Modellierung, Identifikation und Strukturierung betrieblicher Systeme in agilen Kontexten

Beiträge zur komponenten- und serviceorientierten Entwicklung betrieblicher Anwendungssysteme

Dissertation
der Wirtschaftswissenschaftlichen Fakultät
der Universität Augsburg
zur Erlangung des Grades eines Doktors
der Wirtschaftswissenschaften
(Dr. rer. pol.)

vorgelegt
von

Sebastian Klöckner
(Dipl.-Kfm., MBA, M.Sc.)

Augsburg, März 2010

Erstgutachter:	Prof. Dr. Klaus Turowski
Zweitgutachter:	Prof. Dr. Axel Tuma
Vorsitzender der mündlichen Prüfung:	Prof. Dr. Marco C. Meier
Datum der mündlichen Prüfung:	21. Mai 2010

Inhaltsverzeichnis

Verzeichnis der Beiträge

I Einleitung

I.1 Zielsetzung und fokussierte Forschungsfragen

I.2 Fachliche Einordnung und Aufbau

II Beiträge zu Aufgaben und Methoden der betrieblichen Anwendungsentwicklung in agilen Umgebungen

II.1 Beitrag: „An empirical Comparison of the Usability of BPMN and UML Activity Diagrams for Business Users"

II.2 Beitrag: „Experience Report: Appropriateness of the BCI-Method for Identifying Business Components in large-scale Information Systems"

II.3 Beitrag: „Zur systematischen Identifikation von Services: Kriterien, aktuelle Ansätze und Klassifikation"

II.4 Beitrag: „A Survey of Service Identification Approaches - Classification Framework, State of the Art, and Comparison"

III Beiträge zu Strukturen und Architekturen betrieblicher Anwendungen in agilen Umgebungen

III.1 Beitrag: „Enabling Interoperability of Networked Enterprises Through an Integrative Information System Architecture for CRM and SCM"

III.2 Beitrag: „Something is Missing: Enterprise Architecture from a Systems Theory Perspective"

III.3 Beitrag: „FAST ACCESS: A System Architecture for RESTful Business Data"

IV Fazit und Ausblick

IV.1 Fazit und weiterer Forschungsbedarf

IV.2 Ausblick

Anmerkung: Eine fortlaufende Seitennummerierung wird pro Kapitel beziehungsweise pro Unterkapitel des jeweiligen Beitrags vorgenommen. Ein Literaturverzeichnis wird jeweils am Ende eines jeden Kapitels beziehungsweise Beitrags aufgeführt.

Verzeichnis der Beiträge

In dieser Dissertation werden die folgenden, jeweils nach wissenschaftlichen Begutachtungsverfahren veröffentlichten und zur Veröffentlichung angenommenen Beiträge vorgestellt:

II.1 Angenommener Beitrag der Kategorie B, in Abhängigkeit von der Annahmequote ggf. Kategorie A

Birkmeier, D.; Klöckner, S.; Overhage, S. (2010), „An empirical Comparison of the Usability of BPMN and UML Activity Diagrams for Business Users“, Proceedings of the 18th European Conference on Information Systems (ECIS), Association for Information Systems, 6.-9. Juni 2010, Pretoria, South Africa.

II.2 Veröffentlichter Beitrag der Kategorie B

Selk, B.; Klöckner, S.; Bazijanec, B.; Albani, A. (2005), „Experience Report: Appropriateness of the BCI-Method for Identifying Business Components in large-scale Information Systems“, Component-Oriented Enterprise Applications: Tagungsband COEA 2005, Lecture Notes in Informatics P-70, Gesellschaft für Informatik, 20. September 2005, Erfurt: 87-92.

II.3 Veröffentlichter Beitrag der Kategorie B

Birkmeier, D.; Klöckner, S.; Overhage, S. (2008), „Zur systematischen Identifikation von Services: Kriterien, aktuelle Ansätze und Klassifikation“, Modellierung zwischen SOA und Compliance Management: Tagungsband MobIS 2008, Lecture Notes in Informatics P-141, Gesellschaft für Informatik, 27.-28. November 2008, Saarbrücken: 255-272; Best Paper Award.

II.4 Veröffentlichter Beitrag der Kategorie B

Birkmeier, D.; Klöckner, S.; Overhage, S. (2009), „A Survey of Service Identification Approaches - Classification Framework, State of the Art, and Comparison“, Enterprise Modelling and Information Systems Architectures - An International Journal, 4(2): 20-36.

III.1 Veröffentlichter Beitrag der Kategorie B

Selk, B.; Klöckner, S.; Albani, A. (2005), „Enabling Interoperability of Networked Enterprises Through an Integrative Information System Architecture for CRM and SCM“, Business Process Management Workshops: Tagungsband BPM 2005, Springer Lecture Notes in Computer Science 3812, 5. September 2005, Nancy, France: 305-316.

III.2 Angenommener Beitrag der Kategorie B

Klöckner, S.; Birkmeier, D. (2010), „Something is Missing: Enterprise Architecture from a Systems Theory Perspective”, Trends in Enterprise Architecture Research (TEAR) @ ICSOC2009: Tagungsband TEAR 2009, Springer Lecture Notes in Computer Science, 23. November 2009, Stockholm, Schweden.

III.3 Veröffentlichter Beitrag der Kategorie B

Klöckner, S. (2009) „FAST ACCESS: A System Architecture for RESTful Business Data“, Proceedings of the 15th American Conference on Information Systems (AMCIS), Association for Information Systems, 6.-9. August 2009, San Francisco, USA: #748.

Die zu den Beiträgen vermerkten Ranking-Kategorien bestimmen sich nach der *Zeitschriften und Ranking* Tabelle der Wirtschaftswissenschaftlichen Fakultät der Universität Augsburg vom 25. September 2006 und nach den fachspezifischen *WI-Orientierungslisten* in der am 27. Februar 2008 in München von der Wissenschaftlichen Kommission Wirtschaftsinformatik im Verband der Hochschullehrer für Betriebswirtschaft e.V. (WKWI) und dem Fachbereich Wirtschaftsinformatik der Gesellschaft für Informatik e.V. (GI-FB WI) verabschiedeten und am 27. Februar 2009 in Wien erweiterten Fassung.

Für Publikationsorgane, die in mehr als einem Ranking erscheinen, wurde nach dem „best of“-Prinzip jeweils die beste Kategorie eingesetzt.

I Einleitung

Die zunehmende Wandlungsgeschwindigkeit der heutigen Unternehmenspraxis zwingt Unternehmen, ihre inner- als auch zwischenbetrieblichen Geschäftsprozesse in immer kürzer werdenden Abständen anzupassen. Viele dieser Prozesse werden jedoch in nicht unerheblichem Maße durch betriebliche Anwendungssysteme unterstützt bzw. sind von diesen abhängig. In der Konsequenz müssen folglich auch die Informationssysteme eines Unternehmens kontinuierlich an die veränderten Geschäftsprozesse angepasst werden. Diese Anpassung stellt die Verantwortlichen jedoch häufig vor nicht zu unterschätzende Probleme, da Adaptionen von beispielsweise monolithischen Anwendungssystemen zumeist nicht oder nur unter hohem Aufwand möglich sind. Insbesondere die starke Kopplung zwischen Anwendungen und das Fehlen von Standards für den Datenaustausch haben zu starren und unflexiblen IT-Landschaften geführt. Geschäftsprozessrelevante Änderungen können daher oft nur durch aufwändiges Programmieren der Ablauflogik innerhalb der verschiedenen Anwendungen ermöglicht werden. Gleichzeitig kann das Ändern und Ersetzen von Teilsystemen jedoch "gefährliche" Nebenwirkungen (Seiteneffekte) nach sich ziehen, wenn beispielsweise unerwartete Auswirkungen bei nicht betrachteten Drittsystemen auftreten. Die daraus resultierenden hohen Wartungs- und Adaptionskosten und die damit verbundene unzureichende Flexibilität und Agilität der Prozesse werden somit zu einem erfolgkritischen Faktor für die Wettbewerbsfähigkeit von Unternehmen.

Es entstand damit der Wunsch nach neuen Methoden und Strukturen bzw. Architekturen, die deutliche Verbesserungen bei der Entwicklung und Anpassung von Softwaresystemen an sich ändernde Geschäftsprozesse erlauben, um die Wettbewerbsfähigkeit des eigenen Unternehmens erhalten und erhöhen zu können. Hierbei besteht die Hoffnung, dass zunehmend flexiblere und agilere Informationssysteme, im Sinne des vielzitierten „Align and Enable“ – also der engen, wechselseitigen Abstimmung zwischen geschäftlichen Zielvorgaben und IT-Potenzialen – zeitnah an die Veränderungen der Umwelt angepasst werden können (Hanschke 2009, S. 7-55).

Bei dem Versuch, sich der aufgezeigten Problemstellung zu nähern, entsteht relativ schnell die Erkenntnis, dass diese wohl aus den zugrundeliegenden Softwaresystemen, also deren Strukturen und Funktionen, oder aus dem Erstellungsprozess dieser Systeme herrühren müssen. Die essenziellen Schwierigkeiten bei der Lösung dieser Fragestellung im Hinblick auf die zugrundeliegenden Softwaresysteme sind dabei laut Brooks (1987, S. 11-12) die folgenden vier Problemstellungen:

Problem der Komplexität

Softwaregebilde sind für ihre Größe deutlich komplexer als vermutlich jede andere menschliche Konstruktion, da sich keine zwei Teile gleichen (sollten). Komplexität von Software ist allerdings eine essenzielle, und keine unbeabsichtigte Eigenschaft. Folglich wird bei der Reduktion von Komplexität (Abstraktion) gleichzeitig meist auch das Wesentliche „wegabstrahiert“. Zusätzlich besitzen Softwaresysteme eine sehr große Zahl von möglichen Zuständen. Dies macht das Erfassen, Beschreiben und Testen außergewöhnlich schwer. Dementsprechend unterscheiden sich Softwaresysteme elementar von Computern, Gebäuden oder auch Automobilen, bei denen reichlich gleiche Teile verwendet werden und mögliche Zustände im Regelfall erfasst, beschrieben und getestet werden können.

Gleichzeitig bedeutet die zuvor aufgezeigte Logik jedoch, dass bei der Vergrößerung eines Softwaregebildes nicht nur die Wiederverwendung von gleichen Elementen in größerer Anzahl, sondern auch eine Zunahme der Anzahl unterschiedlicher Elemente notwendig ist. Da diese in den meisten Fällen in nicht-linearer Art und Weise miteinander interagieren, steigt die Komplexität des Ganzen auch mehr als linear an.

Mit Hilfe der Tatsachen der essenziellen Komplexität und ihrem nicht-linearen Anwachsen bei zunehmender Größe lassen sich bereits viele der klassischen Probleme bei der Entwicklung und Adaption von Softwareprodukten erklären.

- Die Komplexität erzeugt Schwierigkeiten bei der Aufzählung (Spezifikation) von Zuständen und damit einhergehend Unzuverlässigkeit.
- Die Komplexität von Funktionen erzeugt Schwierigkeiten beim Aufruf von Funktionen und damit einhergehend ist das Programm schwer (wieder) zu verwenden.
- Die Komplexität der Struktur erzeugt Schwierigkeiten, bestehende Programme um neue Funktionen zu erweitern, ohne dabei Seiteneffekte zu erzeugen.
- Die Komplexität der Struktur erzeugt unerwartete Zustände, die sich in Sicherheitsproblemen realisieren.

Mit dieser Komplexität entstehen folglich Kommunikationsprobleme zwischen Teammitgliedern und damit einhergehend Produktfehler, Budgetüberschreitungen und Zeitüberschreitungen.

