lehrende und damit wesentliche Grunddaten. Ergänzungssysteme mit starkem Bezug zu Grunddaten schaffen als isolierte Systeme mehr Probleme als sie lösen.

Dagegen muss es möglich sein, funktionale Bausteine z. B. aus vorhandenen Methodenbibliotheken zu nutzen, etwa für Überschneidungsberechnungen zwischen Lehrangebot und vorläufigen Stundenplänen. Es macht wenig Sinn, ein zentrales System mit der Re-Implementierung spezieller Algorithmen zu überfrachten, wenn es dafür Standardprogramme gibt. Für die Nutzung solcher Zusatzfunktionen benötigt man unbedingt offene Schnittstellen.

[6] Auch der Begriff *Bewegungsdaten* ist üblich; Grunddaten heißen in einem solchen Kontext *Stammdaten*.

Allein wegen der großen Zahl der wichtigsten Stakeholder eines CM-Systems, der Studierenden, gibt es eine Basisfunktion, die in kommerziellen Systemen keine ganz so große Bedeutung hat wie in einem Campus-Management-System: *Kommunikation*. Warum? Der „Kunde“ ist der Teilnehmer einer Veranstaltung. Er befindet sich *im* System und nicht wie sonst in der Umwelt eines Systems. Jeder Dozent muss die Möglichkeit haben, mit genau den Studierenden zu kommunizieren, die für seine Veranstaltung eingetragen sind. E-Mail ist hierfür ein bewährtes Standardsystem, bei dem allerdings hohe Sicherheitsanforderungen etwa an Verteiler gestellt werden müssen. Dies wird in der Bielefelder Entwicklung seit Jahren mit Erfolg genutzt [St+07, S 24].

Es versteht sich, dass alle Funktionen über intuitiv zu erlernende, offene Benutzerschnittstellen erreichbar sind, Buchungen und Auskünfte mit personenbezogenen oder geheimhaltungsbedürftigen Daten aber in geschütztem Umfeld stattfinden.

2.5 Zwischenfazit

Ein Campus-Management-System muss – analog zu betrieblichen Informationssystemen – auf einer zentralen Datenbasis aufsetzen, denn es ist ein primär buchendes System mit leistungsfähigen Auskunft- und Auswertungsfunktionen. Seine Teilsysteme (sog. „Module“) sollten sich um die Grunddaten gruppieren, von denen Dozent, Studierende, Raum und Studiengang die wesentlichen sind. Einer der wichtigsten Vorgangsdatentypen ist *Veranstaltung*. Alle buchenden und auswertenden Operationen auf diesen Datentypen müssen Teil des Systems sein. Eine Ausnahme bilden die Grunddaten entlohnter Personen (Datentyp *Dozent*), die außerhalb des Systems gepflegt werden müssen, denn es macht keinen Sinn, allgemeine Personalfunktionen in ein Campus-Management-System zu implementieren. Hier benötigt das System aus Effizienzgründen die relevanten Daten mit nur *lesenden* Operationen.

Mit den Datentypen eng verbundene Funktionen sollten Teil des Systems sein, Berechnungsfunktionen ohne Gedächtnis sollten aus Standardbibliotheken bezogen werden können. Daher sind ergänzende Funktionen, die selbst keine redundanten Grunddaten pflegen, gerade auch aus dem Repertoire des E-Learnings zu nutzen.

Offene Schnittstellen sind für ein Campus-Management-System ein absolutes Muss, um evolutionäre Erweiterungen und Ergänzungen zu ermöglichen, die im deutschsprachigen Bildungswesen sehr wahrscheinlich sind.

3 Von Pilotprojekten zu Standardprodukten

Nach der vorangegangenen Bestandsaufnahme werden im Folgenden Anforderungen an Standardprodukte formuliert, um danach der Frage nachzugehen, welche Bedingungen erfüllt sein müssen, damit ein nachhaltig wartbarer Standard entstehen kann.

3.1 Landschaft der Standardprodukte

Bick/Börgmann haben aktuell eine europaweite Übersicht von 22 Campus-Management-Systemen erstellt [BB09, S. 109]. Von ihnen sind für den deutschen Sprachraum vor allem die in Tabelle 1 gezeigten fünf Systeme[7] interessant:

Nr	Name	Pilotanwendung	Anmerkung	Hersteller in
1	Campus-Management SAP	FU Berlin	offenbar von SAP keine Fortsetzung	Walldorf
2	Campus Net (Datenlotsen)	Uni Hamburg	Echtbetrieb mit Störungen[8]	Hamburg
3	CampusOnline	TU München	Noch kein Echtbetrieb	Graz / Öst.
4	CAS Campus	RWTH Aachen	Standard (?)	Karlsruhe
5	HISinOne	s. www.hisinone.de	bisher nur Prototyp	Hannover

Tabelle 1: Angebot Campus-Management-Systeme im deutschsprachigen Raum

Von diesen fünf Systemen scheint das erste von SAP nicht fortgeführt zu werden. Zudem befinden sich das zweite und dritte CA-System nicht in dem Zustand, dass sie sich als Referenz eignen. Auch bestehen für das fünfte System vielleicht nicht mehr als Absichtserklärungen. Die Homepage von HISinOne.de enthält eine Landkarte von zehn Universitäten, darunter die RWTH Aachen, die auch Pilotanwender von CAS Campus war (Tabelle 1). Was korrekt ist, lässt sich leicht überprüfen. Obwohl die Idee einer breiten Nutzerbeteiligung für ein solches Vorhaben gut ist, sind Zweifel angebracht, ob die HIS GmbH ein so komplexes Projekt wird managen können. Da HISinOne eine zentrale Datenbasis nicht vermeiden kann, ist es verwunderlich, dass technisch belastbare Angaben zum System auf der Homepage fehlen.

Die CAS Software AG bietet im Rahmen ihrer Internetpräsenz als einziger Hersteller Hinweise auf „lebende“ Installationen, die im Internet einsehbar sind: RWTH Aachen, FH Aachen, TU Kaiserslautern, Universität Bochum. Von Gesprächspartnern, die andere Lösungen bevorzugen, wird angezweifelt, dass der Pilotanwender Aachen eine Standardlösung betreibt. Das lässt sich verifizieren oder widerlegen, indem man die Releasewechsel hinterfragt. Eine „Vorabschau“ im Internet kann natürlich keine seriöse Produkt- und Lieferanten-Überprüfung ersetzen. Mehr hierzu in Abschnitt 3.2.

[7] Drei weitere „deutsche Systeme“ aus der hier zitierten Übersicht sind nicht wirklich umfassende Systeme oder haben keine Referenzen und werden daher im Weiteren nicht berücksichtigt.
[8] s. http://www.info.stine.uni-hamburg.de/news.htm (5.8.2010)

In Hamburg (Universität und Hochschule für Angewandte Wissenschaften) wurde eine telefonische Umfrage eines der Autoren dieses Papiers mit insgesamt zehn Interviews von rund 30 Minuten durchgeführt. Die Mehrzahl der Gesprächspartner war unzufrieden. Das Projekt in der Universität Hamburg wurde 2006 begonnen und sollte nach dem Projektplan 2007 beendet sein. Bis in das Jahr 2009 hinein war der Betrieb von massiven Performance-Problemen begleitet, da offenbar weder das Rechenzentrum der Universität noch der Hersteller Erfahrungen mit einer großen Datenbankinstallation hatte. Die Kosten für die Universität müssen hoch sein[9], werden aber strikt geheim gehalten.

Die Helmut-Schmidt-Universität in Hamburg (Universität der Bundeswehr) setzt Datenlotsen *nicht* ein, sondern HIS-Komponenten. Ihre Entscheidung von 2006 sei hier zitiert:

> „Die funktionalen und strukturellen Vorteile vor allem der Lösung „CampusNet" der Datenlotsen vermögen das finanzielle und planerische Risiko nicht aufzuwiegen." [HSU06, S. 6].

Wenn hier Negatives über einzelne Projekte geschrieben wird, dient das *nicht* dazu, die Produkte schlecht zu machen. Wir haben es hier mit Pilotanwendungen zu tun, in denen Störungen in Projekten dieser Größenordnung durchaus „normal" sind, d. h. sie lassen sich nicht vermeiden. Das war bei den ersten SAP-Installationen vor 25 Jahren nicht anders.

Was jedoch bedenklich stimmt, ist eine aggressive Vermarktungspolitik von Herstellern, während die Pilotinstallation noch nicht beendet ist. Dies beteiligt entweder den Kunden über Gebühr am Entwicklungsrisiko oder es überlastet den Anbieter und bedroht damit seine wirtschaftliche Existenz, auf die jeder Kunde bei nicht selbst entwickelter Software dringend angewiesen ist.

Die Produktlandschaft der *Campus-Management-Systeme* befindet sich in einem weit weniger ausgereiften Zustand als dies bei üblichen betrieblichen Anwendungssystemen der Fall ist. Welchen Anforderungen müssen Softwaresysteme genügen, die mit dem Ziel entwickelt wurden, daraus mehrfach einsetzbare Produkte zu machen?

3.2 Anforderungen an Standardprodukte

Ein Standard-Softwareprodukt muss neben der fachlich umfassenden Abdeckung einer Domäne vor allem zwei Eigenschaften haben, die es von qualitativ hochwertigen Einzelprodukten abhebt: Flexibilität und Nachhaltigkeit. Die beiden zuletzt genannten Eigenschaften sind eine Folge des Anspruchs, ein Problem allgemein gültig zu lösen. Die Abdeckung einer Domäne stellt sich nicht mit dem ersten Pilotprojekt ein, es können sogar Teillösungen entstehen, die sich als Irrweg herausstellen.

Unter Management-Gesichtspunkten lesen sich die Anforderungen so:

> „Damit das IT-Produkt in unterschiedlichen Einsatzfeldern bestehen kann, muss [es] ausgereift sein, d. h. es muss praktisch und technisch funktionieren." [HP09, S. 102].

[9] Ein Zitat: „Der Vertrag ist für die Uni Hamburg eine Katastrophe."

Konsequenterweise betrachten wir hier das Attribut *ausgereift* sowie die *Funktionsfähigkeit* aus einer technischen Sicht. Die technischen Eigenschaften bestimmen maßgeblich die Investitionssicherheit, die der Hersteller festlegt und die den Kunden für die Einsatzdauer eines Produkts binden. Bei Softwareprodukten sind durch die starke Verzahnung mit der Organisation die Abhängigkeiten vom Produkt besonders groß.

Was bedeutet der Begriff *Nachhaltigkeit* in unserem Kontext? Standard-Softwareprodukte haben Lebenszyklen von 30 bis 40 Jahren. Ob 40 Jahre das Maximum ist, wissen wir mangels Historie noch gar nicht. In solchen Zeiträumen spielen sich dramatische Technologiesprünge ab, denen das Produkt folgen können muss. Es gibt viele Beispiele, bei denen das nicht gelungen ist, z. B. der Untergang der Fa. Nixdorf mit ihrem bis in die 90er Jahre sehr erfolgreichen Anwendungssystem COMET. Ein positives Beispiel für ein bekanntes Produkt, inzwischen über 30 Jahre alt:

> Das System R (Realtime) der späteren SAP AG war als R/2 für den Großrechnerbereich in einem Makro-Assembler programmiert, der auf IBM- und SIEMENS-Großrechnern ablauffähig war. Die Portabilität beruhte auf einer weitsichtigen Software-Architektur, damals vom Typ 2-Tier (Schicht). Die ersten Installationen gab es ungefähr 1980. 1986 – 1991 wurde die völlig neue 3 Tier-Architektur (R/3) nicht nur neu entworfen, sondern in der 4GL-Sprache ABAP IV neu implementiert. Heute entwickelt SAP in der Sprache Java.

Das Produkt R, heute ERP, hat auf Grund seiner Architektur und vieler anderer qualitätssichernder Maßnahmen die Flexibilität gehabt, der Technologie zu folgen. Nachhaltig war nicht der Code, sondern das gesamte Entwicklungsumfeld mit Quellcodeverwaltung, Versionsverwaltung, kontrollierten Konventionen, Frameworks, automatischen Tests und nicht zuletzt Offenheit gegenüber dem Kunden. Die Flexibilität gefördert haben sicher auch die im vorigen Abschnitt genannten offenen Schnittstellen.

Eine weitere wichtige Dimension der Flexibilität eines Campus-Management-Systems ist die Unabhängigkeit der Anwendungssoftware vom Betriebssystem [BB09, S. 111], einer der Erfolgsfaktoren schon von R/2. Dies ist besonders wichtig mit Bezug auf Client-Betriebssysteme und geht über das Betriebssystem hinaus. Ein Rich Client, die übliche Architektur der frühen 2000er Jahre [Gr+10, S. 205], darf in einer 3 Tier-Architektur nicht mehr vorkommen.

Damit nicht nur das gerne benutzte Erfolgsprodukt der Fa. SAP als alleiniges positives Beispiel stehen bleibt, sei kurz ein zentral betriebenes Einzelprodukt aus etwa der gleichen Zeit genannt, das sich am Markt als Standardsystem durchgesetzt hat, allerdings mit nur *einem* sehr mächtigen „Server“, einem Großrechenzentrum in Erding bei München.

> Die Entwicklungs- und Firmenhistorie des START-Systems für Reisebuchungen ist anschaulich in [Wiki10] geschildert, seitens der Entwicklung allerdings auf Endnutzer-Hardware beschränkt. Dies geht am hier relevanten, wesentlichen Erfolgsfaktor des START-Systems vorbei. Das System konnte sich von 1978 bis heute nur wegen seiner Software-Architektur evolutionär entwickeln und am Markt durchsetzen.

Denert hatte als Projektleiter diese Entwicklung stark befördert, indem er mit seinem Team erstmalig eine explizit konstruierte Architektur entwarf, die sich bis dahin bei Software eher implizit und beiläufig beim Programmieren ergeben hatte (vgl. [De79]).

Auch die START-Software musste wie das R-System wegen mangelhafter Standardisierung in einer portablen Programmiersprache neu implementiert werden. Dem Erfolg des Produkts hat die konstruktive Pionierarbeit offenbar gut getan.

Von dem umfassenden Kriterienkatalog aus [BB09, S. 113], den Produkte erfüllen müssen, seien hier nur noch diejenigen zusätzlich genannt, die für Standardsysteme ein besonderes Gewicht haben: 'Data Privacy and Reliability', 'Integration with existing Systems'.

Wir fassen zusammen: Software, die mit dem Ziel *Standardprodukt* entwickelt wird, muss in besonderem Maße hohe technische Maßstäbe erfüllen, wie sie heute für die Entwicklung industrieller Qualitätssoftware üblich sind. Dies gilt bereits für die Anforderungen, ganz besonders für den Entwurf, die Implementierung und die den gesamten Prozess begleitende Qualitätssicherung, insb. den Test [Gr+10]. Die Entwicklung von Qualitätssoftware beginnt bei den Anforderungen, die exakt und für den Kunden nachvollziehbar dokumentiert werden müssen, weil sonst das Projekt niemals durch einen Abnahmetest beendet werden kann. Die systematische Qualitätssicherung muss für den Kunden nachvollziehbar sein, d. h. einsehbar. Zertifikate alleine genügen nicht.

Es werden also Mindestqualitäts-Anforderungen sowohl an die erste Version des Produkts als auch an die Leistungsfähigkeit des Herstellers gestellt.

3.3 Wie könnte ein Standard entstehen?

Die Analyse der letzten beiden Abschnitte beschreibt die folgende Situation:

- Die Nachfrage nach einem Standardsystem seitens der Hochschulen ist hoch.
- Es gibt nur die Vorstufe eines Angebots in Form einiger Pilotsysteme, über deren Qualität wenig bekannt ist.

Standardsysteme, die den Anforderungen der vorherigen Ausführungen genügen, müssen erst entstehen. Hierzu müssen Alternativen sowohl bei den Herstellern – die institutionelle Seite – als auch seitens des Produkts – die technische Seite – genauer betrachtet werden.

3.2.1 Institutionelle Alternativen

Wir zählen zunächst die Alternativen auf und bewerten sie dann.

1. *Garage* (Bsp.: Hewlett Packard, Microsoft)
2. *Platzhirsch Typ A* investiert (Bsp.: SAP AG)
3. *Platzhirsch Typ B* ändert sich (Bsp.: HIS GmbH)
4. *Pionier Typ A* mit Venture Capital (Bsp.: Datenlotsen, ca. 80 Mitarbeiter, Einprodukt-Unternehmen)
5. *Pionier Typ B* mit organischem Wachstum (Bsp.: CAS Software AG, ca. 200 Mitarbeiter; Mehrprodukt-Unternehmen)
6. *Kooperation* von Hochschulen (Bsp.: Land Mecklenburg-Vorpommern [MV04]).

Es ist keineswegs so, dass die zuvor genannten Alternativen einander ausschließen, im Gegenteil: Innovation lebt von Pioniergeist *und* Konkurrenz. Sicher ist nur, dass jeder potentielle Kunde – dies ist die oben festgestellte hohe Nachfrage – unbedingt benötigt:

- Präzise, selbst erstellte *Anforderungen*
- geeignetes *Personal*, falls das System später selbst betrieben werden soll
- eine exakte, auf den Anforderungen basierende *Ausschreibung*
- *fachjuristische Unterstützung* für Softwareverträge.

Der Typ *Garage* scheidet heutzutage ziemlich sicher aus, allein deshalb, weil sich wegen der hohen Nutzer-Komplexität keine realistisch große Pilotinstallation betreuen ließe. Pioniere wie die SAP-Gründer waren immerhin vier Personen und hatten mehr Entwicklungszeit, als heute zur Verfügung steht.

Vom realen *Platzhirsch Typ A* (SAP AG) wissen wir im Moment nicht, warum eine Produktlinie *Campus-Management-System* zur Zeit nicht weiter bearbeitet wird. Es könnte die unbefriedigende Ertragssituation in einem zu schwierigen Markt sein. An der Fähigkeit, nachhaltige Software zu entwickeln, mangelt es ganz sicher nicht.

Über den *Platzhirsch Typ B* (HIS GmbH) wurde in diesem Papier schon Verschiedenes gesagt. Es handelt sich um Bewertungen, die sicher nicht jeder teilt und teilen muss. Es gibt Hochschulen, die um die mangelnde Integration der Einzelpakete der Lösung wissen und dies bei der Bewertung der Alternativen in Kauf nehmen [HSS06, MV04]. Schließlich liefert die HIS GmbH seit Jahren benutzbare Software für viele Installationen, wenn auch das Anspruchsniveau bezüglich einer konsistenten Datenbasis reduziert werden muss. Die Autoren machen auch keinen Hehl daraus, dass sie bei den Herstellern Lösungen bevorzugen, die sich privatwirtschaftlich auf Dauer selbst tragen.

Die beiden verbliebenen privatwirtschaftlichen Hersteller gehören zur Klasse der *Pioniere*. Sie werden von den Autoren als Unternehmenstyp für die Herstellung eines Campus-Management-Systems bevorzugt, da hier eher echte Innovationen zu erwarten sind, wie sie zukünftige Produkte benötigen. Pioniere können nur Erfolg haben, wenn sie risikobereite Kunden finden. Dieses Risiko war das Land/die Universität Hamburg im Fall der Datenlotsen offenbar bereit einzugehen. Hinter dem Pilotprojekt stand als Nutzen die Verschmelzung zweier Hochschulen (Harburg und Hamburg). Allerdings muss sich gerade ein Pionier – anders als das in Hamburg wohl der Fall war – einer besonders kritischen Prüfung seiner Verfahren, seines Personals und vor allem der technisch-fachlichen Struktur des Pilotprodukts stellen.

Denkbar ist auch das Modell *Kooperation*. Es ist möglich, dass eine Hochschule mit einer gut funktionierenden Eigenentwicklung eine andere als kooperierenden Kunden aufnimmt. Die Variationsbreite der Art der Zusammenarbeit ist groß und könnte z. B. in die Gründung einer gemeinsamen Servicegesellschaft münden. Wenn man die Evolution der Software auch mit Personal der Partner-Hochschule vorantreibt, wäre ein Schritt zu einer Standardlösung getan, dem man weitere folgen lassen könnte. Diese Richtung deutet die Gründung kooperativer Rechenzentren an, eine mögliche Entwicklung zur wirtschaftlichen Verbesserung für Hochschulen insgesamt.

Für jeden Hersteller-Typ muss dessen Leistungs- und Überlebensfähigkeit dezidiert geprüft werden, aber auch seine Offenheit. Gerade in der Pilotphase – wir sehen Ihr Ende frühestens nach der fünften vollständig abgeschlossenen Installation – ist eine auf Misstrauen gegründete Zusammenarbeit zwischen Hersteller und Kunde desaströs. Erst nach erfolgreich abgeschlossener Pilotphase, belegbar durch Referenzen, sollte man ein neues Softwareprodukt einen *Standard* nennen.

3.2.2 Technische Alternativen

Am Ende von Abschnitt 2 war als Campus-Management-System ein Anwendungssystem gefordert worden, das auf einer integrierten Datenbasis aufsetzt. Diese Anforderung ist nicht dadurch entkräftet, dass die HIS GmbH funktionierende Systeme mit separater Datenbasis pro Baustein liefert. Es herrscht Einigkeit, dass betriebliche Anwendungssysteme dieser Struktur als veraltet gelten [Me04]. Somit ist die bereits in Abschnitt 0 beschriebene technische Grobstruktur alternativlos.

Zudem fassen wir die schon in Abschnitt 0 diskutierten Eigenschaften noch einmal tabellarisch zusammen (Tabelle 2), da diese neben dem Anbieter maßgeblich für Ausschreibung und Entscheidung sind. Jeder Bieter muss die fachliche und technische Qualität seines Produkts offen legen. Dies geschieht durch stichprobenartige Inspektionen der Entwicklungs-Artefakte durch die ausschreibende Institution oder deren Beauftragte. Es ist selbstverständlich, dass Entwicklungsdokumente jeglicher Art dazu gehören und nicht nur Quellcode. Präsentationen sind keine hinreichende Entscheidungsgrundlage.

Als zentrale technische Eigenschaft für eine nachhaltige Entwicklung war die Architektur eines Softwareprodukts genannt und mit den Beispielen R und START untermauert worden. Architekturen werden schon lange über Frameworks implementiert [Sp89, Kap.10; Gr+10, S. 249ff.]. Da ein Framework sich über Skelettprogramme und ablauffähige Bausteine über das gesamte Produkt ausbreitet, kann ein unerprobtes Framework auch sehr nachteilig sein. Deshalb sind ergänzend auch die zu prüfenden Eigenschaften des Quellcodes in Tabelle 2 benannt.

Teilprodukt	Eigenschaft
fachlich	
Anforderungen	Differenziert, präzise & verständlich?
Datenmodell	Sachgerecht & detailliert genug (Attributtypen?)
Anwendungsfälle	Der Domäne angemessen; wesentlich?
Abnahme-Testfälle	Untestbare Anwendungsfälle?
technisch	
Grob-Architektur	Verständlich als Kommunikationsmittel innerhalb des Entwicklungsteams?
Datenbankentwurf	Sachgerecht & SQL-Dialekt-neutral?
Framework	Für die Entwickler verständlich dokumentiert?
Programmstruktur	Nachvollziehbar?
Schnittstellen	Systematisch?
Verständlichkeit	Ja oder Nein?
Fehlerbehandlung	Welches Konzept? Verhalten der Schichten gegenüber dem Nutzer?
Schichtenstruktur	Welche? Ist sie strikt eingehalten?
Client/Präsentation	Betriebssystem-unabhängiger Client?
Anwendung	Strikt getrennt von Präsentation?
DB-Zugriff	Kapselung des SQL-Dialekts?
Code	Nachvollziehbar in Struktur & Namenskonzept?
Variablennamen	Problemadäquat & systematisch?
Entwicklungs-Tickets	Existieren sie & werden sie benutzt?
Schnittstellen	Nachvollziehbar?
Programmaufbau	Verständlich?

Tabelle 2: *Vor* der Auftragserteilung zu prüfende Bestandteile eines neu entwickelten Softwareprodukts

Natürlich erwartet niemand, dass ein Bieter Quellprogramme oder Entwicklungsdokumente elektronisch lesbar oder in Papierkopie verschickt. Es ist aber ein gängiges Procedere – wie auch bei Wirtschaftsprüfern und Zertifizierungs-Auditoren, dass Einblick in jedes vom Prüfer gewünschte Teilprodukt gewährt wird. Die ausschreibende Institution wiederum muss bereit sein, einen gewissen Aufwand zu betreiben, *bevor* sie den Zuschlag erteilt und Verträge unterschreibt. Bei bewährten Standardprodukten reduziert sich dieser Aufwand erheblich, denn dort genügen Kunden-Referenzen.

4 Fazit

Ziel des Beitrags war es herauszuarbeiten, was man unter einem Campus-Management-System verstehen müsste. Es darf nicht der Versuch gemacht werden, alle betrieblichen Anwendungssysteme einer Hochschule dort hinein zu definieren, im Gegenteil: Das System lässt sich auf den Kern eines datenbankgestützten Systems fokussieren.

Der Kern eines Campus-Management-Systems unterstützt den Prozess *Lehre*, gestützt auf eine stets konsistente, transaktionssichere Datenbasis. Ein solches System existiert als Standard derzeit noch nicht, obwohl der Bedarf sehr groß ist. Auch Pilotinstallationen liegen in unterschiedlichem Umfang seitens der Anbieter, aber auch hinsichtlich deren Granularität vor. Dabei muss beachtet werden, dass auch Pilotsysteme ausgesprochen schwergewichtige Organisations- und Softwareprojekte sind.

Mit einer Checkliste am Schluss des Beitrags wird Hochschulen eine Hilfestellung gegeben, *was* bei einer Ausschreibung für ein Campus-Management-Systems zwingend beachtet und geprüft werden muss.

5 Literaturverzeichnis

[AR99] Appelrath, H.-J.; Ritter, J.: R/3-Einführung. Springer, Berlin et al. 1999.

[BB09] Bick, M.; Börgmann, K.: A Reference Model for the Evaluation of Information Systems for an Integrated Campus Management. EUNIS 2009, Santiago de Compostella, Spain, Sept. 2009. (die Quelle wird im Internet gefunden).

[Be09] Bensberg, F.: TCO-Analyse von Campus-Management-Systemen – Methodischer Bezugsrahmen und Softwareunterstützung. In: [HKF09], S. 493-502.

[Br+09] Brune, H.; Jablonski, M.; Möhle, V.; Spitta, T.; Teßmer, M: Ein Campus-Management-System als evolutionäre Entwicklung. In: [HKF09], S. 483-492.

[De79] Denert, E.: Software-Modularisierung. Informatik Spektrum, 2(1979) 4, S. 204-218.

[FH09] Fischer, H.; Hartau, C.: STiNE an der Universität Hamburg – Zur Einführung eines integrierten Campus Management Systems. In: [HKF09], S. 533-542.

[Fi99] Fischer, J.: Informationswirtschaft – Anwendungsmanagement. Oldenbourg, München - Wien 1999.

[Gr+10] Grechenig, T.; Bernhart, M.; Breiteneder, R.; Kappel, K.: Softwaretechnik – Mit Fallbeispielen aus realen Entwicklungsprojekten. Pearson Studium, München 2010.

[HKF09] Hansen, H.R.; Karagiannis, D.; Fill, H-G. (Hrsg.): Business Services – Konzepte, Technologien, Anwendungen (Bd 2). 9. Int. Tagung Wirtschaftsinformatik, Febr. 2009 Wien.

[HSU06] Helmut-Schmidt-Universität Hamburg: Empfehlung für die Auswahl eines Softwaresystems Prüfungsverwaltung / Campusmanagement. Sommer 2006. www.hsu-hh.de/campus-portal/index_V6lDrbV51OIcKbVm.html (wird auch ohne login gefunden: am 4.8.2010).

[HR09] Herzwurm, G.; Pietsch, W.: Management von IT-Produkten. dpunkt Heidelberg 2009.

[Kü97] Küpper, H.-U., Controlling, 2. Aufl. Schäffer-Poeschel, Stuttgart 1997.

[Me04] Mertens, P.: Integrierte Informationsverarbeitung, Bd 1. 14. Aufl. Gabler, Wiesbaden 2004.

[MV04] Mecklemburg-Vorpommern: Landeshochschul-Informationssysteme (2004 bis 2007) http://www.campusmv.de/index.php?id=publikationen (am 02.08.2010).

[Pa72] Parnas, D. C.: On the Criteria to be Used in Decomposing Systems into Modules. CACM 15(1972) 12, pp. 1053-1058.
[Ra09] Radenbach, W.: Integriertes Campus Management durch Verknüpfung spezialisierter Standardsoftware. In: [HKF09], S. 503-512.
[SB08] Spitta, T., Bick, M.: Informationswirtschaft. 2.Aufl., Springer, Berlin et al 2008.
[SG95] Schäfer, J. P.; Grauer, M. (Hrsg.): Universitätsverwaltung und Wirtschaftsinformatik. Proceedings, Siegen Okt. 1995.
[SK95] Sinz, E., Krumbiegel, J.: Gestaltung qualitätsgesicherter Universitätsprozesse am Beispiel des Prozesses 'Lehre und Studium'. In: [SG95], S. 15-32.
[SM95] Spitta, T.; Mordau, J.: Entwicklung und Ergebnisse eines allgemeingültigen Fachkonzeptes für die Prüfungsverwaltung an Hochschulen. In: [SG95], S. 128-147.
[Sp89] Spitta, T.: Software Engineering und Prototyping. Springer, Berlin et al. 1989.
[Sp97] Spitta, T.: Standardsoftware zur Verwaltung und Führung von Fakultäten. Eingeladener Vortrag GI-Jahrestagung `97, Univ. Bielefeld, Fakultät für Wirtschaftswissenschaften, Diskussionspapier Nr. 354, August 1997.
[Sp+10] Sprenger, J.; Klages, M.; Breitner, M. H.: Wirtschaftlichkeitsanalyse für die Auswahl, die Migration und den Betrieb eines Campus-Management-Systems. Wirtschaftsinformatik 52(2010) 4, S. 211-224.
[St+07] Stender, B.; Jablonski, M.; Brune. H.; Möhle, V.: Campus Management von der Hochschule aus gedacht, Wissenschaftsmanagement, (2007) 6, S. 19-26. s. auch: http://www.uni-bielefeld.de/bis/projekt/WiMa-Artikel (am 02.08.2010).
[TUM08] http://portal.mytum.de/pressestelle/meldungen/news_article.2008-01-21.7932373032 am: 08-07-2008. Aktuell: http://portal.mytum.de/iuk/cm/index_html (am 02.08.2010).
[TUM10] http://portal.mytum.de/iuk/cm/hintergrund/ (am 02.08.2010).
[UHH10] Uni Hamburg: http://www.info.stine.uni-hamburg.de/ (am 02.08.2010).
[We96] Weber, J.: Hochschulcontrolling – Das Modell WHU. Stuttgart 1996.
[Wiki10] Wikipedia: Amadeus Germany (Das START System). http://de.wikipedia.org/wiki/Amadeus_Germany (am 02.08.2010).

Projektsteuerung einer inkrementellen Systementwicklung bei gleichzeitiger globaler Produktnutzung

Jaroslav Blaha

Blaha Executive Consulting GmbH
Am Klopferspitz 12
D-82152 Planegg
jbl@blexc.com

Abstract: Die Entwicklung eines zentralen, global genutzten IT-Systems (Transport Management System, TMS) für einen weltweit tätigen Logistik-Konzern erforderte die Segmentierung in fünf Inkremente. Um möglichst frühzeitig produktiven Nutzen zu bewirken wird jedes Inkrement nach Fertigstellung in Produktion gesetzt und Altsysteme entsprechend abgebaut. Die Parallelitäten zwischen der Spezifikation des übernächsten Inkrements, der Entwicklung des aktuellen Inkrements, sowie der Unterstützung und Adaption des jeweils produktiven Inkrements, erfordern ein angepasstes Projektvorgehen. Primäres Ziel und organisatorisches Muss ist die Beherrschung mehrerer paralleler Produktlebenszyklen durch ein geographisch verteiltes Projektteam.

1 Überblick

Die Entwicklung und Einführung eines globalen IT-Systems für die vollständige Abwicklung des Luft- und Seefracht Transport Managements eines der weltweit größten Logistikdienstleister kann aufgrund ihrer Größe, Komplexität und Gesamtdauer von ca. sieben Jahren nur in Inkrementen erfolgen.

Um schnellstmöglich Nutzen zu erreichen wird jedes Inkrement nach einem erfolgreichen Abnahmetest in Produktion gestellt. Während der Projektphase werden aus dem laufenden Betrieb Änderungsanforderungen gestellt. Diese kommen sowohl von den Nutzern, als auch durch legale Notwendigkeiten (z.B. Steueränderungen oder Zollvorschriften); vor Allem letztere müssen zwangsläufig und kurzfristig integriert werden. Jede dieser Änderungen muss allerdings auch parallel in der Entwicklung der nächsten Inkremente berücksichtigt werden, da es Abhängigkeiten zu neuer Funktionalität geben kann.

Insbesondere die Notwendigkeit in eine produktive, global einheitliche Software während der Einführung nationale Besonderheiten einer Vielzahl von Ländern zu integrieren, erfordert ein Team, das diesen multi-nationalen Ansatz wiederspiegelt. Dadurch eröffnet sich neben dem temporalen Problem der Abstimmung von paralleler Inkrement-Entwicklung und Integration von Änderungen aus der produktiven Nutzung, auch noch eine geographische Dimension.

Ein wesentliches Merkmal der Projektphilosophie ist die Beherrschung von Anforderungsänderungen in drei parallel aktiven Projekt-/Produktbaselines: Der Produktionsversion (im Rechenzentrum in Deutschland), der aktuell in der Entwicklung oder Test befindlichen Version (beim Softwareentwickler in Deutschland, Indien und Polen), sowie des gerade spezifizierten Inkrements (in Singapur). Oberste Prämisse ist die Vermeidung von „Requirements Creep". Mit einer Änderungsrate von ca. 2,6% über bis dato vier Jahre scheint dies gelungen.

Das System ist seit drei Jahren - mit zurzeit mehr als 4.200 Nutzern in 92 Ländern - in der produktiven Nutzung. Dabei sind drei von fünf geplanten Inkrementen fertiggestellt. Ermöglicht wurde dies durch ein Projektteam mit ca. 180 Mitgliedern mit 25 Nationalitäten in 12 global verteilten Standorten, sowie einer entsprechend angepassten Projektphilosophie, die den massiv parallelen, technischen und geographischen Randbedingungen genüge trägt.

2 Lösungsstrategien

2.1 Funktionale Separation

Bei der Identifikation des Funktionsumfanges bzw. –zuschnittes der Inkremente zu Projektbeginn wurde eine weitestgehend disjunkte Funktionsmenge für jedes Inkrement festgelegt. Jedes der fünf Inkremente des zukünftigen Systems befriedigt eine klar umgrenzte Menge an zusammengehöriger fachlicher Funktionalität (z.B. Luftfracht Import), die jeweils einen kommerziellen Nutzen für das Unternehmen generieren wird. Die logistischen Funktionen jeden Inkrements bilden zugleich, soweit möglich, die Organisationsstruktur des Unternehmens ab. Z.B. sind Luftfracht Import und Seefracht Import zumeist verschiedene Abteilungen in einer Niederlassung; dadurch können diese Abteilungen unabhängig voneinander in verschiedenen Projektphasen auf das neue System umgestellt werden. Dieses Vorgehen erfüllt zwei Ziele:

- Eine Organisationseinheit des Unternehmens muss nur einmal für die Einführung des Systems umgestellt werden. Neue Funktionen eines nachfolgenden Inkrements betreffen diese Organisationseinheit nur noch in geringem Maße. Dadurch reduzieren sich Aufwände für Schulung, Einführungsunterstützung, Datenmigration, Prozessanpassung etc. bei Folgeinkrementen auf Modifikationen und Weiterentwicklungen des „interessanten" Funktionsumfangs für eine Organisationseinheit.

- Die Spezifikation und Entwicklung eines nachfolgenden Inkrements kann größtenteils unabhängig von der realen Nutzung des Systems erfolgen. Fundamentale Konzepte (z.B. Struktur und Terminologie einer Frachtsendung oder eines Transportweges) werden am Projektanfang definiert und müssen durchgängig genutzt werden; die Besonderheiten von z.B. Import vs. Export können jedoch unabhängig voneinander sequentiell entwickelt werden. Dies ist eine Vorbedingung für die Möglichkeit von bis zu drei parallelen Inkrementen in Spezifikation, Entwicklung bzw. Nutzung.

Funktionale Separation ist bei querschnittlichen Funktionen, wie z.B. den Buchhaltungs-/Abrechnungsmodulen, nur begrenzt möglich. Zusatzaufwände für Weiterentwicklungen (z.B. Datenmodellerweiterung und entsprechende Datenmigration) über alle Inkremente hinweg sind unvermeidbar.

Die Brauchbarkeit dieses Konzeptes wurde vor Allem dadurch nachgewiesen, dass nach Projektbeginn aus unternehmerischen Gründen die Reihenfolge der Inkremente umgestellt werden musste. Diese Änderung war für die Projektdauer und -kosten nahezu neutral.

Anmerkung: Zu Projektbeginn wurde diskutiert, ob anstatt eines inkrementellen auch ein agiler Ansatz sinnvoll wäre. Aufgrund der Größe und Komplexität der abzubildenden logistischen Prozesse wurde dies verneint. Der aktuelle Umfang jeden Inkrements wurde als die Mindestfunktionsmenge betrachtet, die jeweils für eine produktive Einführung (mit allen dazugehörigen organisatorischen Anpassungen in den weltweiten Geschäftsstellen) sinnvoll ist. Fachliche Änderungen zu einem produktiven Inkrement werden jedoch nach Bedarf in sehr kurzen Zyklen zwischen 72 Stunden (von Initiierung bis Produktivsetzung!) und wenigen Monaten agil eingeflochten.

2.2 Projektorganisation

Die Projektorganisation ist aufgrund der Parallelität zwischen Entwicklung und Produktion als „Zwitter“ ausgeprägt:

- Die Projektleitung in Deutschland stellt zugleich in Doppelrolle das Service-management inkl. der Verwaltung von getrennten Projekt- und Servicebudgets, sowie der Administration von Entwicklungs-, Wartungs- und Betriebsverträgen, sicher.

- Die Anforderungsdefinition beschäftigt sich schwerpunktmäßig mit der kompletten Spezifikation eines Inkrementes en bloc. U.a. um die Mitarbeiter dieses Teams von den täglichen Einflüssen aus dem Betrieb (z.B. Incidentanalyse, Klärung von Anforderungen vs. Defekte) zu schützen wurde es in Singapur angesiedelt. Ein kleiner Teil des Teams mit Sitz in Deutschland wird vorwiegend mit der Spezifikation von Änderungen (Changes) ggü. der aktuellen Anforderungsbaseline eingesetzt. Dieses Teilteam fängt die zwangsläufig entstehende Volatilität und Dynamik aus dem Produktivbetrieb auf, isoliert das Singapur-Team von den dadurch entstehenden „Rüstkosten" und ermöglich folglich konzentriertes Arbeiten abseits vom Produktionsalltag. Allerdings ermöglichen das Team in Singapur (Deutschland +6h) und ein kleineres Team in New York (Deutschland -6h) in Notfallsituationen aus der Produktion rund um die Uhr Expertise zu fachlichen Anforderungen einzuholen.
- Die Entwicklung ist vollständig an externe Lieferanten ausgelagert. Auch hier gibt es die Trennung zwischen einem Kernteam für die Entwicklung des nächsten Inkrements und einem Wartungsteam, das Fehlerbehebung und Weiterentwicklung der produktiven Software durchführt. Die wesentliche Schwierigkeit besteht darin mehrere parallele Software-/Releasebaselines zu verwalten und unter Konfigurationskontrolle (inkl. „Branching" und „Merging") zu halten. Die dazugehörigen Methoden und Werkzeuge sind im Software Engineering etabliert und weitestgehend problemlos.
- Ein festes Testteam führt sowohl die Abnahmetests für das nächste Inkrement, als auch Regressionstests für Wartungsreleases der produktiven Software durch. In diesem Team wird eine Trennung in zwei Gruppen vermieden. Stattdessen wird dynamisch und mit Hilfe eines Test Management Tools und einer vorhandenen Menge an Testfällen und –skripten der jeweilige Fokus des Gesamtteams abhängig von der Projektphase zwischen Abnahme- und Regressionstests verschoben. Neben dem zentralen Team existieren nationale Testteams, die aus Sicht der fachlichen Anforderung ihres Landes ebenfalls Regressions- und Abnahmetests („User Acceptance Test") durchführen. Alle Testergebnisse werden im gleichen Test Management Tool zusammengeführt, um einen ganzheitlichen Wissenspool zu erhalten.
- Das Deploymentteam ist für die Einführung der Software (z.B. durch Prozessanpassung und –beratung, initiale und fortlaufende Schulung) in den Zielländern zuständig. Um den lokalen Gegebenheiten Rechnung zu tragen ist das Team auf die drei Regionen Europa, Asien, Nordamerika verteilt, um dort mit lokaler Fachexpertise und Sprachkenntnis zu unterstützen. Dieses Team trägt die Hauptlast des Zielkonflikts zwischen Produktivnutzung und Weiterentwicklung. Da das Team sowohl die fachliche Nutzung der produktiven Software betreuen muss, als auch zeitgleich die Einführung eines neuen Inkrements unter Berücksichtigung der angesammelten Erfahrungen und organisatorischen Anpassungen in den Zielländern vorbereitet und begleitet, ist eine hohe Belastung der Mitglieder unvermeidbar.
- Release Management: Siehe Kapitel 2.4

Eine der wesentlichen Herausforderungen ist es allen Mitarbeitern in allen Teams in allen Projektstandorten das Bewusstsein für die jeweilige Projektphase zu vermitteln. Diese Erfolgen in Wellen: Ein Inkrementzyklus dauert ca. 1,5 Jahre (ca. sechs Monate Spezifikation und ca. ein Jahr Entwicklung, Test, Schulung, Einführung). Dadurch ist jedes Team (außer der Anforderungsdefinition) ca. sechs Monate pro Jahr schwerpunktmäßig mit der Einführung eines neuen Inkrementes beschäftigt, während in der Restzeit die Betreuung des Produktivsystems intensiviert wird.

2.3 System- und Softwarearchitektur

Da jedes Inkrement ca. 1,5 Jahre von Spezifikation (ca. 6 Monate) bis Produktionseinführung benötigt (und die Software damit ca. 2 Jahre Lebensdauer hat bis sie vom nächsten Inkrement abgelöst wird), kann es aus bis zu ca. 30 Softwarereleases bestehen:

- Mehrere Releases, die Abnahmetests bis zur Produktionsreife durchlaufen.
- Reguläre, periodische Releases für die Produktion, die fachliche Änderungen (Changes) implementieren.
- Hot fixes, die kritische Defekte oder Changes in der Produktion beheben. Z.B. kann die Zollverwaltung Vorschriften ändern, die innerhalb von 30 Tagen in einem produktiven TMS umgesetzt werden müssen.

Ein vordefiniertes „Staging Model“ beschreibt, wie neue Releases sequentiell zwischen Entwicklungs-, Test-, Schulungs- und Produktivsystemen verteilt werden.

Rechenzentrumsseitig existiert nur eine (massive redundante) Systeminstanz für die globale, produktive Nutzung. Daneben existieren mehrere (je nach Phase bis zu zwölf) ebenfalls zentrale Systeminstanzen für Entwicklung, Abnahmetests, Lasttests, sowie Schulung. Die Inbetriebnahme eines neuen Softwarereleases erfordert folglich nur die einmalige Installation der Software und ggf. eine dazugehörige Datenmigration pro Instanz. Der fundamentale Nachteil der globalen Nichtverfügbarkeit des Produktivsystems während der Installation wird dadurch aufgewogen, dass alle Nutzer weltweit immer auf einem identischen Softwarestand arbeiten. Unter anderem

- weiß die Supportorganisation immer auf welchen Release sich Nutzerprobleme beziehen und muss jeweils nur Wissen über den aktuellen Release vorhalten;
- da die Nutzerorganisation geographisch verteilt ist, um z.B. regionale Sprachen abzudecken bzw. aus globaler Sicht einen „Follow-the-Sun“ Service anzubieten, kann jede Servicestelle jedem Nutzer weltweit gleichermaßen helfen;
- können langlaufende Transaktionen in Produktion nie mehrere Releases mit potentiell verschiedenen Datenmodellen überspannen;
- müssen nur Testsysteme für zwei Releases (das produktive und das nachfolgende) betrieben werden;
- ist die Behebung von Software- oder Konfigurationsdefekten mit einer zentralen Aktion sofort global wirksam.

Aufgrund des notwendigen Funktionsumfangs ist die Verwendung eines „Thin-Client“ (z.B. browserbasiert) nutzerseitig nicht möglich. Stattdessen wird über den Java WebStart Mechanismus bei jedem neuen Release die jeweils aktuelle Clientsoftware automatisch auf den PC des Nutzers übertragen. Während dies technologisch leicht zu implementieren ist und den Nutzer, sowie den lokalen Support, von der Notwendigkeit jeglicher Installation befreit, gibt es einen Nachteil. In den Stunden nach einem Releasewechsel laden alle Client-PCs bei Starten der Anwendung durch den Nutzer die neue Clientsoftware. Die dadurch entstehende Lastspitze auf dem Kommunikations-netzwerk muss bei der Systemeinführung in einer Geschäftsstelle antizipiert und ggf. durch Bandbreitenanapassung kompensiert werden.

2.4 Release Management Organisation

Der Aufbau eines separaten, vom Projekt unabhängigen, Produkt- oder Release-managements in der Linienorganisation erschien unmöglich, da die oben beschriebenen Abhängigkeiten und die Komplexität des Staging Models von zwei parallelen Organisationen nur unvollständig kontrolliert werden könnten. Deshalb umfasst die Projektorganisation auch das vollständige Management aller Software Releases (Produktiv, Test, Entwicklung) und das Lebenszyklusmanagement, inkl. der dazugehörigen ITIL Prozesse für den Betrieb des Rechenzentrums. Parallel zur Verschiebung des Schwerpunkts von verbleibender Entwicklung zu produktiven Anteilen wird die Struktur des Projektes systematisch und schrittweise hin zu einer Linienorganisation umgebaut. Ziel ist es am Ende des Projektes, mit den gleichen Mitarbeitern, ein Produktmanagement innerhalb der existierenden Aufbauorganisation des Konzerns zu etablieren.

Insbesondere kann dadurch den Mitarbeitern bereits während der Projektphase ein Übergang in Linienfunktionen und dadurch eine Karriereplanung über das Projektende hinaus angeboten werden. Dies stellte sich für die Motivation und Teamstabilität als förderlich heraus.

2.5 Prozesse und Werkzeuge

Für die wichtigsten Prozesse (Change Management, Incident Management, Problem/Defect Management) werden sowohl für die Projektphase als auch für die Produktionsbetreuung die gleichen Werkzeuge und Datenbestände verwendet. Changes, Incidents, Problems werden unabhängig von Ihrer Quelle (Projekt vs. Produkt) in den gleichen Datenbankinstanzen der Tools verwaltet. Die Quellangabe erfolgt durch Verweis auf die Releasenummer. Die Zuordnung von Einträgen (z.B. Defekte) zu Produktivnutzung (z.B. für Garantieansprüche) vs. Projektentwicklung (z.B. zur Bewertung der Abnahmewürdigkeit ggü. einem Lieferanten) erfolgt durch einfache Filterung.

Die Komplexität erhöht sich dadurch, dass sich Einträge inhaltlich auf mehrere Releases und Inkremente beziehen können. Wird z.B. ein Defekt in einer produktiven Softwareversion (aus einem Incident heraus) entdeckt, so ist es wahrscheinlich, dass der gleiche Defekt auch im aktuell in Entwicklung befindlichen Softwarerelease auftritt; dort muss er aber separat bewertet, behoben und nachgetestet werden. Die Verwaltung identischer Einträge für verschiedene Releases mit verschiedenen Bearbeitungszyklen wird von den untersuchten und letztlich ausgewählten Tools nicht unterstützt. Als Behelf werden für solche Einträge Kopien pro Release/Inkrement erstellt und unabhängig verwaltet; die logische Zusammengehörigkeit ergibt sich durch Querverweise auf z.B. die Defektnummern in den Kopien.