Problem der Konformität bzw. Übereinstimmung

Ein Großteil der Softwareentwicklern begegnenden Komplexität ist willkürliche Komplexität, die ohne Grund durch menschliche Institutionen oder Systeme erzeugt wurde oder wird, und deren Schnittstellen entsprochen werden muss. Diese Komplexität kann dabei von Schnittstelle zu Schnittstelle und von Zeitpunkt zu Zeitpunkt variieren. Allerdings entstehen diese Variationen nicht aufgrund von Notwendigkeiten, sondern nur, weil sie von unterschiedlichen Personen entworfen wurden. Ein Großteil der vorhandenen Komplexität

stammt somit aus dem Zwang zur Konformität zu anderen Schnittstellen und diese Komplexität kann nicht alleine durch die Umgestaltung einer Software reduziert werden.

Problem der Änderbarkeit

Softwareeinheiten unterliegen einem kontinuierlichen Veränderungszwang. Obgleich auch Industriegüter (Automobile, Computer, usw.) mitunter nach ihrer Fertigstellung modifiziert werden, so ist die Häufigkeit solcher Modifikationen im Vergleich zu Softwareprodukten zu vernachlässigen. Bei Software kann dies einerseits darauf zurückgeführt werden, dass die Software eines Systems deren Funktion beinhaltet und diese Funktionalität dem größten Wandlungsdruck unterliegt. Andererseits kann Software verhältnismäßig einfach verändert werden, da es sich eigentlich nur um gedankliche Strukturierungen handelt, die unendlich „verformt“ werden können. Bei Gebäuden ist es hingegen verständlich, dass eine Änderung immer mit hohen Kosten verbunden ist.

Bei der Veränderung von Software sind meist zwei Prozesse maßgeblich:

- Eine erfolgreiche Software wird als nutzbringend erkannt und Menschen beginnen diese auch in Bereichen außerhalb der eigentlichen Anwendungsdomäne zu verwenden. Da diese Nutzer aber zusätzlich zu den bestehenden Funktionen gerne über weitere Funktionen verfügen würden, entsteht Änderungsbedarf.
- Eine erfolgreiche Software überlebt die normale Lebenszeit der zugrundeliegenden Maschine, für die sie ursprünglich geschrieben wurde. Wenn es nicht direkt ein neuer Computer ist, so sind es zumindest neue Festplatten, Displays, usw., an die die Software angepasst werden muss.

Zusammenfassend betrachtet bedeutet dies, dass ein Softwareprodukt in eine kulturelle Matrix aus Anwendungen, Benutzern, Gesetzen und Maschinen eingebettet ist. Da sich diese kontinuierlich ändern, besteht auch der Zwang, die Software zu ändern.

Problem der Unsichtbarkeit

Software ist unsichtbar und nicht visualisierbar. Zwar wären geometrische Abstraktionen, wie beispielsweise bei der Aufdeckung von Widersprüchen und Versäumnissen bei Gebäuden durch Grundrisse, kraftvolle Werkzeuge. Die Realisierung von Software ist jedoch von Natur aus nicht im Raum vorhanden und kann folglich nicht auf einfache Art geometrisch dargestellt werden. So endet der Versuch, Softwarestrukturen in Diagrammen darzustellen, meist nicht nur in einem, sondern vielen, einander überlagernden Diagrammen. Diese können den Kontrollfluss, den Datenfluss, das Abhängigkeitsmuster, die Zeitabläufe, die Namensraumbeziehungen, usw. umfassen und sind dabei in den seltensten Fällen ebenflächig, und noch seltener hierarchisch. Ein anschauliches und bestätigendes Beispiel hierzu liefern die verschiedenen Konzepte der Unternehmensarchitektur (Enterprise

Architecture, (Aier et al. 2008; Frank 2002; Gaertner 2004; IEEE Computer Society 2000; IFIPIFAC Task Force 1999; The Open Group 2002; Uhl 2004; Winter und Fischer 2007; Zachman 1987). Und obgleich Fortschritte in der Beschränkung und Vereinfachung von Softwarestrukturen erreicht wurden, bleibt Software weiterhin von Natur aus unsichtbar und erlaubt damit keine Verwendung der wohl kraftvollsten Werkzeuge des Verstandes. Dieses Fehlen beeinträchtigt dabei nicht nur den Entwurfsprozess innerhalb eines Verstandes, sondern behindert auch massiv die Kommunikation zwischen Verständen (von Personen).

Um den von Brooks aufgezeigten Problemstellungen, insbesondere hinsichtlich der zu bewältigenden Komplexität, und den damit verbundenen Herausforderungen im Hinblick auf Prozess- und Systemadaptionen mit einem ganzheitlichen Ansatz begegnen zu können und damit das zuvor genannte „Align and Enable“ realisieren zu können, hat in den vergangenen Jahren insbesondere das komponenten- und serviceorientierte Entwicklungsparadigma (Natis 2003; Rautenstrauch und Turowski 2001; Turowski 2003; Weerawarana et al. 2005) an Bedeutung gewonnen. Speziell die Reduktion der Komplexität aufgrund der geringeren Größe der Softwareartefakte erscheint hierbei vielversprechend.

Die dabei mitunter vorherrschende Konzentration auf die technischen Aspekte der komponenten- und serviceorientierten Architekturen, wie beispielsweise Web Services, hat jedoch dazu geführt, dass organisatorische als auch betriebliche Gesichtspunkte mitunter erneut vernachlässigt wurden (CIO-Worldnews 2008; Picot und Baumann 2009).

Um die fachlichen, und damit die den Erstellungs- und Adaptionsprozess von Softwaresystemen beeinflussenden Aspekte, stärker zu berücksichtigen, gewann in den letzten Jahren vermehrt die Idee der Unternehmensmodellierung sowie -architekturen an Bedeutung (Ferstl und Sinz 2006; Frank 2002; Österle und Blessing 2003; Scheer 2002). Insbesondere die Geschäftsprozessmodellierung und die damit einhergehende Process Governance (COBIT, ITIL, usw.) zählen aktuell zu den Schwerpunkten der Praxis (Hanschke 2009) und lösen dabei nicht unerhebliche Adaptionsanforderungen für die zugrundeliegenden Informationssysteme aus. Um die Komplexität aus Anforderungen aufgrund der betrieblichen Ziele, sowie die vorhandenen Beziehungen der bestehenden bzw. zu entwickelnden Systeme zumindest teilweise bewältigen zu können, verwenden heutige Unternehmen nicht selten das Konzept der Unternehmensarchitektur (Gaertner 2004; Jung 2004; Mertens 2004; Wöbking 2004).

Wie Abbildung 1 zeigt, versucht das Konzept der Enterprise Architecture sowohl geschäfts-orientierte Elemente, wie beispielsweise strategische Ziele, Produkt-/Marktsegmente, Prozesse, Organisationseinheiten, usw., aber auch technische Aspekte, wie beispielsweise Anwendungen, fachliche Services, Informationsobjekte, Softwarekomponenten, usw., zu

verbinden. Dabei stehen insbesondere die Berücksichtigung und Verknüpfung strategischer, personeller als auch technischer Aspekte und Einflussfaktoren im Vordergrund.

Im Rahmen der Konzepte und wissenschaftlichen Beiträge im Bereich der Enterprise Architecture wird erneut deutlich, dass die Reduktion der Komplexität zu den Hauptzielen der jeweiligen Ansätze gehört (z.B. Aier et al. 2008; IEEE Computer Society 2000; The Open Group 2002; Zachman 1987). Damit gewinnt auch aus dieser Perspektive das komponenten- und serviceorientierte Entwicklungsparadigma zunehmend an Bedeutung und Interesse.

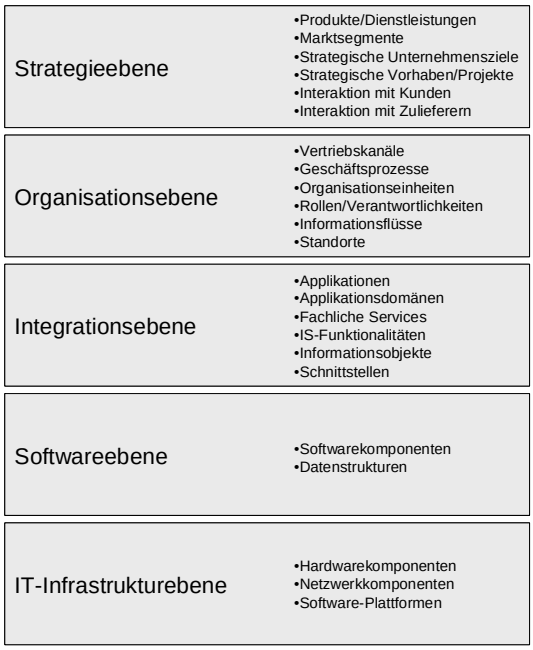

Abbildung 1: Enterprise Architecture nach Aier et. al. (2008)

Die vorliegende Arbeit motiviert sich daher, im strategischen Rahmen des IT-Managements, der Software-Wiederverwendung als auch der damit in Verbindung stehenden Informationsmodellierung, über die zu erwartenden Vorteile des komponenten- und serviceorientierten Entwicklungsparadigmas und der damit einhergehenden, zu prognostizierenden Komplexitäts- und Kostenreduktion. In diesem Kontext will sie sowohl beschreibende, erklärende, als auch gestalterische Beiträge zum Fortschritt bei der Entwicklung betrieblicher Anwendungssysteme beisteuern.

Nachdem nun einleitend die Potenziale der komponenten- und serviceorientierten Entwicklung von Informationssystemen, der Unternehmensmodellierung und -architekturen, sowie ihre Zweckdienlichkeit bei der Adaption und Integration von betrieblichen Informationssystemen motiviert und erläutert wurden, beschreibt Abschnitt I.1 die konkrete Zielsetzung und die untersuchten Forschungsfragen der einzelnen Beiträge. Anschließend wird in Abschnitt I.2 auf die fachliche Einordnung und den Aufbau der Arbeit im Detail eingegangen.

I.1 Zielsetzung und fokussierte Forschungsfragen

Die Wirtschaftsinformatik (WI) versteht sich als Wissenschaft mit einer methodenpluralistischen Erkenntnisstrategie, die sich Instrumenten aus Real-, Formal- und Ingenieurswissenschaften bedient (Wilde und Hess 2007, S. 1; Wissenschaftliche Kommission Wirtschaftsinformatik 1994). Die Teildisziplinen Datenverarbeitung (EDV) und Operations Research (OR) können als wichtige personenbezogene Wurzeln der WI identifiziert werden, wobei die frühen Vertreter der Disziplin auch einen Hintergrund aus der Praxis hatten (Lange 2006, S. 5). Gegenstand der Wirtschaftsinformatik sind hierbei Informations- und Kommunikationssysteme (IKS) in Wirtschaft und Verwaltung (Wissenschaftliche Kommission Wirtschaftsinformatik 1994). Das Ziel wissenschaftlicher Untersuchungen in der Wirtschaftsinformatik kann somit als die Gewinnung von Theorien, Methoden, Werkzeugen und nachprüfbaren Erkenntnissen zu Mensch-Aufgabe-Technik-Systemen und -Infrastrukturen der Information und Kommunikation (sozio-technischen Systemen) in Wirtschaft und Verwaltung beschrieben werden, wobei langfristig die „sinnhafte Vollautomation“ (Mertens 1995, S. 48) angestrebt wird. Zu den Hauptaufgaben der Wirtschaftsinformatik zählen hierbei die Beschreibung, Erklärung und Gestaltung des Untersuchungsgegenstandes, wobei die deutschsprachige WI zu konstruktionsorientierten Methoden und praxisorientierten Arbeiten zur Gewinnung und Validierung von Kenntnissen wie beispielsweise dem Erstellen und Evaluieren von Prototypen neigt (Frank 2006, S. 1; Goeken 2003, S. 9-10; Wilde und Hess 2007) und als notwendig betrachtet (Wissenschaftliche Kommission Wirtschaftsinformatik 1994, S. 81).