Generell sehen die Prozesse verschiedene Bearbeitungspfade vor: Defekte, Incidents, Changes

- die aus dem produktiven Release stammen, müssen in den meisten Fällen kopiert und ebenfalls für in Entwicklung befindliche Folge-Releases initiiert werden;
- die aus einem im Abnahmetest befindlichen Release stammen sind zumeist singulär. Zumindest bei schwerwiegenden Defekte wird zusätzlich geprüft, ob sie nicht auch (bis dato unentdeckt) in der Produktionssoftware auftreten;
- können zudem Auswirkungen auf das übernächste, in der fachlichen Spezifikation befindliche Inkrement haben.

Die Prozesse und Datenstrukturen in den Tools wurden darauf ausgelegt, dass im Extremfall z.B. ein Incident aus dem Produktivsystem zu einem Defekt in zwei Releases (in Produktion = V.x, in Abnahmetest= V.x+1) führt, einen Change (z.B. zur Verbesserung der Ausnahmebehandlung) gegen das in Spezifikation befindliche Inkrement (V.x+2) motiviert, eine vertragliche Garantieleistung ggü. V.x, und eine vertragliche Abnahmeverhinderung ggü. V.x+1 auslöst.

Durch die Bündelung aller Informationen zu allen Inkrementen/Releases in jeweils einer Toolinstanz ergibt sich ein Wissenspool der insbesondere für den Support extrem hilfreich ist. Aktuell wurden mit diesem Ansatz ca. fachliche 650 Changes, 3.000 Incidents und 11.000 Defekte über ca. 80 Softwarereleases miteinander verwoben.

3 Zusammenfassung

Angepasste Mechanismen, Strukturen, Werkzeuge und die Berücksichtigung kritischer Erfolgsfaktoren ermöglichen es ein komplexes, geographisch verteiltes Projekt und dessen Organisation über mehrere Jahre von der reinen Systementwicklung, über eine hybride Phase aus Projekt- und Produktmanagement, hin zu einer Linienorganisation zu überführen.

Entscheidend ist es diese Parameter bei der initialen Projektkonfiguration technologisch, organisatorisch und auch budgetär vorzusehen.

Literaturverzeichnis

[Jws10] Java(TM) WebStart Guide: http://download.oracle.com/javase/6/docs/technotes/guides/javaws/developersguide/contents.html.

[Of07] Office of Government Commerce (ed.): ITIL: The Official Introduction to the ITIL Service Lifecycle, London 2007.

Projekt- und produktorientierte IT-Unternehmen – Einige geschäftsmodellgestützte Überlegungen

Thomas Deelmann

T-Systems International GmbH
Strategy Development
Friedrich-Ebert-Allee 140
53113 Bonn
thomas.deelmann@t-systems.com

Abstract: Die Entwicklung von einem projektorientierten zu einem produktorientierten IT-Unternehmen scheint auf den ersten Blick durch die Möglichkeiten der Wiederverwendung von Software-Artefakten naheliegend und gut umsetzbar zu sein. Bei genauerer Betrachtung existieren zwischen den Geschäftsmodellen eines projekt- und eines produktorientierten Unternehmens jedoch deutliche Unterschiede. Der vorliegende Kurzbeitrag will sie vergleichend gegenüberstellen und so eine Hilfestellung für diesen in der Praxis kritischen Übergang bieten.

1 Einleitung: Motivation, Ziel, Aufbau

Die IT-Industrie befindet sich in einem Prozess der stetigen Weiterentwicklung. Mit Blick auf das Marktvolumen expandieren die meisten Branchensegmente. Gleichzeitig scheint bei Kunden eine Professionalisierung des IT-Managements und bei Lieferanten eine Industrialisierung der Leistungserbringung voranzuschreiten. Eine Komponente dieser Industrialisierung ist die Wiederverwendung von Software. Dies kann auf der Ebene kleinerer Artefakte ebenso erfolgen wie auf der Ebene von Modulen oder kompletten Projektergebnissen. Die Wiederverwendung hilft IT-Anbieter-Unternehmen, getätigte Aufwendungen mehrfach in Umsätze zu verwandeln. Ein konsequentes Verfolgen des Ansatzes führt zu einer Veränderung des Geschäftsmodells: Von der Durchführung von Softwareprojekten hin zu Produktion und Vertrieb von Softwareprodukten.

Durch einen solchen Schritt werden typischerweise weniger Projektmanagementkompetenzen (Führungsaufgaben, -organisation, -techniken und -mittel für die Initiierung, Definition, Planung, Steuerung und Abschluss von Projekten [DIN09]) benötigt, jedoch vermehrt Produktmanagementkompetenzen notwendig. Der Handlungsspielraum des Produktmanagements umfasst allgemein die Spezifikation des Leistungskerns, die Festlegung begleitender Dienste sowie die Bildung und Profilierung von Marken [HH08].

Der Entwicklung der Ausrichtung eines IT-Service-Unternehmens von einer Projektorientierung zu einer Produktorientierung verlangt nach Veränderungen von Geschäftsmodell und verfolgter Strategie der jeweiligen Organisation. Diese Notwendigkeit ist oftmals nicht in allen Facetten direkt erkennbar, jedoch vielfach vorhanden. Die aufgezeigte Entwicklung ist für ein Unternehmen von strategischer Bedeutung (vgl. auch [De10]). Sie mag auf der einen Seite naheliegend und reizvoll sein. Auf der anderen birgt sie auch Risiken, die insbesondere durch eine Nichtberücksichtigung notwendiger struktureller und operativer Veränderungen entstehen und ein sorgfältiges Abwägen verlangen.

Der vorliegenden Kurzbeitrag will Unterschiede zwischen den verschiedenen Ausprägungen der wesentlichen Elemente der Geschäftsmodelle eines projektorientierten und eines produktorientierten IT-Unternehmens aufzeigen, in einem Kurzvergleich gegenüberstellen und so eine Hilfestellung für diesen in der Praxis kritischen Übergang bieten.

Nach diesem einleitenden Abschnitt wird im Hauptteil des Beitrages zunächst das Geschäftsmodell inklusive seiner wesentlichen Bestandteile als Analyseeinheit eingeführt, bevor die Kernbestandteile jeweils für ein projekt- und ein produktorientiertes Unternehmen gegenübergestellt werden. Anschließend werden verschiedene Aspekte im Kontext eines möglichen Geschäftsmodellwechsels skizziert. Eine Zusammenfassung, ein Fazit und ein Ausblick auf weitere Forschungsfragen schließen den Beitrag ab.

2 Geschäftsmodelle: Konzepteinführung und Gegenüberstellung

Geschäftsmodell als Analyseeinheit: Ein Geschäftsmodell kann als „eine abstrahierende Beschreibung der ordentlichen Geschäftstätigkeit einer Organisationseinheit angesehen werden. In der Regel wird bei der Modellkonstruktion auf Organisationseinheiten, Transformationsprozesse, Transferflüsse, Einflussfaktoren sowie Hilfsmittel, oder einer Auswahl hieraus, zurückgegriffen.“ [SDL02]

Der Begriff Organisationseinheit umfasst profitorientierte und nichtprofitorientierte Organisationen sowie Organisationen in ihrer Gesamtheit und Teile hiervon. Zusätzlich kann die sogenannte betrachtete Organisationseinheit und die externe Organisationseinheit (i.e. Kunde, Lieferant, Partner) unterschieden werden. Transferflüsse sind Güter-, Informations- und Finanzflüsse. Transformationsprozess, d.h. die konkrete Leistungserbringung, und Transferfluss sind sogenannte Wertträger. Ihnen kann bei Bedarf ein monetärer Wert zugeordnet werden und so die Wertschöpfung in einer Geschäftsmodellabbildung dokumentiert werden.

Das Geschäftsmodell eines projektorientierten Unternehmens besteht regelmäßig in der Dienstleistungserbringung für einen oder mehrere Kunden in Form von Projekten [Br05]. Demgegenüber besteht das Geschäftsmodell eines produktorientierten Unternehmens darin, ein Werk gegen Zahlung einer Vergütung herzustellen. Verschiedene Unterschiede der Geschäftsmodelle werden nachfolgend entlang der Bestandteile unter Zuhilfenahme einer gewissen Vereinfachung aufgezeigt.

Organisationseinheit: Die Aufbauorganisation eines produktorientierten Unternehmens unterscheidet sich an verschiedenen Stellen von der eines projektorientierten Unternehmens. Das Vorhandensein eines Produktmanagement-Bereiches ist hierbei offensichtlich. Daneben sind Unterschiede im Vertrieb, Personalmanagement und Servicebereich vorhanden. Während Vertriebsmitarbeiter in einem projektorientierten Unternehmen regelmäßig zusätzlich die Rolle eines Projektleiters innehaben, bekleiden sie in einem produktorientierten Unternehmen eine monodimensionalere Funktion. Das Personalwesen in einer Projektumgebung fordert und fördert stärker Mitarbeiter, welche neben technischen Fachkenntnissen auch über gute sogenannte Soft Skills verfügen und in der Interaktion mit Kunden bestehen können. Produktionsorientierte Unternehmen haben hingegen einen ausgeprägten (After Sales-) Servicebereich, da eine gegebene Kundeninteraktion nicht mit der Bezahlung des Hauptproduktes beendet ist (z.B. Bezahlung einer Software oder Bezahlung von Projektleistungen), sondern z.B. durch Anfragen von Kunden bei einer Service-Hotline deutlich länger wirkt.

Transformationsprozess: Der wesentliche Transformationsprozess in einem projektorientierten Unternehmen besteht in der meist kollektiven und teilweise kundengemeinschaftlichen Arbeit an einer individuellen Fragestellung. Diese Arbeit wird durch geeignet zusammengestellte Projektteams erledigt und erfolgt im Anschluss an ein gegebenes Leistungsversprechen. Bei einem produktorientierten Unternehmen steht die antizipierte Lösung von voraussichtlich bei mehreren (potenziellen) Kunden auftretenden Problemen im Mittelpunkt der Aktivitäten. Die Erbringung einer konkreten Leistung wird um die entsprechende Vermarktung ergänzt.

Transferfluss: Transferflüsse können, wie oben beschrieben, aus Gütern, Informationen und Finanzmitteln bestehen. Ein wesentlicher Unterschied zwischen einem produkt- und einem projektorientiertem Unternehmen besteht im Zeitpunkt des Eintretens der Finanzflüsse. Während im Projektgeschäft alle Finanzflüsse zeitlich relativ nahe beieinander liegen können sich im Produktgeschäft deutliche Asynchronitäten ergeben. Im Projektgeschäft erfolgen z.B. Zahlungsströme in Form der Mitarbeiterentlohnung zeitnah zur Rechnungsstellung gegenüber dem Kunden sowie zum korrespondierenden Zahlungseingang. Im Produktgeschäft liegt oftmals eine deutliche zeitliche Lücke zwischen dem Zahlungsstrom in Form der Mitarbeiterentlohnung und der Auslieferung des Produktes an den Kunden und der Bezahlung eines Produktpreises und/oder Servicegebühr.

Hilfsmittel: Sie unterstützen Transformationsprozesse oder Transferflüsse. Durch die explizite Vertriebsorganisation wird ein produktorientiertes Unternehmen Hilfsmittel vornehmlich in diesem Bereich einsetzen (z.B. CRM-Systeme). Ein stärker auf Services und Projekte ausgerichtetes Unternehmen wird versuchen, primär diese Bereiche zu unterstützen, z.B. mit Skill-Management- oder Knowledge-Management-Systemen.

Einflussfaktoren: Beide Geschäftsmodelle werden durch unterschiedliche (externe) Faktoren maßgeblich beeinflusst. Während Angebot und Nachfrage grundsätzlich von Relevanz sind, ist im Projektgeschäft die wirtschaftliche Lage der Kunden von großer Bedeutung. In schlechten Situationen werden Projekte zurückgestellt, zeitlich gestreckt oder inhaltlich eingeschränkt. Zudem verringert sich die Zahlungsbereitschaft nicht nur in der absoluten Höhe der Ausgaben, sondern auch auf Basis von z.B. Tagessätzen.

Das Produktgeschäft reagiert hier – insbesondere in einer Koppelung mit Servicegebühren – weniger volatil, was wiederum geringere Auswirkungen auf das Geschäftsmodell hat. Gleichzeitig besteht durch die ausgewiesenen Produkteigenschaften eine stärkere Vergleichbarkeit gegenüber Wettbewerbsprodukten, Substituten oder Eigenleistungen.

Wertträger: Transformationsprozessen und Transferflüssen können Werte zugewiesen werden. Während im Projektgeschäft der Transformationsprozess (z.B. ‚Projekt') werttreibend ist, kommt im Produktgeschäft den Transferflüssen (z.B. ‚Produktübergang', ‚Bezahlung') diese Eigenschaft zu. Für das jeweilige Unternehmen ist es daher von Relevanz, hieran jeweils weitere Umsatzströme zu koppeln. Bei Projekten gelingt dies durch Change Requests und Folgeprojekte, bei Produkten durch Updates und Servicegebühren.

3 Transition: Übergang und Wechsel von Geschäftsmodellen

Die Unterschiede zwischen projekt- und produktorientierten Geschäftsmodellen gilt es bei einem Übergang zu berücksichtigen. Als Grundlage für die Entscheidung über einen Wechsel kann bei wirtschaftlich ausgerichteten Unternehmen eine Prognose über Gewinn- und gegebenenfalls auch Umsatzverläufe genutzt werden. Sind bei dem zukünftigen produktorientierten Unternehmen die erwarteten Gewinne absolut und/oder als Marge höher als beim projektorientierten Unternehmen und stellt auch die neue Umsatzerwartung kein Hindernis dar, so kann ein Wechsel empfohlen werden. Sind die Gewinnerwartungen geringer, so kann von einem Wechsel abgeraten werden.

Neben diesen idealisierten Szenarien sind weiterhin Fälle denkbar (und treten vermutlich deutlich häufiger auf), in denen eine Entscheidung nicht hinreichend deutlich getroffen werden kann, nicht nachhaltig erscheint oder in denen zunächst bewusst mit zwei Geschäftsmodellen parallel operiert wird. In diesen hybriden Situationen gilt es, die oben aufgeführten Differenzen der Geschäftsmodelle zu berücksichtigen. Es kann in diesen Fällen die Bildung von zwei Geschäftsbereichen, von denen einer ein projektorientiertes und einer ein produktorientiertes Geschäftsmodell verfolgt, in Betracht gezogen werden.

Regelmäßig sollen auch nach der Bildung von Geschäftsbereichen sog. Synergiepotenziale realisiert werden. Unbeschadet dessen ist darauf zu achten, dass bei drei Geschäftsmodell-Bestandteilen, bei denen besonders große Unterschiede bestehen, eine Trennung herbeigeführt wird (vgl. auch den Bericht eines verwandten Praxisbeispiels in [To03]):

(i) Organisationseinheiten: Es ist wichtig, dass die beteiligten Mitarbeiter Kenntnis darüber haben, in welchem Geschäftsmodell sie tätig sind und ggf. bei Parallelaktivitäten bewusste Unterscheidungen herbeiführen, durch z.B. Zeitbuchungen auf unterschiedliche Kostenstellen. Dies kann störende Irritationen und Zielkonflikte vermindern. *(ii) Finanzflüsse:* Zwei Geschäftsmodelle verfügen typischerweise über unterschiedliche Umsatzströme und Gewinnmargen. Der Erfolg des einen Geschäftsbereiches kann bei vorliegen von Quersubventionierungen und Vermischungen nicht hinreichend gewürdigt werden und es kann auch hier zu Verstimmungen bei den Betroffenen führen.

(iii) Transformationsprozesse: Die Arbeitsabläufe und Geschäftsprozesse in den beiden betrachteten Geschäftsmodellen sind regelmäßig unterschiedlich (z.B. hinsichtlich ihrer Wiederholfrequenz) und bilden verschiedene Arbeitskulturen heraus. Eine Trennung bzw. das Bewusstsein über vorhandene Unterschiede hilft, Konflikte zu vermeiden.

Diese Darstellung ist stark vereinfacht. Die Umsetzung erfolgt in der Praxis regelmäßig deutlich differenzierter und auf die jeweilige Unternehmenssituation zugeschnitten. Auch können sich im Zeitverlauf Verschiebungen in der Gewichtung des projekt- und produktorientierten Geschäftsmodells ergeben. Unbeschadet dessen erscheinen die Hinweise für den Umgang mit parallel verfolgten Geschäftsmodellen hilfreich, um eine erfolgreiche Transition zu unterstützen.

4 Abschluss

Der vorliegende Kurzbeitrag hat die Diskussion über eine organisatorische Entwicklung „vom Projekt zum Produkt“ um Überlegungen entlang der Bestandteile eines Geschäftsmodells erweitert. Auch wenn im ersten Schritt ein Übergang naheliegend erscheint und die Arbeitsweisen einer projekt- und einer produktorientierten Organisation Ähnlichkeiten aufweisen, so ist im Rahmen einer Detailbetrachtung anzumerken, dass die jeweiligen Geschäftsmodellkomponenten deutliche Unterschiede aufweisen.

Das Wissen hierüber und ein bewusstes Behandeln der Eigenschaften sind hilfreich, wenn eine Geschäftsmodellweiterentwicklung „vom Projekt zum Produkt“ erfolgen soll. Ergänzende Aktivitäten, wie z.B. das Formulieren und Implementieren einer Strategie, die eine Geschäftsmodell-Erweiterung aufgreift oder die Beachtung und Umsetzung rechtlicher Rahmenbedingungen müssen separat betrachtet werden.

Die oben aufgezeigten Unterschiede sind nur kurz angerissen worden und bedürfen einer weiteren theoretischen Ausgestaltung und praktischen Umsetzung. Ebenso können Fallstudien den Erkenntnisprozess über den (Miss-) Erfolg von Unternehmenstransformationen von einer Projektorientierung zu einer Produktorientierung fördern.

Literaturverzeichnis

[Br05] Brugger, R.: IT-Projekte strukturiert realisieren. Vieweg Verlag, Wiesbaden, 2005, insb. S. 102-113.

[De07] Deelmann, T.: Geschäftsmodellierung: Grundlagen, Konzeption und Integration. Logos Verlag, Berlin, 2007, S. 137.

[De10] Deelmann, T.: Geschäftsmodelldiversifikation von Unternehmensberatungen – Einige Überlegungen zu den Optionen für Beratungs- und Service-Unternehmen im IT-Umfeld. In: Schumann, M.; Kolbe, L. M.; Breitner, M. H. und Frerichs, A. (Hrsg.): Proceedings zur MKWI 2010. Universitätsverlag Göttingen, Göttingen 2010, S. 637-648.

[DIN09] Deutsches Institut für Normung e.V.: DIN-Norm: DIN 69901-5:2009-01, Projektmanagement – Projektmanagementsysteme – Teil 5: Begriffe.

[HH08] Herrmann, A.; Huber, F.: Produktmanagement: Grundlagen, Methoden, Beispiele. Gabler-Verlag, Wiesbaden, 2008, S. 1-2.

[SDL02] Scheer, C.; Deelmann, T.; Loos, P.: Geschäftsmodelle und internetbasierte Geschäftsmodelle – Begriffsbestimmung und Teilnehmermodell. In: Loos, P. (Hrsg.): Working Papers of the Research Group ISYM. Paper 12, Mainz, 2003, S. 22.

[To03] Toffler, B.L.: Final Accounting. Broadway Books, New York, USA 2003.

Produktkernel in der Systemintegration (Erfahrungsbericht aus der Praxis)

Dr.-Ing. Ingo Elsen, Asma Hawari, Uwe Johnen

T-Systems GEI GmbH
Systems Integration Automotive and Manufacturing Industries
Pascalstraße 51
52076 Aachen
{ingo.elsen, asma.hawari, uwe.johnen}@t-systems.com

Abstract: In der Vergangenheit basierten große Systemintegrationsprojekte in der Regel auf Individualentwicklungen für einzelne Kunden. Getrieben durch Kostendruck steigt aber der Bedarf nach standardisierten Lösungen, die gleichzeitig die individuellen Anforderungen des jeweiligen Umfelds berücksichtigen. T-Systems GEI GmbH wird beiden Anforderungen mit Produktkerneln gerecht. Neben den technischen Aspekten der Kernelentwicklung spielen besonders organisatorische Aspekte eine Rolle, um Kernel effizient und qualitativ hochwertig zu entwickeln, ohne deren Funktionalitäten ins Uferlose wachsen zu lassen. Umgesetzt hat T-Systems dieses Konzept für Flughafeninformationssysteme. Damit kann dem wachsenden Bedarf der Flughafenbetreiber nach einer effizienten und kostengünstigen Softwarelösung zur Unterstützung Ihrer Geschäftsprozesse entsprochen werden.

1 Die Idee des Produktkernels

Große Systemintegrationsprojekte im Umfeld der Individual-Software liefen in der Vergangenheit oft so ab, dass, sofern der Typ der Anwendung bekannt war, Wiederverwendung in Form von Kopieren von Source Code betrieben wurde. Mit dem Aufkommen der Enterprise Java Plattform entstanden technische Frameworks, die zwar keinerlei fachliche Funktionalität bereitstellen konnten, den Entwicklern aber sehr viel Arbeit abnahmen und –nehmen. Ein Beispiel hierfür ist das Hibernate-Framework [HIB10]. Neben technischer Funktionalität besteht aber vielfach der Bedarf, über immer wiederkehrende fachliche Funktionalität verfügen zu können.. In der Systemintegration sind Produktlinien in der Regel zu unflexibel, da die Umgebungen, in die die Systeme integriert werden, für einen solchen Ansatz zu heterogen sind. Hier kommt die Idee des Produktkernels zum Tragen: Fachliche Grundfunktionalität, die allen Anwendungen gemeinsam ist, wiederverwendbar bereitzustellen.

In der Literatur [BAS06, BUS01, CLE07, FOW03] finden sich keine Einordnungen von Produktkerneln in die Taxonomie der Elemente einer Softwareanwendung. Wir haben daher einen Vorschlag hierzu entworfen, der sich in Abbildung 1 findet.

Beim Übergang von Bibliotheken zu Frameworks, Produktkerneln, Produktlinien und Produkten steigt jeweils die Komplexität des Elements, während die Flexibilität für den Einsatz in verschiedenen Anwendungsdomänen sinkt. So sind z.B. die Java-Standard-bibliotheken für praktisch jede Anwendungsdomäne einsetzbar; Java-Frameworks für Web-Anwendungen eigenen sich jedoch nicht für reine Client-Anwendungen.

Im Gegenzug sinkt mit steigender Komplexität der Elemente der Aufwand, um einen produktiven Einsatz zu erreichen. Während Off-the-Shelf-Produkte in der Regel sofort einsetzbar sind - maximal ist eine Einstellung der Konfigurationsdateien erforderlich - gilt es, aus Produktkerneln zunächst Anwendungen zu erstellen.

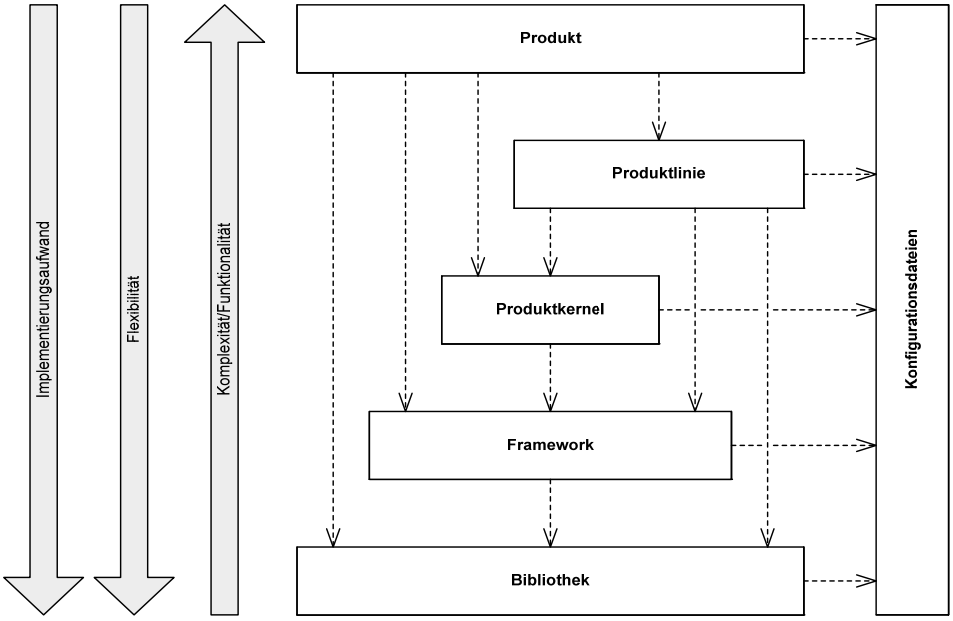

Abbildung 1: Einordnung der Produktkernels in die Taxonomie von Anwendungskomponenten

Produkte werden in ihrer Codebasis nicht von Installation zu Installation verändert. Variabilität in der Funktionalität erzielt der Administrator einer Anwendung durch Konfiguration von z.B. XML Dateien [PAR10]. Hier ist mittlerweile durch Deskriptive Sprachen (z.B. für GUI) eine Anwendungsadaption in weiten Grenzen möglich [QT10]. Hierbei sind die funktionalen Abläufe innerhalb der Anwendung vorgegeben und nicht veränderbar. Inhalte sind konfigurierbar, indem aus einer Gesamtmenge für eine konkrete Anwendung Teilmengen ausgewählt und dem Anwender zur Verfügung gestellt werden [SPR10].

Im Systemintegrationsgeschäft unterscheiden sich die Anforderungen von Kunde zu Kunde, basieren aber zumeist auf einer gemeinsamen Grundfunktionalität der Anwendungsdomäne. Die Abläufe können jedoch unterschiedlich sein, und die Inhalte sind nicht vollständig aus einer Obermenge ableitbar, da hier der Systemkontext, der in aller Regel auf gewachsenen individuell geprägten Anwendungslandschaften und –prozessen basiert, zu berücksichtigen ist.

Ein *Produktkernel* enthält bereits solche Grundfunktionalitäten wie z.B. eine Datenbank, in der sich ein Datenmodell findet, das die Gemeinsamkeiten der Geschäftsprozesse der Anwendungsdomäne enthält. Gleiches gilt für die Geschäftsfunktionalität in den Komponenten des Produktkernels. Auch hier sind komplette Workflows in ihrer Grundfunktionalität bereits implementiert und können durch externe Erweiterungen, z.B. via Shared Libs, unter Anwendung des Dependency Injection Musters [FOW04, NYG07] um kundenspezifische Funktionen ergänzt werden. Hier sind insbesondere Schnittstellen zu Bestandssystemen zu nennen.

Frameworks, die zwar bestimmte Prozessschritte modellieren können [BUS01], aber normalerweise keine Fachlichkeit eines Geschäftsprozesses beinhalten, sind wegen des zu leistenden Aufwands nur dann attraktiv, wenn eine einmalige Anwendungserstellung und Systemintegration gefordert ist. Der Aufwand zur Realisierung einer Fachlichkeit im Vergleich zu einem Produktkernel ist deutlich höher. Auf der anderen Seite eigenen sich Frameworks für die Anwendung in verschiedenen Fachgebieten wie z.B. bei unterschiedlichen Geschäftsprozessen.

In spezifischen Geschäftsdomänen gibt es viele fachliche Gemeinsamkeiten und wiederkehrende Prozesse. Daher ist es sinnvoll, über die Flexibilität von Frameworks hinaus Fachlichkeit so umzusetzen, dass sie in verschiedenen Anwendungsszenarien wiederverwendet werden kann. Gleichzeitig muss die Flexibilität bewahrt werden, so dass eine Anpassung an geänderte prozessuale Abläufe und fachliche Inhalte sowie eine Integration in unterschiedlichste Systemlandschaften gewährleistet ist.

Ein Produktkernel, wie ihn T-Systems einsetzt, besteht im Wesentlichen aus

- fachlichen Elementen, deren Inhalte aber in weiten Grenzen konfigurierbar sind.
- Konfigurationsschnittstellen zur Anpassung des Systemverhaltens in der Integrationslandschaft.
- Schnittstellen zu Frontends verschiedener Hersteller
- Schnittstellen zur Adaption der fachlichen Elemente und Workflows. Hier findet sich primär Dependency Injection (DI) als verbreitetes Architekturmuster.

In Tabelle 1 sind die Nutzen der verschiedenen Ansätze bezogen auf deren organisatorische Umsetzung und wirtschaftliche Anwendung dargestellt.

Kriterium	Bibliothek	Framework	Produktkernel	Produktlinie	Produkt
Realisierungsaufwand bis Inbetriebnahme[1]	--	-	+	+	++
Wartungsaufwand bei großer Basis installierter Anwendungen	--	-	+	+	++
Flexibilität in der Anwendung in verschiedenen Domänen	+	o	o	-	--
Flexibilität in der Anwendung innerhalb einer Domäne bei verschiedenen Szenarien	o	+	++	o	--

Tabelle 1: Eignung der verschiedenen Ansätze aus organisatorischer Sicht.

2 Flughafeninformationssysteme bei der T-Systems GEI GmbH

Die Luftfahrtindustrie hat sich in den vergangenen Jahren rasant verändert. Die Herausforderungen sind gewachsen durch eine steigende Anzahl von Akteuren und Geschäftsprozessen sowie viele neue Leistungen und Angebote, die den Service für den Fluggast verbessern und effizienter gestalten. Um diese zunehmende Komplexität beherrschen zu können und gleichzeitig in der Lage zu sein, sich auf ihre Kernkompetenzen zu konzentrieren, benötigen Flughafenbetreiber eine anspruchsvolle Informations- und Kommunikations-Technologie (ICT).

Zu den Anforderungen der Flughäfen an die ICT gehören u. a. die Bereitstellung auf voll integrierte Lösungen, standardisierte Anwendungen und Systemlösungen, die das zukünftige Wachstum unterstützen.

Das Total Airport Management System (TAMS) der T-Systems GEI ist auf diese Anforderungen ausgerichtet und unterstützt alle wesentlichen Prozesse und Abläufe zum Flughafenbetrieb. Von der Ressourcenplanung über die Disposition bis hin zur Simulation von Passagierströmen ist TAMS modular den Kundenanforderungen angepasst.

[1] Unter der Annahme, dass ein Produkt existiert

Zentraler Bestandteil eines TAMS ist die Airport Operational Database (AODB). Sie stellt sicher, dass alle Informationen, die über die Partnersysteme einfließen oder manuell eingepflegt werden, automatisch eingearbeitet werden und anschließend dem Bedienpersonal und den Kunden des Flughafens adressatengerecht zur Verfügung stehen. Für die Planung von immobilen Ressourcen wie z.B. Check-In-Counter, Abstellpositionen für Flugzeuge u.v.m. bietet T-Systems ein integriertes Resource Management System (RMS). Für übersichtliche Informationsanzeigen sorgt das neue Passagier- und Dienstinformationssystem, Flight Information Display System (FIDS). Dieses System bringt nicht nur großen Nutzen für die Passagiere, sondern auch für die Flughafenbetreiber, die die Anzeigetafeln zusätzlich für Werbung in den öffentlichen Terminalbereichen anbieten können.

Die genannten Lösungen stellen lediglich einen Ausschnitt des gesamten Lösungsportfolio der T-Systems zu TAMS dar (s. folgende Abbildung).

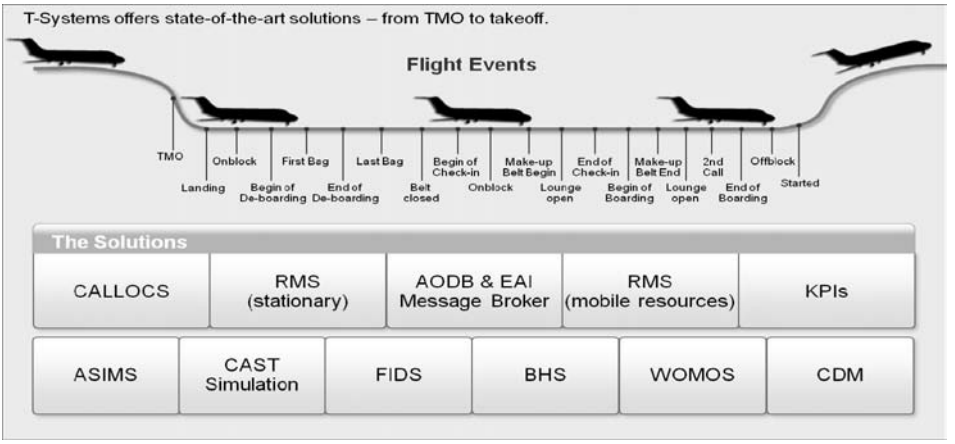

Abbildung 2: Lösungsportfolio der T-Systems zum Total Airport Management System

3 Entwicklung von Produktkerneln für Flughafensysteme

Grundlage für den Einsatz von Produktkerneln ist die Standardisierung. Dabei kann zwischen der **Prozess-Standardisierung**, d. h. der Standardisierung der Entwicklungsmethoden (z. B. Softwareentwicklungs-Prozess, Projekt Management-Prozess/ Projekt-Handling, Entwicklungs-Tools) und der **Technologie-Standardisierung**, d. h. der Standardisierung der technischen Lösung unterschieden werden.

3.1 Standardisierung der Softwareentwicklung (Prozess-Standardisierung)

Die Prozess-Standardisierung bildet die Grundlage für den Wandel von der projektorientierten zur Produktkernel-orientierten Softwareentwicklung. Dabei stehen besonders die Anwendung standardisierter Methoden zum Projekt Management und zum Software-Engineering im Fokus [SEB10, PMB10]. Weitere Aspekte der Prozess-Standardisierung sind in Abbildung 3 dargestellt.

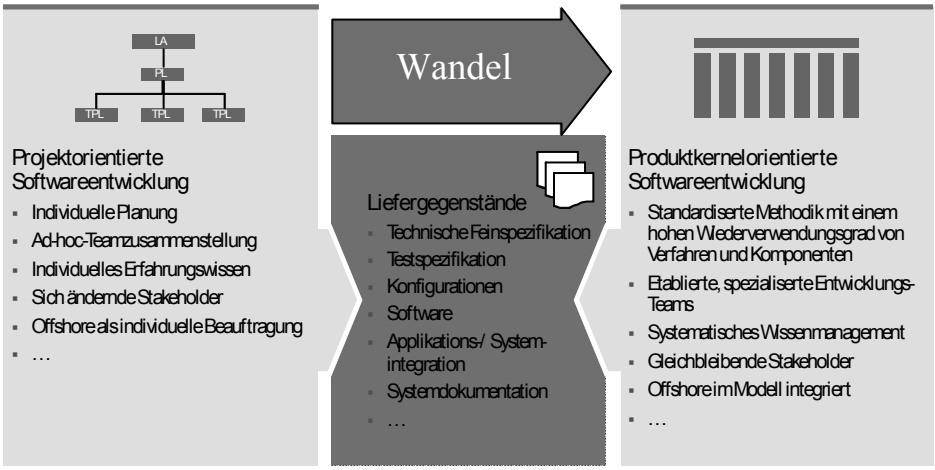

Abbildung 3: Vom Projekt- zum Produktmodell

Zu beachten ist, dass die Produktkernel-orientierte Softwareentwicklung keineswegs im Widerspruch zu der Berücksichtigung kundenindividueller Anforderungen steht.

Die Produktkernel-orientierte Softwareentwicklung führt zur Transparenz im Entwicklungsprozess und der Prozessreife und somit zu einer besseren Vorhersehbarkeit und Messbarkeit der Ergebnisse sowie einem deutlichen Zuwachs an Qualität und Effizienz. Dies ist im Interesse jedes einzelnen Kunden.

3.2 Standardisierung der Lösungen (Technologische Standardisierung)

Der Grad der Prozess-Standardisierung bei der Entwicklung von Flughafensystemen ist in der T-Systems bereits sehr hoch. Zur Bewältigung der oben genannten aktuellen Marktanforderungen der Flughäfen besteht jedoch zusätzlich die Notwendigkeit einer Technologie-Standardisierung.

Dabei ist es nicht das Ziel von T-Systems, das Flughafengeschäft zu einem hundertprozentigen Produktgeschäft zu entwickeln. Vielmehr soll weiterhin auf spezielle und individuelle Kundenanforderungen reagiert werden können. Deshalb bietet sich der Einsatz von Produktkerneln in Kombination mit der Anwendung einer standardisierten Softwareentwicklung für kundenindividuelle Systemanpassungen und Erweiterungen an. Damit werden die Vorteile beider Standardisierungsarten genutzt und gleichzeitig wird die Flexibilität der Systeme garantiert.

Die Entwicklung von Produktkerneln für Flughafensysteme unter Berücksichtigung der laufenden Kundenprojekte und bestehender Lösungen bedarf einer strategischen Planung.

Folgende Maßnahmen sind im vorliegenden Fall für eine Migration vom reinen Projektgeschäft zum Einsatz von Produktkerneln notwendig:

- Definition von Standard Delivery Elementen (SDE), die als Produktkernel angeboten werden.
- Prüfung des aktuellen Abdeckungsgrads (Current Mode of Operations) dieser SDEs bzgl. typischer Kundenanforderungen (funktional und nicht-funktional).
- Aufzeigen eines Ziel-Szenarios (Future Mode of Operations) und der Maßnahmen zur Entwicklung dieses Ziel-Szenarios.
    - Welche Systeme werden zu einem Produktkernel ausgebaut?
    - Welche Systeme werden in Produktkerneln integriert?
    - Welche Systeme werden stillgelegt?
- Definition einer Referenzarchitektur [SUM07] für jeden Produktkernel. Hierzu gehört z. B. die Festlegung von:
    - Entwicklungsplattformen
    - Betriebsplattformen
    - Kernel-Funktionalitäten
    - Nicht-funktionalen Eigenschaften
- Definition eines Life-Cycle-Management für Produktkernel.

3.3 Organisatorische Auswirkungen auf die Zusammenarbeit im Team

Die Bereitstellung von Produktkerneln hat Auswirkungen auf die Organisation und Zusammenarbeit von Projekt- und Entwicklungsteams [MCC96].

Folgende Abbildung stellt eine mögliche Rollenverteilung dar, die die Grundlage einer Zielorganisation sein sollte.

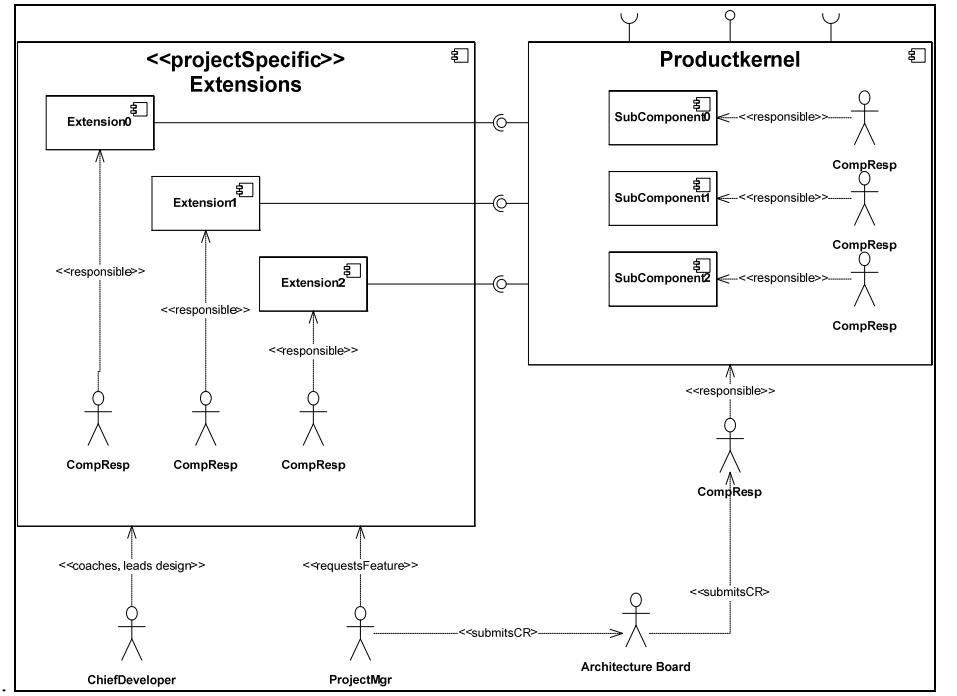

Abbildung 4: Rollenverteilung für die Bereitstellung von Produktkerneln

Relevant ist, dass für jeden Produktkernel ein Entwicklungsteam existiert, das für die domänenspezifische Entwicklung der Grundfunktionalitäten verantwortlich ist. Für die kundenspezifische Integration sind andere Projektteams zuständig.

Diese Trennung der Verantwortlichkeiten führt dazu, dass die Pflege des Source-Code eines Produktkernel nur an einer Stelle stattfindet. Alle Änderungswünsche der Projektteams an diesen Source-Code müssen über ein Architektur-Board bewertet und beim Produktkernel-Entwicklungsteam eingereicht werden. Das Architektur-Board dient somit als Eingangstor für alle Anforderungen an den Source-Code eines Produktkernel.

4 Zusammenfassung und Ausblick

Die Autoren sehen durch die Evolution der Systemintegration hin zu Lösungen auf Basis von Produktkerneln einen essentiellen Beitrag zur Wettbewerbsfähigkeit. Durch den hohen Reifegrad der Lösung kann der Kunde einen höheren Nutzen erzielen. Dies liegt insbesondere daran, dass ein Produktkernel in Abgrenzung zu einem Framework Domänenwissen und somit mehr Fachlichkeit und eine höhere Flexibilität beinhaltet. In Abgrenzung zu einem Produkt bzw. zu einer Produktlinie besteht der Vorteil in der höheren Flexibilität und in der Möglichkeit der Modelarisierung.

Die Mitarbeiter des Anbieters - sowohl im Entwicklungsteam für den Produktkernel als auch im individuellen Projektteam zur Erarbeitung der Kundenlösungen - haben zusätzliches Potential für den Wissensaustausch und den Wissensaufbau. Damit ist die Fokussierung auf ein Lösungsangebot mit Produktkerneln zukunftsträchtig.

Ausgehend von der Idee des Produktkernels und den Anforderungen an Flughafeninformationssysteme hat die T-Systems Systems Integration ein Modell entwickelt, bei dem Produktkernel erfolgreich in Projekten eingesetzt werden. Kernbestandteil sind die technische Standardisierung in Form des Produktkernels und die Standardisierung der Entwicklungsprozesse für ähnlich gelagerte Projekte.

Die positiven Auswirkungen des Kernelansatzes – insbesondere durch die Möglichkeit der Wiederverwendung - auf die Qualität, die Laufzeit und das Budget der Projekte im Flughafenbereich dienen als Referenz für weitere Geschäftsdomänen.

Somit wird dieses erfolgreiche Konzept in Zukunft auf weitere Fachbereiche der T-Systems Systems Integration ausgedehnt. Darüber hinaus sind zukünftig die Messung des Standardisierungsgrades und der technischen und prozessualen Wiederverwendung Bestandteil der Unternehmensentwicklung.

Auch wird durch dieses Konzept eine gute Voraussetzung für ein intelligentes und globales Sourcing- und Delivery-Model gelegt, so dass aktuellen Anforderungen zum Offshoring zur Sicherung der Wettbewerbssituation von Unternehmen begegnet werden kann.

Literaturverzeichnis

[BAZ06] L. Bass, et al., Software Architecture in Practice, Addison-Wesley, 2006.

[BUS01] F. Buschmann et al., Pattern Oriented Software Architecture Vol. 1, Wiley, 2001.

[Az99] Azubi, L. et.al.: Die Fußnote in LNI-Bänden. In (Glück, H.I.; Gans, G., Hrsg.): Formattierung leicht gemacht – eine Einführung. Format-Verlag, Bonn, 1999; S. 135-162

[CLE07] P. Clements, L. Northrop, Software Product Lines, Addison-Wesley, 2007.

[FOW03] M. Fowler, Patterns of Enterprise Architecture, Addison-Wesley, 2003.

[FOW04] M. Fowler, Inversion of Control Containers and the Dependency Injection pattern, http://martinfowler.com/articles/injection.html, 2004

[HIB10] Hibernate Relational Persistence for Java & .NET, http://www.hibernate.org/, 2010.

[MCC96] S. McConnel, Rapid Development, Microsoft Press, 1996.

[NYG07] M. T. Nygard, Release It!, Raleigh, 2007.

[PAR10] T. Parr, Language Implementation Patterns, Raleigh, 2010.

[PMB10] PMBook, T-Systems, 2010.

[QT10] Nokia, Qt Reference Documentation, doc.qt.nokia.com, 2010.

[SPR10] Spring Framework, www.springsource.org/documentation, 2010.

[SEB10] SEBook, T-Systems, 2010.

[SUM07] I. Summerville, Software Engineering, Addison-Wesley, 2007.

Six Sigma for Analyzing Market Preferences

Dr. Thomas Michael Fehlmann

Euro Project Office AG
Zeltweg 50
8032 Zürich
thomas.fehlmann@e-p-o.com

Abstract: The New Lanchester Theory links Business Objectives to market share using a transfer function, known from Six Sigma and Quality Function Deployment (QFD). The transfer function can effectively be used for prioritizing new features for software products. However, it is not easy to define this transfer function. This paper presents current practices and new opportunities for software marketing that arise from recent advances in Six Sigma theory.

1 Introduction

1.1 What are Market's Preferences?

This chapter explains why market's preferences – essential for software products – are on a different level than customer's needs in a project QFD. Market's needs are different – the market decides based on preferences rather than on analyzing individual customer's needs, formulating requirements and assessing whether the software product meets these requirements. Many voices shape such preferences – professional journals, trade magazines, experiences made, assumptions and prejudices play their roles. In ICT, market preferences are expressed as a selection of product features – those features that are relevant when evaluation products and doing buying decisions. Sometimes, this market selection is quite a surprise for the supplier of the product – for instance, when adopting mobile telephones, the market decided for a precedence for the Short Message Service (SMS text messages, originally designed for internal use only when servicing network hubs), it was not expected for inventors and providers of that service.

The *Deming Value Chain* consists of a series of transfers from one value chain topic level into another. It depends on the domain; for instance, with software it looks as shown in Figure 1. Advice how to set up a Deming Value Chain for various deployments is available in the Quality Function Deployment Best Practices [HS06]. W. Edwards Deming published organizational production chain deployment schemes in the early 1930'ies already [De86]. Prof. Akao used similar schemes for "QFD in the Broad Sense" [Ak91]. This is the reason for calling such deployments Deming Value Chain.

However, when no direct customer is involved into development, Voice of the Customer is not as readily available as for customer software projects, project sponsors cannot formulate requirements by analyzing customer's needs; thus product management has to guess somehow what the potential customers want. Nevertheless, it is possible to predict market precedence using the power of Six Sigma and of Deming Value Chains.

1.2 Deming Value Chains

Deming Value Chains link Deming Value Creation Processes, value-added production steps that transform resources into business value.

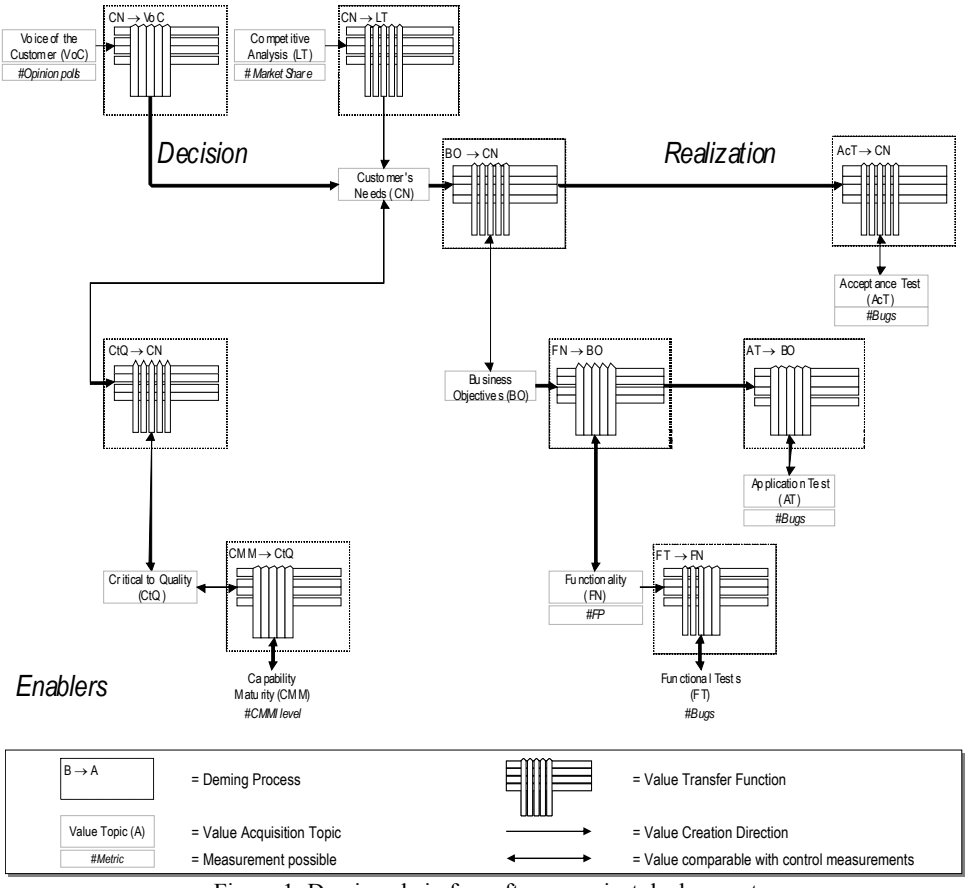

Figure 1: Deming chain for software project deployment

We write **B** → **A** when **B** is a set of resources that result in business values **A**. For instance, Customer's Needs (**CN**) result in Voice of the Customer (**VoC**) that reflects those needs **CN** → **VoC**. Customer's Needs (**CN**) also cause buying decisions (**CN** → **LT**; LT stands for "Lanchester Theory", see below) reflecting the same needs.

In turn, Use Cases (**UC**) must meet Customer's Needs (**UC** → **CN**); i.e. they fulfill those needs what we also understand as sort of production process. In this sense, we adopt and re-use Deming's value chains for software and service purposes.

Obviously, knowing about **VoC** and **LT**, the software supplier should understand Customer's Needs (**CN**), and, if successfully understanding **CN**, he should be able to formulate Use Cases (**UC**) – or User Stories, whatever approach he might take – and finally propose suitable functionality (**FN**). Knowing the results aimed to in the Deming Value Chain does not automatically define which resources provide them.

There is also an upward branch: Test Stories (**TS**), Application Tests (**AT**), and Acceptance Tests (**CT**) support the respective topic according their level; and there is a second downward branch in the value chain that deals with non-functional requirements such as the Critical-to-Quality (**CtQ**) characteristics, and process maturity (**CMM**) needed to meet such quality characteristics.

1.3 Deming Value Chain for Software Products

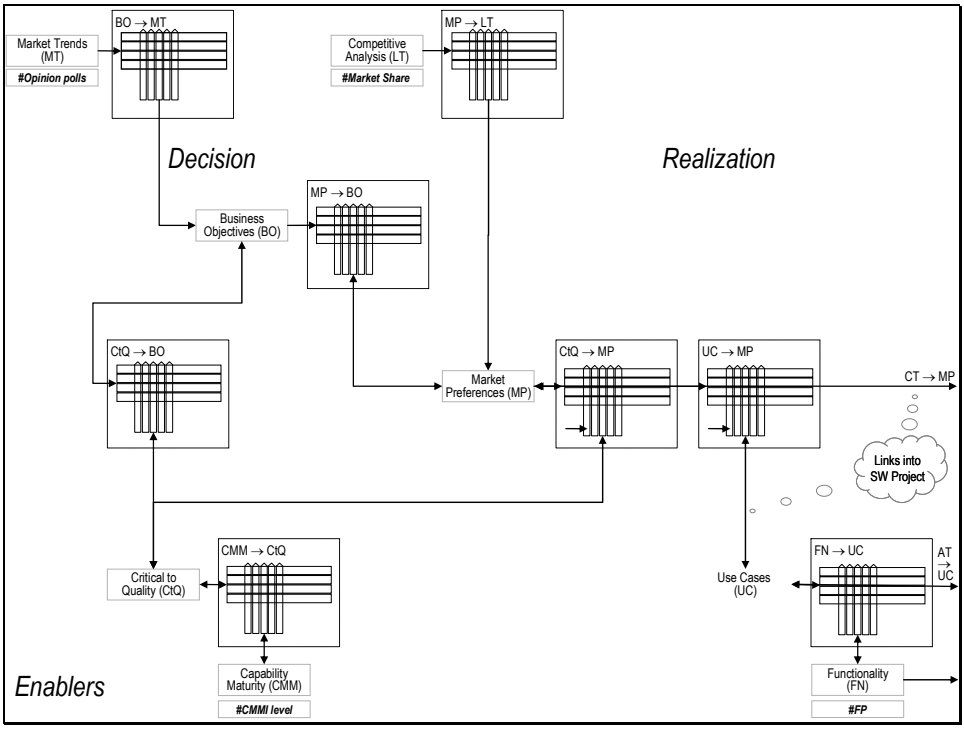

Figure 2: Deming Value Chain for software product deployment

When setting up the Deming value chain for software products, there is a significant difference (Figure 2). Not Customer's Needs (**CN**) are pivotal but *Business Objectives* (**BO**). Market preferences develop around business objectives; the selection and valuation of business objectives changes when market preferences evolve.

The Deming Value Chain starts with *Market Trends* (**MT**). Market trends are harder to assess, they are the result of decisions made by market players about their *Business Objectives* (**BO**) which in turn depend from *Market Preferences* (**MP**) rather than from Software Engineering artifacts directly. Both the functionality offered for Use Cases in the software product, as Critical-to-Quality (**CtQ**) characteristics impact market preferences. This makes the Deming Value Chain for software products deployment significantly more complicated than for software projects.

A software project is needed to create a software product, but the software engineering artifacts such as use cases or test cases link not to requirements or customer's needs, but to market preferences. Moreover, market preferences depend both from critical-to-quality characteristics and from the use case functionality provided by the product. Thus both quality characteristics and functionality are more difficult to align to software product development than in a software project. Moreover, no formal acceptance testing is possible; the market decided probably based on pilot experiences but not on tests.

1.4 Control by Measurements

Six Sigma offers metrics and tools needed to provide *Decision Metrics* to this Deming Value Chain, by linking the controls to observable facts. A decision metric is an evaluation function $\mathbf{A} \rightarrow \mathbf{A}$ that yields stable results, where **A** is some business value topic. This means, repeating the decision method always yields the same results.

Market preferences are not directly measurable; but surveys, market research provide observable measurements, and market share is measurable. Using Six Sigma *Transfer Functions*, we can measure market preferences indirectly; however, we need to know how reliable such indirect measurements are. A Transfer Function is an evaluation of a Deming process $\mathbf{B} \rightarrow \mathbf{A}$.

Competitors excel at different levels with respect to market preferences; this is called a *Competitive Profile*. The difference between market preferences and the competitive profile can be measured using the *Convergence Gap*, which is the length vector difference between the market preference vector and the competitive profile vector, measured in the linear vector space of market preferences.

2 What are Transfer Functions?

In general, observations such as market share, buying decisions, Voice of the Customer, or customer complaints can be measured. However, if those observed and measured facts are not as favorable as they probably should be, if variations are too high, Six Sigma aims at knowing their causes, and becoming able to influence the causes with suitable controls such that future observations are more satisfactory. The dependency of observed facts from controls is called *Transfer Function*.

In mathematical terms, if we have more than one observation and more than one control for the solution, we use the term "*Vector*" for such multidimensional observation profiles and solution profiles. If $\underline{\mathbf{y}}$ denominates the vector profile of the observed facts, and **T** is the transfer function that links the control profile vector $\underline{\mathbf{x}}$ to the target, then $\mathbf{T}(\underline{\mathbf{x}}) = \underline{\mathbf{y}}$ indicates that the controls are capable to always hit the target. Then, hit rate is within the six sigma tolerance range.

T transfers solution vectors $\underline{\mathbf{x}}$ into results $\mathbf{T}(\underline{\mathbf{x}})$, called '*response'* [CSA03]. These responses can be measured and compared with the initial observations $\underline{\mathbf{y}}$. If the variations between observation $\underline{\mathbf{y}}$ and the responses $\mathbf{T}(\underline{\mathbf{x}})$ for all solution vectors $\underline{\mathbf{x}}$ are all within the six sigma tolerance range, the responses become *predictable*.

Transfer Functions are used for prediction. In Six Sigma, prediction models are build for measurable business observations (the '$\underline{\mathbf{y}}$'s, e.g., defect density, project costing) with the aim to control the solution-impacting factors '$\underline{\mathbf{x}}$' so well that the response (i.e., the process results $\mathbf{T}(\underline{\mathbf{x}})$) are forced into the allowable tolerance range around the $\underline{\mathbf{y}}$'s.

The entities $\underline{\mathbf{x}}$ and $\underline{\mathbf{y}}$ are vector profiles since neither observation nor influencing factors come alone, or without constraints. The vector spaces that we consider represent business requirements and solution approaches, respectively. These vector spaces accommodate statistical events, such as requirement definition, or solution design.

Multidimensional vector spaces are much more common that usually felt; for instance, a normal distribution of events is a vector in a multidimensional vector space, its dimension being the degree of freedom, i.e., the number of mutually independent events. Such normal distributions can be quite large and the dimension is not limited.

2.1 Problem Statement

A typical problem statement is: the observation profile $\underline{\mathbf{y}}$ is known; various controls exist but it is unknown which controls are effective and which are not. The control profile $\underline{\mathbf{x}}$ is unknown, even if there are enough candidate controls that seem applicable.

The usual solution approach is to investigate the characteristics of the transfer function **T** that explains how $\underline{\mathbf{x}}$ impacts $\underline{\mathbf{y}}$. In mathematical terms: the problem statement is $\mathbf{T}(\underline{\mathbf{x}}) = \underline{\mathbf{y}}$, where $\underline{\mathbf{x}}$ is unknown.

The number of independent dimensions needed for the control space is equal or higher than for the observation space. Otherwise dimensions wouldn't be independent.

2.2 Matrix Representation of Transfer Functions

If the transfer function **T** is *linear*, **T** can be expressed as a matrix. For this, select a set of *base vectors* both in the space of controls $\underline{\mathbf{x}}$, say $\underline{\mathbf{a}}_1$, …, $\underline{\mathbf{a}}_n$; as in the space of observations $\underline{\mathbf{y}}$, say $\underline{\mathbf{b}}_1$, … $\underline{\mathbf{b}}_m$. Every vector $\underline{\mathbf{x}}$ can be written as $\underline{\mathbf{x}} = \xi_1\underline{\mathbf{a}}_1 + \ldots + \xi_n\underline{\mathbf{a}}_n$, the ξ_i being scalar values, called *components* of $\underline{\mathbf{x}}$. Assuming the base vectors as fixed, the vector $\underline{\mathbf{x}}$ is written simply as $\underline{\mathbf{x}} = < \xi_1, \ldots \xi_n >$. Then a Transfer function **T** can be written as a matrix because every base vector $\underline{\mathbf{a}}_i$ yields a component representation in the space of observations $\underline{\mathbf{y}}$:

$$\mathbf{T}(\underline{\mathbf{a}}_i) = < \tau_{i,1}, \ldots \tau_{i,m} > \tag{1}$$

for all i=1…*n*. The components $\tau_{i,j}$ represent **T** as component matrix (i=1…*n*, j=1…*m*):

$$\mathbf{T} = \begin{bmatrix} \tau_{1,1} & \tau_{1,2} & \cdots & \tau_{1,m} \\ \tau_{2.1} & \tau_{2,2} & \cdots & \tau_{2,m} \\ \cdots & \cdots & \cdots & \cdots \\ \tau_{n,1} & \tau_{n,2} & \cdots & \tau_{n,m} \end{bmatrix} \tag{2}$$

Let $\underline{\mathbf{x}} = < \xi_1, \ldots \xi_n >$ and $\underline{\mathbf{y}} = < \psi_1, \ldots \psi_m >$ for some fixed base vector sets. Then the equation $\underline{\mathbf{y}} = \mathbf{T}(\underline{\mathbf{x}})$ can be written in component form (*matrix multiplication*):

$$\underline{\mathbf{y}} = \left\langle \psi_1 = \sum_{i=1\ldots n} \tau_{i,1}\xi_i, \quad \psi_2 = \sum_{i=1\ldots n} \tau_{i,2}\xi_i, \quad \ldots, \quad \psi_m = \sum_{i=1\ldots n} \tau_{i,m}\xi_i \right\rangle \tag{3}$$

Every transfer function has a transpose, denoted as $\mathbf{T}^{\mathbf{T}}$, whose matrix elements are constructed by switching rows with columns. Assume the matrix representation of **T** has *n* rows and *m* columns; $\mathbf{T}^{\mathbf{T}}$ has *m* rows and *n* columns. *n* is called the *dimension* of the control space, and *m* is the dimension of the observation space.

$$\mathbf{T}^{\mathbf{T}} = \begin{bmatrix} \tau_{1,1} & \tau_{2,1} & \cdots & \tau_{n,1} \\ \tau_{1,2} & \tau_{2,2} & \cdots & \tau_{n,2} \\ \cdots & \cdots & \cdots & \cdots \\ \tau_{1,m} & \tau_{2,m} & \cdots & \tau_{n,m} \end{bmatrix} \tag{4}$$

In more business-related words, *n* is the number of controls in $\underline{\mathbf{x}}$ needed to observe the response $\underline{\mathbf{y}}$ that in turn has profile length *m*. $\mathbf{T}^{\mathbf{T}}$ looks similar to a Transfer function but works on the observation profile $\underline{\mathbf{y}}$: $\mathbf{T}^{\mathbf{T}}(\underline{\mathbf{y}})$ yields a possible solution profile $\underline{\mathbf{x}}' = \mathbf{T}^{\mathbf{T}}(\underline{\mathbf{y}})$ that is in the same space as the unknown "ideal" solution profile $\underline{\mathbf{x}}$, fulfilling $\mathbf{T}(\underline{\mathbf{x}}) = \underline{\mathbf{y}}$; but in contrary to the unknown $\underline{\mathbf{x}}$, it can be computed easily once **T** is known.

Applying **T** to $\underline{\mathbf{x}}$' yields **T**($\underline{\mathbf{x}}$') = $\underline{\mathbf{y}}$'; this is the response that effectively can be achieved with the solution profile $\underline{\mathbf{x}}$'. This response is called *Prediction*. A prediction can be compared with an observation.

The question is: how far away from $\underline{\mathbf{y}}$ is $\underline{\mathbf{y}}$'? The mathematical theory of linear vector spaces gives a straightforward answer: the $\mathscr{L}_2$ vector space norm defines length of a vector, and the distance between two vectors is the length of the vector difference. Vectors can be subtracted by computing the differences between their respective components[1]. In the Euclidian space where we live, $\mathscr{L}_2$ is what we usually call distance between two points in space; a measurable entity, not depending from the measurement direction. In a Six Sigma statistical vector space, the $\mathscr{L}_2$ norm measures distributions.

Let $\underline{\mathbf{y}} = < \psi_1, \ldots \psi_m >$ be the observation and $\underline{\mathbf{y}}' = \mathbf{T}(\underline{\mathbf{x}}') = \mathbf{T}(\mathbf{T}^{\mathbf{T}}(\underline{\mathbf{y}})) = < \psi'_1, \ldots \psi'_m >$ be the prediction. The distance is written with two double-bars ||…||. For normalization reasons[2], we divide the result by square root of the dimension *n*:

$$\|\underline{\mathbf{y}} - \underline{\mathbf{y}}'\| = \sqrt{\frac{\sum_{i=1..n} (\psi_i - \psi'_i)^2}{n}} \tag{5}$$

The difference is called *Convergence Gap*. For instance, take a normal distribution of events $< \xi_1, \ldots \xi_n >$. As seen before, this is a vector in a multidimensional vector space, its dimension being the number of mutually independent events.

Now compare the events vector $< \xi_1, \ldots \xi_n >$ with the vector whose components consist all of the same arithmetic mean $\overline{\xi}$.

$$\overline{\xi} = \frac{\sum_{i=1..n} \xi_i}{n} \tag{6}$$

Take the $\mathscr{L}_2$ vector norm distance between that distribution vector and the arithmetic mean point in the observed vector space of events:

$$\sigma = \sqrt{\frac{\sum_{i=1..n} (\xi_i - \overline{\xi})^2}{n}} \tag{7}$$

[1] It is left to the reader to assess the strange characteristics of $\mathscr{L}_1$, the maximum norm for vectors – the length of the vector is determined by its maximum components – or $\mathscr{L}_3$, replacing squares by cubicles. Comparison of two vectors is useless in $\mathscr{L}_1$, or $\mathscr{L}_3$; but in $\mathscr{L}_2$, normalized vectors can be compared, see [Fe04].

[2] The reason for the division through the square root is discussed in [Fe04].

Formula (7) is the definition of the standard deviation sigma indicating the variance of the normal distribution[3]. Thus, the standard deviation is the vector difference between $< \xi_1, \ldots, \xi_n >$ and $< \bar{\xi}, \ldots, \bar{\xi} >$, since normal distributions are a very simple example of a Transfer Function trying to hit the arithmetic mean point.

Real life transfer functions are more complicated than normal distributions; when studying market preferences, normal distributions are very rare. Actual Convergence Gaps are seldom exactly zero, but it is sufficient if they are small enough.

2.3 Combining Transfer Functions

Given two transfer functions $\mathbf{T}_1$ and $\mathbf{T}_2$ where the goal or observation space of $\mathbf{T}_2$ corresponds to the solution space of $\mathbf{T}_1$, it is common practice to combine a composite transfer function by first applying $\mathbf{T}_1$ and then $\mathbf{T}_2$. The definition for all vectors $\underline{\mathbf{x}}$ is

$$[\mathbf{T}_1 \bullet \mathbf{T}_2](\underline{\mathbf{x}}) = \mathbf{T}_1 (\mathbf{T}_2(\underline{\mathbf{x}}))$$

The combination of two transfer functions corresponds to the sequential chaining in the Deming Value Chain; for instance, when $\mathbf{T}_{\mathbf{MP}\rightarrow\mathbf{LT}}$ denotes the transfer function between market preferences and market share according Lanchester Theory, and $\mathbf{T}_{\mathbf{CtQ}\rightarrow\mathbf{MP}}$ the transfer function between Critical-to-Quality characteristics and market preferences, then $\mathbf{T}_{\mathbf{MP}\rightarrow\mathbf{LT}} \bullet \mathbf{T}_{\mathbf{CtQ}\rightarrow\mathbf{MP}}$ is the combined transfer function that describes the influence of quality characteristics on market share. For business and design decisions, this is very valuable information.

2.4 Consistency of Transfer Functions

However, combinations are not only useful when looking at Deming Value Chains. Using the transpose $\mathbf{T}^{\mathbf{T}}$, we have already seen that the combination $\mathbf{T} \bullet \mathbf{T}^{\mathbf{T}}$ has interesting characteristics.

If $[\mathbf{T} \bullet \mathbf{T}^{\mathbf{T}}](\underline{\mathbf{y}}) \cong \underline{\mathbf{y}}$, $\mathbf{T}$ is a *Consistent Explanation* for $\underline{\mathbf{y}}$ since we can repeatedly apply $\mathbf{T} \bullet \mathbf{T}^{\mathbf{T}}$ to $\underline{\mathbf{y}}$ and always get the same result: $\underline{\mathbf{y}} \cong [\mathbf{T} \bullet \mathbf{T}^{\mathbf{T}}](\underline{\mathbf{y}}) \cong [\mathbf{T} \bullet \mathbf{T}^{\mathbf{T}}]([\mathbf{T} \bullet \mathbf{T}^{\mathbf{T}}](\underline{\mathbf{y}}))$ and so on. This technique has been used extensively with Analytic Hierarchical Process (AHP). [Sa03]; it explains why AHP is a decision metric. Contrary to other decision methods sometimes used in business[4], AHP can be applied repeatedly and still delivers consistent results. This makes AHP valuable and successful when doing decisions based on incomplete knowledge.

If $\mathbf{T}$ is a consistent explanation for $\underline{\mathbf{y}}$, $\mathbf{T}^{\mathbf{T}}(\underline{\mathbf{y}})$ is an approximate solution for the problem $\mathbf{T}(\underline{\mathbf{x}}) \cong \underline{\mathbf{y}}$. The main problem with transfer functions is to solve the problem $\mathbf{T}(\underline{\mathbf{x}}) = \underline{\mathbf{y}}$, that means to find the elements of the profile vector $\underline{\mathbf{x}}$ that describes the relevant market preferences. Such preferences govern buying decisions and are thus decisive for product success on the market.

---

[3] Actually, σ is the *Maximum Likelihood Estimate* for the standard deviation in a normal distribution.

[4] For instance, the popular "Pair-wise Comparison" method is well-known for suggesting wrong decisions.

With the Convergence Gap, there is a simple method to check whether **T** provides a consistent explanation for **y**. This we can use to investigate market preferences based on incomplete data and soft factors.

2.5 A Simple Example

The following simple example may illustrate the problem. It originates from a QFD Workshop analyzing the market share trends for a *customer phone number inquiry help desk* in a former monopolistic telecom company that has moved into a new competitive environment. The market share was measurable; other emerging phone companies with own help desk operations published numbers. However, customer segmentation was unknown – the assumption was people with limited access to Internet.

We build a transfer function between customer preferences and market share that maps the competitive profile of each competitor onto market share. The "Predicted Profile" to the right is the result of multiplying the "Competitive Profile for Market Preference" with the matrix, and normalizing it. It compares with the "Observed Profile".

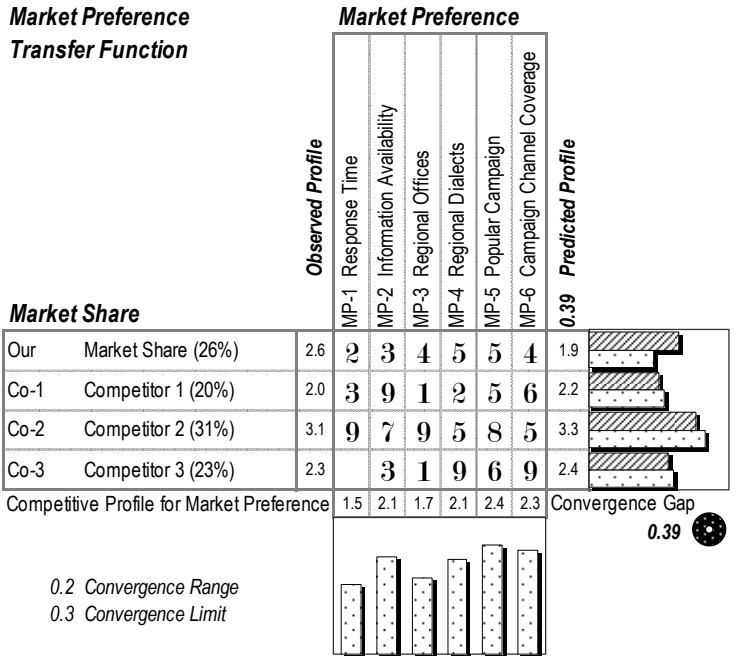

Figure 3: Transfer function not explaining observed market share

In theory, the expectation is that competitors share the market according their specific competitive advantages. In reality, this is not necessarily true.

The matrix cells result from a Quality Function Deployment workshop; they reflect in each column the relative performance of competitors against one market preference topic. The competitive ranking of the controls MP-1: "Response Time" was measurable; Data for MP-2: "Information Availability" was gained with some test calls; MP-3: "Regional Offices" and MP-4: "Regional Dialects" have been measured as well, while MP-5: "Popular Campaign" and MP-6: "Campaign Channel Coverage" were assessed by workshop team agreement.

Nine is highest, zero or empty is least. Equal settings are admissible.

Since the evaluations originate from a real business case, the actual competitors are not shown in Figure 3 until Figure 6. Market shares are shown as a normalized profile vector; for historical reasons [Fe04] the vector components range between 0 and 5.

Clearly there is something missing in Figure 3 that explains why the observed "Our Market Share (26%)" in the observation profile is higher than the predicted market share. The missing market preference factor seemed not related to any technical or organizational feature; rather to the fact that our product was well known and trusted for its long existence. We did call that the *Trust Factor*.

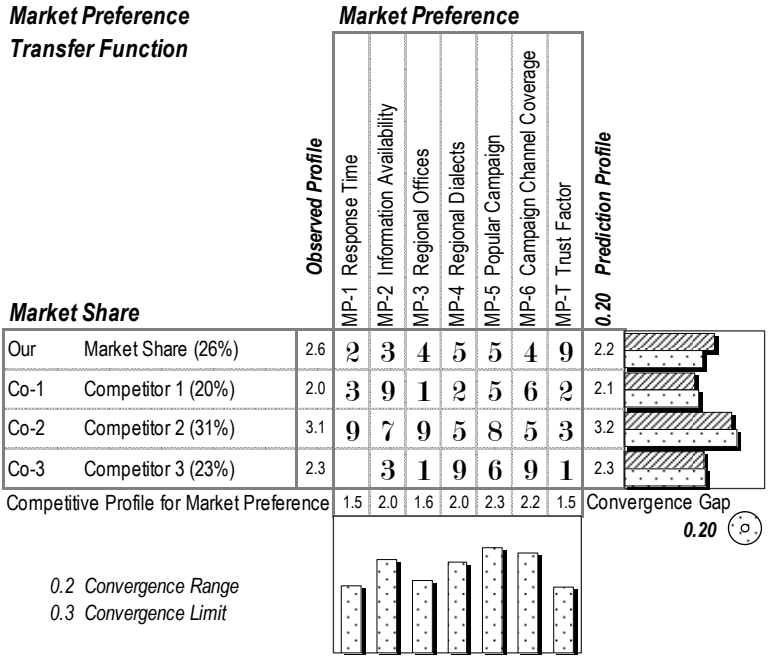

Figure 4: Transfer function better explaining observed market share

Note that the Trust Factor as a market preference has been detected by analyzing the example above only. The competitive ranking was made with a short random survey among people, partly within and partly outside of the organization.

Since the Convergence Gap is now better, there is reason to believe that this extended set of customer preferences – the market preference profile vector **x** with the trust factor added – really reflects market reality. The transfer function makes it possible to quantify the relative importance of the trust factor versus other factors.

However, the Convergence Gap of *0.20* is at the limit of the stated *Convergence Range* – the range of trust defined in view of the data quality – and thus unsatisfactory.

Since market preferences are not customer requirements, it might be interesting to see what happens if we remove some of the Market Preferences. Since these preferences are assumptions not stated requirements, it is tempting to experiment with removing some of the supposed market preferences in order to see what happens.

As candidates for removal, two groups with three topics were identified

1. MP-2: "Information Availability" is not relevant for market preference, because the users of the help desk won't notice if information isn't available.

2. Although the two regional product features (MP-3 and MP-4) were of high political interest and much discussed in the news, typically people aren't ready to pay any price tag for such product features and therefore they are good candidate for removal from the list of market preferences.

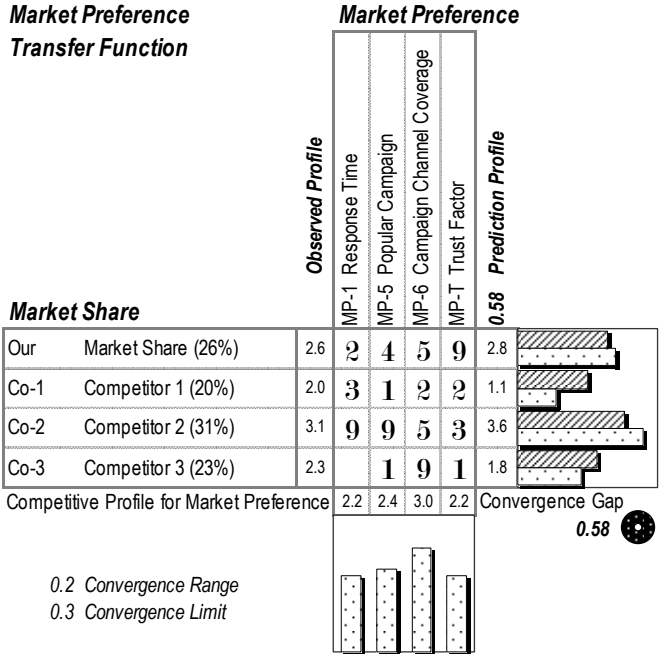

Figure 5: Dismal result when removing some of the supposed market preferences

The result was interesting: the prediction for Competitor 1 and 3 fell below observation, while prediction for Competitor 2's market share was much too high. That gave a hint that something is missing that increases market share for both Competitor 1 and 3.

In fact, both new competitors made heavy use of the evolving Internet search engines and provided full integration of the customer help desk within their web pages. A quick assessment of the team brought a competitive ranking for MP-N: "New Technology" that reflected the actual market share, see Figure 6, and brought the Convergence Gap down to an excellent *0.05*.

In this case, it turns out that with a modified set – the trust factor and the technology factor added – the Convergence Gap is improved and may better reflect market reality.

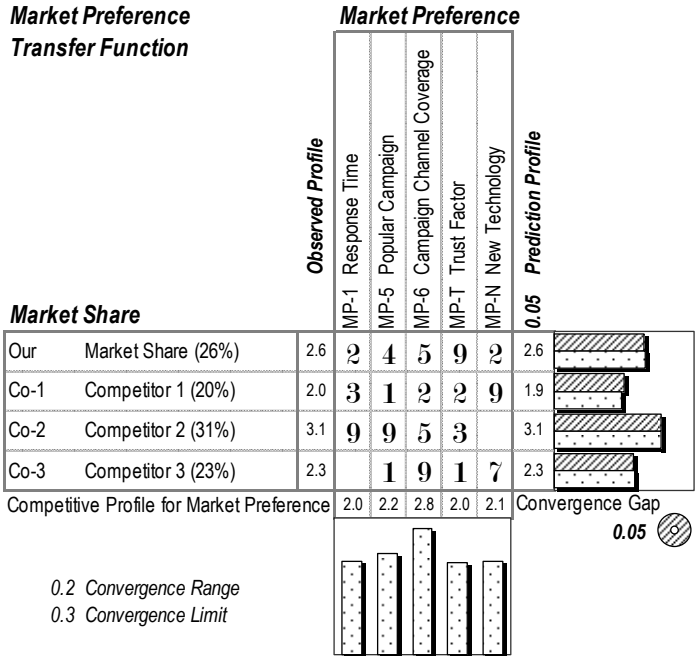

Figure 6: Final result of adjusting the set of market preferences

2.6 Learning from Incomplete Knowledge

Where customers are not readily available to answer questions addressing their needs, Six Sigma offers with transfer functions a nice way of analyzing incomplete information. The Convergence Gap offers a quick means to assess consistency of that analysis; however Six Sigma is no automated problem solver. It requires sound data, domain know-how and excellent QFD practices when predicting market share with Six Sigma.

Clearly, the trust factor is limited in time; as the new competitors consolidate their offerings, its effect will vane and must be replaced by new strong arguments that increase market preference. Recommendations are therefore to watch closely what the competitors do with new technology.

3 The Ideal Situation

3.1 The Lanchester Theory

If the market is in equilibrium, the buying decisions of customers follow a stable pattern. Given a vector of market preferences in the vector space of relevant product features, the transfer function is the matrix of all comparisons buyers made between competitive offerings. This transfer function acts on the market preferences and the results are market shares, again a vector but this time in the vector space of competitive product offerings.

The New Lanchester Theory distinguishes "Strategy of the Strong" for those dominating the market, and "Strategy of the Weak" for those that try to find or create market niches. It deducts a formula drawn from a military model that explains when to use which strategy. For details, see [Fe08].

Ideally, the competitors are distinct only by product features – not by other influence factors such as branding and market image. The target market segment is homogeneous and product features are stable.

The Lanchester "Strategy for the Weak" requires that weak competitors identify areas where they can locally beat the market leader. "Locally" means that they need to focus on some limited market segment.

A formula is derived that basically allows comparing two competitive profile vectors. The so-called "Weapon Strength" E is the ratio between two normalized profile weights (i.e., their vector lengths, see (5)). If E exceeds $\sqrt{3}$ then this competitor is dominant, wins all competitive comparisons and reaches market dominance, see [Fe08].

The New Lanchester Theory has been successfully used under such circumstances for creating world-class products, including software products.

3.2 New Lanchester applied to Help Desk Example

It is tempting to look at the previous example to see what needs to be done for increasing market share. To do this, we calculate the competitive profiles – normally, sum of the competition's profiles against our own competitive profile.

For the above help desk sample, the comparison of competitive profiles shows that the current market is in perfect equilibrium; nobody has a significant competitive advantage:

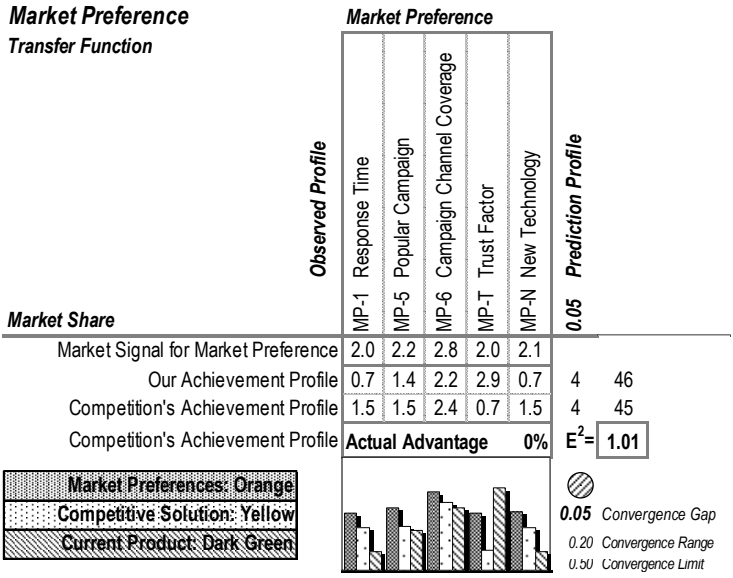

Figure 7: Actual competitive profiles are in equilibrium

The visual profile shows areas for improvement:

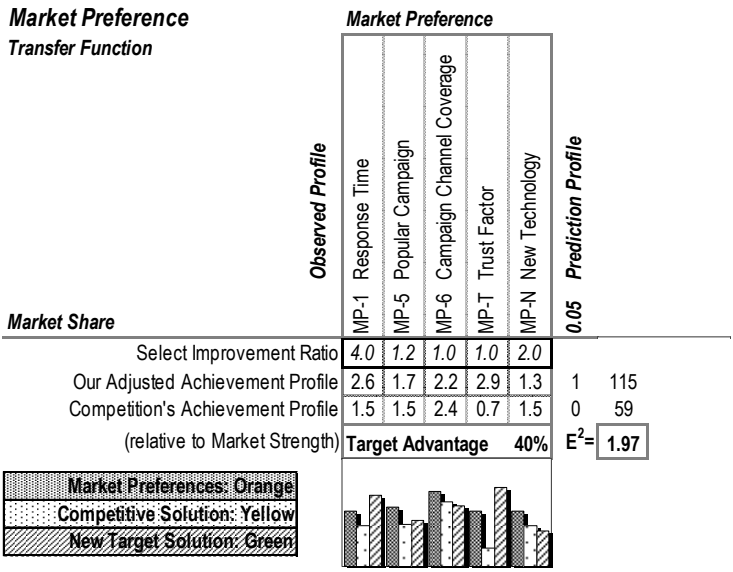

Figure 8: What could be done to win market share

With the New Lanchester theory, it is possible to select topics for product improvement and calculate how well the improved competitive profile of our product matches market preference. Some of the proposed improvements might be costly – especially the increase by a improvement ratio of four (*4.0*) of the MP-1: "Response Time", and the investment needed into MP-N: "New Technology" as well (ratio *2.0*), but an increase of competitive advantage to 40% (corresponding to E2=1.97) is quite rewarding. According the New Lanchester Theory, 41.7% would guarantee market dominance [Fe08].

3.3 Creating a Winning Product

The other relevant transfer functions, according Figure 2: Deming Value Chain for software product deployment, are

1. **UC → MP**: From use case functionality into market preferences – how well does the product meet market preferences? The transfer function $\mathbf{T}_{\mathbf{UC \rightarrow MP}}$ is used to identify the features needed to meet market preferences.

2. **CtQ → MP**: From quality characteristics into market preferences – how well does the product quality meet market preferences, sometimes called "House of Quality"? The transfer function $\mathbf{T}_{\mathbf{CtQ \rightarrow MP}}$ explains the impact of quality characteristics onto market preferences.

Combining the two transfer functions with the Lanchester Theory, it is possible to predict which features need to be added or removed from the software product in order to gain market share. This is the success prediction input needed for release planning.

The first transfer function defines the functionality that is needed to identify the added or superfluous functionality; with the second, quality characteristics are adjusted to better meet market preferences.

4 Overcoming Difficulties

4.1 Identifying Market Preferences

QFD offers many techniques for analyzing information available on the market – such as verbatim analysis, Google search, and many more. The best reference is the QFD Best Practices [HS06] drafted by practitioners of the German QFD Institute.

In particular, in the author's experience, the following approaches provide good results:

- Going to the *Gemba* – observe potential customers how they use the product – is useful for software products even more than for anything else. Software is more easily traceable than hardware, thus software products can easily be traced when used.

- One particular source of information is an ergonomic test – market preferences are more easily detectable when observing what software users are trying to do, than by asking them.

- Another very valuable source of information is building trackers into the software that count how often certain functions or use cases have been activated – care must be taken to get the user's agreement before collecting such information. If doing this, the recommended way is to explicitly ask the user for her or his collaboration before sending collected usage statistics back to the software supplier.

- Asking the product user has the additional advantage that the software product can collect qualified feedback – obviously, such build-in surveys are additional product features and use cases that need to be build into the product.

- Watching evolving technology is a very valuable source for potential market preferences as well – they might point into the future but when correlated with business value they allow reliable predictions.

- Another approach is with classical marketing. If markets are well defined and some customers willing to provide information, a sensing group or a market survey can help.

The preferred selection depends from the Deming Value Chain used for building the software product.

4.2 Using the IT Product Compass

An easy way of detecting unknown market preferences is using some standard model – e.g., the IT Product Compass published by G. Herzwurm and W. Pietsch, p.22 [HP09].

A basic set for selecting market preference topics for investigation in transfer functions consists in this example by the following:

- Development
    - Product Life Cycle status (new product, or refurbished…)
    - Target Definitions (target market, target customers, target problems)
    - Training Needs (depending upon roles, e.g., user, superuser, administrator, …)
    - Installation Ease (interoperability, interfaces, standards, constraints, …)

- Operations
    - Data Security (privacy, redundancy, disaster recovery, …)
    - Operational Ease (service dependency, autonomy, availability, …)
    - Administration (account management, login management, …)

- Application
  - Target Users (office-base, home-based, mobile, …)
  - Consultancy Needs (skills availability, …)
  - Helpdesk Support (channels, competency, response time, …)

- Service and Support
  - Incidence Management (non-conformities, maintenance, …)
  - Defect Prevention (release cycles, regression testing, …)
  - Preventive Maintenance (new feature management, …)
  - Corrective Maintenance (hotfixes, support interventions, …)

This set of standard market preferences topics serve for analyzing market share observations, if none more specific are available. It allows benchmarking software products that do not share common functionality.

4.3 Filling the Matrix

Having a set of potential market preferences is one problem solved; however, setting up the transfer function is not done yet. The transfer function is usually represented as a correlation matrix, i.e., a matrix whose cells indicate the strength of coupling between the respective **x** and **y** component. We have seen that methods exist to measure the matrix components.

Once candidates are selected for the market preference topics, support cases are the most valuable source of information for the **FN** → **UC, UC** → **MP** and **CtQ** → **MP** Transfer functions, see below in 4.4. Trackers, as explained in section 4.1, can also be used to fill in the correlation matrix, since **FN, UC** and **CtQ** are stable at least within one product release.

See [Fe10] for a suggestion how to measure Deming Chains such as **UC** → **MP** when developing software. Note that usually there is functionality in the product that does not contribute to any market preferences topic at all – such as housekeeping functions, security checks, etc., however, market preferences change fast.

4.4 Analyzing Support Cases

A wealth of information is available for most of the companies producing software: support calls. Support cases tell more about changing market preferences than anything else. Customers tell the support people what they are trying to do and cannot achieve, for some reason.

Support calls most often address the **FN** → **UC, UC** → **MP** or **CtQ** → **MP** Transfer function, sometimes also **MP** → **BO**, when users try to explain their business objectives to the help desk. Support data primarily fills the matrix. Collecting support case data successfully involves classifying them into the applicable Deming Process, train support personnel to ask the right questions, and set up the respective data collectors.

For most software product suppliers, this is most rewarding data source and probably the best investment into market research. With such data, trends towards new market preferences are detectable using techniques shown in section 2.5.

4.5 Using Traditional QFD Workshop Techniques

If measuring the matrix components is difficult, traditional QFD workshop techniques can help to "guess" the correlations factors if they aren't measurable. The Convergence Gap is helpful for such workshops as well since even if the accuracy of matrix correlation is uncertain, the uncertainty can be measured – and in fact, it must be measured.

4.6 Closing the Convergence Gap

If the New Lanchester Transfer Function shows a large Convergence Gap, it hints at the possibility that relevant Market Preferences remain undetected. As in example 2.5, additional topics can be constructed from the matrix correlation values observed.

Sometimes it helps reformulating the preference topic statement; sometimes it should be split into two for better modeling the buyer's behavior. The criterion is orthogonality, i.e., the preference topic statements should not be dependent from each other. In the competitive assessment, it should be possible to change one topic's evaluation without affecting others.

However, if this trial and error approach does not lead to tangible results, there is another check that can be performed. Let $\underline{\mathbf{y}}$ denote the observation; a rectangular AHP matrix must exists[5] that has $\underline{\mathbf{y}}$ as its result. To construct this matrix in practice is not difficult; however, the pair-wise comparison involved in its construction will point the team towards the relevant market preferences.

5 Experiences

GMC Software Technology, a world-class provider of personalized customer communication software (Direct mail, bills, statements, etc.) uses New Lanchester Theory to steer its 30% – 50% yearly growth and outperform its competition. New Lanchester Theory was used for selecting new features to be included in new releases. Using the controlled approach, it did not only allow to win market share, but also to avoid winning too much market share, in selecting new features such that GMC remained able to manage its growth.

[5] The AHP matrix is triangular; its sub-diagonal part being completed by reciprocal values. Such a matrix has Eigenvectors, see the Perron Theorem in [Sa03], and its application to Six Sigma in [Fe10].

Since 2008, GMC has become world leader in customer communications software, and the New Lanchester Strategy had being turned from the "Strategy of the Weak" to the "Strategy of the Strong", thus the need to be more innovative than the competition and block possible market niches for new competitors.

Some details of GMC's approach have been published in [Fe04]. Other experience reports are difficult to find, as New Lanchester Theory might be used in corporate strategic decisions that are not widely communicated.

6 Conclusions

Statistical methods, including the Eigenvector theory, open a wide range of application possibilities to product management and product improvement. Finding the best combination of traditional product marketing and Six Sigma approaches is not easy, and experiences are not widely shared.

References

[Ak91] Akao, Y.: Quality Function Deployment: Integrating Customer Requirements Into Product Design (1991)

[CSA03] Creveling, C.M., Slutsky, J.L., Antis, D.: Design for Six Sigma, Prentice Hall, New Jersey (2003)

[De86] Deming, W. E.: Out of the Crisis, Massachusetts Institute of Technology, Center for Advanced Engineering Study (1986)

[Fe04] Fehlmann, Th.: The Impact of Linear Algebra on QFD. In: International Journal of Quality & Reliability Management, Vol. 21 No. 9, pp. 83--96, Emerald, Bradford, UK (2004)

[Fe08] Fehlmann, Th.: New Lanchester Theory for Requirements Prioritization. In: Proceedings of the 2nd International Workshop on Software Product Management, Barcelona, Spain (2008)

[Fe10] Fehlmann, Th.: Requirements Elicitation in Agile Software Development. In: Proceedings of the RePriCo'10 Workshop, ICB Research Report, Institute for Computer Science and Business Information Systems, University of Duisburg–Essen, Germany (2010)

[HS06] Herzwurm, G., Schockert, S.: What are the Best Practices of QFD? In: Transactions from the 12th Int. Symposium on Quality Function Deployment, Tokyo, Japan (2006)

[HP09] Herzwurm, G., Pietsch, W.: Management von IT-Produkten. dpunkt.verlag, Heidelberg, Germany (2009)

[JM08] Johnson, C.M., Mazur, G.: Value Based Product Development – Using QFD and AHP to Identify, Prioritize, and Align Key Customer Needs and Business Goals. In: Transactions from the 20th Symposium on Quality Function Deployment, Santa Fe, NM (2008)

[Sa03] Saaty, Th.L.: Decision-making with the AHP: Why is the principal eigenvector necessary. In: European Journal of Operational Research vol. 145, pp. 85--91, Elsevier Science B.V. (2003)

Wirtschaftlichkeitsbetrachtungen beim Einsatz von Cloud Computing

Sabrina Lamberth, Anette Weisbecker

Fraunhofer IAO
Nobelstraße 12
70569 Stuttgart
{sabrina.lamberth, anette.weisbecker}@iao.fraunhofer.de

Abstract: Cloud Computing gewinnt stetig an Bedeutung, wie am wachsenden Markt von Dienstleistungen in den Bereichen IT-Infrastruktur, Entwicklungsplattformen und Software erkennbar ist. Für Unternehmen ergeben sich daraus neue Möglichkeiten, ihren IT-Bedarf zu decken. Zentrales Bewertungskriterium ist hierbei der ökonomische Nutzen, der durch die Nutzung von Cloud-Angeboten im Vergleich zu Anschaffung und Betrieb unternehmenseigener IT entsteht. Für eine umfassende Betrachtung sind sowohl quantitative als auch qualitative Faktoren einzuschließen, was eine erweiterte Kosten-Nutzen-Analyse für Cloud Computing im Unternehmenseinsatz nahe legt.

1 Möglichkeiten des Einsatzes von Cloud Computing

Cloud Computing steht für die Nutzung von IT-Ressourcen über das Internet, die bedarfsgerecht abgerechnet werden [VR09], [BI09]. Dabei wird unterschieden zwischen Infrastrukturdiensten (Infrastructure-as-a-Service) wie Rechenleistung und Speicherplatz, Plattformen zur Entwicklung von IT-Diensten (Platform-as-a-Service) und Anwendungen (Software-as-a-Service). Für Unternehmen eröffnen sich mit Cloud Computing neue Möglichkeiten, ihren IT-Bedarf zu decken.