Ziel dieser Dissertationsschrift ist es, durch beschreibende, erklärende als auch gestaltende Beiträge den Erkenntnisfortschritt in der Wirtschaftsinformatik zusätzlich zu befördern. Die im Hauptteil dieser Arbeit vorgestellten Beiträge sollen somit durch die Beschreibung und Erklärung der vorhandenen Fragestellungen im Bereich der Geschäftsprozessmodellierung, als Teilbereich der Unternehmensmodellierung und der komponenten- und serviceorientierten Anwendungsentwicklung, sowie das Aufzeigen möglicher neuer Gestaltungs- und Lösungsansätze im Bereich der modularisierten Anwendungsentwicklung zur Beantwortung offener Forschungsfragen auf dem Gebiet der Wirtschaftsinformatik beitragen. Im Bereich der Geschäftsprozessmodellierung und der darauf aufbauenden komponenten- und serviceorientierten Anwendungsentwicklung stehen hierbei die Verwendbarkeit (usability) von Modellierungssprachen, sowie Ansätze zur subsequenten Identifikation von fachlichen Komponenten als auch Services im Zentrum der Untersuchungen. Andererseits soll, auf Basis von Gestaltungsvorschlägen zu Architekturen von Unternehmenssystemen, ein weiterer Schritt in Richtung der „sinnvollen Vollautomation“ gemacht werden. Im Zentrum der Überlegungen stehen hierbei eine Verbesserung der

Kooperation zwischen Unternehmen durch Verwendung einer integrativen Informationssystemarchitektur in den Bereichen Customer Relationship Management (CRM) und Supply Chain Management (SCM), eine systemtheoretische Betrachtung der vorhandenen Unternehmensarchitekturkonzepte sowie mögliche Verbesserungsvorschläge und eine neuartige, REST-basierte Kopplungsarchitektur zur Integration von Unternehmensdaten. In den einzelnen Kapiteln und Beiträgen dieser Arbeit werden daher die folgenden Forschungsfragen genauer untersucht:

II.1 Beitrag: „An empirical Comparison of the Usability of BPMN and UML Activity Diagrams for Business Users"

Das große Interesse an Geschäftsprozessmanagementstrategien hat den Bedarf an integrierten Ansätzen der Geschäftprozessmodellierung stark erhöht. Unter anderem wird die Business Process Modeling Notation (BPMN) als potenzieller Industriestandard in Betracht gezogen. So erklärte beispielsweise die Object Management Group (OMG) BPMN anstatt UML Aktivitätsdiagramme (UML AD) zum Kernstandard für die Erstellung eines Rahmenwerks für die Geschäftsmodellierung. Für Unternehmen ist der Wechsel auf eine neue Geschäftsprozessmodellierungssprache jedoch ein nicht unerheblicher Kostenfaktor. In diesem Beitrag wird daher anhand einer umfassenden empirischen Studie während einer Modellerstellungsaufgabe untersucht, ob BMPN tatsächlich als Industriestandard zu empfehlen ist. Es werden hierbei die folgenden fokussierten Forschungsfragen untersucht:

- Sind UML Aktivitätsdiagramme für Fachanwender mindestens genauso gut verwendbar wie BMPN Modelle?
- Wie kann die Verwendbarkeit von Modellierungssprachen gemessen sowie verglichen werden und welche Faktoren sind dabei ausschlaggebend?
- Welche Schlussfolgerungen lassen sich aus der jeweiligen Verwendbarkeit von UML Aktivitätsdiagrammen und BPMN-Modellen für Forschung und Praxis ableiten?

II.2 Beitrag: „Experience Report: Appropriateness of the BCI-Method for Identifying Business Components in large-scale Information Systems"

Der Einsatz von Fachkomponenten in großen Unternehmensinformationssystemen bietet enormes Potenzial. Dennoch ist sowohl der Findungsprozess als auch die Bestimmung der richtigen Fachkomponenten eine Herausforderung. Dieser Beitrag zielt darauf ab, Erfahrungen zu illustrieren, die während des Modellierungsprozesses einer integrierten Informationssystemarchitektur mit mehr als 500 Funktionen und 1000 Informationsobjekten und unter Verwendung der Business Component Identification

(BCI)-Methode gemacht wurden. Es werden hierbei die folgenden fokussierten Forschungsfragen untersucht:

- Welche Erfahrungen wurden während eines Komponentenfindungsprozesses unter Verwendung der Business Component Identification (BCI)-Methode gemacht?
- Welche Erweiterungen sind aufgrund dieser Erfahrungen notwendig und warum ist dies der Fall?
- Welche Verbesserungen wären aufgrund dieser Erfahrungen möglich und warum ist dies der Fall?

II.3 Beitrag: „Zur systematischen Identifikation von Services: Kriterien, aktuelle Ansätze und Klassifikation“

Die Einführung serviceorientierter Architekturen verspricht eine Vielzahl von Vorteilen für die betriebliche Anwendungsentwicklung. Derzeit steht daher insbesondere die Entwicklung systematischer Methoden für die Identifikation von Services im Mittelpunkt des wissenschaftlichen Interesses. Die in der Literatur vorhandenen Ansätze weisen jedoch hinsichtlich ihrer Konzeption und Vorgehensweise eine starke Heterogenität auf. In diesem, mit einem „Best Paper Award“ ausgezeichneten, Beitrag sollen daher die vorhandenen Ansätze anhand eines detaillierten Klassifikationsschemas einander gegenübergestellt, die jeweiligen Stärken und Schwächen sowie bestehender Forschungsbedarf herausgearbeitet werden. Es werden hierbei die folgenden fokussierten Forschungsfragen untersucht:

- Welche grundlegenden Kriterien können zur Einordnung der verschiedenen Ansätze zur Serviceidentifikation verwendet werden?
- Welche Stärken und Schwächen können bei den einzelnen Ansätzen identifiziert werden?
- Welcher weitere Forschungsbedarf kann aus den vorhandenen Schwächen abgeleitet werden?

II.4 Beitrag: „A Survey of Service Identification Approaches - Classification Framework, State of the Art, and Comparison“

Im Rahmen des Beitrages II.3 wurde ein Klassifikationsschema zur Identifikation von Stärken und Schwächen von Ansätzen zur Serviceidentifikation vorgestellt sowie in der Literatur zu findenden Ansätze gegenübergestellt und vorhandene Schwächen identifiziert. Während der MobIS-Fachtagung 2008 wurden jedoch Fragen hinsichtlich der Abgrenzung zu den Definitionen aus der Service Science Disziplin, der Unabhängigkeit der einzelnen Klassifikationskriterien, der Ausweitungsmöglichkeiten der vergleichenden Diskussion der Ansätze sowie der Schlussfolgerungen geäußert. In diesem Beitrag soll

daher den vorhandenen Schwächen in Beitrag II.3 begegnet und diese behoben werden. Es werden hierbei die folgenden fokussierten Forschungsfragen untersucht:

- Wie können die vorhandenen Definitionen schärfer von den Definitionen der Service Science Disziplin abgegrenzt werden?
- Wie kann die Unabhängigkeit der einzelnen Klassifikationskriterien aufgezeigt werden?
- Welche weiteren Stärken und Schwächen können bei den vorhandenen Ansätzen identifiziert werden und wie können diese im Rahmen des Vergleichs expliziter dargestellt werden?
- Welche weiteren Schlussfolgerungen können gezogen werden und wie können diese expliziter dargestellt werden?

III.1 Beitrag: „Enabling Interoperability of Networked Enterprises Through an Integrative Information System Architecture for CRM and SCM“

Mit dem Einsatz spezialisierter Anwendungssysteme nimmt die Komplexität der Beziehungen zwischen Unternehmen kontinuierlich zu. Gleichzeitig steigt jedoch die Notwendigkeit Informationen zwischen Unternehmen, die Teil von Wertschöpfungsnetzen sind, auszutauschen. Der Einsatz einer integrierten Informationssystemarchitektur (ISA) würde in diesem Fall die interorganisationale Integration deutlich vereinfachen. Dieser Beitrag zielt daher darauf ab, auf Basis zweier Beispiele darzustellen, wie eine integrierte Informationssystemarchitektur für das Customer Relationship Management (CRM) und das Supply Chain Management (SCM) die interorganisationale Integration unterstützen und den kontinuierliche Austausch zwischen Unternehmen eines Wertschöpfungsnetzes ermöglichen kann. Es werden hierbei die folgenden fokussierten Forschungsfragen untersucht:

- Warum ist eine intraorganisationale Integration Voraussetzung für eine interorganisationale Integration?
- Wie kann eine auf Fachkomponenten basierende Informationssystemarchitektur die Interoperabilität in Wertschöpfungsnetzen unterstützen?
- Welche Typen von Fachkomponenten sind für die Unterstützung dieser Interoperabilität notwendig und wie müssen diese bei einem interorganisationalen System zusammengesetzt werden?
- Welche potenziellen Lösungsansätze sind für den kontinuierlichen Austausch von Informationen zwischen Unternehmen eines Wertschöpfungsnetzes darüber hinaus sinnvoll?

III.2 Beitrag: „Something is Missing: Enterprise Architecture from a Systems Theory Perspective"

Die Idee der Unternehmensarchitektur (Enterprise Architecture, EA) wurde in der letzten Dekade, insbesondere von Praktikern aus dem IT-Bereich, zu einem vielversprechenden und umfassenden Ansatz zur Modellierung des aktuellen (IST) oder gewünschten (SOLL) Zustands eines Unternehmens weiterentwickelt. Die bestehenden Ansätze werden jedoch häufig dafür kritisiert, dass sie den fachlichen Nebenbedingungen, Interessen und Zielen eines Unternehmens noch immer nicht gerecht werden. In diesem Beitrag soll aus systemtheoretischer Perspektive analysiert werden, ob, warum und wie weitere Aspekte der Unternehmenswelt in das Konzept der EA integriert werden können. Es werden hierbei die folgenden fokussierten Forschungsfragen untersucht:

- Welches Verständnis und welche Konzepte von Unternehmensarchitekturen (Enterprise Architecure) bestehen bisher in Wissenschaft und Praxis?
- Welche konzeptionellen Schwachstellen können aus systemtheoretischer Perspektive bei den existierenden Unternehmensarchitekturkonzepten identifiziert werden?
- Warum und wie kann/soll insbesondere der menschliche Faktor, als flexibelstes und agilstes Element eines Unternehmens, in das Konzept der EA integriert werden und welche Konsequenzen würden sich daraus ergeben?

III.3 Beitrag: „FAST ACCESS: A System Architecture for RESTful Business Data"

Insbesondere serviceorientierte Architekturen und Web Services sowie ihr Einsatz in Unternehmenssystemen und -architekturen sind Gegenstand einer Vielzahl von Forschungsaktivitäten und erfahren daher große Aufmerksamkeit. Web Services werden, obwohl ihr Aufbau zunehmend komplex wird und somit neue Problemstellungen aufwirft, mitunter sogar als das neue Paradigma für die Entwicklung verteilter Anwendungen und Systeme betrachtet. Interessanterweise wurde jedoch bisher das ressourcenorientierte Paradigma des REpresentational State Transfer (REST), obgleich es einer der Hauptfaktoren für den Erfolg des World Wide Web (WWW) ist, kaum für Verwendung bei Unternehmenssystemen und -architekturen in Betracht gezogen. In diesem Beitrag wird daher eine neue Systemarchitektur vorgestellt, die auf den Grundprinzipien von REST basiert und für die Integration von Unternehmenssystemen verwendet werden kann. Es werden hierbei die folgenden fokussierten Forschungsfragen untersucht:

- Welche Problemstellungen bestehenden bei serviceorientierten Architekturen auf Basis von klassischen Web Services?
- Welche Eigenschaften zeichnen eine REST-basierte Architektur aus und wie könnten diese bei der Integration von Unternehmensdaten hilfreich sein?