Durch die zahlreichen Geschäftsmodelle in Clouds und das zu Grunde liegende dynamische Nutzungsmodell [DL08] sind die Einsatzmöglichkeiten von Cloud Computing in Unternehmen vielfältig. Neben dem Szenario, jegliche Informationstechnik (IT) aus der Cloud zu beziehen, ist eine Kombination mit bereits vorhandenen IT-Ressourcen ebenso denkbar wie eine aufeinander aufbauende Nutzung von Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) und Software-as-a-Service (SaaS). Die grundlegende Frage besteht darin, welche Form eines Cloud-Einsatzes – sei es komplementär oder substitutiv zu Anschaffung und Betrieb unternehmenseigener IT – die langfristig wirtschaftlich vorteilhafteste Situation nach sich zieht. Kernkonstrukt, das zu beleuchten ist, ist daher der Nutzen (ökonomischer Vorteil), der durch den Einsatz von Cloud Computing im Unternehmen generiert werden kann. Für Unternehmen mit (neu hinzugekommenem) IT-Bedarf ergeben sich aus den Faktoren *bereits vorhandene IT-Kapazität* und *(neu) zu deckender IT-Bedarf* aus ökonomischer Sicht drei grundlegende Möglichkeiten, diesen zu decken:

(1) rein über Cloud-Ressourcen (substitutiv), (2) rein über Anschaffung und Betrieb unternehmenseigener Systeme oder (3) über eine Mischform aus Cloud- und unternehmenseigener Umgebung (hybrides Modell bzw. komplementärer Einsatz) [La10]. Für Unternehmen, die über einen Einsatz von Cloud Computing nachdenken, ergeben sich aus den Szenarien (1) bis (3) unternehmensspezifische Bewertungsfragestellungen. Die im Rahmen des Beitrags dargestellte "Erweiterte Nutzwertanalyse" soll es Unternehmen ermöglichen, den Mehrwert von Cloud Computing im Vergleich zu Anschaffung und Betrieb unternehmenseigener IT zu analysieren und ggf. daraus Schlussfolgerungen für das geeignete Mischverhältnis aus unternehmensinternen Systemen und Cloud-Ressourcen zu ziehen. Dabei fällt die Analyse für jedes Unternehmen bzw. jeden Bedarfsfall individuell aus. Bei einem hohen Bedarf an Rechenleistung und schwankenden Kapazitätsanforderungen stellen Cloud-Angebote durch die dynamische Anpassung der Abrechnung an den tatsächlichen Bedarf eine effiziente Alternative dar. Hierzu zählen insbesondere die Branchen Automobil- und Maschinenbau, Finanzen, Pharma und Medien sowie generell kleine und mittelständische Unternehmen bzw. Start-Ups, welche sich nicht auf IT spezialisiert haben [EM09].

Häufig genannte Ziele des Einsatzes von Cloud Computing sind Kosten- und Zeitvorteile [Re09], eine höhere Flexibilität auf operativer und strategischer Ebene (s. Abschnitt 3.1) sowie durch die Vorteile der Ressourcen-Virtualisierung [Th08] eine höhere Leistungsfähigkeit der Unternehmens-IT. Durch Fremdbezug von IT-Ressourcen aus der Cloud sind Kostensenkungen zu realisieren [EM09], sowohl in Bezug auf laufende Kosten als auch durch die Eliminierung von Erstinvestitionskosten. Auch Effekte auf die IT-Kostenstruktur sind zu erwarten, da bei einer Nutzung von Cloud-Diensten eine Verlagerung von fixen zu variablen Kosten erfolgt. Ausmaß und Beitrag dieser Vorteile zum Gesamtnutzen der von Unternehmen genutzten IT-Ressourcen sind im Einzelfall zu überprüfen. Aus diesem Grund sind Ansätze zu formulieren, welche eine Wirtschaftlichkeitsbetrachtung von Cloud Computing im Unternehmenseinsatz ermöglichen und den Vergleich zu Anschaffung und Betrieb unternehmenseigener Systeme unterstützen.

2 Existierende Arbeiten

2.1 Wirtschaftlichkeitsbetrachtungen

Neben klassischen Herangehensweisen bei der Wirtschaftlichkeitsbetrachtung von IT-Anschaffungen existieren Überlegungen, wie Cloud Computing im Vergleich zu Anschaffung und Betrieb unternehmenseigener IT ökonomisch bewertet werden kann. Kosten und Nutzen stellen die primären Wirtschaftlichkeitsfaktoren von IT-Lösungen [Br09] und damit auch den Kern von Wirtschaftlichkeitsbetrachtungen dar. Direkter Nutzen entsteht dabei aus Kostenvorteilen und ist am einfachsten zu ermitteln. Vorhandene Cloud-spezifische Ansätze konzentrieren sich auf den Kostenaspekt und überprüfen hauptsächlich den Kostenvorteil der Nutzung einer Cloud-Lösung gegenüber Anschaffung und Betrieb unternehmenseigener Systeme. In einem *Make-or-Buy*-Ansatz für Cloud Computing [EM09] wird der Fokus auf die Kosten gelegt.

Es wird davon ausgegangen, dass in bestimmten unternehmensspezifischen Ausgangssituationen der Fremdbezug von IT über die Cloud einen Kostenvorteil gegenüber der Eigenerstellung der IT-Leistung (Anschaffung und Betrieb unternehmenseigener IT-Systeme) aufweist. Dieses Modell ermöglicht eine rechnerische Einschätzung ob der Vorteilhaftigkeit eines Cloud-Einsatzes. Ein weiter ausdifferenziertes Rahmenwerk zum unternehmensindividuellen Vergleich von Cloud-Angeboten mit einer unternehmenseigenen IT-Variante bietet ein *TCO-Framework* [KN08]. Als Basis dieses ebenfalls rein kostenorientierten Ansatzes dient die Aufschlüsselung der IT-Kosten nach *Total Cost of Ownership* (TCO). Ergebnis dieser Gegenüberstellung von *Cloud Model* und einem Vergleichsmodell, was sich auf unternehmensinterne IT bezieht, ist ein absoluter Kostenvorteil über die gesamte Nutzungsdauer der betrachteten IT (unternehmensintern oder über einen Cloud-Anbieter bezogen).

Da Kosten als direkt und indirekt quantifizierbare Faktoren nur einen, wenn auch wichtigen, Teilaspekt von Wirtschaftlichkeitsbetrachtungen darstellen und auch qualitative Aspekte einfließen sollen, wird eine Erweiterung der reinen Kostenbetrachtung um qualitative Nutzeneffekte vorgeschlagen, um umfassende ökonomische Auswirkungen von Cloud-Angeboten im Unternehmenseinsatz abzubilden und mit unternehmenseigenen Alternativen zu vergleichen. Hierzu zählt die Bewertung *indirekt* monetär messbarer oder *intangibler* (nicht monetär quantifizierbarer) Nutzenbeiträge von Cloud Computing. Da diese jedoch langfristig einen bedeutenden Einfluss auf die wirtschaftliche Gesamtsituation und Entwicklung eines Unternehmens haben, ist eine Integration wichtig. Zu nennen sind hier qualitative Faktoren mit teilweise erheblicher strategischer Bedeutung wie z.B. Flexibilität durch die Beschaffenheit der zur Verfügung stehenden IT-Ressourcen, Leistungsfähigkeit durch Hochverfügbarkeit (betriebliche Kontinuität, Notfallwiederherstellung) [Th08] oder Nutzerzufriedenheit. Möglich ist eine Kombination kostenfokussierter Ansätze mit qualitativen Methoden der Wirtschaftlichkeitsbetrachtung. Des Weiteren können Überlegungen erfolgen, wie Rentabilitäts- und Effizienzbetrachtungen im Cloud-Fall möglichst flexibel integriert werden können; je nachdem, welche Zielgruppe im Unternehmen Informationen der Wirtschaftlichkeit zur Bewertung von Cloud-Angeboten benötigt oder möglichst schnell und umfassend zur Verfügung stellen kann.

2.2 Methoden der Wirtschaftlichkeitsbetrachtung

Da IT-Investitionen kostenintensiv und irreversibel sind und langfristige unternehmensweite Auswirkungen nach sich ziehen [Hi05], ist eine detaillierte Kosten-Nutzen-Betrachtung unabdinglich – mit dem Ziel des Alternativenvergleichs. Daher existieren bereits zahlreiche Methoden zur Wirtschaftlichkeitsbetrachtung für IT-Anschaffungen. Breite Akzeptanz findet in der Unternehmenspraxis die *Kapitalwertrechnung*, die *Payback-Methode* (Amortisations- bzw. Kapitalrückflussrechnung oder der *Return On Investment* (ROI). Für IT-Investitionen, bei denen ein Großteil des Nutzens nicht monetär quantifizierbar ist, wird einer Investitionsrechnung eine qualitative Methode vorgezogen und in vielen Fällen die *Nutzwertanalyse* eingesetzt [Hi05]. Zur Planung von IT-Projekten bzw. zur Simulation künftiger Kosten-Nutzen-Situationen eignet sich als zusätzliches Instrument z.B. die *Szenariotechnik*.

Die bereits etablierten Methoden der Wirtschaftlichkeitsbetrachtung sind auf das Bewertungsproblem eines Einsatzes von Cloud Computing anzupassen. Da die meisten Cloud-Angebote keine signifikanten Erstinvestitionskosten verursachen, werden klassische Investitionsrechnungen auf den ersten Blick hinfällig und eine generelle Überlegenheit eines Cloud-Einsatzes gegenüber unternehmenseigener IT hinsichtlich der Rentabilität ist anzunehmen. Werden Cloud-Angebote allerdings ergänzend (komplementär) zu bereits bestehenden oder anzuschaffenden unternehmensinternen IT-Komponenten eingesetzt und bestehen hierbei mehrere zu vergleichende Kombinationsmöglichkeiten, macht eine Rentabilitätsbetrachtung wiederum Sinn. Aus diesem Grund wurde in [La10] ein kombinierter Kosten-Nutzen-Bewertungsansatz für Cloud Computing entwickelt, welcher im Nachfolgenden beschrieben und erläutert wird. Die Herangehensweise integriert gleichsam harte und weiche Faktoren, kann flexibel auf das unternehmensspezifische Bewertungsproblem angepasst werden und ist für die ganz unterschiedlichen Kombinationsmöglichkeiten unternehmenseigener mit Cloud-IT-Ressourcen geeignet.

3 Erweiterte Nutzwertanalyse

3.1 Vorgehen / Methode

Da eine reine Kostenbetrachtung als zu einseitig anzusehen ist, andererseits eine rein qualitative Vorgehensweise dem Rationalitätsprinzip der Wirtschaftlichkeit widerspricht, das den Einsatz mengen- und wertmäßiger Bezugsgrößen (sog. harte Faktoren) impliziert, wird hier ein kombinierter Kosten-Nutzen-Bewertungsansatz vorgestellt, welcher möglichst viele harte Faktoren mit einbezieht, gleichzeitig jedoch auch die nicht minder wichtigen weichen (qualitativen) Faktoren integriert. Diese Vorgehensweise ist in der Praxis üblich, wenn der Bewertungsgegenstand viele verschiedene Dimensionen aufweist, wie es bei Cloud-Diensten der Fall ist [WF08]. Den Hauptrahmen des Ansatzes bildet die *Nutzwertanalyse*, da sie unternehmens- bzw. fallspezifisch angepasst werden kann und durch die subjektive Vorgehensweise der Bewertung [Hi05] ein einfach handhabbares Instrumentarium darstellt. Es handelt sich hierbei nicht wie bei den Investitionsrechnungen um ein rechnerisches, sondern ein qualitatives Verfahren, bei dem zunächst Zielkriterien festgelegt werden, an Hand derer Alternativen bewertet werden sollen, sowie Gewichtungsfaktoren, welche die Bedeutsamkeit der Kriterien abbilden. Die (subjektive) Bewertung der Alternativen erfolgt dann über die Vergabe von Punkten (gewichtet: Teilnutzenwerte), der Vergleich über die Summe der gewichteten Punktwerte (Gesamtnutzenwert) [Ho00], [GM06]. Da die Hauptmethode allerdings um einen detaillierten Kostenvergleich von Cloud- und unternehmenseigenen IT-Alternativen sowie – falls möglich – um Kapitalwerte oder Rentabilitätskennzahlen ergänzt wird, erhält der Ansatz die Bezeichnung *Erweiterte Nutzwertanalyse*.

Für das Design der Erweiterten Nutzwertanalyse werden zunächst Zielkriterien festgelegt, welche sich auf die Faktoren *Kosten* und *Nutzen* beziehen (s. Tabelle 1). Die Aufschlüsselung der IT-Kosten zur Aufstellung der Cashflows erfolgt nach den Gesamtkosten der IT über die gesamte Nutzungsdauer bzw. den gesamten Betrachtungszeitraum, um nach Art und Weise des TCO-Frameworks in [KN08] (s. Abschnitt 2.1) eine detaillierte Sicht auf die Kosten von Cloud-Diensten und im Vergleich dazu bei Anschaffung und Betrieb unternehmenseigener IT zu erhalten.

Für die Kostenanalyse sind zunächst einzelne Kostenrechnungen für Cloud- und unternehmenseigene IT-Alternative durchzuführen. Im Anschluss daran ist der jeweilige Nutzen – soweit monetär quantifizierbar – zu ermitteln. Ergebnis der Kosten-Nutzen-Relationen sind Kennzahlen (z.B. ROI, Kapitalwert), welche im Rahmen der Erweiterten Nutzwertanalyse *qualitativ* bewertet werden. Diese stellen die zahlenmäßige Basis der Nutzwertanalyse dar. Zusätzlich werden indirekt monetär bewertbare sowie intangible Nutzeneffekte nach Verfahren der Nutzwertanalyse mit Punkten bewertet. Hierzu zählen bspw. Erhöhung operativer [Br09] und strategischer Flexibilität [Hu08], Zeitvorteile [ID09], [Ya09], Verbesserung der Kontinuität von Geschäftsprozessen (z.B. durch Virtualisierung) [Th08] oder Mitarbeiterzufriedenheit und damit Produktivität [AH09]. Insbesondere die strategische Flexibilität ist unter die intangiblen Nutzeneffekte zu fassen, da die Ergebnisse nicht monetär messbar sind und häufig erst auf lange Sicht zu Tage treten. Dennoch haben sie langfristig eine Bedeutung für die wirtschaftliche Situation von Unternehmen [Po99]. Hierunter fallen Möglichkeiten zum Wachstum von Unternehmen oder zu Aufschub, Abbruch oder Änderungen von ursprünglichen Plänen oder Projekten [Hu08]. Die grobe Vorgehensweise in der Praxis – *Kostenanalyse, Analyse direkten Nutzens, Ermittlung von Kosten-Nutzen-Relationen (Kennzahlen), qualitative Bewertung der Kennzahlen, Analyse und Bewertung qualitativen Nutzens (nicht direkt monetär quantifizierbarer und intangibler Nutzen)* – ist in Abbildung 1 dargestellt.

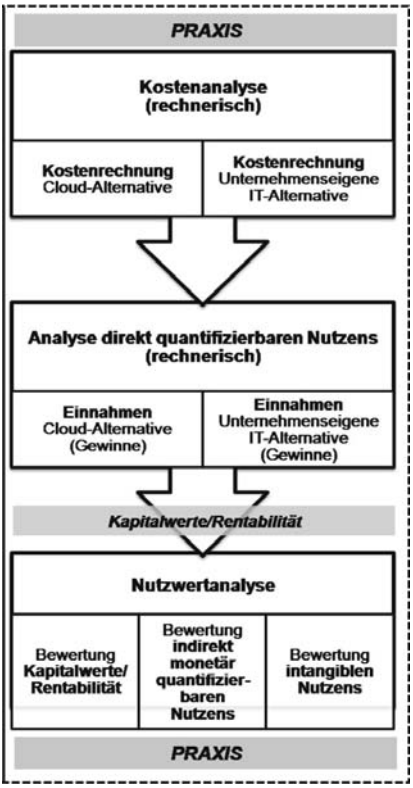

Abbildung 1: Kombinierter Kosten-Nutzen-Bewertungsansatz als Rahmenwerk für eine Erweiterte Nutzwertanalyse (Ausschnitt entnommen aus [La10])

Zusätzlich zu den Kosten- und Nutzenkriterien sind Kriterien zu technischer und rechtlicher Sicherheit der IT-Ressourcen in die Nutzwertanalyse einzuschließen[1]. Dies liegt in der Bedeutung für Unternehmen zusammen, da Cloud-spezifische Risiken zentrale Hinderungsgründe für Unternehmensverantwortliche darstellen, Cloud Computing einzusetzen, ihre Daten in die Cloud und damit in die Hände eines externen Anbieters zu geben. Neben der Angst vor Kontrollverlust über Unternehmensdaten spielen hier die Datensicherheit, die Trennung von Prozessen auf gemeinsam genutzten virtualisierten Ressourcen, der Datenschutz sowie Defizite an Transparenz bezüglich Anbieter-Nutzer-Verhältnisse, Anbieter-Aktivitäten und Preismodelle [CP10], [EN09] eine Rolle. Aber auch Bedenken der Bindung an einen bestimmten Anbieter (sog. Lock-In) oder in Bezug auf Kompatibilität mit der Unternehmens-Compliance [EN09], [CS09] spielen eine wichtige Rolle bei der Auswahl und Bewertung von Cloud-Diensten. In Tabelle 1 ist ein Kriterienkatalog dargestellt, der als Grundlage für die Erweiterte Nutzwertanalyse dient und wirtschaftliche ebenso wie risikobezogene Aspekte umfasst. Dieser ist in der Praxis an Unternehmensprioritäten anzupassen. Steht bspw. IT-Sicherheit an erster Stelle, sieht die Ausgestaltung der Kriterien anders aus, als wenn der Fokus auf dem IT-Betrieb liegt.

Kategorie	*Kriterium*
Übergeordnetes Kriterium	Erfüllung der Unternehmensanforderungen (Spezifität des Cloud-Dienstes)
Direkt monetärer Nutzenbeitrag des Cloud-Dienstes	Rentabilität (ROI)
	Kostenstruktur (fixe/variable Kosten)
	Rentabilität von IT-Sicherheit (ROSI)
Indirekt monetär messbarer Nutzenbeitrag des Cloud-Dienstes	Operative Flexibilität
	Effiziente Gestaltung von Prozessen
	Leistungsfähigkeit der Unternehmens-IT (Qualität)
	Flexibilität des Personaleinsatzes
Intangibler (qualitativer) Nutzenbeitrag des Cloud-Dienstes	Strategische Flexibilität auf Unternehmensebene
	Strategische Flexibilität in der IT (Einführung/Erweiterung von IT im Unternehmen)
	Konzentration auf das Kerngeschäft
	Usability (Beitrag zu Mitarbeiterzufriedenheit und -Wohlbefinden)
Risiken des Cloud-Dienstes (Qualitative Kosten)	Abhängigkeit
	Risiken in Bezug auf Datensensitivität
	Risiken in Bezug auf Kontrollverlust
	Risiken in Bezug auf Unternehmensprozesse
	Einhaltung der Compliance
	Handlungsflexibilität im Worst-Case

Tabelle 1: Zusammenfassender Kriterienkatalog für eine erweiterte Nutzwertbetrachtung von Cloud-Diensten

In Anlehnung an das unternehmensspezifische IT-Kennzahlensystem, das für ein systematisches IT-Controlling idealer Weise im Unternehmen vorliegt [Kü07], kann z.B. detailliert überprüft werden, ob sich bestimmte IT-Bereiche durch einen Einsatz von Cloud-Ressourcen effizienter gestalten lassen. Analysekategorien in weiterer Ausdifferenzierung können z.B. technische Infrastruktur, Software und Systemstruktur (Applikationen) sowie IT-Personal [Kü07], [BR96] sein.

[1] Vgl. hierzu den Abschnitt zur Diskussion.

Gesamtkosten (Cashflow-Analyse)[1]									Spezifika Cloud-Dienst ggü. unternehmenseigener IT	
	Periode (bspw. in Jahren)								Cloud-Alternative	traditionelle IT
Kosten	**0**	**1**	**2**	**3**	**…**	**10**	**Gesamt**	**Barwert**		
DIREKTE KOSTEN										
Externe Kosten aus On-Demand-Nutzung										
Kosten für Rechenleistung[2]									✓	x [3]
Kosten für Bandbreite[2]									✓	x [3]
Kosten für Speicher[2]									✓	x [3]
IT-Investitionskosten										
Hardware (z.B. Zentralrechner, PC)									✓ [4]	✓
Software (z.B. Betriebssystem, Bürosoftware)									✓ [5]	✓
[Implementierungs- und Designkosten]									x	✓
Schulungskosten									✓	✓
Kommunikationskosten									✓	✓
Laufende Kosten (Betrieb, Prozesse)										
Energiekosten									✓	✓
[Hosting]									x	✓
[Jährliche Wartung und Weiterentwicklung]									x	✓
Administration und Support									✓	✓
Kosten für Dienstleistungen									✓	✓
IT-Personalkosten									✓	✓
INDIREKTE KOSTEN										
Opportunitätskosten durch End-User-Operations[6]									✓	✓
Ausfallzeit-bezogene Kosten[7]									✓	✓
GESAMTKOSTEN DER IT										

Tabelle 2: Kostenanalyse-Schema eines Cloud-Angebot im Vergleich zu unternehmenseigener IT

Legende zu Tabelle 2:
[1]ohne Risikobereinigung
[2]nutzungsabhängig, daher variabel (Annahme von Durchschnitts- bzw. Erfahrungswerten notwendig)
[3]externe IT-Kosten in Zshg. mit unternehmenseigener IT: in Investitions- und Betriebskosten enthalten
[4]Basisausstattung Arbeitsplatz-PCs, Internetanschluss etc.
[5]aus SaaS-Angebot (z.B. Lizenzen)
[6]z.B. mangelndes Know-How (unternehmenseigene IT) oder Unerfahrenheit mit Cloud-Technologien
[7]Ausfall unternehmenseigener Server oder Cloud-Serviceausfälle
⬚ Kernschema der Kosten-Cashflows
[…] Verlagerung bei reiner Nutzung von Cloud-Angeboten, da Zuordnung zu Kosten aus On-Demand-Nutzung
✓ relevant
x nicht relevant

Die Kostenaufschlüsselung nach TCO erfordert eine Ermittlung *direkter* und *indirekter* Kosten. Zu direkten Kosten gehören Sach- und Personalkosten, welche unmittelbar im Zusammenhang mit der IT-Infrastruktur, sowohl in Form von Erstinvestitionen als auch während der Betriebsphase anfallen [Br09]. Bei der Betrachtung von Cloud-Angeboten sind Kosten aus der Nutzung des Cloud-Dienstes relevant: Kosten für Rechenleistung, Bandbreite und Speicher [Cu09], [Am10]. Da diese verbrauchsabhängig sind und Anschaffungskosten für physische Hardware entfallen, erfolgt eine Verlagerung von fixen zu variablen Kosten. Die Schwierigkeit bei der Prognose von Cloud-Kosten besteht darin, dass der IT-Bedarf genau geschätzt werden muss. Hier liegt eine doppelte Problematik, die zu Schätzungenauigkeiten führen kann: *(1) Der in der Zukunft liegende Kapazitätsbedarf muss exakt geschätzt werden, damit sich ein möglicher Kostenvorteil durch bedarfsgenaue Abrechnung im Vergleich zu Anschaffung und Betrieb unternehmenseigener IT in der Höhe direkter Kosten niederschlägt und (2) vielen Cloud-Angeboten liegen nicht-lineare Preismodelle zu Grunde (Sensitive Preise [Cu09]), die sich der angefragten Leistung und Menge der Kapazität flexibel anpassen [La10]*. Sensitive Preise bestehen bei nicht-linearen Preismodellen, wenn z.B. Preisstaffelungen in Abhängigkeit vom IT-Bedarf vorliegen. Zur Lösung des Schätzproblems wird ein zweistufiges Verfahren vorgeschlagen, bei dem im ersten Schritt mit Hilfe der *Szenariotechnik* die Schätzung eines Prognose-, Minimal- und Maximalwertes des *IT-Bedarfs* vorgenommen wird und erst im zweiten Schritt die Schätzung eines Prognose-, Minimal- und Maximalwertes der *Kosten* erfolgt. Im Übergang vom ersten zum zweiten Schritt wird eine *Sensitivitätsanalyse* zur Ermittlung der anbieterseitigen Preisstufen durchgeführt, falls Sensitive Preise vorliegen. Im Anschluß daran sind für den Alternativenvergleich die indirekten Kosten zu ermitteln, welche eine Aufsummierung des Werteverzehrs darstellen [WH00]. Sowohl für direkte als auch für indirekte Kosten sind Risikobereinigungen (Risikoaufschläge) durchzuführen, welche sich auf drei Werte (Prognose-, Minimal- und Maximalwert) stützen. Diese Risikowirkung kann bspw. über ein kombiniertes Verfahren aus arithmetischem Mittel der angenommenen Werte und Standardabweichung vorgenommen werden [Fo07]. Auch bei der Ermittlung von Einnahmen (monetärer Nutzen der Alternativen) sind Risikobereinigungen durchzuführen, die als Risikoabschläge von den Prognosewerten abzuziehen sind [BM07].

Tabelle 2 zeigt ein Kostenanalyse-Schema, welches gleichzeitig die Unterschiede in den entstehenden Kosten bei Cloud-Diensten und unternehmenseigener IT aufzeigt; insbesondere die Verschiebung in der Kostenstruktur (fixe zu variable Kosten).

Erweiterte Nutzwertanalyse		**Entscheidungsalternativen**			
		Substitutiver Einsatz von Cloud Computing (Vergleich)		**Komplementärer Einsatz von Cloud Computing (Mischmodell)**	
Entscheidungskriterien	Gewichtungsfaktor	Punktevergleich (Tendenz)	Gewichtete Punktzahl	Punkte	Gewichtete Punktzahl
Direkter Nutzenbeitrag (monetär bewertet: Wirtschaftlichkeit i.e.S.)					
Rentabilität[1] (ROI)					
Kostenstruktur (fixe/variable Kosten)					
Indirekter Nutzenbeitrag (qualitativ bewertet)					
Operative Flexibilität					
Effiziente Gestaltung von Prozessen					
Leistungsfähigkeit der Unternehmens-IT (Qualität)					
Flexibilität des Personaleinsatzes					
Intangibler (qualitativer) Nutzen					
Strategische Flexibilität (gesamtes Unternehmen)					
Strategische Flexibilität (Einführung/Erweiterung von IT im Unternehmen)					
Konzentration auf Kerngeschäft					
Mitarbeiterzufriedenheit (Usability)					
Technische und rechtliche Sicherheit					
Rentabilität von IT-Sicherheit (ROSI)					
Abhängigkeit					
Risiken in Bezug auf Datensensitivität[2]					
Risiken in Bezug auf Kontrollvelust[3]					
Risiken in Bezug auf Unternehmensprozesse[4]					
Einhaltung der Compliance					
Handlungsflexibilität im Worst-Case[5]					
Gesamt (Tendenz)					

Tabelle 3: Beispiel für eine Erweiterte Nutzwertanalyse-Matrix

Legende zu Tabelle 3:
[1]Hier kann alternativ auch der Kapitalwert bewertet werden.
[2]Engstellen bei der Sicherstellung von Vertraulichkeit und Integrität
[3]Cloud: Risikowert umso geringer, je eher Möglichkeiten zur Durchführung von Audits bestehen oder je ausführlicher (unternehmensspezifischer) das SLA gestaltet werden kann (Transparenz, Monitoring)
[4]Indikator: technische Verfügbarkeit
[5]Cloud: z.B. Datenmigration bei Anbieterinsolvenz bzw. –wechsel; bei unternehmenseigener IT: z.B. Datenverlust durch Viren oder Eindringlinge
*Hybride Form aus Cloud- und unternehmenseigenen IT-Ressourcen; es gilt: je größer der Anteil an Cloud Computing im Mischmodell, desto eher gleichen sich die Punktwerte des komplementären Einsatzes von Cloud Computing in der Nutzwertanalyse an die eines substitutiven Einsatzes an.

Nach der Festlegung relevanter Bewertungskriterien (Kosten, Nutzen, Risiken), den einzelnen Kosten- und Nutzenanalysen und der Ermittlung aussagekräftiger Kennzahlen sowie der Gewichtung der einzelnen Kriterien erfolgt zusammenfassend die qualitative Bewertung der Ergebnisse. Das Endergebnis ist in einer Nutzwertanalyse-Matrix festzuhalten, die eine Punktbewertung sowie Gewichtungsfaktoren für die Einzelkriterien enthält (s. Tabelle 3).

3.2 Einbettung der Erweiterten Nutzwertanalyse in den Auswahlprozess von Cloud-Diensten

Die Bewertung der Wirtschaftlichkeit stellt einen bedeutenden Teilschritt in der Übernahme von Cloud-Diensten in Unternehmen dar. Sie ist allerdings einer Vorauswahl *geeigneter* Cloud-Angebote anzuschließen. Die Eignung ist hierbei über Anforderungsanalysen zu ermitteln und kann als dreistufiges Verfahren angelegt werden. Auf Stufe I wird überprüft, welche Cloud-Dienste den benötigten technischen Anforderungen gerecht werden und entsprechende Spezifität aufweisen. Auf Grund der Bedeutung sicherheitstechnischer und rechtlicher Anforderungen sowie der Kontinuität der Dienstgüte [L10], [EN09] ist auf Stufe II ein Kriterienkatalog zu Verfügbarkeit und Sicherheit für eingesetzte Cloud-Dienste zu formulieren und eine Risikobewertung der auf Stufe I vorausgewählten Dienste vorzunehmen. Hier ist unter anderem festzulegen, ob der Cloud-Dienst der Sensitivität betroffener Unternehmensdaten gerecht wird, d.h. ob die Unternehmensdaten „Cloud-geeignet“ sind. Nach diesem Filter-Schritt kann auf Stufe III die Wirtschaftlichkeitsanalyse erfolgen. In Orientierung an den in Tabelle 1 vorgestellten Kriterien ist gegebenenfalls ein unternehmensspezifischer, verfeinerter Kriterienkatalog zu entwickeln und an Hand dessen eine Erweiterte Nutzwertanalyse durchzuführen, wie sie im vorangegangenen Abschnitt vorgestellt wurde. Der Kriterienkatalog ergibt sich aus den auf Stufe I, II und III ermittelten Kriterien zur Spezifität, Sicherheit und Wirtschaftlichkeit. Nach der Erweiterten Nutzwertanalyse, im Rahmen derer eine Nutzwertanalyse-Matrix (vgl. Tabelle 3) mit den o.g. Kriterien (Zielkriterien) erzeugt wird, folgt der Alternativenvergleich von Cloud-Diensten, unternehmenseigenen IT-Alternativen und Mischmodellen. Hieraus ergibt sich eine Endauswahl einer oder mehrere IT-Alternativen - eines einzigen Cloud-Anbieters, verschiedener Dienste oder Kombinationen mit unternehmenseigener IT. Werden Cloud-Dienste gewählt, sind diese in Zusammenarbeit mit dem Anbieter zu konfigurieren.

Zu Garantie der Dienstgüte wird ein Service Level Agreement (SLA) formuliert [Re09]; nicht zuletzt um Risiken und Bedenken des Kontrollverlustes über Unternehmensdaten zu begegnen. Mit dem SLA sollten risikobezogene Maßnahmen allerdings nicht abgeschlossen sein. Vielmehr können über ein systematisches Controlling und Monitoring des Cloud-Dienstes bzw. –Anbieters über die gesamte Nutzungsdauer Cloud-bezogene Risiken eingedämmt werden.

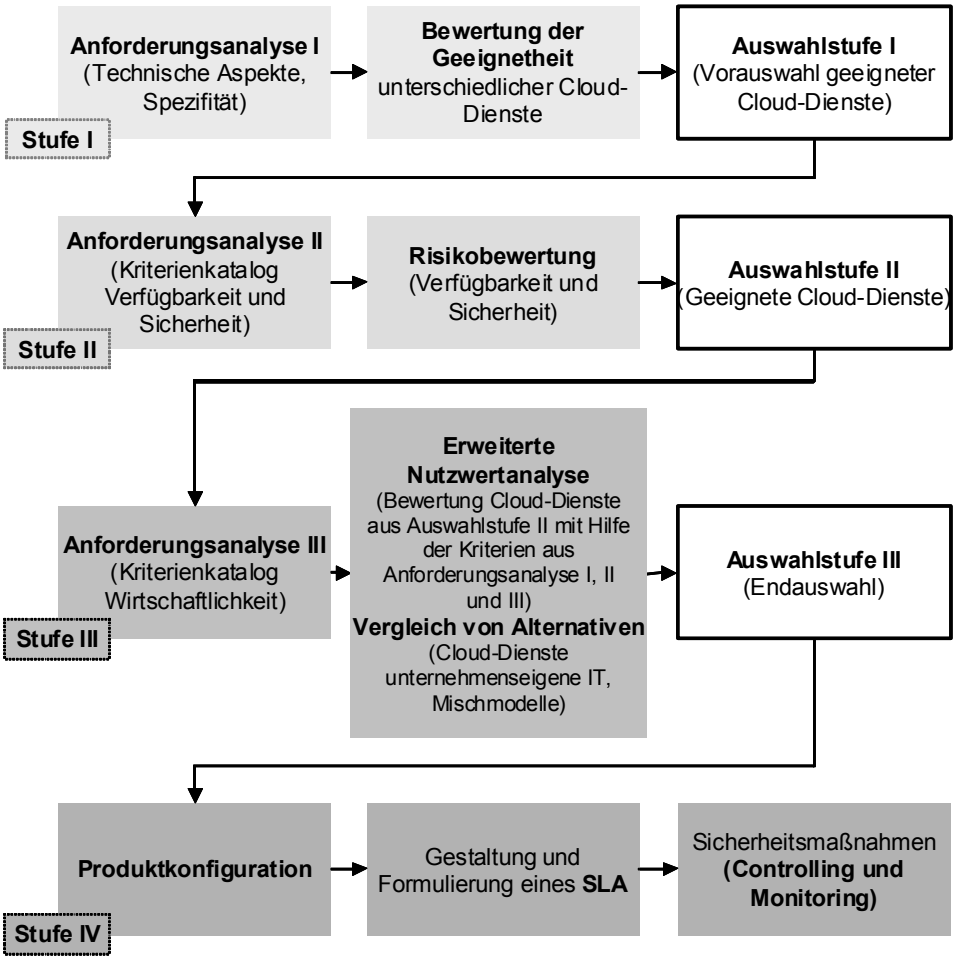

Abbildung 2: Vorgehensmodell zur Auswahl und Bewertung von Cloud-Diensten

4 Diskussion und Ausblick

Mit der Erweiterten Nutzwertanalyse wurde ein Ansatz vorgestellt, der sowohl harte als auch weiche Faktoren integriert und unternehmensspezifisch angepasst werden kann. Es ist jedoch festzustellen, dass es sich beim Bewertungsverfahren zur Ermittlung der Wirtschaftlichkeit unterschiedlicher Cloud-Angebote bzw. Kombinationen mit unternehmenseigener IT um eine komplexe und aufwändige Verfahrensweise handelt; nicht zuletzt, weil die Aufschlüsselung der IT-Kosten in indirekte und direkte Kosten durch mögliche Schätzungenauigkeiten und Verfahren zur Risikobereinigung insbesondere bei der Kalkulation von Cloud-Kosten Engstellen aufweist. Wie es in der Vorgehensweise bei der Auswahl eines Cloud-Dienstes (Abbildung 2) nur angedeutet wurde, spielen zudem neben der Wirtschaftlichkeit des Einsatzes von Cloud Computing in Unternehmen IT-Sicherheit und rechtliche Aspekte eine zentrale Rolle. Auch die Angst der Cloud-Kundenunternehmen vor Kontrollverlust über ihre Daten und Abhängigkeit vom Cloud-Anbieter (Lock-In) lässt sich rein durch wirtschaftliche Vorteile, wenn auch erhebliche, nicht ausgleichen. Da diese Punkte essenziell sind, macht es Sinn, eine Risikoanalyse und vorzeitigem Ausschluss nicht geeigneter Cloud-Dienste oder -Dienstleister im Vorfeld einer Wirtschaftlichkeitsbetrachtung durchzuführen. Zusätzlich können Sicherheits- und rechtliche Kriterien in die Erweiterte Nutzwertanalyse als Zielkriterien einfließen, wenn Daten zwar Cloud-geeignet sind, jedoch bei verschiedenen Cloud-Anbietern oder -Lösungen sehr unterschiedliche Sicherheits- und Servicelevels gegeben sind. Die Wirtschaftlichkeitsanalyse steht somit nie allein und Kosten-Nutzen-Kriterien in Bezug auf Cloud Computing sind immer in einen unternehmensspezifischen Gesamt-Anforderungskatalog einzubetten. Der vorgestellte Ansatz dient als grobes Rahmenwerk und lebt von der unternehmensindividuellen Anpassung. Die Vorgehensweise kann somit nicht pauschalisiert werden. Vorteile sind darin zu sehen, dass auch bei nicht ausreichender Zahlenbasis eine brauchbare Aussage über Eignung und Wirtschaftlichkeit eines Einsatzes von Cloud Computing gemacht werden kann, soweit ein ausführlicher Zielkriterienkatalog – optimaler Weise abgeleitet vom unternehmensindividuellen IT-Kennzahlensystem zum Controlling der IT-Effizienz – formuliert und der Nutzwertanalyse zu Grunde gelegt wird.

Literaturverzeichnis

[AH07] Abele, P.; Hurtienne, J.; Prümper, J.: Usability Management bei SAP-Projekten: Grundlagen – Vorgehen – Methoden. Vieweg+Teubner, Wiesbaden, 2007.

[Am10] Amazon Web Services (AWS): Elastic Compute Cloud (EC2). Preisgestaltung. [Online] 2010. http://aws.amazon.com/de/ec2/pricing/; Abruf am 16. Juni 2010.

[BI09] BITKOM (Hrsg.): Cloud Computing - Evolution in der Technik, Revolution im Business - BITKOM-Leitfaden. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V., Berlin, 2009.

[BM07] Bundesministerium des Innern (Hrsg.): WiBe 4.1 - Empfehlung zur Durchführung von Wirtschaftlichkeitsbetrachtungen in der Bundesverwaltung, insbesondere beim Einsatz in der IT. Bundesministerium des Innern, Referat IT 2 (KBSt), Berlin, 2007.

[Br09] Brugger, R.: Der IT Business Case: Kosten erfassen und analysieren - Nutzen erkennen und quantifizieren. Springer Verlag, Berlin, 2009.

[BR96] Baumöl, U., Reichmann, T.: zit. in [Kü07]. IT-Kennzahlensystem. 1996.

[CP10] Chen, Y.; Paxson, V.; Katz, R.H.: What's New About Cloud Computing Security? University of Berkeley, California, [Online] 2010. http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-5.pdf; Abruf am 8. März 2010.

[CS09] Cloud Security Alliance: Security Guidance for Critical Areas of Focus in Cloud Computing. Cloud Security Alliance. [Online] April 2009. http://www.cloudsecurityalliance.org/csaguide.pdf; Abruf am 2. Januar 2010.

[Cu09] Cumulux (Hrsg.): Demystifying Azure Pricing. Cloud Computing - Business Impact Series. [Online 2009]. http://www.cumulux.com/Azure%20TCO%20-%20Part%201.pdf; Abruf am 18. Januar 2010.

[DL08] Doeffinger, D.; Lutz, F.; Hammermann, D.: Dynamic Services for SAP® Applications. In: Jacob, O. (Hrsg.): ERP Value: Signifikante Vorteile mit ERP-Systemen. Springer-Verlag, Berlin/Heidelberg, 2008; S. 103-115

[EM09] Eymann, T; Matros, R.; von Zydtwyck, N.: Make-or-Buy im Cloud Computing - Ein entscheidungsorientiertes Modell für den Bezug von Amazon Web Services. Bayreuther Arbeitspapiere zur Wirtschaftsinformatik. [Online] Mai 2009. http://opus.ub.uni-bayreuth.de/volltexte/2009/552/pdf/Paper_45.pdf; Abruf am 7. Januar 2010.

[EN09] ENISA (Hrsg.): Cloud Computing. Benefits, risks and recommendations for information security. European Network and Information Security Agency, 2009.

[Fo07] Forrester Consulting (Hrsg.): Total Economic Impact™-Studie zu Avaya IP-Telefonielösungen in einer Zweigstellenumgebung. Unternehmensübergreifende Analyse, Forrester Consulting, 2007.

[GM06] Gadatsch, A.; Mayer, E.: Masterkurs IT-Controlling: Grundlagen und Praxis – IT-Kosten und Leistungsrechnung – Deckungsbeitrags- und Prozesskostenrechnung – Target Costing. 3. Auflage. Vieweg+Teubner, Wiesbaden, 2006.

[Hi05] Hirschmeier, M.: Wirtschaftlichkeitsanalysen für IT-Investitionen. Verlag für Wissenschaft und Kultur Dr. Stein & Brokamp KG, Berlin, 2005.

[Ho00] Hoffmeister, W.: Investitionsrechnung und Nutzwertanalyse. Kohlhammer, Stuttgart/Berlin/Köln, 2000.

[Hu08] Hungenberg, H.: Strategisches Management in Unternehmen. Ziele - Prozesse - Verfahren. 5. Auflage. Gabler-Verlag, Wiesbaden, 2008.

[ID09] IDC: zit. in [Ku09]: Cloud Computing: Kosten-Nutzen-Analyse. In: Computerworld.ch. [Online] 23. November 2009. http://www.computerworld.ch/aktuell/news/49776; Abruf am 20. Januar 2010.

[KN08] Klems, M; Nimis, J; Tai, S.: Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing. FZI Forschungszentrum Informatik Karlsruhe. Karlsruhe: Universität Karlsruhe. [Online] 2008. http://www.cca08.org/papers/Poster12-Markus-Klems.pdf; Abruf am 22. Januar 2010.

[Ku09] Kurzidim, M.: Cloud Computing: Kosten-Nutzen-Analyse. In: Computerworld.ch. [Online] 23. November 2009. http://www.computerworld.ch/aktuell/news/49776; Abruf am 20. Januar 2010.

[Kü07] Kütz, M.: Kennzahlen in der IT. Werkzeuge für Controlling und Management. 2. Auflage. dpunkt.verlag, Heidelberg, 2007.

[La10] Lamberth, S.: Kriterien und Wirtschaftlichkeitsbetrachtung für den Einsatz von Cloud Computing in Unternehmen – Fraunhofer Institut für Arbeitswirtschaft und Organisation IAO, Stuttgart, 2010.

[LW09] Lamberth, S.; Weisbecker, A.; Falkner, J.; Spath, D.: Cloud Computing: Begriff, Geschäftsmodelle, Chancen und Herausforderungen. In: Cloud Computing: Tagungsband des Stuttgarter Softwaretechnik Forums 2009, 01. Dezember 2009. Fraunhofer Verlag, Stuttgart, 2009.

[Po99] Porter, M.: Wettbewerb und Strategie. Econ Verlag, München, 1999; S. 45-82

[Re09] Reese, G.: Cloud Application Architectures. O'Reilly Media, Inc., Sebastopol, 2009.

[Th08] Thorns, F.: Das Virtualisierungs-Buch. Konzepte, Techniken und Lösungen. 2. Auflage. C&L Computer und Literaturverlag, Böblingen, 2009.

[VR09] Vaquero, L., Rodero-Merio, L., Caceres, J., Lindner, M.: A Break in the Clouds: Towards a Cloud Definition. ACM SIGCOMM Computer Communications Review, Vol. 39, No. 1, January 2009, pp. 50-55

[WF08] Weisbecker, A.; Falkner, J; Strauß, O.: Fraunhofer Enterprise Grids: Grid Check. Fraunhofer IRB Verlag, Stuttgart, 2008.

[WH00] Wild, M; Herges, S.: Total Cost of Ownership (TCO) - Ein Überblick. In: Abeitspapiere WI. Lehrstuhl für Allg. BWL und Wirtschaftsinformatik. Johannes-Gutenberg-Universität, Mainz, 2000.

[Ya09] Yara, P.; Ramachandran, R.; Balasubramanian, G.; Muthuswamy, K.; Chandrasekar, D.: Global Software Development with Cloud Platforms. In: Gotel, O.; Mathai, J.; Meyer, B.: Software Engineering Approaches for Offshore and Outsourced Development. Third International Conference, SEAFOOD 2009. Zurich, Switzerland, July 2009. Proceedings. Springer-Verlag, Berlin/Heidelberg, 2009.

Erfolgs- und Misserfolgsfaktoren der Anwendungsentwicklung – eine aktuelle empirische Überprüfung und Einordnung

Dipl.-Inform. Heiner Merz, Dipl.-Kfm. Michael Stewen

Lehrstuhl Wirtschaftsinformatik II, Abteilung VIII des
Betriebswirtschaftlichen Instituts der Universität Stuttgart
Keplerstr. 17
70174 Stuttgart
heiner.merz@bwi.uni-stuttgart.de

Abstract: Dieser Konferenzbeitrag stellt Erfolgsfaktoren und Misserfolgsfaktoren der Anwendungsentwicklung dar. Grundlage hierfür bilden empirische Untersuchungen des Jahres 2009. Diese Erfolgs- und Misserfolgsfaktoren werden anhand der Klassifikation der ISO/IEC 12207 und ISO/IEC 15288 gruppiert und diskutiert.

1 Hintergrund und Basis des vorliegenden Konferenzbeitrags

Dieses Paper basiert auf empirischen Untersuchungen, welche im Sommer 2009 vom Lehrstuhl Wirtschaftsinformatik II der Universität Stuttgart durchgeführt wurden. Vorbereitend der Primärdatenerhebung in Form von Experteninterviews erfolgten Sekundärdatenanalysen, die zur Aufstellung von Erfolgs- und Misserfolgsfaktoren[1] in Softwareentwicklungsprojekten führten. Deren aktuelle empirische Prüfung und Bestätigung erfolgte sowohl in den Experteninterviews [St09], als auch als Zusatzergebnis einer am Lehrstuhl derzeit laufenden Dissertation.

Ziel der empirischen Bestätigung ist die Prüfung der Aktualität und der praxeologischen Relevanz der aus den Sekundärdatenanalysen erstellten Erfolgs- und Misserfolgsfaktoren.
Zur Forschungsmethodik, Auflistung der Interviewpartner, Dokumentation und Auswertung der Interviews sei auf [St09] verwiesen.

2 Terminologie

Manche Begriffe finden in verschiedenen Literaturquellen mehrdeutig Verwendung. Nachstehend erfolgt daher eine kurze Begriffsfestlegung für den vorliegenden Beitrag.

[1] bzgl. Terminologie vgl. 2.2.2 Erfolgsgefährdende Faktoren: ‘Misserfolgsfaktoren‘

2.1 Externe Vergabe

‘Externe Vergabe‘ steht im Folgenden für jegliche Art der Vergabe von Auftragsentwicklung an wirtschaftlich oder rechtlich unabhängige Partner oder an Partner in verschiedenen Rechtsgebieten/Hoheitsbereichen, z.B. auch Outsourcing oder Offshoring.

2.2 Erfolgsfaktoren / Erfolgsgefährdende Faktoren: ‘Misserfolgsfaktoren‘

Ein Faktor[2] kann fördernd, hemmend oder gefährdend in Erscheinung treten. Ein Faktor kann ein Ereignis oder auch ein materielles oder immaterielles Mittel oder eine Leistung sein, auch die Eigenschaften eines Objektes können im Softwareentwicklungsprozess einen Faktor darstellen; selbst die Prozessgestaltung kann als Faktor betrachtet werden (vgl. [St09], S. 30).

Hierbei sind erfolgsgefährdende Faktoren, d.h. ‘Misserfolgsfaktoren‘, keine Umkehrung bzw. Negierung von Erfolgsfaktoren (bzw. umgekehrt). Beide Arten von Faktoren sind in einer Anwendungsentwicklung gesondert zu beachten. Die ideale Anwendungsentwicklung ist bestrebt, alle für sie zutreffenden Erfolgsfaktoren herbeizuführen und / oder diese zu etablieren und sie ist bestrebt permanent zu beachten, dass kein Misserfolgsfaktor auftreten kann. Sie ist weiterhin bestrebt, bei Auftreten eines Misserfolgsfaktors unverzüglich Gegenmassnahmen einzuleiten.

2.2.1 Erfolgsfaktoren

Erfolgsfaktoren können das Erreichen der Ziele einer Anwendungsentwicklung fördern, bewirken können sie einen Erfolg jedoch nicht. Es ist eher die umgekehrte Betrachtung zutreffend: Wenn ein Projekt bzw. eine Anwendungsentwicklung auf die in Kap. 4 gelisteten Erfolgsfaktoren hin geprüft wird und diese fehlen, fehlerhaft sind, nicht zutreffen, nicht gültig oder nicht möglich sind, dann wird ein Erfolg unwahrscheinlich.

Kritische Erfolgsfaktoren stellen hingegen notwendige (jedoch auch nicht hinreichende) Voraussetzungen für das Erreichen der Ziele einer Anwendungsentwicklung dar. Anders ausgedrückt: Ein nicht erfüllter, kritischer Erfolgsfaktor kann zum Scheitern eines Softwareentwicklungsprojektes führen (vgl. [CS10]).

Ein Beispiel für einen Erfolgsfaktor in der Anwendungsentwicklung ist, dass eine Anwendungsentwicklung auf einem Vorgehensmodell basiert, welches eine jeweilige Reaktion auf ein Auftreten eines Misserfolgsfaktors klar definiert – z.B. unmittelbare Eskalation an einen Lenkungsausschuss/Steuerkreis o.ä.

[2] „massgebende Wirkungskraft“, vgl. [BI09]

2.2.2 Erfolgsgefährdende Faktoren: ‘Misserfolgsfaktoren‘

Erfolgsgefährdende Faktoren, d.h. Misserfolgsfaktoren können das Erreichen der Ziele einer Anwendungsentwicklung gefährden, schwieriger oder unmöglich machen. Das Auftreten eines oder mehrerer Misserfolgsfaktoren ist somit kritisch für den unmittelbaren Projektzustand (und somit für einen zukünftigen Projekterfolg) zu sehen.

3 Kategorisierung entsprechend ISO/IEC 12207 und ISO/IEC 15288

Angesichts des Sachverhalts, dass Unternehmen in der Praxis zur Strukturierung, Bewertung und Optimierung ihrer Anwendungsentwicklung oder zur Beurteilung der entsprechenden Qualität ihrer externen Dienstleister auf bewährte Reifegradmodelle, wie z.B. CMMI (vgl. [CM10]) oder SPICE (vgl. [15504]) zurückgreifen, orientiert sich auch die Faktorenanalyse strukturell an einem etablierten Prozessrahmenwerk (vgl. [Hi06], S. 212f. und 218).

Die Kategorisierung der im nachfolgenden Kapitel gelisteten Erfolgs- und Misserfolgsfaktoren in der Anwendungsentwicklung wird daher in Anlehnung an die in nachfolgend abgebildeten Schema der Struktur des Prozessrahmenwerks ISO/IEC 12207 gegliedert.

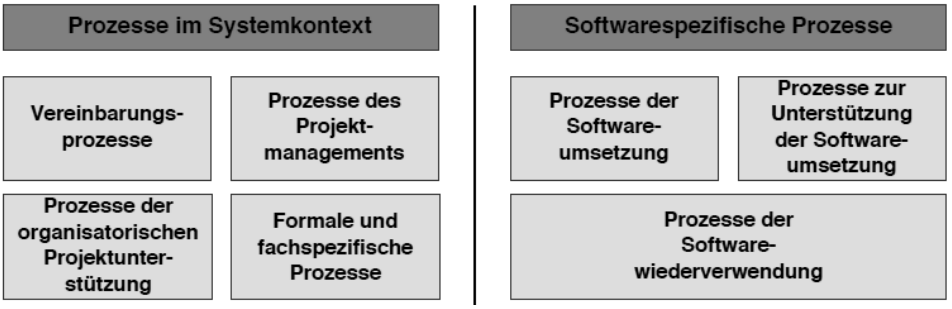

Abbildung 1: Gliederungsschema des Prozessrahmenwerks der ISO / IEC 12207

3.1 ISO/IEC 12207 – Entwicklung, Betrieb und Wartung von Softwaresystemen

Die Norm ISO / IEC 12207 gliedert sich in obig dargestellte sieben Prozessgruppen. Wichtig ist hierbei, dass es sich nicht um Pflichtprozesse handelt, die zu implementieren sind. Vielmehr gilt es bei der Nutzung des Referenzmodells die Prozesse entsprechend den Bedürfnissen der Organisation und in Vereinbarung mit dem Anwendungsbereich auszuwählen und anzupassen.
In ISO / IEC 12207 werden die Bereiche Prozesse im Systemkontext und softwarespezifische Prozesse unterschieden. Die Prozesse im Systemkontext beschäftigen sich mit eigenständigen Softwareprodukten oder Softewaredienstleistungen oder ganzen Softwaresystemen. Die softwarespezifischen Prozesse erweitern den Anwendungsbereich um die Implementierung von Softwareprodukten oder Softwaredienstleistungen in ein grösseres Gesamtsystem (vgl. [12207], S. 14).

3.2 ISO/IEC 15288 – Systementwicklung / Systemlebenszyklus und seine Prozesse

In enger Verbindung zur ISO/IEC 12207 steht die Norm ISO/IEC 15288 mit Fokus auf Prozesse des Systemlebenszyklusses. Beide Normen können parallel eingesetzt werden. Grundsätzlicher Schwerpunkt von ISO/IEC 15288 ist die Bereitstellung eines Rahmens für die Beschreibung des Lebenszyklusses eines künstlich geschaffenen Systems.
Im Verständnis der Norm kann ein System als zentrales Bezugsobjekt sowohl aus Hardware- als auch aus Softwarekomponenten sowie Daten, Menschen, Prozessen, Prozeduren, Anlagen, Materialien und weiteren natürlichen Objekten bestehen. Die ISO/IEC 15288 zielt auf die Bereitstellung eines definierten Satzes an Prozessen für die Kommunikation zwischen Einkäufern, Lieferanten und weiteren Interessengruppen ab (vgl. [15288], S. 1).

Da die beiden Normen ISO/IEC 12207 und ISO/IEC 15288 weitestgehend harmonisiert sind, gleicht die strukturelle Gliederung der Norm ISO / IEC 15288 grossteils den Prozessgruppen der Systemkontextprozesse von ISO/IEC 12207.
Die Kategorisierung nachfolgend gelisteter Erfolgs- und Misserfolgsfaktoren nach ISO/IEC 12207 ist somit übertragbar auf ISO / IEC 15288 und beinhaltet die Anwendungsentwicklung, sowohl aus Softwaresicht als auch aus Systemsicht.
Eine Produktentwicklung kann entsprechend erfolgen.

4 Erfolgs- und Misserfolgsfaktoren der Anwendungsentwicklung

Die prozessorientierte Klassifizierung der Erfolgs- und Misserfolgsfaktoren entspricht i.d.R. den realen Rahmenbedingungen eines Softwareentwicklungsprojektes bzw. einer Anwendungsentwicklung.
So ist z.B. das V-Modell XT "… kompatibel mit aktuellen (Quasi-)Standards … wie zum Beispiel ISO/IEC 15288." ([VM10], S. 1-10)
Eine Übertragung und Anwendung nachfolgend gelisteter Erfolgs- und Misserfolgsfaktoren auf reale Tätigkeiten der Anwendungsentwicklung ist somit durch deren Kategorisierung entsprechend ISO/IEC 12207 und ISO/IEC 15288 unterstützt.

4.1 Erfolgs- und Misserfolgsfaktoren der Prozesse im Systemkontext

4.1.1 Vereinbarungsprozesse

In den Vereinbarungsprozessen erfolgen die Abstimmungen und Vereinbarungen zwischen Anforderer und Leistungserbringer. Bei externer Vergabe sind Ergebnisse der Vereinbarungsprozesse das Pflichtenheft sowie sämtliche abzuschliessende Verträge.[3]

[3] i.d.R. bestehen diese aus dem konkreten Werk-, Dienst- oder Kaufvertrag plus einen Rahmenvertrag, welcher generelle Fragen der Zusammenarbeit (z.B. Haftung, Vertraulichkeitsvereinbarungen, Gerichtsstand usw.) regelt. Bei ersten Aufträgen, unregelmäßiger Zusammenarbeit usw. wird auf den Rahmenvertrag jedoch oft verzichtet so dass dessen Inhalt in den Einzelverträgen berücksichtigt werden muss.

Generelle Erfolgsfaktoren in Vereinbarungsprozessen[4]

- Vereinbarung realistischer Ziele / Definition klarer Projektziele

Übereinstimmend erachten, sowohl die untersuchten Literaturquellen, als auch die Interviewpartner der Experteninterviews, die Vereinbarung realistischer Ziele und Definition klarer Projektziele als den für alle Arten der Anwendungsentwicklung allgemein gültigen Erfolgsfaktor. Dieses eigentlich selbsterklärende Ergebnis bedingt jedoch, dass ein fundiertes Requirements-Engineering der Spezifikation vorausgeht sowie dass ein etabliertes, ebenfalls fundiertes Change-Management von während der Projektlaufzeit / Anwendungsentwicklung oft kommenden Änderungs- oder Anpassungsanforderungen stattfindet.

Vereinbarungsprozesse bei externer Vergabe haben zusätzlich folgende Erfolgsfaktoren

- Umfassende Branchenkenntnisse des Auftragnehmers
- Finanzielle Stabilität des Anbieters
- Angemessene politische und rechtliche Stabilität im Land des Auftragnehmers
- Partnerschaftliche Beziehung zwischen Auftraggeber und Auftragnehmer
- Passende Unternehmensgrösse des Auftragnehmers zum Auftraggeber

Während die ersten vier Erfolgsfaktoren selbsterklärend sind, erklärt sich der Faktor "Passende Unternehmensgrösse" durch geforderte / vorhandene Zertifizierungen, angewendete Vorgehensmodelle und die oft ähnlichen, d.h. dann oft "automatisch passenden" organisatorischen Abläufe und Hierarchien in Organisationen vergleichbarer bzw. 'passender' Grösse.

Misserfolgsfaktoren in Vereinbarungsprozessen[5]

Generelle (d.h. für sowohl intern durchgeführt, als auch extern vergeben, als auch Mischformen) Misserfolgsfaktoren konnten nicht festgestellt werden.

Misserfolgsfaktoren in Vereinbarungsprozessen bei externer Vergabe sind jedoch

- Ungeklärte Eigentumsrechte
- Unklare Verträge
- Festpreisprojekte ohne diesbezügliche Erfahrung
- Vertrauensmangel

[4] Quellen: vgl. [AW05], [Be08], [Jo96], [Ko02], [SG01]
[5] Quellen: vgl. [BJS00], [Jo96], [Ko02]

Der letzte Punkt, "Vertrauensmangel", kann als Umkehrung o.g. Erfolgsfaktors "partnerschaftliche Beziehung" gesehen werden; die ersten zwei Punkte betonen, dass aus nicht besprochenem / vereinbartem (und somit nicht geklärtem) Unklarheiten entstehen, die zu Missverständnissen führen was in Vertragsverhältnissen zur jeweils für sich selbst vorteilhafteren Sicht jedes Partners führt. Streitigkeiten oder gerichtliche Auseinandersetzungen sind hier somit vorprogrammiert. Auftragnehmer, welche Festpreisprojekte akzeptieren ohne diesbezügliche Erfahrung zu haben, tragen evtl. entstehende Mehrkosten (womit o.g. Erfolgsfaktor "finanzielle Stabilität des Anbieters" u.U. gefährdet ist).

4.1.2 Prozesse der organisatorischen Projektunterstützung

Prozesse der organisatorischen Projektunterstützung dienen der Schaffung und Erhaltung der Leistungsfähigkeit einer Organisation. Hierzu gehören z.B. die Bereitstellung notwendiger Infrastruktur und Ressourcen (vgl. [12207], S. 15), das Prozess- und Qualitätsmanagement sowie ein Projektportfoliomanagement (vgl. [12207, S. 25ff.).

Generelle Erfolgsfaktoren in Prozessen der organisat. Projektunterstützung[6]

- Definition von Projektstandards, Einrichtung einer effizienten IT-Infrastruktur
- Aufgabengrösse beherrschbar halten, evtl. Aufteilung in Teilziele
- Frühzeitiges Einleiten eines notwendigen Change-Management Prozesses
- Anhaltende Managementunterstützung
- Geeignetes Projektteam zusammenstellen, einschlägige Erfahrungen der Projektleiter

Während die ersten drei Erfolgsfaktoren konstruktiver Natur sind, d.h. diese Prozesse bzw. Standards werden in der Projekt- und Produktentwicklung entsprechend etabliert, ist der vierte Faktor, "anhaltende Managementunterstützung", eine Führungsfrage psychologischer Art – doch ist er im Falle auftretender Probleme entscheidend wichtig. Der letzte Punkt ist ein Faktor der Kategorie 'Soft Skills'[7], d.h. er betrifft die Eigenschaften, Qualifikationen und das Zusammenarbeiten der Mitarbeiter.

[6] Quellen: vgl. [AW05], [Ke98], [Ko02], [SG01]

[7] „Soft Skills stehen für einen ganzen Katalog von überfachlichen Kompetenzen, die dem persönlichen und zwischenmenschlichen Bereich zuzuordnen sind. Das Spektrum der Soft Skills reicht beispielsweise von Menschenkenntnis und Einfühlungsvermögen sowie Kommunikations- und Kritikfähigkeit bis hin zum Durchsetzungsvermögen und der Fähigkeit, andere Menschen einzubinden und für ein gemeinsames Ziel zu begeistern." [BG10]

Bei externen Vergaben der Anwendungsentwicklung kommen hier noch folgende Erfolgsfaktoren hinzu:

- Angemessenes technisches Verständnis auf Auftraggeberseite
- Hohe Qualifikation der Mitarbeiter auf Auftragnehmerseite
- Sprachkenntnisse, kulturelle Sensibilität und internationale Unternehmenskultur

Während der erste Faktor technisches Know-How auch auf Auftraggeberseite fordert, was bedingt, dass der Auftraggeber fundierter über Probleme oder Fortschritte im Projekt befinden kann und nicht dem Auftragnehmer und 'nur' dessen Darstellungen Glauben schenken muss, beinhaltet der zweite Faktor eine Know-How Forderung auf Auftragnehmerseite – nämlich Fachwissen über die zu entwickelnde Anwendung. Ein gegenseitiges Verstehen des jeweilig anderen Fachwissens ist somit erleichtert. Der letzte Punkt ist wieder eindeutig in der Kategorie 'Soft Skills' (s.o.)

Misserfolgsfaktoren in Prozessen der organisatorischen Projektunterstützung[8]

- Kein handlungsfähiger Lenkungsausschuss
- Keine Kontrolle des Projektes / keine Benutzung automatischer Planungswerkzeuge
- Nicht genügend Ressourcen bzw. unrealistisches Budget
- Projektleitung mit zu wenig operativem Entwicklungs-Know-How
- Personelle Defizite, ungenügendes Wissen oder Erfahrungen bei Mitarbeitern
- Konflikte zwischen Abteilungen

Dass aufgrund des ersten Punktes ein Projekt scheitern kann, erstaunt evtl. etwas – doch führt ein handlungsunfähiger Lenkungsausschuss zu Akkumulation von Problemen. Probleme, welche evtl. einzeln zwar durch Workarounds[9] lösbar sind, in der Summe jedoch ein Projekt zu Fall bringen können. Die restlichen Faktoren sind selbsterklärend, wobei der Teilaspekt "Personelle Defizite" wieder in die Kategorie 'Soft Skills' einzuordnen ist.

[8] Quellen: vgl. [AW05], [Bo91], [BJS00], [Jo96], [Ke98], [Ko02]
[9] hier i.S.v. Umgehung, Ersatzlösung, anderweitige Kompensation

4.1.3 Prozesse des Projektmanagements

Die Prozessgruppe des Projektmanagements umfasst in ISO / IEC 12207 sieben grundlegende Prozesse. So werden beispielsweise in einem Projektplanungsprozess die allgemeinen Projektanforderungen und die Realisierbarkeit eines Projekts festgelegt. Weiterhin werden Vorgehensweisen zur Projektplanung definiert, es wird ein Projektbewertungs- und Projektsteuerungsprozess abgegrenzt, ein Prozess zur Entscheidungsunterstützung, der Risikomanagementprozess, das Konfigurationsmanagement, das Informationsmanagements und ein Messungs- und Bewertungsprozess sind hier ebenfalls mit beinhaltet (vgl. [12207], S. 32ff.).

Generelle Erfolgsfaktoren in Prozessen des Projektmanagements[10]

- Effiziente Organisationsstruktur, schnelle Entscheidungsfindung
- Früher Einsatz von Werkzeugen für Abschätzungen / frühzeitige und regelmässige Kontrolle der Projekt(teil)ergebnisse / verlässliche Schätzungen
- Formales Berichtswesen / standardisierte, dokumentierte Prozesse / stringente Projektmanagement-Methoden
- Permanentes Risikomanagement während der gesamten Projektlaufzeit

Während die ersten beiden Punkte offensichtlich sind, etabliert der dritte ein Mindestmass an Bürokratie. Ein formales Berichtswesen sorgt jedoch dafür, dass Information regelmässig und an die richtigen Stellen fliesst. Und so kann z.B. ‘versehentlich‘ nichts verschwiegen werden, das Management muss permanent involviert bleiben. Der Punkt “permanentes Risikomanagement“ begründet sich vor allem durch die, während der Projektlaufzeit i.d.R. unweigerlich kommenden Änderungen. Das Risikomanagement sollte Teil des Change-Managements sein oder zumindest von diesem bei jedem Change automatisch angestossen werden.

Hinzukommende Erfolgsfaktoren bei externen Vergaben

- Geografische Nähe des Anbieters
- Umfassende Erfahrung mit IT-Outsourcingprojekten

Während der zweite Punkt offensichtlich und einleuchtend ist, ist der erste Erfolgsfaktor, die geografische Nähe des Anbieters, zu Zeiten des Internets, Telekonferenzen usw. zunächst verblüffend. Doch kulturelle und mentale Unterschiede wachsen oft mit der geografischen Entfernung und die Auswirkung von Arbeitszeiten in unterschiedlichen Zeitzonen (mit der für Echtzeitkommunikation dann oft nur kleinen möglichen Kommunikationszeit-Schnittmenge) wird nach wie vor stark unterschätzt.

[10] Quellen: vgl. [AW05], [Be08], [Bo91], [BJS00], [Fe04], [Jo96], [Ko02]

Misserfolgsfaktoren in Prozessen des Projektmanagements[11]

- Andauernde Änderungen von Anforderungen, ineffektives Change-Management
- Fehlendes Risikomanagement / keine Kontrolle der Kosten, Kostenüberschreitungen
- Unrealistische Aufwandsabschätzungen / Planungen (Zeit, Kosten, Ressourcen usw.)
- Kein angemessenes Konfigurationsmanagement
- Zu grosser Projektumfang

Ein fundiertes und etabliertes Change-Management wurde in allen, die Sekundärdatenanalyse bestätigenden Interviews (vgl. [St09], S. 63ff.) als einer der wichtigsten Erfolgsfaktoren, sowohl in der Entwicklung, als auch im Betrieb von Anwendungen genannt. Andauernde Änderungen von Anforderungen zeugen entweder von einem unzureichenden Requirements-Engineering oder stellen zumindest den derzeitigen Zweck und das Ziel der Anwendung überhaupt in Frage. Zudem führt eine permanente ad-hoc Berücksichtigung von Änderungen, neben dem Aufwand (Chaos?) auch zur Frustration der Entwickler. Kostenüberschreitungen, welche i. d. R. aus fehlendem oder mangelhaftem Risikomanagement, aus zu vielen Änderungen oder aus unrealistischen Aufwandsabschätzungen entstehen, führen zwar nur selten zum gänzlichen Misserfolg, sie bewirken jedoch oft das Fertigstellen einer Version ohne 'Features' / Begeisterungsmerkmale.

Misserfolgsfaktoren speziell bei externer Vergabe von Anwendungsentwicklungen

- Verspätung extern entwickelter Komponenten bzw. extern durchgeführter Aufgaben
- Konflikte zwischen internen und externen Mitarbeitern

Externe Vergabe erzeugt Abhängigkeiten – und was nützt es z.B. später vor Gericht Recht zu bekommen, wenn ein komplettes Grossprojekt aufgrund eines kleinen, extern vergebenen Teilprojektes scheiterte? Der zweite Punkt benennt die bereits erwähnten 'Soft Skills'.

4.1.4 Formale (technische) und fachspezifische Prozesse

Diese Prozessgruppe umfasst den Prozess zur Definition der Anforderungen aller Anspruchsgruppen an das Softwaresystem, die Ableitung einer Systemspezifikation aus den Anforderungen sowie die Überprüfung derselben auf ihre Realisierbarkeit.

[11] Quellen: vgl. [Bo91], [BJS00], [FLS02], [Jo96], [Ke98], [Ko02]

Weiterhin umfasst sie den Prozess des Entwurfs und der Überprüfung einer Systemarchitektur auf Basis der Spezifikationen. Der Realisierungsprozess ist hier ebenso beschrieben wie der Systemintegrationsprozess, welcher einzelne Systemelemente zu einem Gesamtsystem zusammenführt. Ergänzt werden diese Prozesse durch Systemeignungstests und die Installation des geprüften Systems (vgl. [12207], S. 42ff.).

Generelle Erfolgsfaktoren in formalen (technischen) und fachspez. Prozessen[12]

- Detaillierte Spezifikation, Erstellen eines umfassenden Business-Case
- Einbindung von Benutzern / Akzeptanz bei Benutzern
- Stabile Basisanforderungen, die den 'minimalen' Funktionsumfang ergeben
- Abstimmung der Implementierungsmethode auf die Organisation
- Verwendung von Standardelementen

Ebenso wichtig wie eine umfassende, detaillierte Spezifikation ist die fachliche Beschreibung der Funktionalität der späteren Anwendung. Hierdurch wird vermieden, dass Entwickler den Kontext der Funktionalität missverstehen und dass die Entwickler somit, trotz formal korrekter einzelner Umsetzungen, das ganze Produkt am Bedarf vorbei entwickeln. Die Einbindung von Benutzern in einen Anwendungsentwicklungsprozess bewirkt i.d.R. dasselbe. Zum dritten Punkt: Bei der Erhebung der Anforderungen an die Software sollte eine Konzentration auf stabile Basisanforderungen erfolgen, d.h. zunächst nur wenige, jedoch grundlegenden Anforderungen, die schnell umsetzbar sind. Dies präsentiert Nutzern oder Kunden schon früh ein verifizierbares Ergebnis. Der vierte Punkt ist offensichtlich – eine Verwendung von Standardelementen erspart Entwicklungsaufwand sowie fast sämtliche sonstige Tests (ausser Integrationstests) und weitere analytische Qualitätssicherungsmassnahmen einer Neuentwicklung.

Misserfolgsfaktoren in formalen (technischen) und fachspezifischen Prozessen[13]

- Andauernde Änderung der Anforderungen / keine Kontrolle der Anforderungen
- Anforderungen fehlen oder wurden nicht bzw. falsch verstanden
- Erwartungen der Anwender nicht beachtet / erfüllt, fehlende Benutzerakzeptanz
- Zu hohe Komplexität der Software
- Inkompatibel zu bestehenden Anwendungen / Nichtbeachtung von Industriestandards
- Unklare oder unvollständige Spezifikationen, Fehlen eines Pflichtenhefts
- Fehler in der Implementierung

[12] Quellen: vgl. [AW05], [Fe04], [Jo96], [Ke98], [Ko02], [SG01]
[13] Quellen: vgl. [Bo91], [BJS00], [FLS02], [Jo96], [Ke98], [Ko02]

Während der erste Punkt wieder die Bedeutung eines ‘gelebten‘ Change-Managements unterstreicht, belegen die weiteren fünf Punkte die fundamentale Bedeutung eines fundierten und etablierten Requirements-Engineerings. Lediglich der letzte Punkt dieser Liste, welche formale (technische) Prozesse betrifft (sowie teilweise der Punkt “Zu hohe Komplexität der Software“) beinhaltet tatsächlich Probleme der technischen Umsetzung.

4.2 Softwarespezifische Prozesse

Die nun nachfolgenden Erfolgs- und Misserfolgsfaktoren sind in der Anwendungsentwicklung aus der Entwicklung von Software hergeleitet. Ob eine Übertragbarkeit auf weitere Bereiche, wie z.B. eine Produktentwicklung stattfinden kann, muss durch jeweilige Betrachtung der durchzuführenden Entwicklung festgestellt werden. Eine solche Übertragbarkeit kann in vielen Fällen jedoch durchaus gegeben sein, zudem stellt die Softwareentwicklung einen entscheidenden Anteil aller Entwicklungstätigkeiten dar.

4.2.1 Prozesse der Softwareumsetzung

Die Prozessgruppe zur Softwareumsetzung besteht aus sieben Prozessdefinitionen, die die Herstellung spezifischer Systemelemente im Sinne von Softwarekomponenten und deren Implementierung in das gesamte Softwaresystem beschreiben (vgl. [12207], S. 57). Die Prozessempfehlungen dieses Bereichs weisen starke Analogien zu Teilprozessen aus der Gruppe der formalen (technischen) und fachspezifischen Prozesse auf und deckt sich in Bezug auf relevante Erfolgs- und Misserfolgsfaktoren mit denen jener Prozessgruppe (vgl. [St09], S. 52).

4.2.2 Prozesse zur Unterstützung der Softwareumsetzung

Generelle Erfolgsfaktoren der Prozesse zur Unterstützung der Softwareumsetzung[14]

Die betrachteten Literaturquellen benennen als Erfolgsfaktoren zur Unterstützung der Softwareumsetzung analytische Massnahmen (vgl. [SBM83], S. 38ff.). Insbesondere werden hier genannt

- Code-Inspektionen, Design-Reviews und formale Testmethoden

Analytische (Qualitätssicherungs-)Massnahmen sind diagnostischen Massnahmen und messen das Qualitätsniveau. Diese statischen (“prüfen“) und dynamischen (“testen“) Massnahmen sammeln gezielt und mit analytischen Mitteln Informationen. Sie geben somit den Status der Anwendungsentwicklung zum jeweiligen Prüf-/Testzeitpunkt wieder.

[14] Quellen: vgl. [Jo96], [Ko02]

Misserfolgsfaktoren der Prozesse zur Unterstützung der Softwareumsetzung[15]

- Fehlen von Qualitätssicherung (insbesondere fehlende Integrationstests, Audits und Reviews), keine entwicklungsbegleitenden Code-Inspektionen und Design-Reviews
- Nicht vorgesehene Abnahmekriterien
- Hoher Grad an Inkompatibilität

Misserfolgsfaktoren speziell bei externer Vergabe von Anwendungsentwicklungen

- Defizite in extern ausgeführten Aufgaben oder extern hergestellten Komponenten

Der erste Punkt genereller Misserfolgsfaktoren der Prozesse zur Unterstützung der Softwareumsetzung stellt hier die Umkehrung obigen Erfolgsfaktors dar; Methoden der analytische Qualitätssicherung fehlen und fundierte Aussagen über den Zustand oder Fortschritt der Anwendungsentwicklung können somit nicht erfasst und dokumentiert werden. Der zweite Punkt benennt einen Teilbereich dieses Themas: Tests o.ä. können nicht sinnvoll durchgeführt werden, wenn ein Sollzustand undefiniert oder unbekannt ist, gegen diesen nämlich das 'Ist' getestet werden soll. Hieraus, sowie aus fehlerhaften oder undefinierten Anforderungen entstehen Inkompatibilitäten, insbes. bei externer Vergabe.

4.2.3 Prozesse der Softwarewiederverwendung[16]

Die Prozesse zur Softwarewiederverwendung nehmen, im Vergleich zu anderen Prozessen, im Rahmen von Softwareentwicklungsprojekten eine besondere Stellung ein, da sie über die Grenzen eines einzelnen Projektes hinweg Anwendung finden. Deren Intention ist es, Organisationen bei der projektübergreifenden Wiederverwendung (ein Aspekt einer nachhaltigen Anwendungs- und Produktentwicklung) einzelner Softwareelemente zu unterstützen (vgl. [12207], S. 17).
Der Domain Engineering Prozess unterstützt auf Basis aktueller und angenommener Bedürfnisse von Stakeholdergruppen die Softwarewiederverwendung durch die Ausgestaltung und Verwaltung thematisch abgegrenzter Bereiche, auch Domänen genannt. Aufgabe des Prozesses ist, die Festlegung der Struktur und des Umfangs solcher Domänen sowie der entsprechende Aufbau und die Pflege der darin zusammengefassten Objekte, wie zum Beispiel Softwarecode oder Dokumentationsmuster (vgl. [12207], S. 78ff.). In direktem Zusammenhang damit steht der Reuse Asset Management Prozess, welcher die Verwaltung einzelner Wiederverwendungsobjekte übernimmt. Aufgaben dieses Prozesses sind zum Beispiel die Implementierung einer adäquaten Speicherungslösung für die zu verwaltenden Objekte sowie die objektbezogene Überwachung der Wiederverwendung. Sollten Probleme mit einzelnen Wiederverwendungsobjekten auftreten, trägt der Prozess auch eine Informations- beziehungsweise Berichtsfunktion (vgl. [12207], S. 80ff.).

[15] Quellen vgl. [Bo91], [BJS00], [FLS02], [Ko02]
[16] vgl. [St09], S. 55f.

Zudem zählt der Reuse Program Management Prozess zu dieser Gruppe. Seine Aufgabe ist die Planung und Steuerung der Softwarewiederverwendungstätigkeiten des Unternehmens. Auch sind Überwachungsaktivitäten und die gezielte Förderung der Softwarewiederverwendung Gegenstand des Prozesses (vgl. [12207], S. 82).

Der gesamte vorliegende Bereich der Softwarewiederverwendung ist als ein Erfolgsfaktor in Softwareentwicklungsprojekten zu berücksichtigen (vgl. [Va05], S. 29; [Jon96], S. 6).
Werden einzelnen Codebausteine oder ganze Softwarekomponenten projektübergreifend wiederverwendet, kann dies eine verkürzende Wirkung auf die Entwicklungszeit haben. Grundlage hierfür ist die gute Dokumentation der entsprechenden Komponenten, so dass die Entwickler notwendiges Wissen für die Implementierung leicht erlangen können (vgl. [Va05], S. 29).

5 Fazit

Die Erkenntnisse, welche dieser Beitrag der doch bereits reichlich vorhandenen Literatur zu Erfolgs- bzw. Misserfolgsfaktoren hinzufügt, bestehen in der Prüfung dieser Faktoren auf praxeologische Relevanz sowie Kategorisierung / Einordnung der, in den Experteninterviews empirisch als wichtig und besonders praxisrelevant begründeten Faktoren, direkt in ein Schema, welches gängigen Vorgehens- und Reifegradmodellen oft zugrunde liegt.

Literaturverzeichnis

[AW05] Amberg, M. und Wiener, M. (2006). Kritische Erfolgsfaktoren für Offshore-Softwareentwicklungsprojekte, eine explorative Studie des Lehrstuhls Wirtschaftsinformatik III, Friedrich-Alexander-Universität Erlangen-Nürnberg; URL: http://www.competence-site.de/it-outsourcing/Kritische-Erfolgsfaktoren-fuer-Offshore-Softwareentwicklungsprojekte (Zugriff am 30. Sep. 2010)

[Be08] Bennicke, M., Hofmann, A., Lewerentz, C. und Wichert, K. H. (2008), Software Controlling, in: Informatik-Spektrum, 31, 2008, 6, S. 556–565

[Bo91] Boehm, B. (1991), Software risk management: principles and practices, in: IEEE Software, 8, 1991, 1, S. 32–41

[BJS00] Boos, M., Jonas, K. J. und Sassenberg, K. (Hrsg.) (2000), Computervermittelte Kommunikation in Organisationen, Bd. 3, Internet und Psychologie: Neue Medien in der Psychologie, Göttingen, Bern u. a. 2000.

[BG10] Bologna-Glossar (2010). Universität Paderborn, URL: http://www2.uni-paderborn.de/~Studienreform/Einrichtung%20STUGA/Bologna-Glossar.htm (Zugriff am 19. Juni 2010)

[BI09] Bibliographisches Institut & F. A. Brockhaus AG (2009), Brockhaus Lexikon; URL: http://www.brockhaus.de/suche/index.php?begriff=Faktor&bereich=mixed&x=0&y=0 (Zugriff am 24. Jan. 2009)

[CS10] California State University Monterey Bay (2010), Information Technology – Data Warehouse Glossary, URL: www.csumb.edu/site/x7101.xml (Zugriff am 30. Sep. 2010)

[CM10] Carnegie Mellon University (2008). WebPage des Software Engineering Institute der Carnegie Mellon University, Pittsburgh (USA); URL: http://www.sei.cmu.edu/cmmi/ (Zugriff am 29. Sep. 2010)

[Fe04] Feyhl, A. W. (2004), Management und Controlling von Softwareprojekten. Software wirtschaftlich auswählen, entwickeln, einsetzen und nutzen, 2. Aufl., Wiesbaden 2004.

[FLS02] Frühauf, K., Ludewig, J. und Sandmayr, H. (2002), Software-Projektmanagement und -Qualitätssicherung, 4. Auflage, Zürich 2002.

[Hi06] Hindel, B., Hörmann, K., Müller, M. und Schmied, J. (2006). Basiswissen Software-Projektmanagement: Aus- und Weiterbildung zum Certified Professional for Project Management nach iSQl-Standard, 2., überarb. und erw. Aufl., dpunkt.verlag GmbH, Heidelberg, ISBN: 3-89864-390-5

[Jo96] Jones, C. (1996), Patterns of Software System Failure and Success, London und Boston 1996.

[Ke98] Keil, M., Cule, P. E., Lyytinen, K. und Schmidt, R. C. (1998), A framework for identifying software project risks, in: Commun. ACM, 41, 1998, 11, S. 76–83

[Ko02] Kotulla, A. (2002), Management von Softwareprojekten: Erfolgs- und Misserfolgsfaktoren bei international verteilter Entwicklung, Diss.: Universität Hohenheim, Wiesbaden 2002.

[SBM83] Schmitz, P., Bons, H. und van Megen, R. (1983). Software-Qualitätssicherung - Testen im Software-Lebenszyklus, 2., durchgesehene Auflage, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig und Wiesbaden 1983.

[SG01] Standish Group International (2001), Extreme Chaos, URL: http://www.standishgroup.com/sample_research/showfile.php?File=extreme_chaos.pdf, (Zugriff am 2. März 2009)

[St09] Stewen, Michael (2009), Erfolgsfaktoren in extern vergebenen Softwareentwicklungsprojekten Diplomarbeit am LSt. Wirtschaftsinformatik II, Universität Stuttgart, 2009.

[Va05] Valtinat, T. (2005), Entwurf von Softwareanpassungsprojekten bei gescheiterter Inbetriebnahme, Diss.: RWTH Aachen, Düsseldorf 2005.

[VM10] V-Modell XT, Version 1.3 (2010). Dokumentation des V-Modell XT als .pdf (Hrsg.): Der Beauftragte der Bundesregierung für Informationstechnik; URL: http://ftp.tu-clausthal.de/pub/institute/informatik/V-Modell-xt/Releases/1.3/V-Modell-XT-Gesamt.pdf (Zugriff am 30. Sep. 2010)

[12207] ISO/IEC 12207:2008(E), Systems and Software Engineering – Software Life Cycle Processes, 2008.

[15288] ISO/IEC 15288:2008(E), Systems and Software Engineering – System Life Cycle Processes, 2008.

[15504] ISO/IEC 15504-5:2006 Process Assessment Model

Service Marktplätze als Bestandteil des Cloud Computing - Ein Reifemodell zur Kategorisierung und Abgrenzung von eServices -

Norman Pelzl, Georg Herzwurm

Lehrstuhl für ABWL und Wirtschaftsinformatik II (Unternehmenssoftware)
Betriebswirtschaftliches Institut
Universität Stuttgart
Keplerstr. 17
70174 Stuttgart
{pelzl, herzwurm}@wi.uni-stuttgart.de

Abstract: Das Interesse der Öffentlichkeit an Cloud Computing ist nach wie vor sehr hoch, da die Anzahl der Cloud Computing Angebote trotz der frühen Entwicklungsphase des Marktes überproportional gestiegen sind. Daher versprechen auch viele große IT-Anbieter aus Deutschland und Amerika eServices anzubieten. Aus diesem Grund werden in diesem Beitrag eServices im Kontext von Cloud Computing von IT-Anbieter hinsichtlich deren Integrierbarkeit untersucht und ein Ausblick auf Service Marktplätze gegeben. Wir konnten feststellen, dass nur sehr wenige Cloud-eServices sofort und integrativ online nutzbar sind.

1 Einleitung und Ausgangssituation

Cloud Computing gilt als Business-Innovator und Hoffnungsträger in der IT-Branche [Mü09]. Sowohl Kunden als auch Software-Partner suchen heute zunehmend Dienstanbieter von Software welche standardisierte Dienste und eServices online anbieten können [VJ10]. Diese eServices sollen mit automatisierter Abwicklung hinsichtlich Bestellung und Zahlungsmodalitäten sowie in bestehende Systeme integriert funktionieren [Kr09]. Daher stehen Services-Anbieter vor der Herausforderung, geeignete Cloud-eServices zu entwickeln, um dadurch Kundenbindungen und neue Vertriebskanäle aufzubauen [Mü09]. Dazu eröffnen die neuen eServices-Dienste den Services-Anbietern die Möglichkeit, neue Geschäftsmodelle für Service Marktplätze zu erschließen und existierende Dienste zu vertreiben [Bö09]. Als Grundlage dessen, sollen neben einer eindeutigen Definition von eService im Kontext von Cloud Computing der Status quo der derzeitigen online angebotenen, bestellbaren und nutzbaren Cloud-eService-Angebote in einem Reifemodell abgebildet werden. Hierbei werden relevanten IT- und Software-Großunternehmen aus Deutschland und Amerika hinsichtlich der eService-Angebote untersucht. Anschließend werden Anregungen für ein Geschäftsmodell für Service Marktplätze mit Cloud-eServices diskutiert.

2 eServices im Cloud Computing

2.1 Cloud Computing und Service Marktplätze

Vor dem Hintergrund zahlreicher bestehender Definitionen zum Begriff Cloud Computing (CC) und dem Schwerpunkt des Artikels auf eServices zitieren wir die Cloud Computing Definition vom National Institute of Standards and Technology (NIST[1]) „Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (for example networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." und von Böhm, Leimeistert, Riedl und Krcmar [Bö09]: „Cloud Computing ist ein auf Virtualisierung basierendes IT-Bereitstellungsmodell, bei dem Ressourcen sowohl in Form von Infrastruktur als auch Anwendungen und Daten als verteilter Dienst über das Internet durch einen oder mehrere Leistungserbringer bereitgestellt wird. Diese Dienste sind nach Bedarf flexibel skalierbar und können verbrauchsabhängig abgerechnet werden.". Ergänzend fügen wir hinzu, dass die Ressourcen nicht nur *über das* Internet bereitgestellt, sondern auch *im* Internet selbst genutzt werden können. Service Marktplätze werden zumeist in der Literatur mit dem Zusammentreffen mehrerer Anbieter und mehrerer Nachfrager begründet welche, mithilfe eines festgelegten Preisbildungsmechanismus Produkte und Dienstleistungen, handeln [Ko07]. Darüber hinaus definiert sich ein elektronischer Marktplatz über die Unterstützung mindestens einer Transaktionsphase (Information, Vereinbarung oder Abwicklung) [Pe00; Sc00].

2.2 Kategorisierung und Definition von Cloud-eServices

Unter eService wird das elektronische Erbringen von Dienstleistungen über das Internet verstanden [Br02; We02; Re03; WE10]. Genauso wie es zahlreiche Definitionen über eServices und Cloud Computing gibt, existieren beim Cloud-eService zahlreiche Definitionen welche entweder zum eServices [Ös01] oder Web Services [Re03] Bezug nehmen [WE10]. Nach Hünerberg/Mann und Wegmann [HM02; We02] können typische eServices beispielsweise folgende Anwendungsformen umfassen: Web-Hosting, Mail-Hosting, Chat-Foren, Online-Lotto usw.

Bruhn typologisiert eServices nach folgenden beiden Typologieansätzen [Br02]. Zuerst auf Basis generischer Dienstleistungsdimensionen, wie Integrationsgrad (Autonom, Integrativ) vs. Immaterialitätsgrad (Materiell, Immateriell) und Individualisierungsgrad (Standardisiert, Individualisiert) vs. Interaktionsgrad (Unabhängig, Interaktiv). Anschließend auf Basis speziellerer Leistungsdimensionen, wie Empfänger des E-Service (Privater Nachfrager, Gewerblicher Nachfrager) vs. Marktstellung der Dienstleistung () und Marktstellung der Dienstleistung vs. Kostenstruktur der Dienstleistung (Variable Kosten, Nicht Variable Kosten).

[1] http://csrc.nist.gov/groups/SNS/cloud-computing/

Auf der Grundlage, dass Herstellereigene eServices von Unternehmen nur mittels eines fixen Preises Kunden angeboten werden können, kommen noch zwei weitere Dimensionen in Betracht. Zum Einen der Grad der Individualisierung bzw. Standardisierung der eServices, welche in einem hohen Grad standardisiert sein müssen. Zum Anderen die Marktstellung der Dienstleistung, hierbei ist es von Bedeutung, dass der eService den Nutzen und somit den wirtschaftlichen Mehrwert einer bestehenden Leistung/Anwendung erhöht [HP09]. Wir gehen deshalb in diesem Beitrag davon aus, das eServices im Cloud Computing Umfeld ein standardisierter-, internetbasierter-, Value-added-Service mit fixen Preis darstellt und sich damit von den generischen eService-Dienstleistungs-Definitionen u.a. nach Bruhn abgrenzt.

Nach der Aufstellung einer Rastermatrix kommen, auf Basis von Hünerberg/Mann, Wegmann und Bruhn [HM02; We02; Br02], somit die eServices für potenzielle Kunden in Betracht, welche sich in dem grünen (rechts unten) Quadranten befinden.

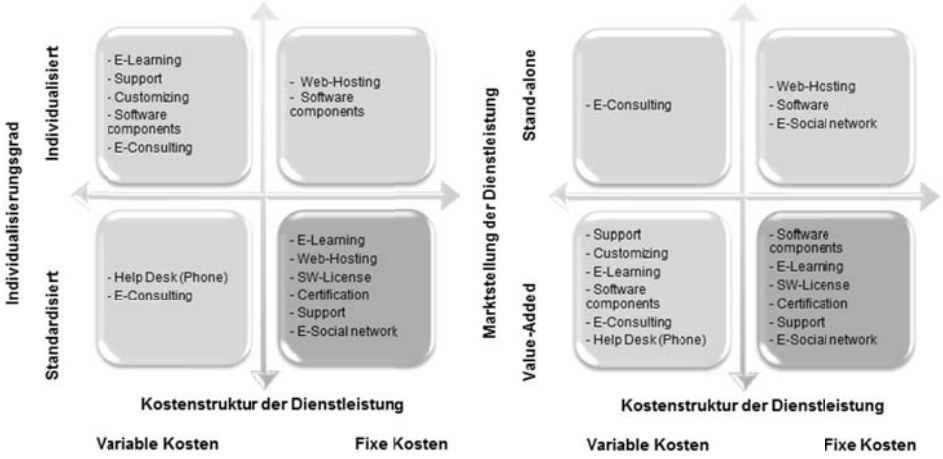

Abbildung 1: Rastermatrix von Anwendungsformen für Service Marktplätze[2]

Zusammenfassend definieren wir eServices im Kontext von Cloud Computing (Cloud-eServices), als eine standardisierte-, Value added-Software Applikation und/oder Dienstleistung mit fixen Preis, die online bestellt, bezahlt und in bzw. mit Anwendungen (in den bestehenden Geschäftsprozessen) integriert und genutzt wird.

[2] In Anlehnung an Bruhn02.

3 Ein Reifemodell von eServices

3.1 eServices-Analyse und Methodik

Im Rahmen dieses Beitrags wurden von den Autoren die eServices von elf Unternehmen systematisch untersucht. Der Fokus der zu untersuchenden Unternehmen wurde auf führende IT- und Software-Großunternehmen gelegt, welche Cloud Computing eServices anbieten und sowohl in Deutschland als auch in Amerika vertreten sind.

Die auszuwertenden eServices und die Datenerhebung wurden im Laufe des März bis Mai 2010 anhand der aktuell auf den Webseiten der jeweiligen Unternehmen veröffentlichten Angebote erarbeitet. Hierbei wurden keine Unterscheidung hinsichtlich Public oder Private Cloud eServices-Anwendungen gemacht. Die einzelnen Kategorien wurden aufgrund der angebotenen eServices der Unternehmen angelegt. Ausgehend von den unterschiedlich angebotenen eServices der Unternehmen lassen sich diese in ein Reifemodell übertragen [SSB05]. Hierbei werden die einzelnen eServices nach Übereinstimmung mit der Cloud-eServices Definition aus Kapitel 2.2 kategorisiert. Somit ist es mit diesem Reifemodell möglich, einzelne Angebotstypen der eServices zuzuordnen und die Anbieter abzugrenzen.

Für die Übersichtlichkeit der Ergebnisse wurden drei unterschiedliche Tabellen bzw. Reifestufen erstellt, die aufeinander aufbauen. Im ersten Schritt werden die angebotenen eServices aufgezeigt. Darauffolgend werden die eServices dargestellt, welche zusätzlich auch online im Internet, beispielsweise durch ein integriertes Formular, bestellt und damit auch verwaltet werden können. Im letzten Schritt werden die im Kapitel 2.2 definierten Cloud-eServices aufgezeigt, die sofort in der bestehende Anwendung bzw. Geschäftsprozess integriert genutzt werden kann.

3.2 eServices Ergebnisse

Im Folgenden werden die Ergebnisse der Untersuchung vorgestellt. Tabelle 1 zeigt die angebotenen eServices auf.

	Google	Amazon	Salesforce	IBM	Telekom	SAP	Oracle	Microsoft	Netsuite	HP	Apple
Support		x	x	x	x	x	x	x	x	x	x
SW-License			x	x	x	x	x	x	x	x	x
E-Learning			x	x	x	x	x	x	x	x	x
Web-Hosting		x		x	x		x	x	x	x	
Certification			x	x		x	x	x		x	x
Office[3]	x							x			x
ERP-SW[4]			x			x	x	x	x		
Hardware		x		x							

Tabelle 1: Angebote von eServices

In Tabelle 2 werden die eServices der Unternehmen aufgezeigt, die auch gleich online bestellt werden können. Hier wird deutlich, dass rund 45-50 % der angebotenen eServices aus Tabelle 1 nicht online bestellt werden können. Diese Unternehmen verweisen entweder auf einen persönlichen Kontakt (Telefon oder Termin) an das Call-Center des Kundenservice oder geben nur eine E-Mail-Adresse an.

	Google	Amazon	Salesforce	IBM	Telekom	SAP	Oracle	Microsoft	Netsuite	HP	Apple
Support		x	x				x	x		x	x
SW-License			x				x	x			x
E-Learning			x	x				x		x	x
Web-Hosting		x					x	x			
Certification			x	x			x	x		x	
Office	x							x			x
ERP-SW			x				x	x	x		
Hardware		x		x							

Tabelle 2: Angebote mit möglicher Bestellung von eServices

[3] Die Kategorie „Office“ umfasst in diesem Artikel gebräuchliche Bürosoftware wie beispielsweise Textverarbeitung und Tabellenkalkulation

[4] Die Kategorie „ERP-SW“ umfasst in diesem Artikel die mögliche Anwendungssoftware eines Unternehmens.

In der letzten Tabelle werden die Anbieter aufgeführt, welche die im Abschnitt 2.2 definierten Cloud-eServices anbieten. Im direkten Vergleich mit Tabelle 2 ergeben sich deutliche Unterschiede. Insbesondere der Cloud-eService „Certification“ wird bei keinem Unternehmen als sofort nutzbar angeboten. Die Kategorie „E-Learning“ ist nur bei IBM und HP sofort verwendbar. Außerdem besteht momentan nur bei Google („Google docs“) und Microsoft („Office Live“) die Möglichkeit, Office-Produkte sofort online zu bestellen und zu nutzen. Im Gegensatz dazu nehmen zurzeit die Angebote von sofort nutzbaren ERP-Lösungen, wie von Microsoft und Oracle, zu.