- Wie müsste eine REST-basierte Kopplungsarchitektur für die Integration von Unternehmensdaten aufgebaut sein und welche Vorteile würden sich daraus ergeben?

Auf Basis der zuvor dargestellten Zielsetzung der Arbeit sowie den fokussierten Forschungsfragen werden in dem folgenden Abschnitt die fachliche Einordnung der Arbeit sowie der Aufbau der Arbeit dargestellt.

I.2 Fachliche Einordnung und Aufbau

Die vorliegende Arbeit greift die Grundidee der Wirtschaftsinformatik auf und versucht sowohl im Bereich des Software Engineering (Sommerville 2001) als auch der Unternehmensmodellierung (Business Engineering, (Ferstl und Sinz 2006, S. 185-186; Frank 2002; Österle und Blessing 2003, S. 81; Scheer 2002)) einen Beitrag zu leisten. Hierbei möchte sie sowohl Methoden als auch Werkzeuge, im Sinne von Architekturen, evaluieren bzw. validieren sowie präsentieren.

Sie greift einerseits die Konzepte und Methoden der modularen Anwendungsentwicklung auf und liefert Ergebnisse, die bei der Gestaltung und Entwicklung solcher Systeme sowohl notwendig wie auch hilfreich sind. Im Rahmen der Beiträge werden daher Methoden der Prozessmodellierung als auch der Identifikation von Komponenten und Services genauer untersucht.

Andererseits präsentiert diese Arbeit verschiedene Architekturmodelle, die eine „sinnhafte Automatisierung“ von zwischen- als auch innerbetrieblicher Kooperation unterstützen können und stellt damit neuartige Artefakte im Bereich der (Unternehmens-)architekturen, sowohl auf technischer als auch auf fachlicher Seite, zur Verfügung. Die vorgestellten Architekturen betrachten einerseits die interorganisationalen Integrations- und Realisierungsmöglichkeiten für eine integrierte Informationssystemarchitektur für das CRM und SCM als auch stärker technisch orientierte Architekturen im Bereich der verteilten Datenhaltung auf Basis der REpresentational State Transfer (REST). Darüber hinaus wird aus systemtheoretischer Perspektive das, der Unternehmensmodellierung zuzurechnende Konzept der Enterprise Architecture genauer untersucht.

Die vorliegende Arbeit beruht dabei prinzipiell auf der Grundidee des Design-Science-Ansatzes (Hevner et al. 2004) und folgt damit auf abstraktem Niveau einem iterativ-inkrementellen Vorgehen mit den Elementen Problemstellung, Konzeption, Lösungsansatz, Evaluation und der sich daraus ergebenden neuen Problemstellung. Da diese Arbeit jedoch nur ausgewählte und veröffentlichte bzw. zur Veröffentlichung angenommene Arbeiten enthält, können einzelne Zwischenschritte mitunter nicht direkt aus dieser Arbeit nachvollzogen werden. Vielmehr müssten auch weitere, bisher unveröffentlichte Arbeiten

des Autors hinzugezogen werden, um die Einzelschritte vollständig nachvollziehen zu können. Dennoch tragen bereits die hier dargestellten Arbeiten zu einem Erkenntnisfortschritt im Forschungsgebiet der Wirtschaftsinformatik bei. Abbildung 2 stellt den Gesamtaufbau der Arbeit grafisch dar.

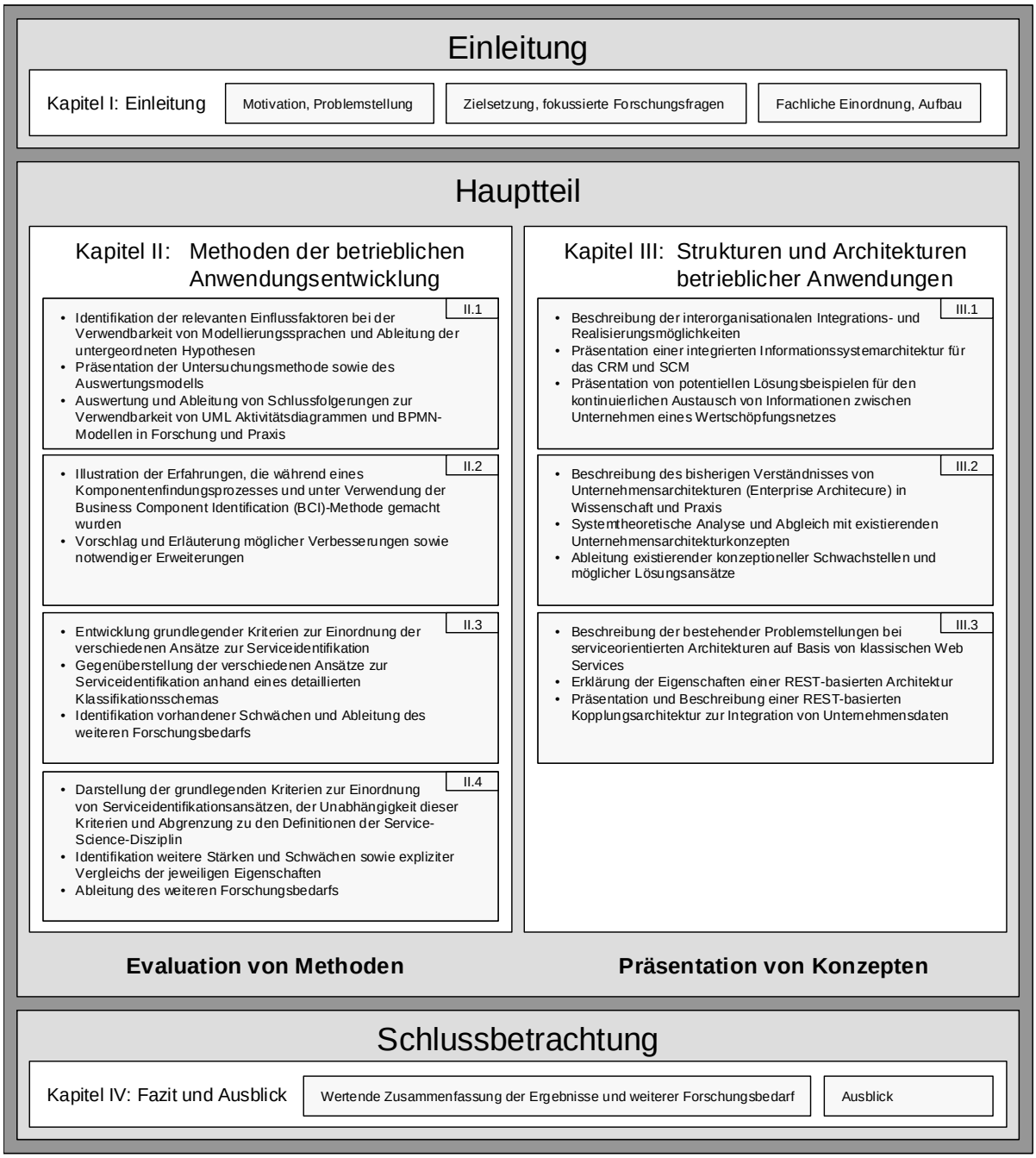

Abbildung 2: Aufbau der Arbeit

Kapitel II beschäftigt sich vorrangig mit der Evaluation von Methoden im Bereich der Unternehmensmodellierung und des Software Engineering.

Da im Rahmen des Requirements Engineering die Ist-Analyse der Prozesse und die richtige und vollständige Ermittlung der Anforderungen (Becker et al. 1995) von elementarer Bedeutung für die nachfolgenden Schritte ist, wird in Beitrag II.1 ein Auszug aus einer breit

angelegten empirischen Untersuchung zur Verwendbarkeit von Modellierungssprachen (UML AD, eEPK, BPMN, Petrinetze und Fachnormsprache) bei Modellerstellungsaufgaben präsentiert. Der Beitrag fokussiert dabei auf die Verwendbarkeit von UML Aktivitätsdiagrammen (UML AD) und BPMN für Fachanwender.

Die anschließenden Beiträge II.2, II.3 und II.4 fokussieren auf die, dem Requirements Engineering folgenden, Phase der Identifikation und Spezifikation von Komponenten und Services. Im Mittelpunkt steht dabei vornehmlich die Bestimmung der optimalen Granularität, d.h. Komplexität der Funktion, von Softwarekomponenten und Services, die insbesondere für die Praxis grundlegende Auswirkungen auf die Wiederverwendbarkeit als auch Adaption dieser Anwendungssystembausteine und die damit verbundenen Kosten hat. Die zunehmende Bedeutung dieser Thematik ist dabei nicht überraschend, da die ansteigende Wandlungsgeschwindigkeit der heutigen Unternehmenspraxis Unternehmen zur kontinuierlichen Anpassung ihrer Geschäftsprozesse, und damit einhergehend der sie unterstützenden Anwendungssysteme, zwingt. Die Adaption dieser Anwendungssysteme gestaltet sich dabei, wie zuvor bereits dargestellt, jedoch meist schwierig und kostenintensiv.

In Beitrag II.2 werden daher Erfahrungen, die während eines Komponenten-findungsprozesses mit mehr als 500 Funktionen und 1000 Informationsobjekten im Bereich des CRM und SCM und unter Verwendung der Business Component Identification (BCI)-Methode gemacht wurden, illustriert und mögliche Verbesserungen sowie notwendige Erweiterungen vorgeschlagen.

Im Rahmen des mit einem Best-Paper-Awards ausgezeichneten Beitrags II.3 werden grundlegende Kriterien zur Einordnung verschiedener Ansätze zur Serviceidentifikation entwickelt und 13 in der Literatur vorhandene Serviceidentifikationsansätze anhand dieser Kriterien gegenübergestellt. Basierend auf dieser Gegenüberstellung werden vorhandene Stärken und Schwächen der Identifikationsansätze aufgezeigt, sowie der daraus resultierende weitere Forschungsbedarf aufgezeigt.

Beitrag II.4 baut schließlich auf Beitrag II.3 auf und versucht, die während der Präsentation des Beitrages II.3 auf der MoBIS 2008 geäußerten, Fragestellungen und Kritikpunkte zu beantworten bzw. zu beseitigen. Hierzu wird die Unabhängigkeit der Klassifikationskriterien deutlicher herausgearbeitet, eine schärfere Abgrenzung zu den Definitionen der Service-Science-Disziplin vorgenommen sowie weitere Stärken und Schwächen bei den vorhandenen Ansätzen identifiziert und im Rahmen des Vergleichs expliziter dargestellt.

Kapitel III beschäftigt sich vorwiegend mit der Präsentation von Konzepten (Architekturen) im Bereich der Unternehmensmodellierung und des Software Engineering. Hierbei werden sowohl Systemarchitekturen im Bereich der betrieblichen Anwendungssysteme als auch theoretische Betrachtungen bestehender Konzepte aus dem Forschungsgebiet der Unternehmensarchitektur vorgestellt und resultierende Verbesserungsvorschläge präsentiert.