	Google	Amazon	Salesforce	IBM	Telekom	SAP	Oracle	Microsoft	Netsuite	HP	Apple
Support		x	x				x			x	x
SW-License			x				x				x
E-Learning				x						x	
Web-Hosting		x					x				
Certification											
Office	x							x			
ERP-SW			x				x	x	x		
Hardware		x		x							

Tabelle 3: Angebote mit möglicher Bestellung und sofortige integrierte Nutzung von Cloud-eServices

Das dargestellte Reifemodell zeigt, dass sich die erarbeitete Kategorisierung aus Kapitel 2.2 genau in den Angeboten der gesamten Unternehmen wiederspiegelt. Medienbrüche treten insbesondere bei der Bestellung und sofortigen Nutzung auf und dadurch besteht die benötigte integrative Nutzung von Cloud-eServices für Unternehmen nur teilweise oder gar nicht.

4 eService Marktplätze im Cloud Computing

Elektronische Marktplätze sind generell nach der Internet-Blase um das Jahr 2000 nicht mehr in der großer Anzahl, wie zur Boom Zeit vorhanden [IB08]. Eine Vielzahl von vertikalen und horizontalen Marktplätzen waren als Geschäftsmodell nicht ausgereift genug und es kam zu einer notwendigen Konsolidierung. Mittlerweile hat sich aber die Art der angebotenen Güter von beispielsweise C-Gütern hin zu reinen digitalen integrierten Gütern (Cloud-eServices) gewandelt. Neben diesen „neuen“ Gütern könnten gerade die neuen Technologien und Methoden wie Breitband-Internet, Virtualisierung und Skalierbarkeit entscheidend für mögliche eService Marktplätze sein [FL05].

Denn neben der Informationsphase können jetzt auch die Phasen Vereinbarung, Abwicklung und Bezahlung elektronisch in den Geschäftsprozessen integriert, abgewickelt werden [MS08]. Somit nimmt der Anteil des elektronischen Handels in den letzten Jahren wieder stark zu. eService Marktplätze können sowohl für Kunden, Partner aber auch Endkunden entwickelt werden. In der Preisgestaltung sind neben dem bisherigen Festpreis dann auch variable Preise mittels einer Ausschreibung oder Auktion möglich und besonders reizvoll. Hierbei ergibt sich die Chance einen zentralen übergreifenden Marktplatz zu entwickeln, welcher die neu entstandenen Technologien integriert und von vielen Anbietern Cloud-eServices anbietet. Darüber hinaus könnten die Arten des Cloud Computing kombiniert und unabhängig vom Anbieter angeboten werden, beispielsweise könnte die CRM-Software vom Anbieter A, aber die benötigte Hardware dazu vom Anbieter B gewählt werden. Die Vereinheitlichung und Abbildung der Cloud-eServices auf Service Marktplätzen ermöglichen nicht nur Transparenz sondern auch Vergleichbarkeit und Übersichtlichkeit bei der Auswahl und Verwendung der Cloud-eServices.

5 Zusammenfassung und Ausblick

Unsere Reifemodell-Analyse hat gezeigt, dass bisher nur sehr wenige Anbieter von Cloud-eServices in der Lage sind Ihre Angebote komplett online anzubieten. Gerade die Bereiche „Certification“ und „E-Learning“ sind bisher gar nicht oder wenig vorhanden. Dazu sollte in zukünftigen Analysen aber auch vorhanden Partnerportale der Unternehmen mitbetrachtet werden. So könnten künftig auch individuelle Reports zu verschiedene Anwendungen oder integrierter Support, beispielsweise bei einer Implementierung, angeboten werden. Schließlich wäre es möglich, dass die unterschiedlichen Unternehmen Ihren Kunden, durch die Entwicklung eines Service Marktplatzes für eServices bei der Bestellung und Benutzung, entgegenkommen.

Literaturverzeichnis

[Bö09] Böhm, M.; Leimeister, S.; Riedl, C.; Krcmar, H.: Cloud Computing: Outsourcing 2.0 oder ein neues Geschäftsmodell zur Bereitstellung von IT-Ressourcen? In: Information Management und Consulting 24, 2009.

[Br02] Bruhn, M.: Electronic Services - eine Einführung in den Sammelband. In: Electronic Services - Dienstleistungsmanagement Jahrbuch 2002, Wiesbaden, 2002, S. 3-42.

[FL05] Forzi, T.; Laing, P.: Leitfaden: Praxisnahe Vorgehensweise zur Bewertung und Auswahl elektronischer Marktplätze; Aachener Competence Center – Electronic Commerce (ACC-EC).

[HP09] Herzwurm, G.; Pietsch, W.: Management von IT-Produkten - Geschäftsmodelle, Leitlinien und Werkzeugkasten für softwareintensive Systeme und Dienstleitungen, dpunkt-Verlag, Heidelberg, 2009, S. 206-210.

[HM02] Hünerberg, R.; Mann, A.: Das Dienstleistungspotenzial des Internet. In: Electronic Services - Dienstleistungsmanagement Jahrbuch 2002, Wiesbaden, 2002, S. 43-66.

[IB08] IT und E-Business im Mittelstand 2008, IBM-Studie, URL: http://www.impulse.de/downloads/impulse_IBM_Studie_2008.pdf, Abruf vom 13.09.2010.

[Ko07] Kollmann, T.: E-Business. 2. Aufl., Wiesbaden : Gabler, 2007.

[Kr09] Kraus, M.: Cloud Computing und Services - Status quo und Trends in Deutschland, 2009.

[MS08] Meier, A.; Stormer, H.: eBusiness & eCommerce: Management der digitalen Wertschöpfungskette, Springer, Berlin.

[Mü09] Münzel, G. et. al.: Cloud Computing - Evolution in der Technik, Revolution im Business, BITKOM-Leitfaden, Berlin, 2009; S. 13-17.

[Ös01] Österle, H.; Fleisch, E.; Alt, R.: Business Networking: Shaping Collaboration between enterprises, Springer, Berlin, 2001.

[Pe00] Peters, R.: Elektronische Märkte und automatisierte Verhandlungen. In: Wirtschaftsinformatik, 2000, 42. Jg., Nr. 5, S. 413-421.

[Re03] Reichmayr, C.: Collaboration und WebServices: Architekturen, Portale, Techniken und Beispiele, Springer, Berlin, 2003.

[Sc00] Schneider, D.; Schnetkamp, G.: E-Markets, Wiesbaden, 2000.

[SSB05] Schneider K.; Scheer, A.-W.; Bullinger, H.-J.: Service Engineering: Entwicklung und Gestaltung innovativer Dienstleistungen, Springer, Berlin, 2005; S. 469-470.

[VJ10] Velten, C.; Janata, S.: Cloud Vendor Benchmark 2010 - Cloud Computing Anbieter im Vergleich Deutschland, 2010.

[We02] Wegmann, C.: Der E-Services Marketingmix. In: Electronic Services - Dienstleistungsmanagement Jahrbuch 2002, Wiesbaden, 2002, S. 243-262.

[WE10] Wenzel, S; Neumann, S.; Bandulet, F.; Faisst, W.: Electronic Business Services and their Role for Enterprise Software 2009: In Benlian, A.; Hess, T. Busmann; P.: Sofware-as-a-Service: Anbieterstrategien, Kundenbedürfnisse und Wertschöpfungsstrukturen, Gabler, Wiesbaden, 2010.

Strategische Steuerung der Transition von Projekten und Produkten durch Application Lifecycle Management

Stephan Salmann, Christian Horstmann

Management Consulting
Corporate Quality Consulting GmbH
Industriestraße 37
53721 Siegburg
{Stephan.Salmann, Christian.Horstmann}@corporatequality.de

Abstract: Die größte Herausforderung im IT-Management ist heute nicht die „richtige Durchführung" von Projekten, sondern das Erkennen der „richtigen Projekte". Wann ist ein Reengineering-Projekt notwendig und in welchen Situationen ist ein Konvergenzprojekt zielführend? „Richtige Projekte" sind Projekte, die dazu führen, dass der Wertbeitrag der IT für die Unternehmung erhöht wird. In diesem Beitrag wird ein Ansatz zum Application Lifecycle Management vorgestellt, der die rationale Ableitung von Projekten aus den existierenden Produkteigenschaften der IT und somit die strategische Steuerung der Transition von Projekten und Produkten ermöglicht. Dieser Ansatz basiert auf einem dreistufigen Vorgehen aus Messung, Bewertung und der Definition von Maßnahmen. Die im letzten Schritt definierten Projekte bzw. Programme führen zu neuen Produkten oder zur Veränderung der Eigenschaften von bestehenden Produkten.

1 Einführung

1.1 Motivation

Bei der Transition von Projekten und Produkten, sind zwei unterschiedliche Richtungen zu betrachten. Zum einen geht es um den Übergang aus dem Projektlebenszyklus in die Phasen des Produktlebenszyklus. Für die Richtung Projekt-Produkt existieren in Literatur und Praxis eine Vielzahl von Vorgehensweisen und Standards, beispielsweise der Projektmanagementstandard PRINCE2[1] oder der ITIL-Standard für das IT-Service Management[2].

[1] In PRINCE2 wird der Übergang von Projekt zum Produkt im Prozess „Managing Product Delivery" beschrieben [Bu06, S. 35 f.].
[2] In ITIL wird dieser Übergang im Abschnitt „Service-Transition" behandelt [Bo08, S. 21].

Im Gegensatz dazu gibt es für die andere Wirkrichtung vom Produkt zum Projekt, hinter der die Frage steht, wann es sinnvoll ist Projekte anzusetzen, relativ wenige theoretische Ansätze und in der Praxis anerkannte Vorgehensweisen. Dieser Umstand stellt für die Leiter der IT-Abteilungen ein erhebliches Problem dar, da es somit noch nicht möglich ist, die Weiterentwicklung der IT konsequent an strategischen Zielen auszurichten.

Das in diesem Artikel dargestellte Application Lifecylce Management (ALM) ist ein viel versprechender Lösungsansatz zu der oben beschriebenen Problematik. Erst durch die Etablierung eines umfassenden ALMs wird eine Unternehmung in die Lage versetzt das Projektportfolio und Programme anhand rationaler Entscheidungen zu definieren und den Wertbeitrag der IT für das Business kontinuierlich zu erhöhen.

1.2 Vorgehen des Application Lifecycle Managements

Das grundsätzliche Vorgehen zum ALM zeigt die folgende Abbildung 1. Im ersten Schritt werden Messwerte erhoben. Aus den erhobenen Messwerten müssen Bewertungen abgeleitet werden (Schritt 2), die wiederum Grundlage für die Definition von Maßnahmen sind (Schritt 3).

Abbildung 1: Vorgehen des Application Lifecycle Managements (ALM) [Cq09]

1.3 Gliederung des Beitrages

Die Abschnitte zwei, drei und vier dieses Dokumentes orientieren sich an den genannten Vorgehensschritten des Application Lifecycle Managements.

2 Messen der Produkteigenschaften der Anwendungslandschaft

Bei der Auswahl und Definition der Messwerte ist auf folgende Punkte zu achten:

1. Aufwand für die Erhebung,
2. Genauigkeit und Objektivität der Messung,
3. Möglichkeiten die Messwerte zu bewerten.

Zu 1.): ALM kann nur erfolgreich sein und Akzeptanz in der IT-Organisation erlangen, wenn der Aufwand für die Beteiligten in einem vernünftigem Rahmen bleibt. In den meisten Fällen existieren bereits einzelne Kennzahlen oder Kennzahlensysteme. Diese sind bei der Auswahl für ALM zu berücksichtigen und wenn möglich zu verwenden. Beispielsweise liegen im Bereich IT-Operations oftmals Daten über die Anzahl der Change Requests und auch über Ausfallzeiten und Auslastungsgrade vor.

Zu 2.): Da anhand der Ergebnisse des ALMs Entscheidungen begründet werden, die für die Mitarbeiter der IT-Organisation tief greifende Veränderungen bedeuten können, ist es von großer Bedeutung, dass die zu Grunde liegende Messwerte genau und objektiv erhoben werden. Automatisch, unter Zuhilfenahme von Tools gewonnene Messwerte, wie beispielsweise Komplexitätsmaße (cyklische Komplexität, lines of Code), sind nicht durch die subjektive Einschätzung von Individuen beeinflussbar und sind im Hinblick auf Objektivität überlegen. Allerdings ist die Aussagekraft dieser Messwerte auch nicht vollständig objektiv, da die Definition und der Algorithmus für ihre Gewinnung eine subjektive Festlegung darstellt. Bei sehr komplexen Messwerten, die sehr schwierig automatisch zu erheben sind, ist es von daher auch möglich auf die subjektive Einschätzung der IT-Architekten und Entwickler zurückzugreifen (zum Beispiel bei Wartbarkeit oder Flexibilität). Eine möglichst hohe Objektivität und Genauigkeit muss dann durch präzise Definition der Messwerte und der Fragestellung erreicht werden.

Zu 3.): Aus der Vielzahl der möglichen Messwerte sind nur diejenigen sinnvoll, aus denen in den weiteren Schritten des ALM auch normative Aussagen abgeleitet werden können. Dies setzt voraus, dass man sich bereits zum Zeitpunkt der Erhebung über die Bewertungskriterien im Klaren ist.

Das übergeordnete Ziel des ALM ist die Erhöhung des Wertbeitrages der IT für die Unternehmung, welches sich aus dem Verhältnis von Kosten und Nutzen der Anwendungslandschaft ergibt. Von daher sind Kosten und Nutzen Messwerte, die auf jeden Fall für das ALM erhoben werden müssen. Darüber hinaus ist es allerdings auch sinnvoll weitere Messwerte zu integrieren, solange diese auf strategischen Leitlinien basieren, für die Entscheidungskriterien definiert werden können.[3]

Die diskutierten Punkte gelten allgemein für alle möglichen Messwerte. Nachfolgend diskutieren wir die Erhebung von Kosten und Nutzen detaillierter. Zuvor gehen wir auf die notwendigen Vorraussetzungen zur Durchführung einer Messung ein.

2.1 EAM als Vorraussetzung zur Messung der Produkteigenschaften

Die Enterprise Architecture ist nach *Krallmann* und *Schönherr* als Verbindung von Geschäfts- bzw. Organisationsarchitektur und IT-Architektur zu verstehen (siehe Abbildung 2). Da wir mit ALM das Ziel haben die Wirkung der IT auf das Geschäft zu verbessern, ist eine beide Bereiche überspannende Sicht, wie sie durch eine Enterprise Architecture gegeben ist, eine unabdingbare Vorrausetzung.

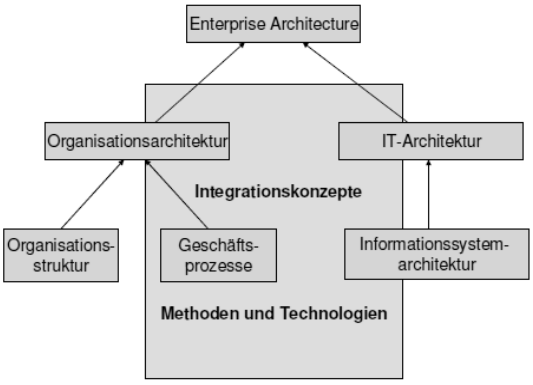

Abbildung 2: Einordnung Enterprise Architecture [KRAL08, S. 104]

Die in der Enterprise Architecture definierten Entitäten sind die Messobjekte des ALM. In der **IT-Architecture** sind dies Anwendungen, wobei hier Aufmerksamkeit auf die Festlegung der angemessenen Granularitätsebene gelenkt werden muss. In modernen Architekturen in denen wiederverwendbare Services sowie übergreifende Systeme zum Masterdata-Management genutzt werden, bilden Anwendungen keine monolithischen Blöcke mehr, die ohne weiteres voneinander abgegrenzt werden können.

Eine allgemeingültige Empfehlung zur Definition der richtigen Granularitätsebene kann nicht gegeben werden, da hier zu viele Situationsparameter zu beachten sind. Mögliche Kriterien zur Definition der Messobjekte sind:

[3] Ein Beispiel für ein System aus Leitlinien, Kriterien und Messwerten wird in Gliederungspunkt 2.1 vorgestellt.

- Was stellt sich aus Sicht des Nutzers als abgeschlossene Einheit (Anwendung) dar?

- Was ist im Hinblick auf die später zu fällenden Entscheidungen ein sinnvolles Handlungsobjekt?

Auf der Seite der **Geschäfts-** bzw. **Organisationsarchitektur** werden Prozess- oder Domänenmodelle verwendet.

Hinter den Domänenmodellen steht die Idee der modularen Organisation, in der jedes einzelne Modul eine autonome und redundanzfreie Einheit bildet [PI03, S. 230 f.]. Im Gegensatz dazu liegt der Fokus bei den Prozessmodellen auf einer vollständigen Modellierung der Leistungsprozesse und ihrer konsequenten Ausrichtung auf die Wünsche des Kunden [HA93, S. 32].

Da die Gestaltungsprinzipien für Domänenmodelle den Gestaltungsprinzipien auf der Ebene der IT stark ähneln, sind Domänenmodelle besonders geeignet, um eine Verbindung zwischen IT- und Geschäftsarchitektur herzustellen.

Im Enterprise Architecture Management wird diese Verbindung durch Bebauungspläne realisiert, in denen angegeben wird, in welchen Domänen der Unternehmung die Anwendung verwendet wird. In folgender Abbildung 3 ist ein Beispiel für einen Bebauungsplan angegeben.

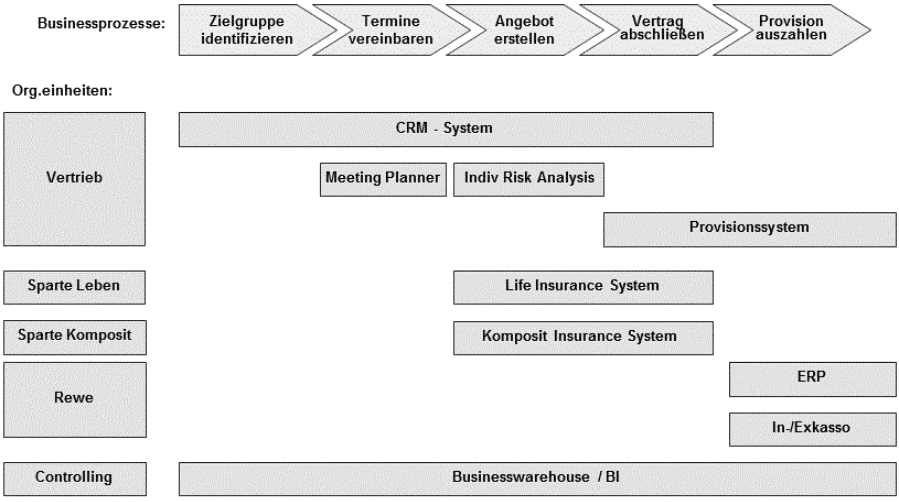

Abbildung 3: Beispiel eines Bebauungsplans: Vertretervertrieb

Die dem Bebauungsplan zu Grunde liegenden Verbindungen werden verwendet, um den Wertbeitrag der IT für das Business zu ermitteln (siehe Abschnitt 2.3). Das EAM bildet mit Modellen und Objekten, die sich im EAM-Repository befinden, den zentralen Datenpool zur Erfassung der Messwerte.

2.2 Messung der Kosten

Wenn es möglich ist, die Kosten für Anwendungen vollständig und richtig zu erfassen, ist ein wesentlicher Grundstein für ALM gelegt. Obwohl in den allermeisten Fällen bereits ein Kostenrechnungssystem in der IT vorliegt, sind für eine Kostenrechnung, die den Anforderungen des ALMs genügt, einige zusätzliche Punkte zu überwinden. Das Grundprinzip der Kostenrechnung im ALM wird in der Abbildung 4 dargestellt.

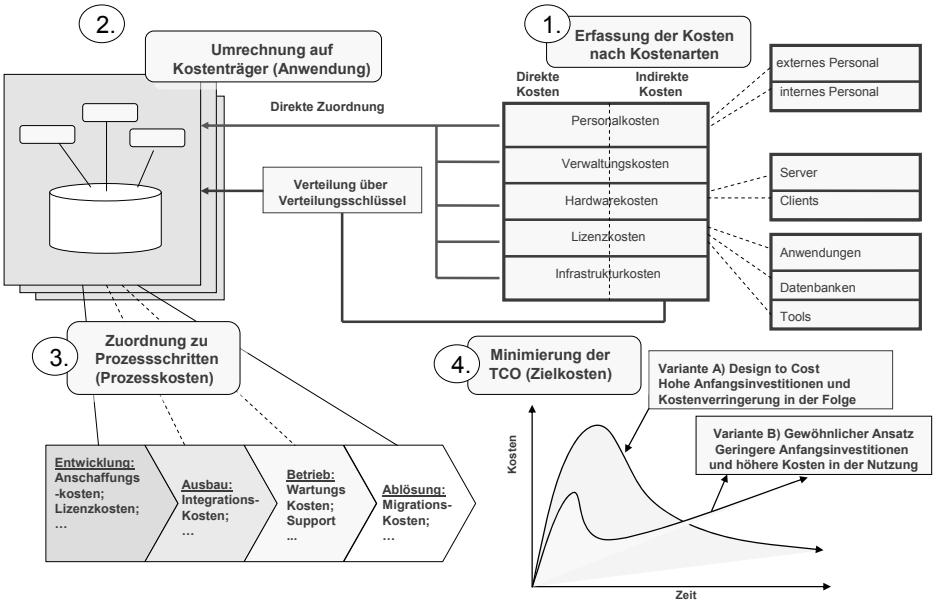

Abbildung 4: Kostenrechnung im ALM [Cq09]

Im ersten Schritt werden die angefallenen Kosten nach Kostenarten erfasst. Wichtige Kostenarten sind Personalkosten, Hardware- und Infrastrukturkosten sowie Lizenzkosten. Bei den Personalkosten sind sowohl die Kosten für externes Personal, als auch die Kosten für internes Personal über kalkulatorische Kostensätze, zu erfassen.

Die direkten Kostenanteile dieser Kostenarten können unmittelbar auf die Kostenträger, die einzelnen Anwendungssysteme, umgelegt werden (zum Beispiel beim Einsatz eines externen Programmierers, der eine Erweiterung an einer Anwendung vornimmt). Für die Zuordnung der indirekten Kosten müssen Umrechnungsschlüssel definiert werden.

Ein möglicher Umrechnungsschlüssel für Kosten, die nicht direkt umgelegt werden können, wie beispielsweise Kosten für Management, ist die Größe der Anwendung. Ein in Wissenschaft und Praxis etabliertes Maß hierfür sind Function Points [Ga06, S. 242]. Die Umrechnung von indirekten Kosten auf die Kostenträger erfolgt dann im Verhältnis zu der Größe der Anwendung gemessen in Function Points. Die Verwendung von Function Points bietet auch bei der späteren Bewertung große Vorteile, da es die Möglichkeit liefert Messwerte auf ein einheitliches Maß zu skalieren.

Im Schritt drei werden die Kosten im Hinblick auf die Lebenszyklusphase der Anwendung differenziert. Da die Kostenarten oftmals nach Betriebs-, Entwicklungs- und Ablösungskosten differenziert werden, ist hierfür meistens kein besonderer Aufwand notwendig.

Das übergeordnete Ziel auf der Kostenseite besteht darin, die TCO, also die Kosten über den gesamten Lebenszyklus der Anwendung zu minimieren (Schritt vier). Hier verbirgt sich eine zentrale Problematik des ALM, nämlich die Tatsache, dass man nur eine Momentaufnahme der aktuellen Kostensituation (oder auch bezogen auf andere Messwerte) vorliegen hat. Auf diese Problematik wird im Gliederungspunkt 3 näher eingegangen.

2.3 Messung des Nutzens

Die Messung der Kosten erfolgt auf Basis von objektiven, quantitativen Messwerten, den pagatorischen oder auch kalkulatorischen Kostengrößen, die teilweise direkt zugeordnet werden können. Die Bewertung des Nutzens einer Anwendung ist nicht so ohne weiteres möglich, da alle Erlöse bzw. Erträge der Unternehmung nur indirekt der IT zugeordnet werden können. Ein Ansatz zur Ermittlung des Nutzens einer Anwendung, der auf einer serviceorientierten Architektur aufbaut, ist in folgender Abbildung 5 dargestellt.

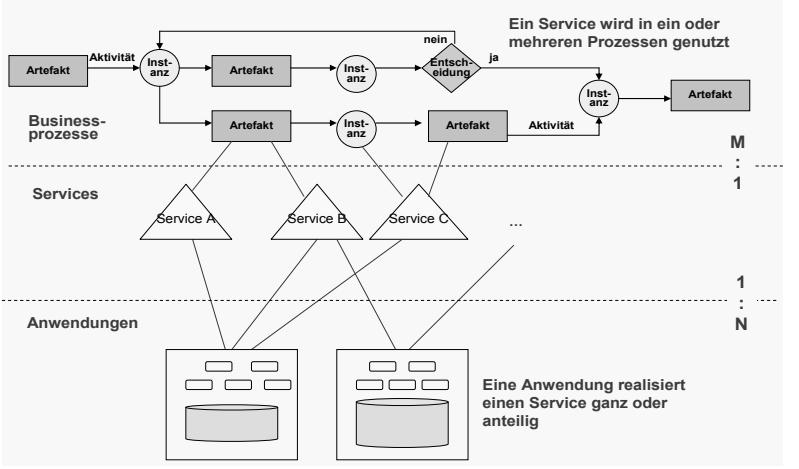

Abbildung 5: Nutzenmessung [Cq09]

Die Berechnung des Nutzens erfolgt hier in zwei Schritten. Im ersten Schritt wird die Ergebnissverbesserung in den Geschäftsprozessen durch die Nutzung der von der IT bereitgestellten Services ermittelt. Eine Ergebnisverbesserung ergibt sich entweder durch eine Erhöhung des Umsatzes oder durch eine Verringerung der Prozesskosten, wobei die IT-Kosten bei den Prozesskosten nicht eingerechnet sind.

In der nachfolgenden Abbildung 6 ist ein Beispiel für die Umrechnung der Ergebnisverbesserung bzw. des Nutzens auf einzelne Services angegeben.

Prozesse / Services		Material-wirtschaft	Produktion	Rechnungs-wesen	Einkauf	Vertrieb	**Ergebnisverbesserung in den Prozessen ohne IT-Kosten**
Berechnung Umsatzsteuer	Personalkosten			100 000	30000	20000	**150 000**
	Kosten Büromat.			5000	1000	1000	**7000**
	Summe der Ergebnisverbesserungen durch Service „Umsatzsteuer“						**157 000**
Portfoliodarstellung der Assets	Personalkosten			100 000			**100 000**
Kundenaufträge im Internet	Umsatzzuwachs	Über alle Bereiche 500 000 bei einer Umsatzrendite von 20 %					**100 000**
Verwaltung Materialstamm-daten	Personalkosten	200000	50000				**250 000**
	Verringerung Ausschuss		150000				**150000**
	Summe der Ergebnisverbesserungen durch Service „Materialstamm“						**400 000**

Verteilung des Nutzens auf Service

Abbildung 6: Umrechnung des Nutzens von Prozessen auf Services [Cq09]

Der Service „Berechnung Umsatzsteuer“ führt im Prozess Rechnungswesen zu einer Ergebnisverbesserung von 100 000 GE für Personalkosten und von 5000 GE für Büromaterial. Insgesamt liefert der Service über alle Prozesse zu einer Ergebnisverbesserung von 157 000 GE. Im nächsten Schritt müssen die durch die Ergebnisverbesserung ausgedrückten Nutzenwerte der einzelnen Services auf die Anwendungen umgelegt werden, die die Services bereitstellen.

Hierzu ist es notwendig anzugeben, welche Services eine Anwendung in welchem Ausmaß realisiert. In Abbildung 7 ist dargestellt wie diese Umrechnung erfolgt und wie schließlich ein Gesamtwert für den Nutzen einer Anwendung ermittelt wird.

Services / Applika-tionen	Berechnung Umsatzsteuer	Portfolio-darstellung der Assets	Kundenaufträge Im Internet annehmen	Verwaltung Materialstamm-daten	**Nutzen der Anwendung**
ERP Anwendung	0,7 * 157 000 = 109 900	0,5 * 100 000 = 50 000	0	0,4 * 400 000 = 160 000	**319 900**
Internet Anwendung	0,1 * 157 000 = 15 700	0	1,0 * 100 000 = 100 000	0	**15 700**
Supply-Chain System	0,2 * 157 000 = 31 400	0,5 * 100 000 = 50 000	0	0,6 * 400 000 = 240 000	**321 400**

Verteilung des Nutzens auf Anwendungen

Abbildung 7: Umrechnung Nutzen auf Anwendungen [Cq09]

3 Bewerten

Es wurde bereits darauf hingewiesen, dass die Bewertung von Messwerten nur anhand definierter Kriterien erfolgen kann. Die Ableitung von Bewertungskriterien aus Leitlinien thematisieren wir im ersten Teil dieses Abschnitts. Dann zeigen wir, wie wir aus eindimensionalen und mehrdimensionalen Kriterien bewertende Aussagen gewinnen können.

3.1 Ableitung von Bewertungskriterien aus Leitlinien

Die Erhebung von Messwerten ist kein Selbstzweck. Der oftmals geäußerte Wunsch nach mehr Transparenz, ist alleine keine Begründung. Transparenz über gewisse Sachverhalte macht nur Sinn, wenn dieses mehr an Informationen zu Verbesserungen von Prozessen oder Entscheidungen führt. Damit dies erreicht wird, müssen die zu erhebenden Messwerte logisch aus übergeordneten Leitlinien abgeleitet werden. Dieser Zusammenhang wird in Abbildung 8 anhand eines Beispiels verdeutlicht.

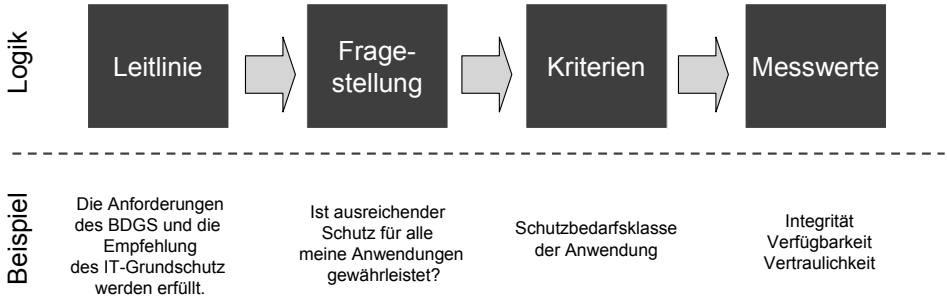

Abbildung 8: Ableitungslogik zur Erhebung von Bewertungskriterien und Messwerten [Cq09]; vgl. Goal-Question-Metrics Paradigma [Wall01, S. 35 f.]

Ausgangspunkt ist eine strategische Leitlinie des IT-Managements, die besagt, dass alle relevanten Compliance-Anforderungen, wie beispielsweise das Bundesdatenschutzgesetz, erfüllt werden sollen. Aus dieser Leitlinie leitet sich eine konkrete Fragestellung ab: Ist ausreichender Schutz für alle Anwendungen gewährleistet? Aus dieser Fragestellung wiederum ergibt sich das Kriterium, welches am Ende zur Entscheidung herangezogen wird.

Bezogen auf das genannte Beispiel ist dies die Zugehörigkeit einer Anwendung zu einer Schutzbedarfsklasse, da sich aus dieser die notwendigen Schutzbedarfsmaßnahmen ableiten lassen. Die Schutzbedarfsklasse schließlich erfordert die Erhebung der Messwerte Integrität, Verfügbarkeit und Vertraulichkeit. Das verwendete Beispiel aus dem Bereich Compliance-Anforderungen steht in keinem unmittelbaren Zusammenhang mit dem ALM, welches sich vor allem auf die Wirtschaftlichkeitskriterien Kosten bzw. Nutzen bezieht.[4]

Allerdings ist es aus folgenden Gründen empfehlenswert bei der Durchführung der Datenerhebung für das ALM einen möglichst breiten Blickwinkel einzunehmen. Zum einen können die Leitlinien nicht isoliert betrachtet werden, da sie bei der späteren Entscheidungsfindung alle zu beachten sind. Zum anderen ist auch im Hinblick auf den Aufwand der Erhebung und die Akzeptanz eine einmalige Erhebung pro Periode wünschenswert.[5]Insgesamt ergibt sich eine Gesamtkonzeption aus Prinzipien, Fragestellungen, Kriterien und Messwerten (siehe Abbildung 9).

[4] Siehe hierzu Gliederungspunkt 2.
[5] Hinter dem integrierten System aus Leitlinien, Kriterien und Messwerten steht auf der Ebene der Toolunterstützung ein integriertes EAM-Repository (siehe Gliederungspunkt 2.1).

Nr.	Kriterium	Fragestellungen	Auswertungen	Messwerte
1.	**Strategiekonformität** gewährleistet eine optimale Unterstützung des Business durch die IT und bestimmt die Ausrichtung am Geschäft.	Unterstützt die IT die Anforderungen der Unternehmensstrategie?	Matrix Unterstützung Strategische Ziele/Anwendungen	Unterstützung strategischer Unternehmensziele
2.	**Wirtschaftlichkeit** in der IT optimiert den Wertbeitrag im Business.	Liefert die Applikation einen positiven Wertbeitrag für die Geschäftprozesse?	Kosten - Nutzen Portfolio	Domänenvalue, Anzahl Nutzer, Businessvolumen, Automationsgrad, Wartungskosten Betrieb, Wartungskosten Entwicklung, Kosten Betrieb Netze, Infrastruktur und Hardware
2b	Die Anwendungen aus den vier **Anwendungsklassen** sollen proportional jeweils gleich hohe Kosten verursachen.	Ist der Aufwand, der jeweils in die Anwendungsklassen Gold, Silber, Bronze und Blau gesteckt wird, angemessen?	Chart zur Aufwandsverteilung in den vier Kategorien	Kritikalität der Anwendung; Aufwand für Wartung und Betrieb
3.	Durch **Redundanzfreiheit** werden die Kosten in der IT verringert.	Welche Systeme / Anwendungen weisen Redundanzen bei Funktionen und Daten auf?	Bebauungsplan und Konvergenzanalyse	Funktionale Redundanz, Datenredundanz
4.	Durch eine hohe **Wartbarkeit** kann auf die Anforderungen des Kunden flexibel und wirtschaftlich reagiert werden.	Ist die Wartbarkeit einer Anwendung, im Hinblick auf die Häufigkeit von Änderungen, angemessen?	Portfolio Wartbarkeit vs. Änderungshäufigkeit	Skillverfügbarkeit, Dokumentation, Parametrisierungsgrad, Modularisierungsgrad, Struturierungsgrad, Anzahl der Schnittstellen, Kopplungsgrad der Schnittstellen, Komplexität, Aufwand je FP, Anzahl Change Request je Function Point, Größe der Anwendung nach Function Point
5.	Kernsysteme sind nach Möglichkeit frei von **externen Abhängigkeiten** zu gestalten.	Wie hoch ist der Anteil an Fremdfertigung bei Kernanwendungen und Nicht-Kernanwendungen?	Anteil an Fremd- und Eigenentwicklung in Kern- und Nicht-Kernsystemen	Ist Kernsystem/Nicht Kernsystem, Eigenfertigung/Fremdfertigung
6	**Standardsoftwareprodukte** sind, wenn sie den Anforderungen genügen, im Bereich der Nicht Kernanwendungen bevorzugt einzusetzen.	Wie hoch ist der Anteil von Standardsoftware im Bereich der Nicht-Kernanwendungen	Verhältnis Standardsoftware zu Individualsoftware	Ist Standardsoftware/Individualentwicklung
7.	**Architekturkonformität** und die damit verbundene Standardisierung verringern die Kosten bei Änderungen.	Folgt die Applikation den Architekturstandards? Verwendet die Applikation den definierten Technologie Warenkorb (siehe Komponentenkatalog Systemarchitektur)?	Standardisierungsgrad je Applikation und für alle Applikationen	Konformität bzgl. Architekturvorgaben Konformität bzgl Technologie Warenkorb
8	Ein ausgewogenes **Build/Run Verhältnis** eröffnet der IT neue Handlungsspielräume.	Werden 30% der IT-Aufwendungen zum Change der Systeme verwendet? Bei welchen Systemen ist dieses Verhältnis nicht gegeben?	Übersicht Aufwände für Build und für Run	Budget für Wartung; Budget für Entwicklung
9	Das **Lifecycle - Portfolio** ist ausgewogen zu gestalten, da sonst Investetionstaus entstehen.	Verteilen sich die Anwendungen ausgewogen über das Lifecycle-Portfolio oder bestehen Investitionsstaus?	Portfolio Änderungsvolumen über die Lifecyclephasen	Lifecycle Phase
10	Für die Unterstützung des Business von "morgen" ist die **Zukunftsfähigkeit** der Applikationen unerlässlich.	Sind die Anwendungen auf bereits erkennbare zukünftige Veränderungen hin ausgerichtet?	Chart zur Zukunftsfähigkeit der Anwendungen	Zukunftsfähigkeit

Abbildung 9: Konzeption von Leitlinien, Fragestellungen und abgeleiteten Messwerten[6] [Cq09]

[6] In der Abbildung werden in der dritten Spalte keine Kriterien genannt sondern Auswertungen aufgeführt. Dies bedeutet inhaltlich das gleiche, da hinter den Auswertungen die Kriterien stehen, die bei der Entscheidungsfindung helfen sollen.

3.2 Bewertung anhand eindimensionaler Kriterien

Die Bewertung von erhobenen Messwerten ist am einfachsten, wenn für den jeweiligen Messwert absolute Ausprägungen als Bewertungskriterien herangezogen werden. Die von uns bereits als Beispiel angeführte Erfüllung von Sicherheitsanforderungen ist ein eindimensionales Kriterium, bei dem die Anforderungen entweder erfüllt sind oder nicht. Die Entscheidungen leiten sich hier ebenfalls einfach ab. Falls die Anforderungen nicht erfüllt sind, müssen Maßnahmen ergriffen werden, damit diese erfüllt werden.

3.3 Bewertung anhand eines Portfolios (mehrdimensionale Kriterien)

In vielen Fällen ist eine Entscheidung nur unter Nutzung von mehrdimensionalen Kriterien möglich, beispielsweise bei der Identifikation der Lebenszyklusphase, dem Nukleus des ALMs. Ausgangspunkt sind die erhobenen Kosten und Nutzerwerte.[7]

Eine Anwendung ist dann als wirtschaftlich einzustufen, wenn sie mehr Nutzen liefert als sie Kosten verursacht, sie somit eine hohe Effizienz aufweist. Dies führt zu der in der Abbildung 10 dargestellten Auswertung.

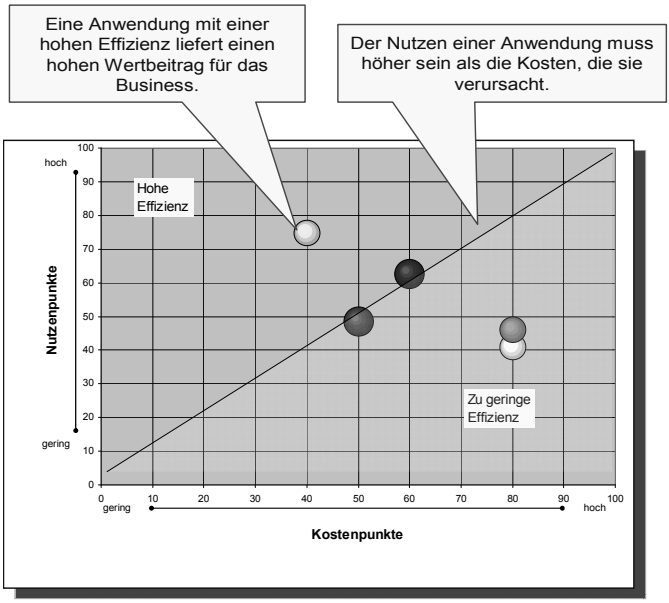

Abbildung 10: Auswertung Wirtschaftlichkeit [Cq09]

[7] Siehe Gliederungspunkt 2.2 und 2.3.

Diese Bewertung der Kosten- und Nutzenmesswerte ist aber noch kein ausreichender Indikator zur Ableitung von Entscheidungen, da es sich lediglich um eine Momentaufnahme handelt. Anwendungen durchlaufen einen Lebenszyklus, wobei sie in den unterschiedlichen Phasen einen unterschiedlichen Wertbeitrag liefern (siehe Abbildung 11). Damit die Lebenszyklusphase einer Anwendung identifiziert werden kann, müssen die zukünftigen Veränderungen des Nutzen-Kosten Verhältnis antizipiert werden.

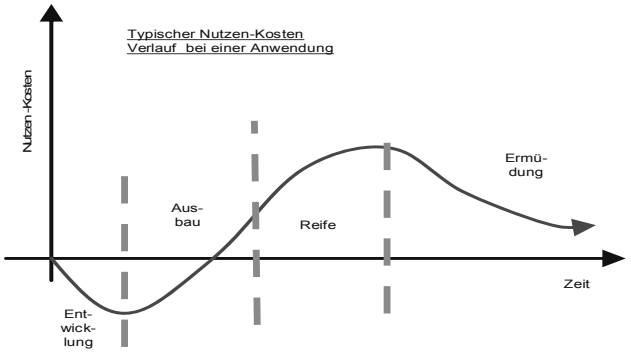

Abbildung 11: Lebenszykluskurve einer Anwendung [Ba06, S. 159]

Dies geschieht durch eine Messung der Veränderung des Verhältnisses in der Vergangenheit, was eine mehrperiodische Messung voraussetzt. Nehmen wir den obigen Graph des Lebenszyklusverlaufes als gegeben an, so können wir anhand des aktuellen Nutzen/Kostenverhältnis als auch über die Veränderung des Verhältnisses im Vergleich zur Vorperiode (erste Ableitung) folgende Aussagen treffen (siehe Abbildung 12).

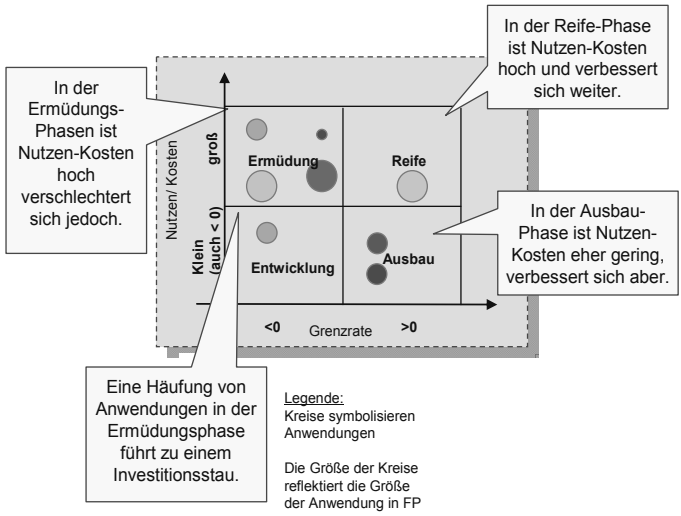

Abbildung 12: Ableitung LC-Portfolio [Cq09]

Die Einschätzung der zukünftigen Entwicklung der Anwendung und somit ihre Lebenszyklusphase kann durch die Definition von konkreten Fragen bzw. Messwerten zur Zukunftsfähigkeit ergänzt und unterstützt werden. Beispiele für derartige Fragen sind:

1. Ist die Anwendung darauf vorbereitet neuen Vertriebspartner einen schnellen und sicheren Zugriff auf das System zu ermöglichen?

2. Ist das System auf die Einführung von neuen bzw. die Änderung von bestehenden Produkten vorbereitet?

3. Ist die Anwendung auf die Migration von Daten aus Anwendungen von übernommenen Unternehmen vorbereitet?

4. Sind absehbare Änderungen von Vorschriften und Gesetzen in der Anwendungen umsetzbar?

Die Bewertung der Zukunftsfähigkeit kann entweder in einer eigenen Auswertung erfolgen oder als gewichteter Faktor in die Abszisse des LC-Portfolios mit einfließen.

4. Maßnahmen ableiten

Auf Basis der vorhergehenden Analyse lassen sich Projekte und Programme – zwar nicht automatisch ableiten – aber zumindest rational begründen. Wir gehen von der Situation aus, dass sich aus der vorhergehenden Analyse ein erheblicher Handlungsbedarf aufgrund von Defiziten in mehreren Kriterien ergibt. Damit nun die richtigen Projekte definiert werden, die dazu führen, dass strategische Ziele erreicht werden müssen folgenden Punkte beachtet werden:

1. Zusammenfassung von Projekten zu einem Programm mit einer strategischen Stoßrichtung;

2. Definition von Ziel-Messwerten und Evaluierung des Fortschrittes.

4.1 Definition von Projekten und Programmen

Bei der Definition von Projekten können nicht alle aufgezeigten Defizite zusammen behoben werden. Dies liegt zum einen daran, dass Komplexitätsgrenzen vorliegen. Zum anderen aber auch daran, dass die Leitlinien und Kriterien oftmals konkurrierende Ziele darstellen, beispielsweise die Forderung nach Kosteneinsparung und einer Erhöhung der Regelkonformität bzgl. der gültigen Compliance-Anforderungen.

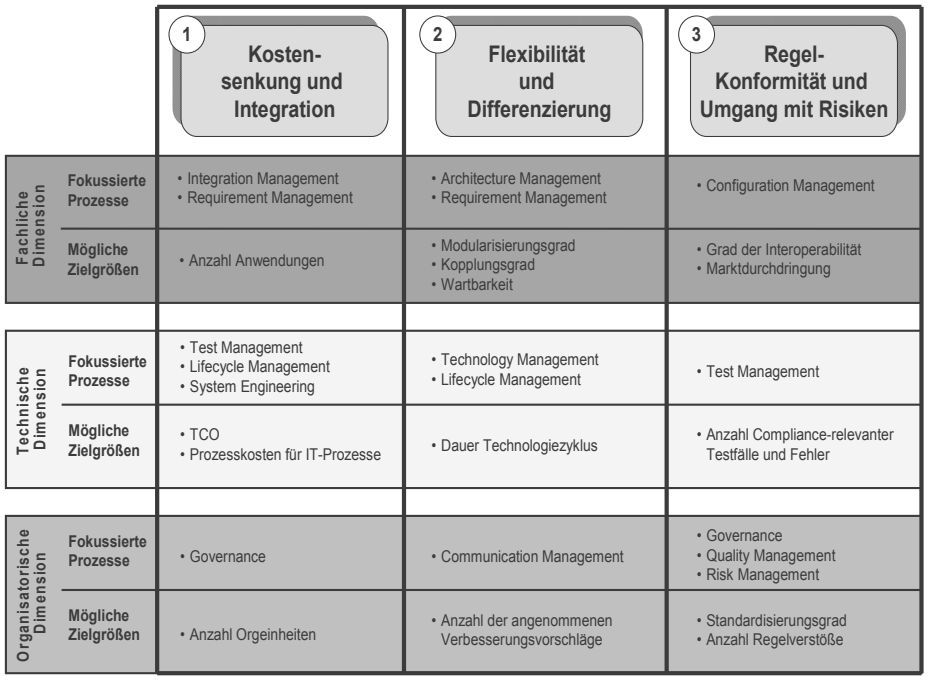

Abbildung 13: Standardprogrammtypen [Cq10]

Abbildung 13 zeigt die wesentlichen Standardprogrammtypen mit den jeweils zu betonenden Managementprozessen und einer sinnvollen Kombination von Zielgrößen, die in einem Programm erreicht werden können.

Auch wenn Anpassungen oder Erweiterungen dieser Standardtypen möglich sind, sollte die strategische Fokussierung nicht aufgegeben werden. Der strategische Fokus sollte dort gelegt werden, wo sich aus den Ergebnissen des ALMs der größte Handlungsbedarf ergeben hat, beispielsweise bei Integration und Kosten.

4.2 Umsetzung des Programms

Alle im Programm definierten Projekte müssen einen definierten Beitrag zur Erreichung der strategischen Ziele haben. Die Überprüfung des Zielerreichungsgrades des Programmes muss über alle Phasen vorgenommen werden (siehe Abbildung 14).

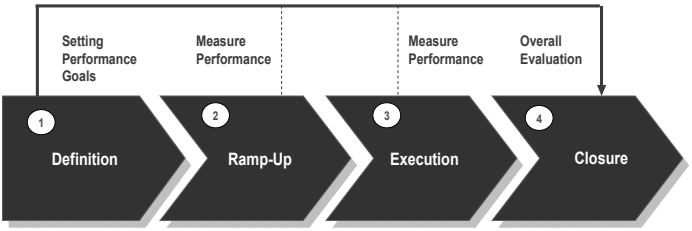

Abbildung 14: Umsetzung von Programmen [Cq10]

Als Grundlage für die Messung können die bereits im ALM definierten Messwerte herangezogen werden. Sollen konkurrierende Ziele in der IT erreicht werden, wie beispielsweise Kostensenkung und Flexibilisierung, ist es sinnvoller zwei getrennte Programme aufzusetzen.

Literaturverzeichnis

[Ba06] Babu, M.: Offshoring IT Services: A Framework for Managing Outsourced Projects. Tata McGraw-Hill, New Delhi 2006.

[Be06] Bentley, C.: Prince2 revealed. Butterworth-Heinemann, Oxford 2006.

[Bo08] Bon, J. et al.:Service Transition basierend auf ITIL V3 – Eine Management Guide. Van Haren, Zaltbommel 2008.

[Cq09] CQ Consulting GmbH: Foliensatz zum ALM für einen Versicherungskonzern. Unveröffentlicht, Siegburg 2009.

[Cq10] CQ Consulting GmbH: Foliensatz zum Programm Management. Unveröffentlicht., Siegburg 2010.

[Ga06] Galorath, D.; Evans, M.: Software Sizing, Estimation, and Risk Management, Auerbach Publications, Boca Raton 2006.

[Ha93] Hammer, M.; Champy, J.: Reengineering the Corporation. Harper-Business, New York 1993.

[Kral08] Krallmann, H.; Schönherr, M.: Nachhaltige Flexibilisierung von Unternehmen durch das Management Service-orientierter Architekturen als Instrument der Beschäftigungssicherung. In: Zülich, G.(Hrsg.): Beiträge der Arbeits- und Betriebsorganisation zur Beschäftigungssicherung. Gabler, Wiesbaden 2008, S. 101-125.

[Pi03] Pico, A. et al.: Die grenzenlose Unternehmung, 5. Aufl., Gabler-Verlag, Wiebaden 2003.

[Wall01] Wallmüller, E.: Software Qualitätsmanagement in der Praxis. 2. Aufl., Hanser, München 2001.

Wertgetriebene Softwarewartung

Harry M. Sneed

Testabteilung
ANECON GmbH, Wien
harry.sneed@t-online.de

Abstract: Der folgende Beitrag überträgt das Konzept von "value-based software engineering" wie vom Boehm ursprünglich vorgeschlagen auf die Softwarewartung. Zunächst werden verschiedene Ansätze zur Bewertung eines Softwaresystems und zur Berechnung eines "return of investment (ROI)" vorgestellt. Anschließend schlägt der Autor eine Bewertungsmethodik vor, die auf Hayek's Theorie der Werterhaltung von Kapitalgüter in einer schnell wandelnden Wirtschaft fasst. Einerseits werden die Kosten von Wartungsarbeiten auf der Basis einer Impactanalyse kalkuliert. Anderseits wird der Nutzen, der durch die Wartungsarbeiten entsteht, mit Hilfe der Hayekischen Wertsteigerungstheorie berechnet. Kosten und Nutzen fließen in die Kalkulation der Wartungsamortisationsberechnung ein. Eine Fallstudie aus der industriellen Praxis illustriert die Anwendung des Ansatzes.

1 Value driven Software Engineering

Der "value-based" Software-Engineering Ansatz kombiniert eine Reihe altbewährter Software-Engineering- Prinzipien darunter "participatory design", "user engineering", "cost estimation", "software economics", "software investment analysis" und "software engineering ethics". Wertgetriebene Software-Engineering umfaßt u.a. folgende wertgetriebene Aktivitäten:

- Wertgetriebene Requirements-Engineering mit der Gewichtung und der Prioritätssetzung der Anforderungen nach Wirtschaftlichkeitskriterien
- Wertgetriebene Softwarearchitektur mit der Versöhnung der Systemziele mit erreichbaren architektonischen Lösungen.
- Wertgetriebene Software-Design, mit Schwerpunkt auf die Auslegung des Designs um den Projektzielen und Wirtschaftlichkeitsüberlegungen gerecht zu werden.
- Wertgetriebener Test nach dem die Funktionen mit dem höchsten am intensivsten getestet werden.
- Wertgetriebene Projektplanung- und -steuerung mit dem Vorziehen jener Aufgaben die den höchsten Kundenwert versprechen.
- Wertgetriebene Risikoanalyse bei der jene Risiken mit dem höchstmöglichsten betriebswirtschaftlichen Verlustpotential identifiziert werden.

- Wertgetriebenes Qualitätsmanagement durch die Bestimmung der Qualitätsziele nach ihrem Nutzen für den Kunden.
- Wertgetriebene Entwicklungsprozesse in denen der Kundenwert im Mittelpunkt der Arbeitsaufteilung steht.

Diese Forschungsagenda wird von der "Economics-driven Software Engineering research community" mit der Website (www.edser.org) vorangetrieben [Blom08].

Der Kerngedanke der wertgetriebenen Softwareentwicklung ist, dass jeder einzelne Arbeitsschritt ein Ergebnis hervorbringen sollte, der an und für sich einen Wertgewinn für den Auftraggeber darstellt, d.h. jedes Dokument, jedes Werkzeug, jedes Teilmodell, jede Codekomponente und jeder Testfall soll sich wirtschaftlich rentieren. Es soll ein positives „Return on Investment" haben nach der Gleichung:

$$ROI = (benefit - cost) / costs$$

Jedes Arbeitsergebnis, egal wie klein, stellt einen Wert dar. Es verursacht auch Kosten. Seine Kosten dürfen seinen Wert niemals überschreiten. Ist dies der Fall, darf das Ergebnis nicht erst produziert werden. Es rentiert sich nicht. Ergebnisse mit dem höchsten ROI sind vorzuziehen. Jene mit einem niedrigen ROI sind zurückzustellen. Gerade im Zeitalter der agilen Softwareentwicklung sollen Projektverantwortlichen eine Leitlinie haben nach der sie entscheiden können was als nächstes zu machen ist. Diese Leitlinie ist der Wertbeitrag der jeweiligen Aufgaben [BoTu05].

In einem iterativen Entwicklungsprojekt gehören jene Bausteine mit dem höchsten Wertbeitrag in der ersten Iteration. Nach jeder Iteration werden die anstehenden Aufgaben neu bewertet um den Inhalt der nächsten Iteration zu bestimmen. Der Kundenwertbeitrag ist das Kriterium nach dem die Ziele einer jeden Iteration, bzw. eines jeden Sprints, gesetzt werden. Dies schließt technische Überlegungen nicht aus, z.B. wenn eine Komponente die technische Voraussetzung für die Nächste ist, aber die wirtschaftliche Überlegung soll immer im Vordergrund stehen. Sie bestimmt die Reihenfolge der Implementierung und die Intensivität des Tests, die sich in der Testüberdeckung ausdruckt [Mell05].

In seinem Buch "Value-based Software Engineering" fasst Biffl die Hauptfaktoren bei der Bewertung einer Software-Engineering Aktivität zusammen. Diese sind:

- der Nutzen eines Produktes oder einer Dienstleistung minus die Kosten zur Bereitstellung derselben.
- der prozentuale Beitrag der Aktivität zum Nutzen des Gesamtproduktes bzw. zur gesamten Dienstleistung
- die Kosten der Aktivität relativ zu den Gesamtkosten des Produktes, bzw. der Dienstleistung.

Da Nutzwert relativ ist, muss der Nutzwert einer jeden Aktivität im Bezug zum Nutzwert des Ganzen betrachtet werden. Aktivitäten, die mehr Nutzen bei weniger Kosten liefern, sind vorziehen. Dies ist auch ein Grundsatz der agilen Softwareentwicklung [Biff06].

Die Voraussetzung für wertgetriebene Software-Engineering ist die Fähigkeit Nutzen, Kosten und Risiken zu quantifizieren. Wenn Nutzen nicht quantifizierbar ist, ist sie auch nicht mit den Kosten vergleichbar. Demzufolge, ist es kaum möglich wertgetriebene Software-Engineering ohne Vermessungssystem einzuführen. Es muss möglich sein, die Funktionalität und Qualität der Software die man bereits hergestellt hat zu messen, um die Funktionalität und Qualität der geplanten Software zu projektieren. Erst wenn diese Voraussetzung erfüllt ist, kann man dazu übergehen Kosten, Nutzen und Risiken miteinander zu vergleichen. Für den wertgetriebenen Ansatz muss sich die Software in Zahlen auslegen lassen [Soli04].

2 Softwarewartung als Werterhaltung von Kapitalgütern

Die Theorie der relativen Werterhaltung geht zurück auf die Forschung des deutschen Volkswirt Franz Schmidt im 19. Jahrhundert. Schmidt vertrat die These, dass der wahre Wert eines Kapitalguts nur relativ zu den anderen Konkurrenzgütern auf dem Markt zu ermitteln wäre. Das hängt wiederum von anderen Faktoren ab wie die Inflationsrate, die Wirtschaftslage und die Marktentwicklung. Nach Schmidt hängt der Wert eines Wirtschaftsgutes von der Verfügbarkeit vergleichbarer Güter ab, vorausgesetzt es hat überhaupt einen Wert. Wo keine Nachfrage ist, ist auch keinen Wert. Ein Produkt hat nur so viel Wert, wie der Benutzer des Produktes ihm beimisst [Schm22].

Im 20. Jahrhundert griff der österreichische Volkswirt Friedrich Hayek Schmidt's These auf und erweiterte sie auf die Erhaltung von Gütern. Hayek interessierte sich für den Wert der Kapitalerhaltung. Ihn ging es im Gegensatz zu den anderen Volkwirten seiner Zeit weniger um die Schaffung neuer Kapitalgüter als viel mehr um die Erhaltung bestehender Kapitalgüter. Er wollte wissen, was es kostet die Güter zu erhalten relativ zu den Kosten einer Neubeschaffung. Hayek schrieb "The question is, how much of the gross receipts from his capital income does the entrepreneur have to reinvest in order to ensure a constant flow of income? Either he may reduce his investment to a minimum until the value of his product has been depreciated by change, or he may reinvest amortization quotas of the same on an increasing magnitude as before." [Haye39]

Auf die Erhaltung von Softwaresystemen übertragen, heißt das, die Besitzer eines Softwaresystems können die Wartungskosten auf ein Minimum reduzieren und zuschauen wie das System im Sinne der Gesetze der Softwareevolution von Belady und Lehman immer weniger Wert wird, oder sie können Kapital verhältnismäßig zu dem was sie in die Entwicklung gesteckt haben in die Verbesserung des Systems investieren. Durch regelmäßige Sanierungen und funktionale Erweiterungen können sie den Wert ihrer Software nicht nur erhalten sondern auch über die Zeit steigern. Ansonsten nimmt der Wert ihres Systems ständig ab. Zum Einen, fehlt die Funktionalität immer weiter hinter den Anforderungen der Benutzer zurück. Um die dringlichsten Anforderungen gerecht zu werden, wird die Software notdürftig geändert und erweitert. Es werden aber auch immer neue Mängel entdeckt. Diese Mängel müssen beseitigt werden.

Die Behebung der Mängel und die Änderung bzw. Erweiterung der Funktionalität werden als Softwarewartung bezeichnet. Mit jedem Eingriff in die Software, ob zur Korrektur eines Fehlers oder zur Änderung einer Funktion, steigert die Komplexität jener Software. Es entstehen neue Beziehungen zwischen Codebausteinen und dadurch mehr Abhängigkeiten relativ zur Codegröße. Gleichzeitig sinkt die Qualität in dem Maße wie die Software sich immer weiter von ihrer ursprünglich beabsichtigten Struktur entfernt. Steigende Komplexität und sinkende Qualität bedeuten Wertverlust. Die Software wird immer weniger Wert [BeLe85].

Hayek erkannte die Verantwortung der Güterbesitzer für die Werterhaltung ihrer Güter. In diesem Zusammenhang betonte er die Notwendigkeit voraus zu planen, wenn es um die Erhaltung der Güter geht. Er fasst die Pflicht zur Werterhaltung wie folgt zusammen: "All this means that the mobility of capital, i.e. the degree to which it can be maintained in a changing world depends on the foresight of the entrepreneurs…It also means that the amount of capital available at any moment in a dynamic society depends much more on the foresight of the entrepreneurs in constructing maintainable goods than on current cost savings or on time to market" [Haye41].

Der Softwaremarkt ist zweifelsohne ein dynamischer Markt und Software als Gut nimmt schnell an Wert ab, wenn sie nicht ständig erneuert wird. Nach der Lehre von Hayek ist der Wert der Softwarewartung gleich der Höhe der Produktabwertung wenn es nicht gewartet wird. Ein Softwaresystem mit einem momentanen Nutzwert von € 10 Million und einer jährlichen Abwertung von 10% wird schon nach 5 Jahren die Hälfte seines Wertes verloren haben und nur noch € 5 Million wert sein wenn es nicht erneuert wird. Um diesen Wertverfall zu verhindern musste in der gleichen Zeit € 5 Million in die Softwarewartung reinvestiert werden. Die gängigen Wartungsgebühren in der Höhe von 15-25% des Kaufwertes bestätigen die Notwendigkeit dieser Investition. Wenn schon die Hälfte dieser Gebühren gegen die ursprünglichen Entwicklungskosten gebucht wird, bleiben noch circa 10% für die Investition in die Werterhaltung, bzw. in die Evolution, übrig. Ein Anwender selbstentwickelter Softwaresysteme müsste demzufolge bereit sein mindestens 10% vom dem was er in die Entwicklung jener Systeme investiert hat, jährlich für die Erhaltung jener Systeme bereitzustellen. Diese Erhaltungskosten müssen auch in die Berechnung der Amortisation der Produkte einfließen.

Der deutsche Volkswirt Lachmann hat die Lehre von Hayek durch den Begriff der Komplementarität ergänzt. Güter sind komplementär wenn der Wert des einen Gutes vom Wert der anderen Güter abhängt. Die Abwertung von Software ist mit dem Begriff komplementärer Systeme zu erklären, d.h. Systeme die gegenseitig abhängig sind. In der Betriebswirtschaft sind die meisten Produkte komplementär. Sie stützen sich aufeinander. Wenn ein Produkt an Wert abnimmt, zieht er den Wert der anderen Produkte mit herunter. Es geht also nicht nur um den Wert eines einzelnen Systems. Sofern die Systeme mit einander integriert sind, geht es um den Wert aller Systeme. Es muss nur ein System an Wert verlieren um den Wert des ganzen Betriebes herunterzuziehen. Dies ist der Preis eines hohen Integrationsgrades.

Software existiert und funktioniert in einer komplementären Umgebung in der fast alles von allem abhängig ist. Diese Umgebung, bzw. die Systemlandschaft eines Unternehmens ist eine Kapitalstruktur die sich mit der Zeit wandelt. Neue Systeme werden eingeführt, andere Systeme werden ersetzt oder erneuert. Wenn ein System stehen bleibt, hat dies eine negative Auswirkung auf die Andere. Ihr Wert wird dadurch vermindert. Da die Systeme komplementär sind müssen, um den Wert des Ganzen zu erhalten, alle Teilsysteme in etwa den gleichen Stand behalten. Zur Werterhaltung von Software gehört die gleichmäßige Evolution aller komplementären Systeme. Lachmann schreibt „for any particular type of capital good, maintenance is a matter of maintaining its complementarity to the rest of the changing capital structure. Hence, maintenance may mean not only preventing any change through deterioration, but actually changing a good directly, in a manner that adapts it to the changing capital structure around it, thereby delaying obsolescence and increasing usefulness." [Lach75].

Weil Wandel in einem dynamischen Markt unausweichlich ist, müssen die Kapitalgüter sich auch wandeln. Hayek weist darauf hin, dass die Erhaltung von Kapitalgütern mehr darauf ausgerichtet ist den Wert jener Güter zu erhalten als den Verfall vorzubeugen. Sowohl Hayek als auch Lachmann sind der Meinung dass komplementärer Produkte mit der Zeit zunehmend komplexer werden. Sie stellen fest: „that over time there develops "an increasing degree of complexity of the pattern of complementary displayed by the capital structures..". Diese Beschreibung passt genau zu Softwareprodukten. Software altert wenn sie ihre Umgebung nicht weiter komplementiert. Der Wert eines Softwaresystems ist direkt proportional zu dem Grad in dem andere Systeme, einschließlich die manuellen Systeme, vom ihm abhängig sind. Wert hängt von der Bedeutung eines Produktes ab und Bedeutung hängt von der Abhängigkeit anderer Akteure im Umfeld des Produktes von diesem Produkt ab. Die Erhaltung eines Kapitalgutes in einer wandelenden Kapitalstruktur, verlangt dass das Gut nicht nur in seinem bisherigen Zustand bleibt, sondern weiterentwickelt wird um mit seiner veränderten Umgebung Schritt zu halten [Baet98].

Nach Hayek lässt sich Kapital als verpacktes Wissen definieren. Softwareprodukte sind demnach Kapitalgüter bei denen das Wissen im Code verpackt ist. Das Wissen das im Code steckt ist das Ergebnis eines organisatorischen Lernprozesses, der sich über Jahre hinstrecken kann. Durch gemeinsames Probieren kommt eine Arbeitsgruppe endlich zu einem annehmbaren Ergebnis. Deren Zwischenergebnisse verkörpern das Wissen, das bis zu diesem Zeitpunkt gesammelt wurde [Tock05]. Deshalb haben sie einen Nutzwert auch wenn sie nicht voll einsatzfähig sind. Der lange Prozess bis hin zu einem brauchbaren Softwareprodukt ist sehr kapitalintensiv. Er bindet Geld und Ressourcen. Drum müsste es ein Hauptanliegen des Kapitalbesitzers sein, dieses Wissen zu erhalten und zu erweitern. Wer dies nicht tut verschwindet Kapital. Der Zweck der Softwarewartung ist die Erhaltung und Vermehrung der Funktionalität eines Softwareproduktes. Der Zweck von Software-Reengineering ist die Erhaltung und Vermehrung der Produktqualität [Snee91].

3 Anwendung wertgetriebener Software Engineering für Softwarewartung

Softwarewartung so wie sie von der IEEE Standard-1219 formuliert ist "includes modification of a software product after delivery or to adapt the product to a modified environment" [IEEE93]. Die Literatur zum Thema Softwarewartung identifiziert vier Klassen von Wartungsaufgaben:

- korrektive Wartung
- adaptive Wartung
- perfektive Wartung
- enhancive Wartung.

Korrektive Wartung befasst sich mit der Behebung von Fehler und sonstiger Mängel. Adaptive Wartung umfasst die Änderungen bestehender Funktionen und Daten, also die Anpassung der Software an geänderte Anforderungen, bzw. die Bearbeitung von Change Requests. Perfektive Wartung zielt auf eine Steigerung der Softwarequalität bei gleichbleibender Qualität. Restrukturierung und Refaktorierung gehören zur perfektiven Wartung. Enhansive-Wartung bedeutet Erweiterung der Software. Zusätzliche Funktionen und/oder Daten werden hinzugefügt, die Größe der Software wächst [Chap01].

Wartungsaufgaben werden durch Fehlermeldungen, Änderungsanträge oder durch neue Anforderungen ausgelöst. Korrektive Wartung wird durch Fehlermeldungen bzw. Error-Reports, adaptive sowie perfektive Wartung durch Änderungsanträge, bzw. Change-Requests und erweiternde Wartung durch neue Anforderungen ausgelöst. Die drei Arten von Wartungsaufträgen sind demnach wie folgt:

Auftragsart		Wartungsart
error report	=	Korrektive
change request	=	Adaptive oder Perfektive
new requirement	=	Erweiterung

Wartungsaufgaben werden in Releases zusammengefasst. Jedes Release, bzw. jede Freigabe einer neuen Version, ist ein Teilprojekt im Lebenszyklusprojekt des Produktes. Zu entscheiden welche Aufgaben einem Release zugewiesen werden, müssen die Aufgaben einzeln bewertet werden, denn da das Produkt bereits im Einsatz ist, hat man immer die Wahl welche Änderungen man vornimmt und welche nicht. Hierbei kann wertgetriebene Software-Engineering eine Rolle spielen. Jede einzelne Aufgabe, sprich Fehlerkorrektur, Änderung, Nachbesserung und Erweiterung wird einer Nutzwertanalyse unterzogen um sie relativ zu den anderen anstehenden Aufgaben zu priorisieren. Dazu wird ihre ROI berechnet [Snee04].

Zur Kalkulation der RoI einer Wartungsaufgabe gehören drei Voraussetzungen:

a) die Aufwandsschätzung jener Aufgabe
b) die Risikoanalyse jener Aufgabe
c) der Nutzwertanalyse jener Aufgabe

4 Aufwandsschätzung von Wartungsaufgaben

4.1 Expertenschätzung

Les Hatton hat einen Artikel in der IEEE Software mit dem Titel “How accurately do Engineers predict Software Maintenance Tasks” veröffentlicht. Der Artikel basiert auf einer Studie von 957 Wartungsaufgaben bezogen auf 13 verschiedene Softwareprodukte in der Zeit vom März 2001 bis November 2005. 30% der untersuchten Wartungsaufgaben waren Korrekturen, 20% waren Änderungen und 50% waren Nachbesserungen oder Erweiterungen. Zwischen den letzten zwei Aufgabenarten hat Hatton nicht unterschieden. Die große Mehrzahl der Aufgaben, etwa 77%, nahm weniger als einen Personentag in Anspruch. Am anderen Ende der Aufwandsskala brauchten weniger als 5% der Wartungsaufträge mehr als vier Personentage. Der höchste Aufwand für eine Wartungsaufgabe betrug 44 Stunden.

Der Zweck der Studie war heraus zu bekommen wie gut Wartungsaufgaben von dem Wartungspersonal geschätzt werden. Hatton stellte fest, dass 75% der Aufgaben korrekt geschätzt und nur 16% grob unterschätzt worden. Das verbleibende 9% wurde leicht unterschätzt. Daraus folgt, dass nur eine von vier Wartungsaufgaben falsch eingeschätzt wurde, aber dass bei diesen die Mehrzahl der Schätzungen völlig daneben lag. Diese Fehlschätzungen waren fast alle von kleinen Aufgaben mit einem Aufwand von wenigen Stunden. Ergo, kann man daraus schließen, dass Wartungsingenieure ihre Aufgaben selbst ausreichend gut schätzen können [Hatt07].

4.2 Werkzeugschätzung

Ein automatisierter Vorgang zur Schätzung einzelner Wartungsaufgaben wurde vom Sneed im Rahmen des GEOS Projektes in Wien implementiert [Snee01]. Dieser begann mit der Textanalyse eines Wartungsauftrages – Fehlermeldung oder Change Requests. Aufgrund der erkannten Objekte im Text wurde der Wartungsauftrag einem bestehenden Anwendungsfall in der Software-Repository zugeordnet. Falls er sich nicht automatisch zuordnen lies, musste der zuständiger Analytiker ihn manuell zuordnen. Nachdem der betroffene Anwendungsfall einmal identifiziert wurde, konnte dieser über eine automatisierte Impaktanalyse zu den dazu gehörigen Modulen und Klassen zurückverfolgt werden. Die Beziehungen der Anwendungsfälle und Datenobjekten zu den Klassen und Datenbanktabellen waren in der Repository gespeichert.

Waren die betroffenen Komponenten einmal erkannt, konnte der zuständige Analytiker schätzen um wie viel Prozent sie geändert werden müssten. Der Änderungsanteil reichte in der Regel von 2 bis 40%. Mit einem Änderungsanteil von über 40% muss man sowieso davon ausgehen, dass der Modul oder Klasse ganz zu ersetzen ist. Der prozentuale Änderungsanteil wurde anschließend mit der Anzahl Anweisungen, bzw. Datenelemente, in diesem Baustein multipliziert um die Größe der Änderung zu projektieren.

Da die betroffenen Klassen auch von anderen Klassen abhängig sein können, entweder durch Vererbung oder durch Assoziation, werden die abhängigen Klassen auch in den Wirkungsbereich (Impact Domain) des Wartungseingriffes einbezogen. Bei ihnen wird allerdings der prozentuale Änderungsanteil geringer angesetzt als den der vollbetroffener Klassen. Dieser ist eine Funktion der Entfernung von den vollbetroffenen Klassen. Die Formel für die Ermittlung des gesamten Wirkungsbereiches ist:

$$\begin{aligned} \textit{Raw change size} = {} & \textit{(directly impacted class size * change rate)} \\ & + \textit{(}\Sigma\textit{ indirectly impacted class sizes * (change rate/4))} \end{aligned}$$

Die rohe Änderungsgröße wird sowohl durch die Komplexität als auch durch die Qualität der direkt und indirekt betroffenen Klassen justiert. Die Komplexität ist der gewichtete Mittelwert aller Komplexitätsmaße der betroffenen Klassen. Dazu gehören solche einzelne Komplexitätsmaße wie Ablaufkomplexität, Schnittstellenkomplexität, Zugriffskomplexität, Sprachkomplexität und Verknüpfungskomplexität, die auf einer rationalen Skala von 0 bis 1 eingestuft werden. Hohe Komplexität ergibt einen Wert über 0,5 während niedrige Komplexität einen Wert unter 0,5 ergibt. Zum Zweck der Vereinigung werden sämtliche Komplexitätsmaße wie Kopplungsgrad, Vererbungstiefe, zyklomatische Komplexität und Sprachvolumen in ein rationales Maß überführt. Die Komplexitätsjustierung ergibt sich aus der Teilung der gemessenen Komplexität durch den Mittelwert 0,5.

$$\begin{aligned} \textit{ComplexityAdjustment} = {} & \textit{measured Complexity/} \\ & \textit{medianComplexity} \end{aligned}$$

Danach folgt aus einem Komplexitätskoeffizient von 0,7 einen Justierungsfaktor von 1,4.

Das gleiche Verfahren wird auch für die Qualität des Wirkungsbereiches angewandt. Hier ist die Qualität des betroffenen Codes der gewichtete Mittelwert aller Einzelqualitäten – Modularität, Wiederverwendbarkeit, Flexibilität, Testbarkeit, Konformität, usw. Zum Zweck der Vergleichbarkeit werden auch sie auf ein rationales Skala konvertiert. Eine überdurchschnittliche Qualität ergibt einen Wert größer als 0,5, eine unterdurchschnittliche Qualität einen Wert darunter. Die Qualitätsjustierung folgt aus der Teilung der mittleren Qualitätsnote 0,5 durch die gemessene Qualität.

$$\begin{aligned} \textit{QualityAdjustment} = {} & \textit{MedianQuality/} \\ & \textit{measuredQuality} \end{aligned}$$

Danach ergibt einer Qualitätskoeffizient von 0,6 einen Justierungsfaktor von 0,83.

Die endgültige justierte Größe des Wirkungsbereiches der geplanten Änderung ist die rohe Anzahl betroffener Anweisungen justiert durch die Komplexität und Qualität des Codes.

$$\textit{adjusted impact size} = \textit{raw–size * (complexity / 0.5) * (0.5 /quality)}$$

Angenommen, die direkt betroffenen Klassen haben 800 Anweisungen und die indirekt betroffenen Klassen 4000 Anweisungen und die Änderungsrate wird auf 10% geschätzt. In dem Fall umfasst der Wirkungsbereich 180 Anweisungen.

*(800 + (4000/4) * 0.1 = 180 statements.*

Mit einem Komplexitätsjustierungsfaktor von 0,7 und einem Qualitätsjustierungsfaktor von 0,6 ist die justierte Größe der betroffenen Codemenge 209 Anweisungen.

*180 * [0.7/0.5] * [0.5/0.6] = 209 statements.*

Die justierte Größe des Wirkungsbereiches wird anschließend durch die bisherige Produktivität in Anweisungen pro Personentag dividiert um den Aufwand für die Wartungsaufgabe in Personentage zu ermitteln. Bei einer Produktivität von 20 Anweisungen pro Personentag wären das hier 209/20 = 10,5 Personentage für die Durchführung der Änderung.

Die Kosten eines neuen Releases ist die Summe der Kosten aller einzelnen Wartungsaufträge – Fehlerkorrekturen und Änderungen. Falls sich die Wirkungsbereiche zwei oder mehr Wartungsaufträge überschneiden werden die gemeinsam betroffenen Anweisungen nur einmal gezählt. Dadurch fallen die Gesamtkosten um etwas weniger als die Summe aller Einzelkosten aus, denn eine Modul oder Klasse wird nur einmal gezählt unabhängig davon wie viele Wartungseingriffe sie betreffen. Der Wirkungsbereich eines Releases ist die vereinigte Menge der Wirkungsbereiche aller einzelner Wartungsaufträge. Dieser wird durch die mittlere Produktivität dividiert um die Bruttokosten des Release zu ermitteln. Die endgültigen Nettokosten sind die Bruttokosten justiert durch die projektspezifischen Bedingungen aus dem COCOMO-Modell wie Teamzusammengehörigkeit, Prozessreife, Werkzeugunterstützung und Erfahrung mit dem Produkt. Hinzu kommt ein Overhead-Faktor für das Produktmanagement um noch 20 bis 30%. Zusammenfassend sind die Kosten eines neuen Releases:

*{Betroffene Anweisungen / Produktivitätsrate} * Einflussfaktoren * Overheadfaktor*

Zu empfehlen ist es, zweigleisig vorzugehen und die Schätzung aus der automatisierten Impakt-Analyse mit der Expertenmeinung der Wartungsingenieure zu vergleichen. Man benutzt die eine Methode um die Andere zu bestätigen. Durch diesen dualen Ansatz kommt man zu einer plausiblen Vorhersage der Wartungskosten.

5 Nutzwertanalyse der Softwarewartung

Die Kosten und Risiken der Wartungsaufgaben sind die eine Seite der RoI Kalkulation. Der Nutzen der Wartung steht auf der anderen Seite. Zu unterscheiden ist zwischen den Nutzen der vier Wartungsarten – der Nutzen der Fehlerbeseitigung, der Nutzung der Änderungen, und der Nutzen der Nachbesserungen. Jede Wartungsart hat seinen eigenen Nutzen. Es ist deshalb erforderlich sie differenziert zu betrachten.

5.1 Nutzen der korrektiven Wartung

Der Nutzwert einer Fehlerkorrektur ist das Inverse von dem was der Fehler später kostet. Die Kosten eines Fehlers sind zweierlei. Zum Einen gibt es die Kosten der Fehlerbeseitigung in der Produktion, die nach Boehm und Anderen das Vierfache mehr ist als die Kosten der Fehlerbeseitigung in der Wartung. Der Nutzen der früheren Korrektur ist also die Differenz zwischen den Kosten der jetzigen Korrektur in der Wartung und der späteren Korrektur in der Produktion. Zum Zweiten gibt es die Kosten der Produktionsausfälle. Jede Arbeitsstunde, die aufgrund eines Fehlers zusätzlich entsteht, sei es ein durch einen Systemausfall oder durch die Umgehung eines falschen Ergebnis verursacht einen Geldverlust. Dieser Geldverlust entspricht den Kosten der verlorenen Arbeitsstunden.

Die verlorene Arbeitszeit ist nur die eine Seite des Schadens der durch Softwarefehler entsteht. Hinzu kommt der Verlust an Vertrauen sowie an verlorenen Geschäftsgelegenheiten. Dieser Verlust kann in die Millionen gehen, je nachdem wie kritisch das System ist. Ein ernsthafter Fehler kann zur Produktionsunterbrechung, zum Kundenverlust oder gar zu einem Gerichtsprozess führen. Es ist schwer solche Kosten im Voraus zu entziffern. Boehm und Huang schlagen eine negative Pareto Verteilung für die Schätzung derartiger Verluste vor [BoHu06].

Die hohen potentiellen Kosten von Fehlern in der Produktion erklären warum korrektive Wartung immer den Vorrang vor den anderen Wartungsarten hat. Zunächst muss die Qualität des Produktes gesichert werden, dann folgt die Funktionalität.

5.2 Nutzen der adaptiven Wartung

Der Nutzen einer Änderung ist die Differenz in dem Wert des Produktes ohne die geänderte Eigenschaft und dem Wert mit der geänderten Eigenschaft. Jede Änderung müsste zu einer Wertsteigerung führen. Die Höhe der Wertsteigerung ist gleich der Nutzwert. Manche Softwaresysteme verlieren ihren ganzen Wert wenn sie nicht geändert werden, z.B. im Falle einer Gesetzesänderung oder eines Währungswechsels. In solchen Fällen ist der Wert der Änderung gleich dem Wert des Gesamtproduktes.

Andere Änderungen wie die Verschönerung der Benutzeroberfläche bringen nur marginale Wertsteigerungen. Es muss nachgewiesen werden, dass solche Änderungen zu einer Produktivitätssteigerung führen. Wenn vorher eine Kassiererin 20 Kunden pro Stunde abfertigt, muss sie nachher mindestens 21 Kunden pro Stunde abfertigen können. Demnach ist der Wert der Änderung 5% des Wertes der Kundenabfertigungstransaktion. Um ihre Kosten zu rechtfertigen musste jede Änderung entweder zu einer Produktivitätssteigerung oder zu einer Kostenersparnis führen. Ausnahmen sind Änderungen, die nicht quantifizierbaren Nutzen wie die Steigerung der Benutzerzufriedenheit bringen, aber auch solche Änderungen sollten mit einem fiktiven Nutzwert versehen werden. Die Höhe der Wertsteigerung ist der entscheidende Faktor bei der Rechtfertigung von Änderungsanträgen [Mock00].

Wertsteigerung	=	*Wert des geänderten Systems*
	-	*Wert des ursprünglichen Systems*

5.3 Nutzen der perfektiven Wartung

Kent Beck schreibt "the cost of a piece of code over its many-year life is dominated by how well it communicates to others… Every method name and every class name is an opportunity to communicate what is going on…" [Beck95]. Effektive Softwarewartung setzt voraus dass die Software verständlich und handhabbar ist. Perfektive Wartung zielt darauf, das Produkt in diesem Sinne nachzubessern. Die Namen sollten sprechender sein, die Kommentare ausführlicher, die Codestruktur transparenter, die Codebausteine mehr von einander isoliert sein. Kurzum, es sollte einfacher und weniger gefährlich sein den Code zu ändern, bzw. zu erweitern. Auch perfektive Wartungsmaßnahmen müssen einen Nutzwert haben. Messbar ist dieser Wert nur anhand der reduzierten Wartungskosten. Man musste vergleichen, was die Durchführung einer Änderung vor der Restrukturierung/Refaktorierung kostet und was sie nachher kostet. Der Wert der Perfektionsaktion ist die Differenz zwischen den Kosten vorher und den Kosten nachher. Man wird aber selten die Gelegenheit haben, einen derartigen empirischen Vergleich durchzuführen. Ergo ist man auf die Messung von Systemeigenschaften die den Wartungsaufwand treiben angewiesen. Diese Eigenschaften sind jene Komplexitäten und Qualitäten die mehr oder weniger Arbeitsaufwand verursachen.

Leider ist noch nie bewiesen, welche Eigenschaften das sind. Viele Forscher haben sich mit dieser Frage beschäftigt aber noch keiner konnte bisher eine eindeutige Korrelation zwischen Codeeigenschaft und Wartungsaufwand nachweisen. Das Ersparnis an Wartungsaufwand ist sogar mit relativ banalen Eigenschaften wie Goto freier Code nicht eindeutig demonstrierbar. Dass die Erhaltung objektorientierter Systeme weniger kostet als die von prozeduralen Systemen ist auch noch nie bewiesen worden [Eier07]. Ergo können wir nur vermuten welche die wahren Wartungskostentreiber sind.

Dieser Autor hat vorgeschlagen, die Größe, Komplexität und Qualität des Produktes vor und nach der perfektiven Wartung zu messen. Das Ziel einer Sanierung müsste sein, die Größe und die Komplexität der Software zu reduzieren und die Qualität zu steigern.

Die Wertsteigerung ergibt sich aus der Summe dieser Faktoren. Sollte aber die Größe und Komplexität steigen und die Qualität sinken folgt eine Wertminderung [Snee05]. Die Differenz wird wie folgt errechnet:

Added Value = *(1 - NewSize/OldSize))*
+ *(OldComplexity - NewComplexity)*
+ *(NewQuality - OldQuality)*

Im Falle eines Systems mit 100.000 Anweisungen, eine Komplexität von 0,70 und eine Qualität von 0,45 vor der Sanierung und 90.000 Anweisungen, eine Komplexität von 0,60 und eine Qualität von 0,55 nach der Sanierung wäre die Wertsteigerung:

(1 – (90.000 / 100.000)) +
(0,70 – 0,60) +
(0,55 – 0,45) = 0,30 = 30% Wertsteigung

Eine negative Wertsteigung würde folgen, wenn ein System bei gleichbleibender Funktionalität an Codemenge und Komplexität zunimmt. Steigt die Größe durch ein Refactoring von 100 auf 110 Anweisungen und die Komplexität von 0,70 auf 0,80, käme auch bei einer Steigerung der Qualität um 0,10 einen Wertverlust zustande.

(1 – (110.000 / 100.000)) +
(0,70 – 0,80) +
(0,55 – 0,45) = -0.10 = 10% Wertverlust

Dies ist das Gefahr bei Reengineering Projekten bei denen die Codemenge aufgebläht wird, um die Codequalität zu erhöhen. Der Gewinn an Qualität wird durch die vermehrte Quantität negiert, es sei denn man behauptet die Größe des Codes spiele keine Rolle. Wir wissen jedoch, je mehr Code zu warten ist, desto höher die Wartungskosten. Es muss ein Anliegen sein, bei gleichbleibender Funktionalität die Codemenge zu reduzieren, z.B. Durch die Entfernung von Klonen.

Ein ähnlicher Ansatz zur Rechtfertigung der Migration konventionell verteilter Komponente in Web Services wurde von Tilley, Huang und andere vorgeschlagen [TGH04].

6 Projektierte Kosten und Nutzen der Wartung

6.1 Feststellung des Nutzens

Der Nutzwert eines Wartungsprojekts entspricht dem Nutzwert des neuen Releases, das aus dem Projekt hervorgeht. Der Wert des Release ist der Wert des Produktanteils, welches von dem Projekt betroffen ist. Wenn das Gesamtprodukt einen Wert von € 1 Million hat und 5% der Anweisungen geändert oder hinzugefügt werden, hat dieses Release den Wert von € 50,000. Das entspricht dem proportionalen Anteil am Gesamtwert.

Wenn der Aufwand für das Release drei Personenmonate beträgt und der Mannmonat € 10.000 kostet, sind die Kosten des Releases 3 x 10,000 = € 30.000. Die RoI für dieses Wartungsprojekt ist dann:

$$(50.000 – 30.000) /30.000 = 0.66$$

Eine Möglichkeit den Wert eines IT-Systems zu fixieren ist der Kostenersparnisansatz. Demzufolge ist der Wert des Systems die Differenz zwischen den Kosten des betroffenen Geschäftsprozesses ohne das IT-System und den Kosten desselben Prozesses mit dem System [Levy87].

$$Value = (Operation\ without\ IT) – (Operation\ with\ IT)$$

Mit dieser Gleichung kommt man auch zu einem negativen Nutzen, nämlich dann wenn der Prozess mit IT mehr kostet als der Prozess ohne IT. Dies war z.B. der Fall beim Arbeitslosengeld-II System. Dies ist aber eine grobe Vereinfachung weil ein IT-System in der Regel Nutzen hat, die nicht in gesparten Kosten auszudrucken sind, z:B. bequemer arbeiten oder schneller abfertigen. Solche qualitative Nutzen sind jedoch schwer zu erfassen. Deshalb bleiben wir lieber beim Kostenersparnis.

Die Kosten sollen außerdem noch durch die Risiken ergänzt werden. Jeder Eingriff in ein bestehendes System birgt Risiken in sich. Die Zuverlässigkeit der Software könnte durch die Änderungen beeinträchtigt werden oder es kommt zum Verlust an Performanz. Es gilt diese Risiken zu identifizieren und zu gewichten – nach Wahrscheinlichkeit und nach Schadensausmaß. Es gilt ferner die potentieller Kosten dieser Risiken zu schätzen. Danach wird die Summe der potentiellen Risikokosten zu den Kosten des Projektes selbst hinzugefügt.

$$Value = Benefit – (Costs + Risks)$$

Diese Gleichung wurde angepasst und angewandt in einer Doktorarbeit von dem Betriebswirt Eckhart von Hahn mit dem Titel “Preservation of Software Value” [Hahn05]. Von Hahn befasst sich darin mit der Werterhaltung von Softwareprodukten in einem dynamischen Markt. Der Wert der Produkte sollte nicht nur erhalten sondern auch noch gesteigert werden. Reengineering ist ein Mittel dieses Ziel zu erreichen. Durch ein Reengineering Projekt sollte die Qualität eines Produktes steigen und die Komplexität sinken. Dadurch sollten die Wartungskosten zurückgehen. Ein Produkt mit reduzierten Wartungskosten steigt im Wert. Von Hahn weist darauf hin, dass der Wert eines Produktes nie gleich bleiben kann. Entweder er sinkt weil die Funktionalität hinter den Erwartungen hinterher hängt oder er sinkt weil die steigende Komplexität oder Größe zu höheren Wartungsaufwänden führt. Mit jedem neuen Release muss also die Funktionalität erweitert und die Komplexität relativ zur Größe reduziert werden. Wenn nicht immer Mehr in ein Produkt investiert wird, verliert es an Wert. Dies führte SAP dazu ihre Wartungsgebühren von 17 auf 22% vom Kaufpreis zu erhöhen, eine Erhöhung die von den Anwendern abgelehnt wurde. Diese Erhöhung war aber notwendig um die Kosten der Produktweiterentwicklung zu finanzieren. Leider ist es der SAP nicht gelungen dies den Kunden klar zu machen.

6.2 Schätzung der Kosten

Die Schätzung von Wartungskosten kann sowohl auf der Mikroebene als auch auf der Makroebene stattfinden. Auf der Mikroebene werden die Kosten der einzelnen Wartungsaufträge geschätzt. Wie dies erreicht wird, wurde im 3. Abschnitt geschildert. Auf der Makroebene werden die jährlichen Wartungskosten zum Beginn eines jeden Jahes projektiert. Hierfür hat Boehm in dem originalen COCOMO Model folgende Formel vorgeschlagen:

*AnnualMaintEffort = Systemtype * [(AnnualChangeRate * DevelopmentEffort) * QualityAdjustment]*

Der Systemtyp könnte Standalone, Integriert, Verteilt oder Embedded sein. Jeder Typ hat einen anderen Multiplikationsfaktor. Die jährliche Änderungsrate ist der Anteil geänderter und hinzugefügter Anweisungen relativ zu allen Anweisungen. Der Entwicklungsaufwand ist die Anzahl Personenmonate, die gebraucht worden um das System bis zur ersten produktiven Freigabe zu bringen. Der Qualitätsjustierungsfaktor ist ein Multiplikator der den Qualitätsstand der Software wiederspiegelt. Eine überdurchschnittliche Qualität mindert den Wartungsaufwand und ist deshalb < 1. Eine unterdurchschnittliche Qualität steigert den Wartungsaufwand und ist deshalb > 1. Der Qualitätsfaktor ist also genau inverse zur gemessenen Qualitätsnote. Boehm hat die Komplexität außeracht gelassen. Vielleicht meinte er sie wäre in dem Entwicklungsaufwand eingeschlossen. Dies ist aber wie wir inzwischen wissen eine Fehlannahme, denn Komplexität bleibt nicht konstant, sie steigt in dem Maße wie die Software sich von seiner ursprünglichen Form entfernt [SnHT04].

Deshalb hat dieser Autor die Boehm Formel um einen Komplexitätsjustierungsfaktor ergänzt. Die erweiterte Formel sieht so aus.

*AnnualMaintEffort = Systemtype * [(AnnualChangeRate * DevelopmentEffort) * QualityAdjustment * ComplexityAdjustment]*

Wie die Komplexitäts- und Qualitätsjustierungsfaktoren zustande kommen wurde schon beschrieben. Der Entwicklungsaufwand dürfte aus der Zeiterfassung zu holen sein. Die jährliche Änderungsrate wird einfach auf der Basis der bisherigen Änderungsraten hochgerechnet [Snee05].

7 Berechnung der RoI eines Wartungsprojektes

Als Beispiel für die Berechnung einer RoI für die Softwarewartung wird ein Testwerkzeug herangezogen. In einem Testprojekt für ein Webportal hatte der Autor die Aufgabe gehabt, ein Anforderungsdokument zu analysieren um daraus anforderungsbasierte Testfälle abzuleiten. Es ist in Testprojekten üblich dies manuell zu tun. Dafür waren 18 Personentage erforderlich. Bei einem Tagessatz von € 800,- kamen die Kosten für die manuelle Testfallermittlung auf € 14,400. Später erstelle der Autor einen Textanalysator, um diese Arbeit zu automatisieren. Mit dem Textanalysator konnte die gleiche Anzahl an Testfällen aus demselben Dokument innerhalb 3 Personentage gewonnen werden. Der Werkzeugeinsatz brachte damit ein Ersparnis von 15 Personentage = € 12.000 gegenüber dem manuellen Verfahren. Dies stellt den Nutzwert dieser Software für ein Testprojekt dar. Seitdem werden mit dem Werkzeug im Durchschnitt zwei Testprojekte pro Jahr unterstützt. Das macht einen jährlichen Nutzwert von € 24.000 aus.

Die Kosten zur Entwicklung des Werkzeuges betrugen 40 Personentage, bzw. € 32.000. Es dauerte also 1,5 Jahre bis die Entwicklungskosten sich amortisiert haben. Danach fielen rund € 8.000 pro Jahr – ein Viertel der Entwicklungskosten - für die Wartung und Weiterentwicklung an. Dies scheint etwas hoch zu sein, ist aber eine Folge der ständigen Anpassungen an neuen Anforderungstypen. Der Risikofaktor bei der Wartung dieses Produktes ist vernachlässigbar weil das Produkt keine kritische Anwendung ist, die dauernd im Betrieb sein muss. In einem Zeitabschnitt von einem Jahr rechnet sich die RoI der Erhaltung wie folgt:

$$RoI = [24.000 – 8000] / 8.000 = 2$$

Demnach erwirtschaftet der Textanalysator einen Gewinn von € 16.000 pro Jahr oder das Doppelte der Investitionshöhe. Würde die Nutzung auf vier Testprojekte pro Jahr bei gleichbleibender Wartungskosten, steigern wäre die RoI = 5 oder das Fünffache der Investition in der Wartung. In diesem Falle lohnt es sich sehr wohl das Produkt weiter zu warten. Würde aber die Nutzung auf ein Testprojekt pro Jahr zurückfallen und die Wartungskosten auf € 12.000 steigen, wäre der Gewinn = Null. Hier musste man sich die Frage stellen ob es lohne dieses Produkt weiter zu erhalten.