In Beitrag III.1 werden daher zunächst die interorganisationalen Integrations- und Realisierungsmöglichkeiten einer integrierten Informationssystemarchitektur für das CRM und SCM genauer beleuchtet und im weiteren Verlauf ein potenzieller Lösungsansatz für den kontinuierlichen Austausch von Informationen zwischen Unternehmen eines Wertschöpfungsnetzes präsentiert.
Beitrag III.2 wechselt auf die theoretische Ebene der Systemtheorie der Technik. Auf Basis einer Analyse des bisherigen Verständnisses von Unternehmensarchitekturen (Enterprise Architecure) in Wissenschaft und Praxis werden in Verbindung mit den allgemeinen Konstrukten der Systemtheorie der Technik existierende konzeptionelle Schwachstellen aufgezeigt und mögliche Lösungsansätze, insbesondere unter Berücksichtigung des menschlichen Faktors, präsentiert.
Beitrag III.3 stellt schließlich eine neuartige, und mitunter mit der Grundidee klassischer Web Services in Konflikt stehende Kopplungsarchitektur auf Basis des Architekturansatzes des REpresentational State Transfers (REST) vor. Hierbei werden zunächst bestehende Problemstellungen bei serviceorientierten Architekturen auf Basis von klassischen Web Services sowie die Eigenschaften einer REST-basierten Architektur beschrieben und erklärt. Darauf aufbauend wird dann die entsprechende REST-basierten Kopplungsarchitektur zur Integration von Unternehmensdaten vorgestellt.

Die vorliegende Arbeit ordnet sich somit inhaltlich innerhalb der Wirtschaftsinformatik in das grundsätzliche Leitbild der komponenten- und serviceorientierten Entwicklung betrieblicher Anwendungssysteme ein (Turowski 2003, S. 9-15) und versucht dabei sowohl Aspekte des Software Engineering als auch der Unternehmensmodellierung zu berücksichtigen. Das strategische Thema der Wiederverwendung, welche Hand in Hand mit dem komponenten- und serviceorientierten Entwicklungsparadigma einhergeht, bildet dabei den Ausgangspunkt und Hintergrund des Hauptteils der Arbeit. Wie sich aus dem Aufbau der Arbeit mitunter erkennen lässt, wurden die einzelnen Beiträge jeweils durch Ergebnisse der vorhergehenden Untersuchungen und Erfahrungen bzw. Prototypen und Architekturkonzepte beeinflusst bzw. angestoßen.

Literatur

Aier S, Riege C, Winter R (2008) Unternehmensarchitektur – Literaturüberblick und Stand der Praxis. WIRTSCHAFTSINFORMATIK 50(4):292-304

Becker J, Rosemann M, Schütte R (1995) Grundsätze ordnungsmäßiger Modellierung. WIRTSCHAFTSINFORMATIK 37(5):435-445

Brooks FP (1987) No Silver Bullet Essence and Accidents of Software Engineering. IEEE Computer 20(4):10-19

CIO-Worldnews (2008) SOA growth projections shrinking. http://www.cio.de/862080. Abruf am 2009-11-04

Ferstl OK, Sinz EJ (2006) Grundlagen der Wirtschaftsinformatik, 5 Aufl. Oldenbourg, München

Frank U (2002) Multi-perspective Enterprise Modeling (MEMO) - Conceptual Framework and Modeling Languages. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02), Hawaii, USA

Frank U (2006) Towards a Pluralistic Conception of Research Methods in Information Systems. ICB Research Report No. 7, Universität Duisburg-Essen

Gaertner W (2004) Ansatz für eine erfolgreiche Enterprise Architecture im Bereich Global Banking Division/Global Transaction Banking IT and Operations der Deutschen Bank. WIRTSCHAFTSINFORMATIK 46(4):311-313

Goeken M (2003) Die Wirtschaftsinformatik als anwendungsorientierte Wissenschaft. Symptome, Diagnose und Therapievorschläge. Arbeitsbericht des Instituts für Wirtschaftsinformatik, Philipps-Universität Marburg

Hanschke I (2009) Strategisches Management der IT-Landschaft. Ein praktischer Leitfaden für das Enterprise Architecture Management. Hanser Fachbuch, München

Hevner AR, March ST, Park J, Ram S (2004) Design Science in Information Systems Research. MIS Quarterly 28(1):75-105

IEEE Computer Society (2000) Recommended Practice for Architectural Description of Software-Intensive Systems.

IFIPIFAC Task Force (1999) GERAM: Generalised Enterprise Reference Architecture and Methodology Version 1.6.3. http://www.cit.gu.edu.au/~bernus/taskforce/geram/versions/geram1-6-3/GERAMv1.6.3.pdf. Abruf am 2009-09-03

Jung E (2004) Ein unternehmensweites IT-Architekturmodell als erfolgreiches Bindeglied zwischen der Unternehmensstrategie und dem operativen Bankgeschäft. WIRTSCHAFTSINFORMATIK 46(4):313-315

Lange C (2006) Entwicklung und Stand der Disziplinen Wirtschaftsinformatik und Information Systems – Teil III: Ergebnisse zur Wirtschaftsinformatik. ICB Research Report No. 4, Universität Duisburg-Essen

Mertens P (1995) Wirtschaftsinformatik – Von den Moden zum Trend. In: König W (Hrsg) Tagungsband der Wirtschaftsinformatik `95 – Wettbewerbsfähigkeit Innovation Wirtschaftlichkeit, Frankfurt am Main

Mertens P (2004) Diskussionsrunde zum Thema „Unternehmensarchitekturen in der Praxis“. WIRTSCHAFTSINFORMATIK 46(4):315

Natis YV (2003) Service-Oriented Architecture Scenario. http://www.gartner.com/resources/114300/114358/114358.pdf. Abruf am 2009-02-16

Österle H, Blessing D (2003) Business Engineering Model. In: Österle H, Winter R, Baumöl U (Hrsg) Business Engineering: Auf dem Weg zum Unternehmen des Informationszeitalters. Springer, Berlin, 65-85

Picot A, Baumann O (2009) The Relevance of Organisation Theory to the Field of Business and Information Systems Engineering. Business & Information Systems Engineering 1(1):62-69

Rautenstrauch C, Turowski K (2001) Common Business Component Model (COBCOM): Generelles Modell komponentenbasierter Anwendungssysteme. In: Buhl HU, Huther A, Reitwiesner B (Hrsg) Tagungsband der 5. Internationalen Tagung Wirtschaftinformatik: Information Age Economy, Augsburg

Scheer A-W (2002) ARIS – Vom Geschäftsprozess zum Anwendungssystem, 4 Aufl. Springer, Berlin

Sommerville I (2001) Software Engineering, 6 Aufl. Pearson Studium, München

The Open Group (2002) TOGAF "Enterprise Edition" Version 8.1. http://www.opengroup.org/architecture/togaf8-doc/arch/. Abruf am 2009-09-03

Turowski K (2003) Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme. Shaker Verlag, Aachen

Uhl J (2004) „Unternehmensarchitekturen“ ist ein Dauerthema – aber die Ziele bzw. Motivation und damit Schwerpunkte ändern sich, vor allem mit wirtschaftlichen Randbedingungen. WIRTSCHAFTSINFORMATIK 46(4):317

Weerawarana S, Curbera F, Leymann F, Storey T, Ferguson DF (2005) Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice Hall PTR, Upper Saddle River

Wilde T, Hess T (2007) Forschungsmethoden der Wirtschaftsinformatik – Eine empirische Untersuchung. WIRTSCHAFTSINFORMATIK 49(4):280–287

Winter R, Fischer R (2007) Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. Journal of Enterprise Architecture 3(2): 7-18

Wissenschaftliche Kommission Wirtschaftsinformatik (1994) Profil der Wirtschaftsinformatik. WIRTSCHAFTSINFORMATIK 36(1):80-81

Wöbking F (2004) Unternehmensarchitekturen in der Praxis – Architekturdesign am Reißbrett vs. situationsbedingte Realisierung von Informationssystemen. WIRTSCHAFTSINFORMATIK 46(4):319-320

Zachman JA (1987) A framework for information systems architecture. IBM Systems Journal 26(3):277-293

II Beiträge zu Aufgaben und Methoden der betrieblichen Anwendungsentwicklung in agilen Umgebungen

Die Evaluation bzw. Validierung existierender Methoden und Strukturen gehört zu den Kernaufgaben der Wirtschaftsinformatik. Hierbei sind sowohl Aspekte der Unternehmensmodellierung als auch des Software Engineering von Bedeutung.

Daher werden in diesem Kapitel ein Beitrag zur Evaluation von Modellierungssprachen und drei Beiträge zur Identifikation von Softwarekomponenten und -services vorgestellt. Unterkapitel II.1 stellt den Beitrag „An empirical Comparison of the Usability of BPMN and UML Activity Diagrams for Business Users“ vor. Dieser präsentiert dem Leser eine empirische Untersuchung zur Verwendbarkeit (Usability) von UML Aktivitätsdiagrammen (UML AD) und der Business Process Modelling Notation (BPMN). Im Rahmen dieser Untersuchung hat sich gezeigt, dass es empirisch nicht widerlegbar ist, dass UML Aktivitätsdiagramme für Fachanwender nicht mindestens genauso gut verwendbar (usable) sind wie BPMN.

Darüber hinaus werden in den Unterkapiteln II.2, II.3 und II.4 Methoden zur Komponenten- und Serviceidentifikation evaluiert. Unterkapitel II.2 präsentiert mit dem Beitrag „Experience Report: Appropriateness of the BCI-Method for Identifying Business Components in large-scale Information Systems“ einen Erfahrungsbericht zur Anwendbarkeit der Business Component Identification (BCI) Methode bei einer Vielzahl von Funktionen und Informationsobjekten. Hierbei stellte sich unter anderem heraus, dass die Startlösung einen nicht unerheblichen Einfluss auf die von der BCI-Methode gelieferten Ergebnisse hat.

Die Beiträge II.3 und II.4 präsentieren schließlich eine umfangreiche Untersuchung von 13 in der Literatur vorhandenen Ansätzen zur Serviceidentifikation. Im Rahmen der Untersuchung stellte sich heraus, dass einerseits die den Ansätzen zugrundeliegenden Servicedefinitionen als auch der jeweilige Formalisierungsgrad mitunter stark variieren. Andererseits besteht erheblicher Forschungsbedarf hinsichtlich der Weiterentwicklung der Ansätze zu ausgereiften Methoden.

II.1 Beitrag: „An empirical Comparison of the Usability of BPMN and UML Activity Diagrams for Business Users”

Autoren: Dominik Birkmeier, Sebastian Klöckner, Sven Overhage
Alle Lehrstuhl WI-SE, Universität Augsburg,
Universitätsstraße 16, D-86135 Augsburg,
Email: dominik.birkmeier @wiwi.uni-augsburg.de
sebastian.kloeckner@wiwi.uni-augsburg.de
sven.overhage@wiwi.uni-augsburg.de

Erscheint in: Proceedings of The European Conference on Information Systems 2010 (ECIS 2010).

Das große Interesse an Geschäftsprozessmanagementstrategien hat den Bedarf an integrierten Ansätzen der Geschäftprozessmodellierung, die es allen beteiligten Interessensgruppen ermöglicht daran teilzunehmen und die Geschäftsprozesse des Unternehmens aktiv zu gestalten, stark erhöht. Dies war unter anderem die Ursache für die Entwicklung der Business Process Modeling Notation (BPMN) als potenzieller Industriestandard. Sie offeriert dabei nicht nur technische Vorteile, wie beispielsweise die Unterstützung der service-orientierten Anwendungsentwicklung, sondern soll sich auch durch eine einfache Verwendbarkeit für Fachanwender auszeichnen. Dieser Annahme folgend wird BPMN sogar von der Object Management Group (OMG) genutzt. Sie erklärte BPMN anstatt Aktivitätsdiagramme (UML AD) zum Kernstandard für die Erstellung eines Rahmenwerks für die Geschäftsmodellierung. Für Unternehmen ist der Wechsel auf eine neue Geschäftsprozessmodellierungssprache jedoch ein nicht unerheblicher Kostenfaktor. Gleichzeitig fehlen jedoch zuverlässige Untersuchungen, ob BPMN tatsächlich für Fachanwender besser und leichter verwendbar ist als UML AD. Dieser Beitrag präsentiert daher eine umfassende empirische Untersuchung, bei der die Verwendung der Sprachen durch Fachanwender während einer Modellerstellungsaufgabe untersucht wurde. Die Ergebnisse deuten darauf hin, dass UML AD mindestens genauso gut verwendbar ist wie BPMN, da BPMN weder bei Effektivität, Effizienz noch der Nutzerzufriedenheit signifikante Abweichungen zeigen konnte.