8 Zusammenfassung

Softwarewartung darf nicht allein als Kostenfaktor angesehen werden. Er stellt einen Wert dar, nämlich was es kosten würde die gleiche Aufgabe ohne die Software zu bearbeiten oder was es kosten würde eine neue Lösung zu schaffen. Der Nutzwert einer Neuentwicklung muss im Bezug zum Nutzwert der bestehenden Lösung angesehen werden. Wenn er den Nutzen der vorhandenen Lösung bei nicht mehr als 50% übertrifft, empfiehlt es sich bei der alten Lösung zu bleiben. Das liegt daran, dass die Risiken einer Neuentwicklung sich zwischen 10 und 50% der Entwicklungskosten bewegen. Man braucht einen Mehrwert von mindestens 50% um diese Risiken abzufangen. Dies ist der Hauptgrund warum die Entwicklung neuer Ablösesysteme immer wieder verschoben wird. Die Erhaltung der bestehenden Systeme hat also sehr wohl einen quantifizierbaren Nutzwert, nämlich die ersparten Kosten einer Neuentwicklung. Es sei erforderlich diesen zu ermitteln um die Kosten der Wartung zu rechtfertigen.

Viele IT-Manager meinen die Wartung der bestehenden Systeme kostet zu viel. Sie fragen, warum ist Softwarewartung so teuer? Die Antwort von Tom DeMarco ist „relative to what?“ [DeMa97] Wenn man die Wartungskosten mit dem Wert des gewarteten Gutes vergleicht, denn sind sie eher zu niedrig.

Literaturverzeichnis

[Baet98] Baetjer, H.: Software as Capital – An Economic Perspective on Software Engineering, IEEE Computer Society Press, Los Alamitos, 1998, p. 87

[Beck95] Beck, K.: Clean Code – Pipe Dream or State of Mind, Smalltalk Report Nr. 4, June, 1995, p. 20

[BeLe85] Belady, L./Lehman, M.: Laws of Software Evolution, in Software Evolution, Academic Press, London, 1985.

[Biff06] Biffl et a.: Value-based Software Engineering, Springer Pub., Berlin, 2006, p. 21

[Blom08] Blom, S. /Gruhn, V./Koehler, A./Schaefer, C.: Methoden und Grundlagen der wertebasierten Softwareentwicklung, Objektspektrum, Nr. 1, Feb 2008, p. 12

[BoHu06] Boehm, B./Huang, L.: How much Software Quality investment is enough – A value-based Approach, IEEE Software, Sept. 2006, p. 88

[BoTu05] Boehm, B./Turner, R.: Management Challenges to implementing Agile Processes, IEEE Software, Sept. 2005, p. 30

[Chap01] Chapin, N./Hale, J./Kahn, K./Ramil, J./Tan,W.: Types of Software Evolution and Maintenance, Journal of Software Maintenance and Evolution, Vol. 13, Nr. 1, Jan. 2001, p.3

[DeMa97] DeMarco, T.: Warum ist Software so teuer, Hanser Verlag, München/Wien, 1997

[Eier07] Eierman, M./Dishaw, M.: Comparison of object-oriented and third generation development languages, Journal of Software Maintenance and Evolution, Vol. 19, No. 1, Jan. 2007, p. 33

[Hahn05] von Hahn, E.: Werterhaltung von Software, German University Press, Wiesbaden, 2005, p. 19

[Hatt07] Hatton, L.: How accurately do Engineers predict Software Maintenance Tasks, IEEE Computer, Feb. 2007, p. 64

[Haye39] Hayek, F.: The Maintenance of Capital, in Profits, Interest and Investment, Routledge & Sons, London, 1939.

[Haye41] Hayek, F.: The Theory of Capital, University of Chicago Press, Chicago, 1941, p. 93
[IEEE93] IEEE: ANSI/IEEE Standard 1219-1993, Standard for Software Maintenance, IEEE Press, New York, 1993, p. 15
[Lach75] Lachmann, L.: Reflections on the Hayekian Capital Theory, in Proc. of Allied Social Science Association, Dallas, 1975, p. 132
[Levy87] Levy, L.: Taming the Tiger – Software Engineering and Software Economics, Springer Verlag, London, 1987, p. 111
[Mell05] Mellor, S.: Adapting Agile Approaches to your Project Needs, IEEE Software, May, 2005, p. 17
[Mock00] Mockus, A./Votta, L.: Identifying reasons for Software Change using Historic Databases, in Proc. of 16th ICSM, IEEE Press, San Jose, 2000, p. 120
[Schm22] Schmidt, F.: Organische Tageswertbilanz, Herder Verlag, Leipzig, 1922, s. 133
[SnHT04] Sneed,H./Hasitschka,M./Teichmann,M.T.: Software-Produktmanagement, dpunkt Verlag, Heidelberg, 2004, p. 18
[Snee91] Sneed, H.: Economics of Software Reengineering, Journal of Software Maintenance, Vol. 3, Nr. 1, Sept. 1991, p. 163
[Snee01] Sneed, H.: Impact Analysis of Maintenance Tasks, in Proc. of 17th ICSM, IEEE Press, Florence, 2001, p. 180
[Snee04] Sneed, H.: A Cost Model for Software Maintenance and Evolution, in Proc. of 20th ICSM, IEEE Press, Chicago, 2004, p. 264
[Snee05] Sneed, H.: Estimating the Costs of Reengineering Projects, in Proc. of 12th WCRE, IEEE Press, Pittsburgh, 2005, p. 111
[Snee05] Sneed, H.: Software-Projektkalkulation, Hanser Verlag, München, 2005, p. 117
[Soli04] Solingen, R.: "Measuring the ROI of Software Process Improvement", IEEE Software, May 2004, p. 32
[TGH04] Tilley,S./Gerdes,J./Hamilton,T./Huang,S./Mueller,H./Smith,D./Wong,K.: On the Business Value and technical Challenge of adopting web services, Journal of Software Maintenance and Evolution, Vol. 16, Nr. 1-2, Jan. 2004, p. 31
[Tock05] Tockey, S.: Return on Software – maximizing the Return on your Software Investment, Addison-Wesley, Boston, 2005, p 211

Teil IV
Tutorien

Produktmanagement in der IT: Geschäftsmodelle und Produktpositionierung

Georg Herzwurm

Universität Stuttgart
Lehrstuhl für Allgemeine Betriebswirtschaftslehre und Wirtschaftsinformatik II
(Unternehmenssoftware)
Keplerstraße 17
D-70174 Stuttgart
herzwurm@wi.uni-stuttgart.de

Abstract: IT-Produktmanagement als Querschnittsfunktion mit betriebswirtschaftlichen und technischen Aspekten stellt eine besondere Herausforderung dar. Die Abgrenzung von IT-Produkten gegenüber industriell erzeugten Produkten oder Dienstleistungen ist vielschichtig, da IT-Produkte häufig komplexe Services beinhalten und selten Massenkonsumgüter sind. Das Tutorium vermittelt Themengebiete des Produktmanagements, die speziell für IT-Produkte relevant sind. Der Fokus liegt auf der kundenorientierten Gestaltung des IT-Produktmanagements sowie generischen Geschäftsmodellen IT-Produkt-anbietender Unternehmen. Anhand von ausgewählten Praxisbeispielen wird das Wissen zum IT-Produktmanagement vertieft, damit die Teilnehmer es in der eigenen Praxis anwenden können.

1 Bedeutung des IT-Produktmanagements

IT-Produkte werden nicht nur von speziellen IT-Unternehmen angeboten, vielmehr organisieren sich interne IT-Abteilungen verstärkt in marktlichen Strukturen. Dabei sind IT-Produktanbieter mit der Herausforderung konfrontiert, dass Kunden einen steigenden Bedarf an individuellen Lösungen haben, aber auch vermehrt Standardprodukte verlangen [HP09].

IT-Produkte sind daher nicht mehr individuell zu planen und zu vermarkten, sondern müssen zu marktgerechten Lösungen gebündelt und standardisiert werden. Der Ansatz, vom Projekt kommend zum kundenorientierten (Standard-)Produkt zu gelangen, ist ein Ansatz des IT-Produktmanagements [HP09]: gerecht werden können IT-Produktanbietende Unternehmen diesen Herausforderungen, die Chancen zugleich sind, nur durch eine klare Sicht auf die eigenen Unternehmensziele bei herausragender Kundenorientierung [He10]. Die Kenntnis der eigenen produkt(gruppen)bezogenen Geschäftsmodelle stellen eine Möglichkeit dar, die Herausforderungen des IT-Produktmanagements zu identifizieren und auf den dem IT-Markt und den IT-Produkten unterliegenden Wandel zu reagieren.

2 Ziele und Inhalte des Tutoriums

Den Teilnehmern wird ein grundlegendes Verständnis des IT-Produktmanagements vermittelt ohne zu sehr in Details zu verfallen. Von besonderer Bedeutung sind die kundenorientierte Gestaltung des IT-Produktmanagements sowie generische Geschäftsmodelle IT-Produkt-anbietender Unternehmen, die die Grundlage der Ausrichtung des IT-Produktmanagements legen [Pi08].

Anhand von Praxisbeispielen wird die Positionierung des IT-Produktmanagements in ausgewählten Unternehmen vorgestellt. Die folgende Abbildung stellt eine Basisanwendung (Core) mit ergänzenden Funktionalitäten (Extend) dar; Ziel des IT-Produktmanagements des betrachteten Unternehmens ist die Implementierung der Anwendung anhand von Beratungsprojekten (Integration project). Eingebettet sind die IT-Produkte bzw. die IT-Dienstleistung in den sogenannten IT-Produktkompass [HP09,Pi08], der im Rahmen der Veranstaltung eingeführt und erläutert wird.

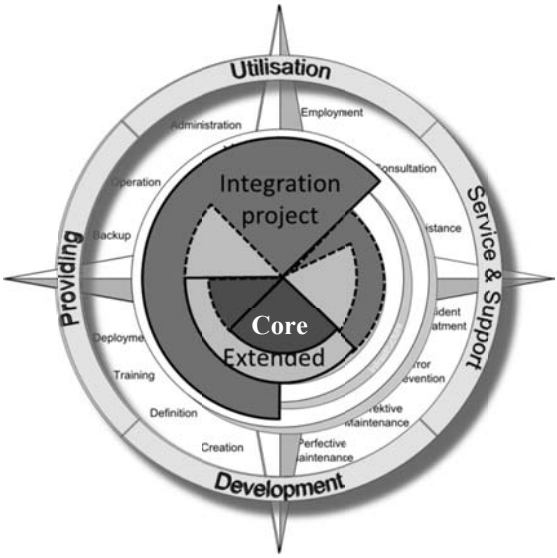

Abbildung 1: IT-Produktpositionierung im Produktkompass [He10]

Während des Tutoriums wird das Potential der jeweiligen Positionierung in Abhängigkeit des Geschäftsmodells und des IT-Produktspektrums diskutiert: welche Geschäftsstrategien sind für das Produktmanagement ableitbar?

Die Vorstellung eines in der IT-Projektpraxis bewährten Werkzeugs zur konsequenten Kundenorientierung im IT-Produktmanagement rundet das Tutorium ab: im Rahmen eines Workshops wird den Teilnehmern die Ermittlung von und der Umgang mit langfristigen Kundenanforderungen aufgezeigt, um IT-Produkte anwendergerecht zu entwickeln.

Das Tutorium richtet sich selbstredend an IT-Produktmanager. Darüber hinaus sind Projektleiter angesprochen, die Unternehmens- und Teilbereiche des IT-Produktmanagements verantworten. IT-domänenunabhängig richtet sich das Tutorium an Personen, die mit Anforderungsermittlung, -analyse und -management im Rahmen Ihrer berufliche Tätigkeit in Kontakt kommen und das Wesen kundenorientierter IT-Produktentwicklung kennenlernen wollen, um so einen Mehrwert für ihre tägliche Arbeit zu erhalten.

Literaturverzeichnis

[He10] Herzwurm, G.; Krams, B.; Mautsch, L.O.; Schockert, S.: QFD for planning the marketing mix. In: Proceedings of the 16th International Symposium on QFD, Portland, USA; S. 233-244

[HP09] Herzwurm, G.; Pietsch, W.: Management von IT-Produkten. dpunkt.verlag, Heidelberg 2009.

[Pi08] Pietsch, W.: Der IT-Produktkompass – Ein Instrument für die Analyse von Geschäftsmodellen und die strategiekonforme Positionierung von IT-Produkten. In: Tagungsband Fachtagung Software Management 2008. Köllen Verlag, Bonn 2008; S. 73-87

Zertifizierung zum "Certified Professional for Requirements Engineering" (CPRE) des International Requirements Engineering Board (IREB e.V.): Praxisorientierte Hinweise aus dem Schulungsalltag eines Trainingsproviders

Stefan Jesse, Sixten Schockert

Nathan Expertise
Peter-Schumacher-Straße 50
D-50171 Kerpen
{jesse, schockert}@nathan-expertise.de

Abstract: Das Tutorium zeigt die Grundzüge der Zertifizierung zum „Certified Professional for Requirements Engineering" (CPRE) anhand des aktuell gültigen Lehrplans. Die Zertifizierung stellt die erste Stufe einer Ausbildung dar, die ganzheitlich Requirements Engineering behandelt und den Fokus auf die Ermittlung, die Dokumentation, die Prüfung/ Abstimmung sowie die Verwaltung von Anforderungen legt. Im Rahmen der Veranstaltung erhalten die Teilnehmer Informationen zu den Ursprüngen der Zertifizierungsidee sowie der Vision des IREB e.V. und erfahren Wesentliches zu den Inhalten des Lehrplans. Ferner wird die Zertifizierungsprüfung durchleuchtet und die Teilnehmer erfahren aus der Schulungspraxis des Referenten, welche Hürden es bei der Zertifizierungsprüfung zu nehmen gilt.

1 Bedeutung des Requirements Engineering

Noch immer zeigen diverse Studien, dass IT-Projekte nur zu einem Bruchteil erfolgreich bzw. ohne Erreichung der veranschlagten Ziele abgeschlossen werden oder gar nie realisiert werden.[1] Gründe dafür liegen unter anderem in einem unzureichenden Requirements Engineering zu Anfang der Entwicklung, etwa weil die Beteiligten – vereinfacht die Kunden- und die Entwicklungsseite – unzureichend und missverständlich kommunizieren und dadurch Anforderungen unvollständig oder nicht hinreichend spezifiziert werden.

[1] Vgl. z.B. Chaos Summary Report der Standish Group [Sg09]

Im Rahmen der Softwareentwicklung hat sich das Requirements Engineering etabliert, um die Gefahr des Scheiterns von IT-Projekten durch strukturierte Anforderungsermittlung zu reduzieren. Requirements Engineering bietet die Möglichkeit, Software konsequent nach den Bedürfnissen der Kunden zu erstellen und deren Anforderungen in Softwareentwicklungsprojekten umzusetzen.

2 Die CPRE-Initiative

Ausgehend von einer Vision engagierter Wissenschaftler und Experten aus der Praxis über die standardisierte Wissensvermittlung zu Requirements Engineering hat sich eine Initiative entwickelt, der weltweit seit 2007 mehr als 3000 Personen (Stand 2009) in Form einer erfolgreich abgelegten Zertifizierungsprüfung gefolgt sind [SQ09].

Ursächlich für diese Erfolgsgeschichte zeichnet sich das IREB e. V.: das International Requirements Engineering Board. Die Vertreter des Boards haben die Professionalisierung des Requirements Engineering zu Ihrem obersten Ziel gemacht. Dabei bedeutet Professionalität der bewusste Einsatz anerkannter Methoden und Verfahren, die Förderung einer standardisierten Begriffsbildung und –verwendung, Führung des Nachweises, ob Methoden, Verfahren und Begriffe von verantwortlichen Personen korrekt eingesetzt werden sowie die Publikation und Vermittlung neuer Erkenntnisse, Wissen sowie Best Practices im Requirements Engineering an die interessierte Community [IREB].

Die Mitglieder des IREB e. V. definieren Lehrpläne, arbeiten die Prüfungen zum Erwerb des Zertifikats aus, arbeiten mit anerkannten Zertifizierungsstellen zur Durchführung zusammen und sorgen durch Akkreditierung von Schulungsanbietern (anerkannter Trainingsprovider) dafür, dass die Ausbildung vorbereitend zu den Zertifizierungsprüfungen den Zielen und Qualitätsstandards des IREB e.V. entsprechen [IREB].

3 Merkmale der Ausbildung zum Certified Professional for Requirements Engineering

Das Kernstück der Ausbildung stellt der Lehrplan dar, der 2010 in der überarbeiteten und aktuell gültigen Version 2.1 veröffentlicht wurde und auch in englischer Sprache verfügbar ist. Dieser gilt für die erste Stufe der Zertifizierung, dem Foundation Level.

3.1 Lehrplan

Der Lehrplan besteht aus einzelnen Lerneinheiten denen unterschiedliche Lernziele zu Grunde liegen. Bei der Formulierung der Lernziele wird zwischen Kennen eines Inhalts und der Anwendung eines Lehrinhalts differenziert. Der Lehrplan enthält folgende übergeordnete Lerneinheiten die überwiegend chronologisch aufeinander aufbauen [Lp10]:

- Einleitung und Grundlagen
- System und Systemkontext eingrenzen
- Anforderungen ermitteln
- Dokumentation von Anforderungen
- Natürlichsprachige Dokumentation von Anforderungen
- Anforderungen modellbasiert dokumentieren
- Anforderungen prüfen und abstimmen
- Anforderungen verwalten
- Werkzeuge

3.2 Das CPRE-Modell: die Zertifizierungsstufen

Das CPRE-Modell ist in drei Zertifizierungsstufen unterteilt. Der Foundation Level bildet die erste Stufe und enthält die in Kapitel 4.1 dargelegten Inhalte. Darauf aufbauend werden in Zukunft der Advanced Level und der Expert Level entstehen. Die folgende Abbildung gibt einen Überblick über die – beim Advanced Level vorgesehenen – Inhalte der ersten beiden Zertifizierungsstufen.

Abbildung 1: Inhalte des CPRE Foundation Level und geplante Module des CPRE Advanced Level (in Anlehnung an [GRS09])

Die Zertifizierung im Foundation Level erhalten Interessenten durch Bestehen der entsprechenden Zertifizierungsprüfung, die als Voraussetzung für die Erlangung der Stufe des Advanced Levels gilt. Die Module des Advanced Levels tauchen in ausgewählten Themengebieten tief in die jeweilige Materie ein und bilden Spezialisten aus. Nach Erlangung von drei oder mehr Advanced Level Zertifikaten können Absolventen ein Expert Level Zertifikat beantragen.

3.3 Zertifizierungsprüfung

Die Zertifizierungsprüfung zum Foundation Level dauert 75 Minuten in denen 45 Multiple Choice Fragen beantwortet werden müssen. Die Prüfung gilt als bestanden, sobald der Prüfling mehr als 60% aller zu erreichenden Punkte erlangt.

Die Prüfung wird durch Mitarbeiter des international Software Quality Institute (iSQI) zu festen Terminen abgenommen, kann im Rahmen von Inhouse-Schulungen im Anschluss einer Schulung eines Trainingsproviders erfolgen und wird seit 2010 auch online angeboten. Die Online-Prüfungen werden durch Pearson VUE koordiniert und über diesen Anbieter auch gebucht: für diese Art der Prüfung begibt sich der Prüfling in die Räumlichkeiten eines anerkannten Schulungszentrums um dort terminlich flexibel eine Prüfung ablegen zu können. Auf Wunsch kann die Prüfung auf Englisch abgelegt werden.

4 Ziele des Tutoriums

Das Tutorium vermittelt den Teilnehmern Hintergründe zu den Schulungen zum Certified Professional for Requirements Engineering (CPRE), den Initiatoren und deren Vision, der Ausgestaltung der Lehrinhalte des CPRE-Lehrplans und es werden Hinweise zu der Zertifizierungsprüfung gegeben. Den Teilnehmern wird ausreichend Gelegenheit gegeben Fragen zu stellen, die moderiert in der Gruppe diskutiert werden können.

Die Ausbildung zum Certified Professional for Requirements Engineering richtet sich zunächst domänenunabhängig an Personen, die mit Anforderungsanalyse und -management im Rahmen ihrer beruflichen Tätigkeit in Kontakt kommen. Dazu zählen unter anderem Software-Entwickler, -Produktmanager, Mitarbeiter in Forschung & Entwicklung. Damit Anforderungen gemeinsam aufgenommen werden können und ein gemeinsames Verständnis für die Kommunikation herrscht ist die Ausbildung ebenfalls für Mitarbeiter in der Qualitätssicherung, Consultants/Berater und Mitarbeiter der im Lehrplan enthaltenen Ausschüsse (z. B. des Change Control Boards) relevant. Da Berührungspunkte mit Reifegradmodellen bzw. Best Practices und CPRE vorhanden sind, profitieren auch Anwender von CMMI und ITIL.

Darüber hinaus bietet die Ausbildung für Anforderungsmanager, System- und Software-Analytiker (System Engineers), Software-Architekten, Projektleiter sowie Software-Tester als Beteiligte im Softwareentwicklungsprozess Mehrwert in der täglichen Arbeit.

Im Rahmen des Tutoriums werden Erfahrungen mit der Foundation Level-Prüfung aus der Perspektive eines vom IREB e. V. anerkannten Schulungsanbieters vermittelt. Das Tutorium ersetzt aber in keinem Fall eine dezidierte Prüfungsvorbereitung in Form einer Schulung. Zur Vorbereitung auf die Zertifizierungsprüfung des Foundation Level empfiehlt das IREB e. V. eine mindestens dreitägige Schulung. Überdies gibt es ein Buch [PR10] zweier Mitglieder des Boards, Klaus Pohl und Chris Rupp, welches auf den Lehrplan zum Foundation Level abgestimmt ist.

Literaturverzeichnis

[GRS09] Grau, R.; Rupp, C.; Schüpferling, D: Die RE-Völkerverständigung: Über das Zertifikat „Certified Professional for Requirements Engineering“. In (OBJEKTspektrum 03/2009); S. 44-49

[IREB] Homepage des International Requirements Engineering Board e.V., http://www.certified-re.de, abgerufen am 15.09.2010

[PR10] Pohl, K; Rupp, C: Basiswissen Requirements Engineering – Aus- und Weiterbildung nach zum CPRE Foundation Level nach IREB-Standard. 2. Aufl., dpunkt.verlag, Heidelberg, 2010.

[Sg09] Standish Group, Chaos Summary 2009 Report, 2009.

[SQ09] SQ Magazin, Interview mit Chris Rupp, Ausgabe 13/ Dezember 2009; S. 6-7

[Lp10] Lehrplan des Foundation Levels Version 2.1 des IREB e.V., IREB e.V. 2010. Download möglich über http://www.certified-re.de/lehrplaene.html.

GI-Edition Lecture Notes in Informatics

P-1 Gregor Engels, Andreas Oberweis, Albert Zündorf (Hrsg.): Modellierung 2001.

P-2 Mikhail Godlevsky, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications, ISTA'2001.

P-3 Ana M. Moreno, Reind P. van de Riet (Hrsg.): Applications of Natural Language to Information Systems, NLDB'2001.

P-4 H. Wörn, J. Mühling, C. Vahl, H.-P. Meinzer (Hrsg.): Rechner- und sensorgestützte Chirurgie; Workshop des SFB 414.

P-5 Andy Schürr (Hg.): OMER – Object-Oriented Modeling of Embedded Real-Time Systems.

P-6 Hans-Jürgen Appelrath, Rolf Beyer, Uwe Marquardt, Heinrich C. Mayr, Claudia Steinberger (Hrsg.): Unternehmen Hochschule, UH'2001.

P-7 Andy Evans, Robert France, Ana Moreira, Bernhard Rumpe (Hrsg.): Practical UML-Based Rigorous Development Methods – Countering or Integrating the extremists, pUML'2001.

P-8 Reinhard Keil-Slawik, Johannes Magenheim (Hrsg.): Informatikunterricht und Medienbildung, INFOS'2001.

P-9 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Innovative Anwendungen in Kommunikationsnetzen, 15. DFN Arbeitstagung.

P-10 Mirjam Minor, Steffen Staab (Hrsg.): 1st German Workshop on Experience Management: Sharing Experiences about the Sharing Experience.

P-11 Michael Weber, Frank Kargl (Hrsg.): Mobile Ad-Hoc Netzwerke, WMAN 2002.

P-12 Martin Glinz, Günther Müller-Luschnat (Hrsg.): Modellierung 2002.

P-13 Jan von Knop, Peter Schirmbacher and Viljan Mahni_ (Hrsg.): The Changing Universities – The Role of Technology.

P-14 Robert Tolksdorf, Rainer Eckstein (Hrsg.): XML-Technologien für das Semantic Web – XSW 2002.

P-15 Hans-Bernd Bludau, Andreas Koop (Hrsg.): Mobile Computing in Medicine.

P-16 J. Felix Hampe, Gerhard Schwabe (Hrsg.): Mobile and Collaborative Busi-ness 2002.

P-17 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Zukunft der Netze –Die Verletzbarkeit meistern, 16. DFN Arbeitstagung.

P-18 Elmar J. Sinz, Markus Plaha (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2002.

P-19 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund.

P-20 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund (Ergänzungsband).

P-21 Jörg Desel, Mathias Weske (Hrsg.): Promise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen.

P-22 Sigrid Schubert, Johannes Magenheim, Peter Hubwieser, Torsten Brinda (Hrsg.): Forschungsbeiträge zur "Didaktik der Informatik" – Theorie, Praxis, Evaluation.

P-23 Thorsten Spitta, Jens Borchers, Harry M. Sneed (Hrsg.): Software Management 2002 – Fortschritt durch Beständigkeit

P-24 Rainer Eckstein, Robert Tolksdorf (Hrsg.): XMIDX 2003 – XML-Technologien für Middleware – Middleware für XML-Anwendungen

P-25 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Commerce – Anwendungen und Perspektiven – 3. Workshop Mobile Commerce, Universität Augsburg, 04.02.2003

P-26 Gerhard Weikum, Harald Schöning, Erhard Rahm (Hrsg.): BTW 2003: Datenbanksysteme für Business, Technologie und Web

P-27 Michael Kroll, Hans-Gerd Lipinski, Kay Melzer (Hrsg.): Mobiles Computing in der Medizin

P-28 Ulrich Reimer, Andreas Abecker, Steffen Staab, Gerd Stumme (Hrsg.): WM 2003: Professionelles Wissensmanagement – Erfahrungen und Visionen

P-29 Antje Düsterhöft, Bernhard Thalheim (Eds.): NLDB'2003: Natural Language Processing and Information Systems

P-30 Mikhail Godlevsky, Stephen Liddle, Heinrich C. Mayr (Eds.): Information Systems Technology and its Applications

P-31 Arslan Brömme, Christoph Busch (Eds.): BIOSIG 2003: Biometrics and Electronic Signatures

P-32 Peter Hubwieser (Hrsg.): Informatische Fachkonzepte im Unterricht – INFOS 2003

P-33 Andreas Geyer-Schulz, Alfred Taudes (Hrsg.): Informationswirtschaft: Ein Sektor mit Zukunft

P-34 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 1)

P-35 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 2)

P-36 Rüdiger Grimm, Hubert B. Keller, Kai Rannenberg (Hrsg.): Informatik 2003 – Mit Sicherheit Informatik

P-37 Arndt Bode, Jörg Desel, Sabine Rathmayer, Martin Wessner (Hrsg.): DeLFI 2003: e-Learning Fachtagung Informatik

P-38 E.J. Sinz, M. Plaha, P. Neckel (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2003

P-39 Jens Nedon, Sandra Frings, Oliver Göbel (Hrsg.): IT-Incident Management & IT-Forensics – IMF 2003

P-40 Michael Rebstock (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2004

P-41 Uwe Brinkschulte, Jürgen Becker, Dietmar Fey, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle, Thomas Runkler (Edts.): ARCS 2004 – Organic and Pervasive Computing

P-42 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Economy – Transaktionen und Prozesse, Anwendungen und Dienste

P-43 Birgitta König-Ries, Michael Klein, Philipp Obreiter (Hrsg.): Persistance, Scalability, Transactions – Database Mechanisms for Mobile Applications

P-44 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): Security, E-Learning. E-Services

P-45 Bernhard Rumpe, Wofgang Hesse (Hrsg.): Modellierung 2004

P-46 Ulrich Flegel, Michael Meier (Hrsg.): Detection of Intrusions of Malware & Vulnerability Assessment

P-47 Alexander Prosser, Robert Krimmer (Hrsg.): Electronic Voting in Europe – Technology, Law, Politics and Society

P-48 Anatoly Doroshenko, Terry Halpin, Stephen W. Liddle, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications

P-49 G. Schiefer, P. Wagner, M. Morgenstern, U. Rickert (Hrsg.): Integration und Datensicherheit – Anforderungen, Konflikte und Perspektiven

P-50 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 1) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-51 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 2) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-52 Gregor Engels, Silke Seehusen (Hrsg.): DELFI 2004 – Tagungsband der 2. e-Learning Fachtagung Informatik

P-53 Robert Giegerich, Jens Stoye (Hrsg.): German Conference on Bioinformatics – GCB 2004

P-54 Jens Borchers, Ralf Kneuper (Hrsg.): Softwaremanagement 2004 – Outsourcing und Integration

P-55 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): E-Science und Grid Adhoc-Netze Medienintegration

P-56 Fernand Feltz, Andreas Oberweis, Benoit Otjacques (Hrsg.): EMISA 2004 – Informationssysteme im E-Business und E-Government

P-57 Klaus Turowski (Hrsg.): Architekturen, Komponenten, Anwendungen

P-58 Sami Beydeda, Volker Gruhn, Johannes Mayer, Ralf Reussner, Franz Schweiggert (Hrsg.): Testing of Component-Based Systems and Software Quality

P-59 J. Felix Hampe, Franz Lehner, Key Pousttchi, Kai Ranneberg, Klaus Turowski (Hrsg.): Mobile Business – Processes, Platforms, Payments

P-60 Steffen Friedrich (Hrsg.): Unterrichtskonzepte für inforrmatische Bildung

P-61 Paul Müller, Reinhard Gotzhein, Jens B. Schmitt (Hrsg.): Kommunikation in verteilten Systemen

P-62 Federrath, Hannes (Hrsg.): „Sicherheit 2005“ – Sicherheit – Schutz und Zuverlässigkeit

P-63 Roland Kaschek, Heinrich C. Mayr, Stephen Liddle (Hrsg.): Information Systems – Technology and ist Applications

P-64 Peter Liggesmeyer, Klaus Pohl, Michael Goedicke (Hrsg.): Software Engineering 2005

P-65 Gottfried Vossen, Frank Leymann, Peter Lockemann, Wolffried Stucky (Hrsg.): Datenbanksysteme in Business, Technologie und Web

P-66 Jörg M. Haake, Ulrike Lucke, Djamshid Tavangarian (Hrsg.): DeLFI 2005: 3. deutsche e-Learning Fachtagung Informatik

P-67 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 1)

P-68 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 2)

P-69 Robert Hirschfeld, Ryszard Kowalcyk, Andreas Polze, Matthias Weske (Hrsg.): NODe 2005, GSEM 2005

P-70 Klaus Turowski, Johannes-Maria Zaha (Hrsg.): Component-oriented Enterprise Application (COAE 2005)

P-71 Andrew Torda, Stefan Kurz, Matthias Rarey (Hrsg.): German Conference on Bioinformatics 2005

P-72 Klaus P. Jantke, Klaus-Peter Fähnrich, Wolfgang S. Wittig (Hrsg.): Marktplatz Internet: Von e-Learning bis e-Payment

P-73 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): “Heute schon das Morgen sehen“

P-74 Christopher Wolf, Stefan Lucks, Po-Wah Yau (Hrsg.): WEWoRC 2005 – Western European Workshop on Research in Cryptology

P-75 Jörg Desel, Ulrich Frank (Hrsg.): Enterprise Modelling and Information Systems Architecture

P-76 Thomas Kirste, Birgitta König-Riess, Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Informationssysteme – Potentiale, Hindernisse, Einsatz

P-77 Jana Dittmann (Hrsg.): SICHERHEIT 2006

P-78 K.-O. Wenkel, P. Wagner, M. Morgenstern, K. Luzi, P. Eisermann (Hrsg.): Land- und Ernährungswirtschaft im Wandel

P-79 Bettina Biel, Matthias Book, Volker Gruhn (Hrsg.): Softwareengineering 2006

P-80 Mareike Schoop, Christian Huemer, Michael Rebstock, Martin Bichler (Hrsg.): Service-Oriented Electronic Commerce

P-81 Wolfgang Karl, Jürgen Becker, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle (Hrsg.): ARCS´06

P-82 Heinrich C. Mayr, Ruth Breu (Hrsg.): Modellierung 2006

P-83 Daniel Huson, Oliver Kohlbacher, Andrei Lupas, Kay Nieselt and Andreas Zell (eds.): German Conference on Bioinformatics

P-84 Dimitris Karagiannis, Heinrich C. Mayr, (Hrsg.): Information Systems Technology and its Applications

P-85 Witold Abramowicz, Heinrich C. Mayr, (Hrsg.): Business Information Systems

P-86 Robert Krimmer (Ed.): Electronic Voting 2006

P-87 Max Mühlhäuser, Guido Rößling, Ralf Steinmetz (Hrsg.): DELFI 2006: 4. e-Learning Fachtagung Informatik

P-88 Robert Hirschfeld, Andreas Polze, Ryszard Kowalczyk (Hrsg.): NODe 2006, GSEM 2006

P-90 Joachim Schelp, Robert Winter, Ulrich Frank, Bodo Rieger, Klaus Turowski (Hrsg.): Integration, Informationslogistik und Architektur

P-91 Henrik Stormer, Andreas Meier, Michael Schumacher (Eds.): European Conference on eHealth 2006

P-92 Fernand Feltz, Benoît Otjacques, Andreas Oberweis, Nicolas Poussing (Eds.): AIM 2006

P-93 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 1

P-94 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 2

P-95 Matthias Weske, Markus Nüttgens (Eds.): EMISA 2005: Methoden, Konzepte und Technologien für die Entwicklung von dienstbasierten Informationssystemen

P-96 Saartje Brockmans, Jürgen Jung, York Sure (Eds.): Meta-Modelling and Ontologies

P-97 Oliver Göbel, Dirk Schadt, Sandra Frings, Hardo Hase, Detlef Günther, Jens Nedon (Eds.): IT-Incident Mangament & IT-Forensics – IMF 2006

P-98 Hans Brandt-Pook, Werner Simonsmeier und Thorsten Spitta (Hrsg.): Beratung in der Softwareentwicklung – Modelle, Methoden, Best Practices

P-99 Andreas Schwill, Carsten Schulte, Marco Thomas (Hrsg.): Didaktik der Informatik

P-100 Peter Forbrig, Günter Siegel, Markus Schneider (Hrsg.): HDI 2006: Hochschuldidaktik der Informatik

P-101 Stefan Böttinger, Ludwig Theuvsen, Susanne Rank, Marlies Morgenstern (Hrsg.): Agrarinformatik im Spannungsfeld zwischen Regionalisierung und globalen Wertschöpfungsketten

P-102 Otto Spaniol (Eds.): Mobile Services and Personalized Environments

P-103 Alfons Kemper, Harald Schöning, Thomas Rose, Matthias Jarke, Thomas Seidl, Christoph Quix, Christoph Brochhaus (Hrsg.): Datenbanksysteme in Business, Technologie und Web (BTW 2007)

P-104 Birgitta König-Ries, Franz Lehner, Rainer Malaka, Can Türker (Hrsg.) MMS 2007: Mobilität und mobile Informationssysteme

P-105 Wolf-Gideon Bleek, Jörg Raasch, Heinz Züllighoven (Hrsg.) Software Engineering 2007

P-106 Wolf-Gideon Bleek, Henning Schwentner, Heinz Züllighoven (Hrsg.) Software Engineering 2007 – Beiträge zu den Workshops

P-107 Heinrich C. Mayr, Dimitris Karagiannis (eds.) Information Systems Technology and its Applications

P-108 Arslan Brömme, Christoph Busch, Detlef Hühnlein (eds.) BIOSIG 2007: Biometrics and Electronic Signatures

P-109 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 1

P-110 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 2

P-111 Christian Eibl, Johannes Magenheim, Sigrid Schubert, Martin Wessner (Hrsg.) DeLFI 2007: 5. e-Learning Fachtagung Informatik

P-112 Sigrid Schubert (Hrsg.) Didaktik der Informatik in Theorie und Praxis

P-113 Sören Auer, Christian Bizer, Claudia Müller, Anna V. Zhdanova (Eds.) The Social Semantic Web 2007 Proceedings of the 1st Conference on Social Semantic Web (CSSW)

P-114 Sandra Frings, Oliver Göbel, Detlef Günther, Hardo G. Hase, Jens Nedon, Dirk Schadt, Arslan Brömme (Eds.) IMF2007 IT-incident management & IT-forensics Proceedings of the 3rd International Conference on IT-Incident Management & IT-Forensics

P-115 Claudia Falter, Alexander Schliep, Joachim Selbig, Martin Vingron and Dirk Walther (Eds.) German conference on bioinformatics GCB 2007

P-116 Witold Abramowicz, Leszek Maciszek (Eds.) Business Process and Services Computing 1st International Working Conference on Business Process and Services Computing BPSC 2007

P-117 Ryszard Kowalczyk (Ed.) Grid service engineering and manegement The 4th International Conference on Grid Service Engineering and Management GSEM 2007

P-118 Andreas Hein, Wilfried Thoben, Hans-Jürgen Appelrath, Peter Jensch (Eds.) European Conference on ehealth 2007

P-119 Manfred Reichert, Stefan Strecker, Klaus Turowski (Eds.) Enterprise Modelling and Information Systems Architectures Concepts and Applications

P-120 Adam Pawlak, Kurt Sandkuhl, Wojciech Cholewa, Leandro Soares Indrusiak (Eds.) Coordination of Collaborative Engineering - State of the Art and Future Challenges

P-121 Korbinian Herrmann, Bernd Bruegge (Hrsg.) Software Engineering 2008 Fachtagung des GI-Fachbereichs Softwaretechnik

P-122 Walid Maalej, Bernd Bruegge (Hrsg.) Software Engineering 2008 - Workshopband Fachtagung des GI-Fachbereichs Softwaretechnik

P-123 Michael H. Breitner, Martin Breunig, Elgar Fleisch, Ley Pousttchi, Klaus Turowski (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Technologien, Prozesse, Marktfähigkeit
Proceedings zur 3. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2008)

P-124 Wolfgang E. Nagel, Rolf Hoffmann, Andreas Koch (Eds.)
9th Workshop on Parallel Systems and Algorithms (PASA)
Workshop of the GI/ITG Speciel Interest Groups PARS and PARVA

P-125 Rolf A.E. Müller, Hans-H. Sundermeier, Ludwig Theuvsen, Stephanie Schütze, Marlies Morgenstern (Hrsg.)
Unternehmens-IT: Führungsinstrument oder Verwaltungsbürde
Referate der 28. GIL Jahrestagung

P-126 Rainer Gimnich, Uwe Kaiser, Jochen Quante, Andreas Winter (Hrsg.)
10th Workshop Software Reengineering (WSR 2008)

P-127 Thomas Kühne, Wolfgang Reisig, Friedrich Steimann (Hrsg.)
Modellierung 2008

P-128 Ammar Alkassar, Jörg Siekmann (Hrsg.)
Sicherheit 2008
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
2.-4. April 2008
Saarbrücken, Germany

P-129 Wolfgang Hesse, Andreas Oberweis (Eds.)
Sigsand-Europe 2008
Proceedings of the Third AIS SIGSAND European Symposium on Analysis, Design, Use and Societal Impact of Information Systems

P-130 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
1. DFN-Forum Kommunikations-technologien Beiträge der Fachtagung

P-131 Robert Krimmer, Rüdiger Grimm (Eds.)
3rd International Conference on Electronic Voting 2008
Co-organized by Council of Europe, Gesellschaft für Informatik and E-Voting.CC

P-132 Silke Seehusen, Ulrike Lucke, Stefan Fischer (Hrsg.)
DeLFI 2008:
Die 6. e-Learning Fachtagung Informatik

P-133 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 1

P-134 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 2

P-135 Torsten Brinda, Michael Fothe, Peter Hubwieser, Kirsten Schlüter (Hrsg.)
Didaktik der Informatik – Aktuelle Forschungsergebnisse

P-136 Andreas Beyer, Michael Schroeder (Eds.)
German Conference on Bioinformatics GCB 2008

P-137 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2008: Biometrics and Electronic Signatures

P-138 Barbara Dinter, Robert Winter, Peter Chamoni, Norbert Gronau, Klaus Turowski (Hrsg.)
Synergien durch Integration und Informationslogistik
Proceedings zur DW2008

P-139 Georg Herzwurm, Martin Mikusz (Hrsg.)
Industrialisierung des Software-Managements
Fachtagung des GI-Fachausschusses Management der Anwendungsentwick-lung und -wartung im Fachbereich Wirtschaftsinformatik

P-140 Oliver Göbel, Sandra Frings, Detlef Günther, Jens Nedon, Dirk Schadt (Eds.)
IMF 2008 - IT Incident Management & IT Forensics

P-141 Peter Loos, Markus Nüttgens, Klaus Turowski, Dirk Werth (Hrsg.)
Modellierung betrieblicher Informations-systeme (MobIS 2008)
Modellierung zwischen SOA und Compliance Management

P-142 R. Bill, P. Korduan, L. Theuvsen, M. Morgenstern (Hrsg.)
Anforderungen an die Agrarinformatik durch Globalisierung und Klimaveränderung

P-143 Peter Liggesmeyer, Gregor Engels, Jürgen Münch, Jörg Dörr, Norman Riegel (Hrsg.)
Software Engineering 2009
Fachtagung des GI-Fachbereichs Softwaretechnik

P-144 Johann-Christoph Freytag, Thomas Ruf, Wolfgang Lehner, Gottfried Vossen (Hrsg.)
Datenbanksysteme in Business, Technologie und Web (BTW)

P-145 Knut Hinkelmann, Holger Wache (Eds.)
WM2009: 5th Conference on Professional Knowledge Management

P-146 Markus Bick, Martin Breunig, Hagen Höpfner (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Entwicklung, Implementierung und Anwendung
4. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2009)

P-147 Witold Abramowicz, Leszek Maciaszek, Ryszard Kowalczyk, Andreas Speck (Eds.)
Business Process, Services Computing and Intelligent Service Management
BPSC 2009 · ISM 2009 · YRW-MBP 2009

P-148 Christian Erfurth, Gerald Eichler, Volkmar Schau (Eds.)
9th International Conference on Innovative Internet Community Systems
I^2CS 2009

P-149 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
2. DFN-Forum Kommunikationstechnologien
Beiträge der Fachtagung

P-150 Jürgen Münch, Peter Liggesmeyer (Hrsg.)
Software Engineering
2009 - Workshopband

P-151 Armin Heinzl, Peter Dadam, Stefan Kirn, Peter Lockemann (Eds.)
PRIMIUM
Process Innovation for Enterprise Software

P-152 Jan Mendling, Stefanie Rinderle-Ma, Werner Esswein (Eds.)
Enterprise Modelling and Information Systems Architectures
Proceedings of the 3rd Int'l Workshop EMISA 2009

P-153 Andreas Schwill, Nicolas Apostolopoulos (Hrsg.)
Lernen im Digitalen Zeitalter
DeLFI 2009 – Die 7. E-Learning Fachtagung Informatik

P-154 Stefan Fischer, Erik Maehle Rüdiger Reischuk (Hrsg.)
INFORMATIK 2009
Im Focus das Leben

P-155 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2009:
Biometrics and Electronic Signatures
Proceedings of the Special Interest Group on Biometrics and Electronic Signatures

P-156 Bernhard Koerber (Hrsg.)
Zukunft braucht Herkunft
25 Jahre »INFOS – Informatik und Schule«

P-157 Ivo Grosse, Steffen Neumann, Stefan Posch, Falk Schreiber, Peter Stadler (Eds.)
German Conference on Bioinformatics 2009

P-158 W. Claupein, L. Theuvsen, A. Kämpf, M. Morgenstern (Hrsg.)
Precision Agriculture Reloaded – Informationsgestützte Landwirtschaft

P-159 Gregor Engels, Markus Luckey, Wilhelm Schäfer (Hrsg.)
Software Engineering 2010

P-160 Gregor Engels, Markus Luckey, Alexander Pretschner, Ralf Reussner (Hrsg.)
Software Engineering 2010 – Workshopband
(inkl. Doktorandensymposium)

P-161 Gregor Engels, Dimitris Karagiannis Heinrich C. Mayr (Hrsg.)
Modellierung 2010

P-162 Maria A. Wimmer, Uwe Brinkhoff, Siegfried Kaiser, Dagmar Lück-Schneider, Erich Schweighofer, Andreas Wiebe (Hrsg.)
Vernetzte IT für einen effektiven Staat
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2010

P-163 Markus Bick, Stefan Eulgem, Elgar Fleisch, J. Felix Hampe, Birgitta König-Ries, Franz Lehner, Key Pousttchi, Kai Rannenberg (Hrsg.)
Mobile und Ubiquitäre Informationssysteme
Technologien, Anwendungen und Dienste zur Unterstützung von mobiler Kollaboration

P-164 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2010: Biometrics and Electronic Signatures Proceedings of the Special Interest Group on Biometrics and Electronic Signatures

P-165 Gerald Eichler, Peter Kropf, Ulrike Lechner, Phayung Meesad, Herwig Unger (Eds.)
10th International Conference on Innovative Internet Community Systems (I^2CS) – Jubilee Edition 2010 –

P-166 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
3. DFN-Forum Kommunikationstechnologien
Beiträge der Fachtagung

P-167 Robert Krimmer, Rüdiger Grimm (Eds.)
4th International Conference on Electronic Voting 2010
co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC

P-168 Ira Diethelm, Christina Dörge, Claudia Hildebrandt, Carsten Schulte (Hrsg.)
Didaktik der Informatik
Möglichkeiten empirischer Forschungsmethoden und Perspektiven der Fachdidaktik

P-169 Michael Kerres, Nadine Ojstersek Ulrik Schroeder, Ulrich Hoppe (Hrsg.)
DeLFI 2010 - 8. Tagung der Fachgruppe E-Learning der Gesellschaft für Informatik e.V.

P-170 Felix C. Freiling (Hrsg.)
Sicherheit 2010
Sicherheit, Schutz und Zuverlässigkeit

P-171 Werner Esswein, Klaus Turowski, Martin Juhrisch (Hrsg.)
Modellierung betrieblicher Informationssysteme (MobIS 2010)
Modellgestütztes Management

P-172 Stefan Klink, Agnes Koschmider Marco Mevius, Andreas Oberweis (Hrsg.)
EMISA 2010
Einflussfaktoren auf die Entwicklung flexibler, integrierter Informationssysteme
Beiträge des Workshops der GI-Fachgruppe EMISA (Entwicklungsmethoden für Informationssysteme und deren Anwendung)

P-173 Dietmar Schomburg, Andreas Grote (Eds.)
German Conference on Bioinformatics 2010

P-174 Arslan Brömme, Torsten Eymann, Detlef Hühnlein, Heiko Roßnagel, Paul Schmücker (Hrsg.)
perspeGKtive 2010
Workshop „Innovative und sichere Informationstechnologie für das Gesundheitswesen von morgen“

P-175 Klaus-Peter Fähnrich, Bogdan Franczyk (Hrsg.)
INFORMATIK 2010
Service Science – Neue Perspektiven für die Informatik
Band 1

P-176 Klaus-Peter Fähnrich, Bogdan Franczyk (Hrsg.)
INFORMATIK 2010
Service Science – Neue Perspektiven für die Informatik
Band 2

P-177 Witold Abramowicz, Rainer Alt, Klaus-Peter Fähnrich, Bogdan Franczyk, Leszek A. Maciaszek (Eds.)
INFORMATIK 2010
Business Process and Service Science – Proceedings of ISSS and BPSC

P-178 Wolfram Pietsch, Benedikt Krams (Hrsg.)
Vom Projekt zum Produkt
Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschafts-informatik (WI-MAW), Aachen, 2010

P-179 Stefan Gruner, Bernhard Rumpe (Eds.)
FM+AM`2010
Second International Workshop on Formal Methods and Agile Methods

The titles can be purchased at:

Köllen Druck + Verlag GmbH
Ernst-Robert-Curtius-Str. 14 · D-53117 Bonn
Fax: +49 (0)228/9898222
E-Mail: druckverlag@koellen.de