1 INTRODUCTION

Successfully implementing a Business Process Management (BPM) strategy considerably depends on establishing an integral approach to business process modelling, which allows diverse parties, such as managers, analysts, business users, and information system designers to participate and together optimize a company's business processes (Weske 2007). In such an approach, stakeholders require a process modelling language that can easily be used and understood by business and IT parties in order to communicate relevant process semantics. This demand, amongst others, led to the development of the Business Process Modeling Notation (BPMN). BPMN supporters claim that the language is not only well suited for system development purposes, but is also usable and understandable for "all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally to the business people who will manage and monitor those processes" (OMG 2006).

Following this argument, the Object Management Group's (OMG) Business Modeling & Integration Domain Task Force recently adopted BPMN as the core standard to develop a new business modeling framework around. This activity comprises, amongst others, the specification of a BPMN meta-model as well as the standardization of means to model business rules, organizational structures, business goals, etc. (OMG 2007). With its turn to BPMN, the OMG deliberately decided not to make use of the Unified Modeling Language (UML) and its process modeling notation, the UML Activity Diagram (UML AD). The UML AD was deemed as being too technically oriented (White 2004). However, as BPMN also has technical roots and the adoption of a newly proposed modeling language is a significant expense factor for companies, the promised advantages for business users have to be backed with solid arguments. While BPMN's technical features (e.g. the integration into the service-oriented computing technology) are unquestioned, the claimed advantages for business users remain to be proven, however. Why should BPMN be better usable for business users? Where did this opinion originate from? And is it justified?

To the best of our knowledge, the presumed superiority of BPMN over UML AD has not been substantiated with sound theoretical arguments or consolidated empirical findings. Instead, several authors who conducted analytical comparisons have highlighted considerable similarities between the languages (White 2004, Wohed & van der Aalst & Dumas & ter Hofstede & Russell 2006) or found BPMN to be more complex (Recker & zur Muehlen & Siau & Erickson & Indulska 2009). Therefore, it ought to be evaluated thoroughly whether BPMN is really more usable for business users and for which reasons this might be the case. In this paper, we examine and compare the usability of BPMN 1.1 and UML AD (UML 2.x) for

business users on the basis of an empirical study. Thereby, we test the conservative hypothesis that UML AD is at least as usable as BPMN during a model creation task. We try to falsify the proposition and to confirm that BPMN is indeed more usable than UML AD. The empirical comparison is based upon a set of process models that has been created in a laboratory experiment. Building upon such a confirmatory, quantitative-positivistic approach (Creswell 2008, Popper 1980), we will proceed as follows: after presenting related work in the next section, we introduce relevant theories and concepts for our study in order to derive and refine our proposition. We then present our study in which we compared BPMN and UML AD to test our hypothesis. Finally, we present results from the conducted study, discuss them, and introduce implications for practice and academia.

2 RELATED WORK

The evaluation and comparison of conceptual modeling languages in general and process modeling languages in particular has frequently been addressed in literature. To get a complete picture, recommendations have been made to combine analytical with empirical approaches (Gemino & Wand 2003). So far, BPMN and its claim to be more usable for business users have mainly been analyzed from an analytical perspective, though. Some authors used a semiotic quality framework with linguistic evaluation categories to analyze BPMN (Nysetvold & Krogstie 2005, Wahl & Sindre 2005). On that basis, Wahl and Sindre (2005) concluded that BPMN "is easily learned for simple use", although especially its advanced modeling concepts (e.g. the variety of event types) are likely to compromise the usability for business users. Nysetvold and Krogstie (2005) did not only analyze BPMN but also compared it to UML AD. They found BPMN to be superior with respect to learnability, precision, and its language patterns. However, BPMN and UML AD were judged to be equally suited "to improve communication between the IT-department and the business departments". Their findings are limited in significance though, since both languages were ranked against a very simplistic weighting scheme.

As part of their survey, Recker et al. (2009) analyzed BPMN against the Bunge-Wand-and-Weber ontology. Overall, they confirmed BPMN to be a mature language, which is well suited for modeling business processes. Identified weaknesses referred to ambiguous language elements and ontological shortcomings, which, however, were classified to be not of immediate practical relevance. White (2004) and Wohed et al. (2006) used the workflow patterns as introduced by van der Aalst et al. (2003) to examine the expressive power of BPMN. They have shown the expressive power of BPMN to be comparable to those of established modeling languages and furthermore agree that there is a notable similarity between BMPN and UML AD constructs. White considered BPMN constructs to be more usable for business users since "although the UML 2.0 development included a more

focused effort to upgrade the Activity Diagram in terms of its use for business people, it is still more technically oriented" (White 2004). However, he did not elaborate on why this might be the case.

Empirical evaluations of BPMN and especially on its usability for business users are still rare. While Recker and Dreiling (2007) have evaluated BPMN versus Event-Driven Process Chains (EPC, Dumas & Aalst & Hofstede 2005), they had a specific focus: to test teaching effects. They trained participants of the EPC group and tested their performance against untrained BPMN modelers, which makes it difficult to generalize their results. To the best of our knowledge, there is no empirical evaluation that focuses on confirming the claim that the usability of BPMN for business users is (a) higher than that of other process modeling notations in general and (b) higher than that of UML AD in particular.

3 THEORY AND PROPOSITIONS

A switch to a new process modeling language always results in significant investments into new tools, training of employees, translation of existing process models, etc. When taking into account that BPMN borrowed many concepts from existing languages as, e.g., UML AD, Event-Driven Process Chains, and Petri Nets (Weske 2007), it has to be questioned where the claimed better usability (Weske 2007, White 2004) comes from and if it really exists.

To identify the underlying reasons for a better usability of BPMN, the understanding of the term usability has to be clarified. In some cases usability is defined as a broad concept comparable to quality in use (Bevan 1995), while in other interpretations it is understood quite narrowly and distinguished from, for example, utility (Nielsen 1994). We adopted the definition of usability from the International Organization for Standardization (ISO), which explains the benefits in terms of user performance and satisfaction (ISO 1998). Furthermore, ISO defines usability as the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". Those aspects are further defined as follows (ISO 1998):

Effectiveness is the accuracy and completeness with which the users achieve the specified goals. Accuracy can be measured by the extent to which the quality of the output corresponds with the specified criteria, while completeness can be measured as the proportion of the target quantity, which has been achieved.

Efficiency is the level of effectiveness in relation to the expenditure of resources. These resources can include mental or physical effort, time, materials or financial cost.

Satisfaction is defined as the extent to which users are free from discomfort and their attitudes towards the use of the products. Amongst others, it can be assessed by asking the user to give a number corresponding to the strength of their feeling at any particular moment, or by asking users to rank products in order of preference.

Therefore, if a process modeling language is characterized as having a better usability, it has to be shown that the language is significantly better in at least one of those aspects. Efficiency, as relation between effectiveness and used resources, and satisfaction can be evaluated straightforwardly. In order to assess the effectiveness of a model creation task using a certain modeling language, the meaning of effectiveness and reasons for variations of effectiveness in the context of conceptual modeling have to be clarified and refined. As stated by the ISO, effectiveness in the sense of accuracy can be measured as quality of the outputs corresponding with specified criteria and therefore has to be interpreted as quality of the model in the context of conceptual modeling. Completeness, on the other side, does not seem to be directly applicable, as it is one criterion of model quality and the proportion of the target quantity cannot be directly determined for conceptual models.

According to Hadar and Soffer (Hadar & Soffer 2006, Soffer & Hadar 2003) and based on a model of Topi and Ramesh (2002), the quality of the model and variations of this quality have several determinants. The influencing factors and their interactions are shown in Figure 1 and briefly recapitulated in the following section as they can bias the results of empirical studies.

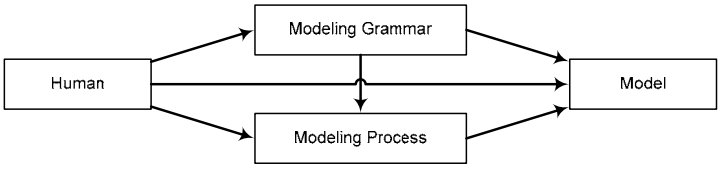

Figure 1. Factors that affect a conceptual model (Hadar & Soffer 2006).

The human factor that could influence the model results from the individual’s perception and interpretation of reality, professional experience and the perception of model quality. But as long as differences in the human factor can be minimized or statistically balanced, e.g. by sample size, the human factor of individuals should not have an impact on the results of a study comparing the effectiveness of two given modeling languages.

The modeling grammar itself can cause different variations due to differences in its set of constructs involved and its expressive power. Expressive power describes completeness (i.e., including all the constructs required for representing the domain) and clarity (i.e., without problems of construct redundancy, excess and overload) (Wand & Weber 1993). When comparing the modeling elements (set of constructs) of UML AD and BPMN, many similarities become obvious. There are, however, also differences, particularly regarding the modeling of data objects, events, or the data flow between process steps. For example, if data elements are included in a BPMN model, which is essential for most business processes, they have to be separated from the control flow (OMG 2006). Moreover, BPMN generally contains fewer graphical constructs than UML AD and instead uses variations of

them to support similar process patterns. E.g., BPMN uses similar elements to model events of different types or to depict parallel, exclusive, and inclusive gateways. Especially this reduced set of graphical constructs in combination with the clear separation of control and data flow in BPMN are often emphasized in literature and used as a rationale to claim its better usability for business users (Weske 2007, White 2004). Yet, it has to be questioned, if these similar constructs might have a negative impact on the expressive power of BPMN. As some constructs of BPMN can be regarded as overloaded, the clarity of the language might be reduced (Wand & Weber 1993). Therefore, it has to be validated if the claim of better usability, due to the reduced set of graphical constructs, outweighs the reduction of expressive power. For a comprehensive comparison of both languages, the interested reader is referred to Wohed et al. (2006) and White (2004).

The modeling process can be divided into two main phases: the perception of reality and the representation of the perceived reality in the model. The perception of reality can be influenced by human factors, as discussed above. The representation of the mental model of the application domain then depends on the mapping of reality into modeling constructs. Imprecise semantics of modeling constructs, like BPMN’s intermediate events or the variety of gateway semantics, and vague rules defining how to map real world phenomena into the modeling constructs are likely to have an impact on model quality. In addition, when taking the limited cognitive capacity of humans (Gemino & Wand 2005) and the problems of apparent complexity (Gemino & Wand 2003) into account, the separation of control and data flow in BPMN could have a negative influence on resulting models quality, as information objects can easily be forgotten by the modeller.

As there are reasonable doubts deduced from theoretical concepts, the claim of superior usability of BPMN (Nysetvold & Krogstie 2005, Weske 2007, White 2004) might be contested. In order to substantiate this claim, an intergrammar comparison (Gemino & Wand 2004) would have to result in a falsification of the proposition:

P: For business users, UML Activity Diagrams are at least as usable as BPMN models.

Based on the discussed differentiation of usability, the stated proposition P can be further refined into:

P1: Business users will be at least as effective in modeling with UML AD as with BPMN,
P2 Business users will be at least as efficient in modeling with UML AD as with BPMN,
P3: Business users will be at least as satisfied with modeling with UML AD as with BPMN.

If a test of the stated propositions leads to a falsification of any single one of them and hence BPMN proofs to be better than UML AD in at least one category, it would justify a shift to BPMN.

4 RESEARCH METHOD

The experiment conducted to examine the stated propositions followed the design used by Batra et al. (1990). In a related research topic they evaluated representations with different data modeling techniques in an empirical examination. The main factor examined in the experiment is the notation used for model creation by the participants. Besides the modeling language, we additionally introduced three different training levels to simulate an environment of different experiences as it is usually found in practice (Recker 2008). The analysis, however, concentrates on one source of variation, namely the modeling method, within a completely randomized design (Cobb 1998, Dean & Voss 1999).

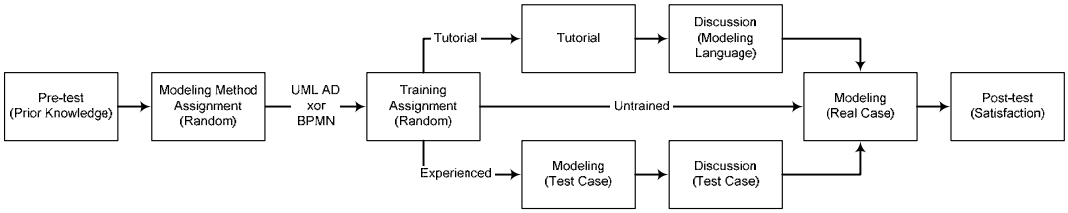

Figure 2. Design of experiment.

Figure 2 depicts the design of the experiment. It starts with a pre-test on prior domain and modeling knowledge, whose results were used to identify and exclude possible outliers afterwards. Two equally large groups were selected at random and allocated to different modeling techniques. Next, the participants in each group were randomly assigned to undergo different trainings. Depending on the assignments, short tutorials in BPMN or UML AD were provided for one third of the participants. Each of them took 45 minutes, included one small test case, was held by the same instructor and both were congruent with regard to their content and explanations. To simulate a second level of training, another third of the participants had to create a model of a complex test case without a detailed tutorial and were urged to discuss their individual experiences afterwards. Finally, the remaining participants did not receive any special training. The actual case was modeled by all participants at the same time. There was no time restriction; however, time was recorded for the following analysis. A survey on user satisfaction completed the experiment. Parts of the experiment were repeated with the same participants on a control case, which has marginal differences from the main case in its representational complexity (Bodart & Patel & Sim & Weber 2001).

Participants in the experiment were 30 graduate students in their final year with major or minor in business administration. They were randomly chosen out of over 80 volunteers for the study and split into two groups with 15 subjects each. Ex-ante interviews revealed that all of them had similar, only slightly differing backgrounds in conceptual modeling, mainly based on an undergraduate course that included modeling with Event-Driven Process Chains. Following Gemino and Wand (2005), we agree that the use of students is appropriate for

such a type of study and can in fact be beneficial, since "prior knowledge on the problem solving (domain understanding) […] might have confounded the results". Furthermore, the incentive scheme used in the study aligns with Batra et al. (1990).

Several materials, in the interest of brevity not depicted in the paper, were provided to the participants of the study. For the pre-test, a survey, aiming on identifying prior knowledge, had to be filled out. During the model creation task, participants were supplied with one sheet of detailed working instructions, a complete description of the process in natural language and several large empty pages to prepare the models. In addition, each participant received four pages of information on the application of their assigned modeling method, containing the available modeling primitives and common patterns, as well as one end-to-end example. These had been independently checked, compared and validated by several faculty members with experience in both modeling techniques.

Prior knowledge on the utilized business domain by some participants "might create substantial difficulties in an experimental study" (Gemino & Wand 2004). Thus, we decided to choose a process from a domain that was equally well-known to all participants. As shown in Table 1, the case contains all basic control flow patterns as introduced by van der Aalst et al. (2003), and one of their structural patterns. Its complexity is comparable to most of the reference business processes found in the German standard reference on business information systems for e-commerce (Becker & Schütte 2004).

Primitive		**Basic control flow pattern**		**Structural pattern**	
18	Flow element	3	Sequence (regular)	1	Arbitrary cycle
12	Data element	1	Sequence (conditional)		
		1	Parallel split		
		4	Synchronization		
		3	Exclusive choice		
		3	Simple merge		

Table 1. Case characteristics (after van der Aalst et al. (2003)).

To gather information on the user satisfaction with their respective modeling notation, a post-test survey was conducted. Figure 3 depicts the respective four questions of the post-test survey. On the first question, a high value indicates a highly satisfied user, whereas on the last three questions small values are favourable.

Q1 - Do you think you have understood the modeling language thoroughly?	(not at all) 1 … 2 … 3 … 4 … 5 … 6 (completely)
Q2 - Do you think the modeling language is challenging for you?	(not at all) 1 … 2 … 3 … 4 … 5 … 6 (highly)
Q3 - Do you think the concept of the modeling language is difficult?	(not at all) 1 … 2 … 3 … 4 … 5 … 6 (highly)
Q4 - Do you think the application of the modeling language is difficult?	(not at all) 1 … 2 … 3 … 4 … 5 … 6 (highly)

Figure 3. Post-test survey on user satisfaction.

During the statistical analysis we mainly applied the programming language and software environment R (Crawley 2007) for statistical computing and graphics. In addition, we also utilized SPSS (Norusis 2008) for various calculations and GGobi (Cook & Swayne & Buja 2007) for interactive graphics. Hypothesis testing was primarily performed via Student's t

tests. Where necessary, Kolmogorow-Smirnow tests and Bartlett's test helped to check for normality and equal variance conditions.

5 RESULTS

5.1 Data scoring

Three raters graded the prepared models from each subject for correctness by comparing it with the adequate solution (Batra et al. 1990). Sample solutions were prepared by four experienced modelers independently and afterwards discussed and matched against each other. Since every model is prepared from a subjective view and hence there is no one solution (Hadar & Soffer 2006), tolerated variations from the sample solutions were defined. Only differentiations exceeding the defined tolerance are marked as failures. The inter-rater reliability was close to a hundred percent and the few remaining differences were discussed until a common grading was found.

In order to assess the quality of the resulting models the grading of the prepared models had to be operationalized and concrete failure types needed to be defined. Gemino and Wand (2004) propose accuracy, correctness, detail, completeness, quality and discrepancies, rated by experts, as possible measures for affected (outcome) variables. Yet, they do not describe how the criteria can be measured. The "Guidelines of Modeling (GoM)", proposed by Schuette and Rotthowe (1998), on the other side suggest construction adequacy, language adequacy, economic efficiency, clarity, systematic design and comparability. Both propositions cannot be exactly matched with each other and not all of the proposed criteria are applicable to the experiment. Based on our experience in evaluating exams of courses on Event-Driven Process Chains as well as an analysis of errors found in these solutions, we developed a grading scheme for evaluating the conceptual models (as Batra et al. 1990). We decided to split our evaluation criteria into a language part, for all language-related errors, and an application part, for all errors due to a wrong application of the modeling language. This splitting follows the model of Hadar et al. (2006), who use modeling grammar and modeling process as factors influencing the model quality. In addition, most of the criteria presented by Gemino and Wand (2004) as well as Schuette and Rotthowe (1998) can be mapped onto our criteria. The composition of the criteria as used in the grading scheme is shown in Figure 4. The scheme has been evaluated by further faculty members, experienced in teaching process modeling (Batra et al. 1990). Any necessary changes were discussed and implemented.

As with all languages, violations of the notational correctness generally become manifest in syntax, semantic or pragmatic errors (Silverstein 1972) and were therefore attributed to the language part. For the language criteria syntax and semantic, we separated between single and repetitive errors. More than three consecutive, equal errors were counted as one

repetitive error. An incorrect modeling of a flow pattern that is due to a wrong, while allowed, combination of language elements (e.g. a wrongly modeled loop) formed a pragmatic error.

For the application part, we merged the proposed criteria of Gemino and Wand (2004), Schuette and Rotthowe (1998), as well as our own experience into adequacy, consistency, strictness, granularity and data handling as evaluation criteria for the successful application of the modeling language. A model can be defective with respect to its adequacy by comprising redundancies or a limited flexibility. The flexibility of a process is compromised, whenever independent activities are or have to be described as sequences. The consistency is compromised by building contradictions or differing denominations into the model and can be compared to naming conventions and clarity (Schuette & Rotthowe 1998), as well as discrepancies (Gemino & Wand 2004). Data handling was introduced to account for informational equivalency (Gemino & Wand 2004), as BPMN emphasizes the separation of data flows, while UML does not make such a distinction. It is biased by including superfluous data elements, leaving out relevant data elements or omitting data flows. The granularity is impacted by either leaving out or unnecessarily splitting up individual flow elements. It accounts for errors concerning minimalism and degree of abstraction in the sense of Schuette and Rotthowe (1998), as well as detail and completeness in the view of Gemino and Wand (2004). Strictness collects the errors, which are due to wrong mappings of reality onto modeling constructs, e.g. by introducing wrong constraints for workflows or omitting workflows, and could be compared to correctness (Gemino & Wand 2004).

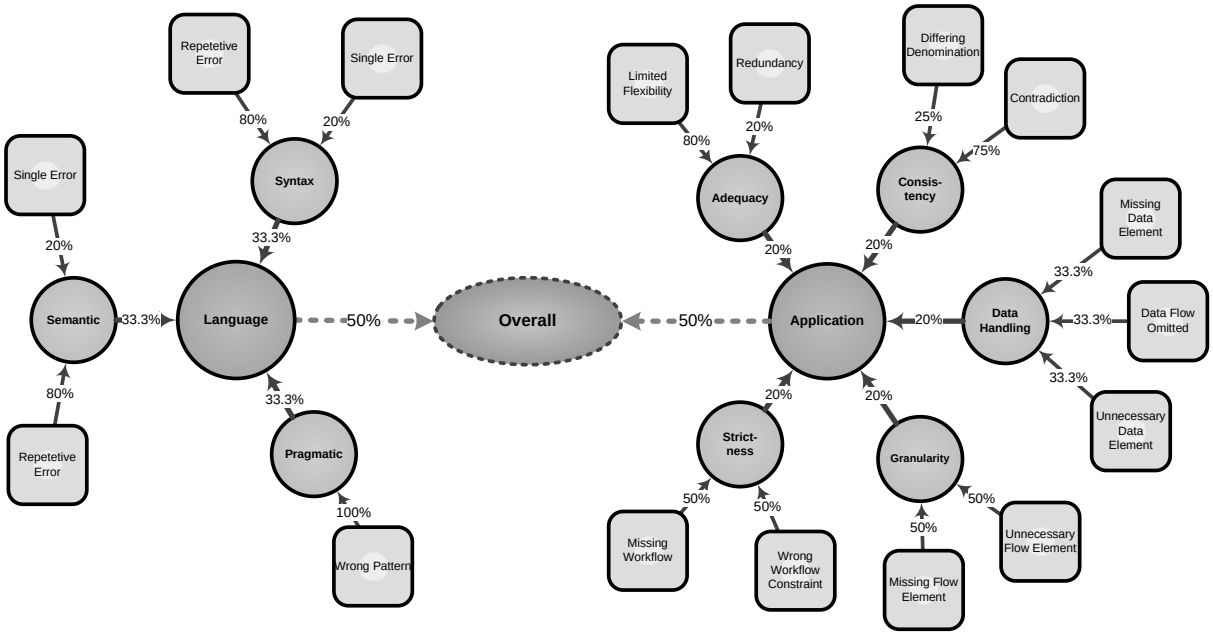

Figure 4. Criteria and aggregation.

To evaluate and compare usability, measures for effectiveness, efficiency and satisfaction were defined. As argued, the effectiveness is determined by the model quality, which can be measured in several predefined criteria. In each model, we identified and counted single

failures. However, to evaluate the model quality through the main criteria, aggregated scores had to be built. First, the counted occurrences of errors were scaled to each form a score between 0 (worst) and 100 (best). In doing so, a score of 100 in one of the single failure categories is equal to zero errors. A score of 0 is assigned for the highest amount of single errors in one category over all models. Such a score is assigned to each model in all of the single failure categories. The approach uses a linear transformation and, thus, no information is changed or lost. Since the single failure categories are less capable to provide an overview on the different modeling techniques, they are aggregated to form the defined evaluation criteria depicted in Figure 4. Most of the applied weights are balanced, except where equal weights would be unreasonable. In a second step, efficiency, as defined above, can be measured by the achieved score-points per time. Since the time needed to complete the model creation task was recorded (in minutes), the efficiency can be calculated for every criterion by dividing the effectiveness scores through time. The corresponding unit is “points per minute”. Furthermore, each question of the post-test survey was recorded on a scale from 1 to 6 and, hence, can be directly used to measure user satisfaction.

5.2 Data analysis

The three propositions, concerning effectiveness, efficiency and satisfaction, were tested on the collected data. The t test for two independent samples of equal size assumes Gaussian variables in both groups. Furthermore, it is applicable for equal and unequal variances in the groups, though the former situation is preferable. Therefore, Bartlett’s test can be performed to prejudge the equality of variances, but it strongly relies on the condition of both groups following a normal distribution. Hence, if the normality assumption is violated, equality of variances cannot be reliably evaluated (marked with n/a in the following analysis). In such cases, the more conservative version of the t test for unequal variances is applied. Although t tests are rather robust towards violations of its preconditions (Boneau 1960), test results have to be interpreted more carefully whenever an assumption is violated.

The effect of intervening variables was examined using an analysis of covariance (ANCOVA) technique (Kutner & Nachtsheim & Neter & Li 2005). Two possible covariate factors might have an influence on the analysis: prior domain knowledge and the assigned training. Domain knowledge information was recorded in the surveys. The analysis showed that domain knowledge was comparable between subjects and had no significant influence on the results. Therefore, it is safe to be left disregarded in the further analysis. The different experience levels, which were intentionally introduced and monitored through the assigned trainings, constitute the second possible covariate factor. The analysis revealed that the experience levels have neither an influence on the effectiveness nor on the satisfaction scores. On the other hand, in compliance with our expectations, the experience has a significant influence on the time needed for the model creation task. Consequently, the

training has an impact on the efficiency scores (points per minute) as well. Nevertheless, as our analysis showed, the influence of the factor modeling method is independent from a possible inclusion of the experience level. Thus, it is safe to be removed from the further analysis as well.

		Effectiveness									**Efficiency**								
Variable		ND [1]	EV [2]	t test [3]		Summary Statistics [6]					ND [1]	EV [2]	t test [3]		Summary Statistics [7]				
				t	p-value	Min	Max	Mean	Med [4]	SD [5]			t	p-value	Min	Max	Mean	Med [3]	SD [4]
Overall	AD	✓	✓	-1,331	0,903	95	99	97,3	97	1,29	✓	✓	-0,825	0,792	0,97	2,14	1,39	1,25	0,373
	BPMN	✓				94	98	96,7	97	1,45	✓				0,84	1,92	1,28	1,27	0,305
Language	AD	✓	✓	-0,541	0,703	94	100	98,1	99	2,09	✓	✓	-0,806	0,786	0,97	2,16	1,40	1,26	0,377
	BPMN	✓				92	100	97,6	99	2,61	✓				0,82	1,94	1,29	1,27	0,309
Application	AD	✓	✓	-1,241	0,888	94	99	96,8	97	1,52	✓	✓	-0,856	0,800	0,98	2,12	1,38	1,25	0,371
	BPMN	✓				93	99	96,1	96	1,71	✓				0,87	1,89	1,27	1,27	0,299
Syntax	AD	×	n/a	0,318	0,376	94	100	98,5	99	1,60	✓	✓	-0,712	0,759	0,99	2,16	1,40	1,26	0,373
	BPMN	✓				94	100	98,7	99	1,84	✓				0,83	1,96	1,31	1,30	0,318
Semantics	AD	✓	n/a	0,741	0,233	94	100	98,8	100	2,11	✓	✓	-0,704	0,756	0,97	2,16	1,40	1,28	0,373
	BPMN	×				97	100	99,3	100	1,22	✓				0,86	1,96	1,32	1,30	0,309
Pragmatics	AD	✓	✓	-0,970	0,830	86	100	96,8	100	5,03	✓	✓	-0,985	0,833	0,93	2,17	1,38	1,27	0,386
	BPMN	✓				83	100	94,9	97	5,85	✓				0,76	1,89	1,25	1,21	0,306
Adequacy	AD	✓	✓	-2,131	0,979	91	100	97,4	98	2,44	✓	✓	-0,973	0,830	0,99	2,07	1,38	1,26	0,364
	BPMN	✓				94	100	95,7	95	1,99	✓				0,88	1,85	1,27	1,27	0,285
Consistency	AD	✓	n/a	1,125	0,136	91	100	97,9	100	3,08	✓	✓	-0,654	0,741	0,97	2,17	1,39	1,27	0,378
	BPMN	×				94	100	99,0	100	2,00	✓				0,88	1,96	1,31	1,27	0,311
Strictness	AD	×	n/a	-1,987	0,971	97	100	99,1	100	1,39	✓	✓	-0,877	0,806	1,00	2,10	1,41	1,27	0,379
	BPMN	✓				93	100	97,8	98	2,04	✓				0,87	1,96	1,30	1,30	0,318
Granularity	AD	✓	✓	0,603	0,276	81	100	94,0	95	4,47	✓	✓	-0,658	0,742	0,90	2,10	1,34	1,19	0,378
	BPMN	✓				88	98	94,9	95	3,31	✓				0,81	1,86	1,26	1,23	0,300
Data Handling	AD	✓	✓	-2,033	0,974	90	99	96,1	98	3,44	✓	✓	-1,119	0,864	0,98	2,15	1,36	1,27	0,365
	BPMN	✓				82	99	93,1	93	4,56	✓				0,86	1,83	1,23	1,23	0,290

[1] Normally Distributed, [2] Equal Variances, [3] One-tailed, [4] Median, [5] Standard Deviation, [6] Unit: points (0-100), [7] Unit: points per minute

Table 2. Effectiveness and Efficiency – Tests and summary statistics.

Proposition 1. The left side of Table 2 contains an overview of descriptive statistics and testing results for the effectiveness measures. Considering the p-values for the one-tailed t test, it becomes obvious that the results are unambiguous, as for none of the criteria the difference is close to being significant. Thus, the stated proposition cannot be rejected and it can be concluded that business users are at least as effective in modeling with UML AD as with BPMN. A closer look at the results reveals that the UML AD group has higher means in language and application scores, as well as in four of the lower aggregated criteria. Wherever differences in medians are present, UML AD scores are higher as well.

While the language scores are rather equal, Figure 5 reveals that especially the application of UML AD seems to be more successful. An examination of the contrary hypothesis, of BPMN being at least as effective for business users as UML AD, leads to deeper insights. Such a hypothesis will be significantly rejected for the criteria adequacy, strictness and data handling.

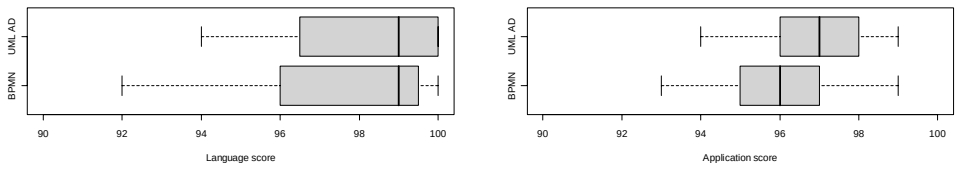

Figure 5. Effectiveness – Boxplots of language and application scores.

Proposition 2. As discussed, the basis of all efficiency indicators is the time needed to finish the model creation task. The amount of time needed is more spread out for UML AD modelers than for participants using BPMN. Means and medians of both groups are between 74 and 80 minutes. A t test showed no significant difference in model creation time between BPMN and UML AD (p-value: 0.492). The statistical results on efficiency shown in the right side of Table 2 are even more distinct than those for effectiveness. All reported p-values are larger than 74% and thus, they are far from rejecting the claimed proposition. For every single criterion, the means in the UML AD group are higher than those in the BPMN group. Overall, business users modeling with UML AD are at least equally efficient as with BPMN.

Proposition 3. Finally, the examination of the post-test survey results (detailed table not depicted due to reasons of brevity) gives information about the average satisfaction of business users with the different modeling notations. As discussed, a high value on question 1 and low values on the other three are favourable. For all questions the means and medians of BPMN and UML AD are relatively close to each other and thus no clear tendency can be seen. This observation is validated by the p-values of one-tailed t tests, which range between 0.39 and 0.79 and, hence, are far from indicating significant differences. However, since the reliability of the t test is reduced, due to the interval scaled data (levels 1 to 6), the insignificance of the results was additionally confirmed by ordinal logistic regression tests (Kutner et al. 2005). Consequently there is no reason to discard the stated proposition and it can be concluded that business users are at least as satisfied with UML AD as with BPMN.

As we failed to reject any of the refined propositions (P1, P2 and P3), the main proposition P (UML Activity Diagrams are at least as usable as BPMN models) cannot be falsified as well. An examination of the control process supported these results. The study we performed can easily be replicated and, hence, an independent confirmation of the results is possible. However, as for any empirical study there are some limitations as to what extent the results can be generalized. First of all, the number of samples could be larger. We plan to increase the sample size and validate our results in further experiment settings in order to increase the external validity of our findings. Additionally, more complex cases might be different in usability for business users, although in our opinion this would only strengthen our observations and might even support the findings in categories, where the UML AD is already significantly better than BPMN.

6 CONCLUSIONS AND IMPLICATIONS

In this paper we evaluated the usability of BPMN and UML AD based on a comprehensive empirical comparison. Starting with a comparative discussion of both languages, we deduced several doubts concerning the claimed superior usability of BPMN compared to UML AD for business users. In order to favour BPMN, the stated proposition would have to be falsified in

at least one of the deduced aspects of usability (being effectiveness, efficiency and satisfaction). After describing the study design, we provided several findings that are suited to judge the proposition. The discussed empirical results show that BPMN was not able to falsify any of the sub-propositions. In contrast, extended examinations revealed that UML AD was significantly more effective in the criteria data handling and adequacy.

With respect to the modeling of flexible processes, in which self-contained activities should preferably be allowed to run in parallel, UML AD turned out to be superior. The usage of BPMN instead promoted a rather sequential modeling style in which unrelated activities run one after the other. Such a modeling unnecessarily degrades the process flexibility. Another remarkable observation concerns the separation of control and data flow in BPMN, which apparently mislead participants to leave out parts of the data flow. Originally being introduced as a means to separate concerns and reduce the modeling complexity (Weske 2007), this concept turned out to be inferior to a combined flow modeling as present in UML AD. Admittedly, we could not confirm our theoretically deduced doubt that the decision of reducing the set of modeling constructs and instead introducing variants, would lead to a higher rate of mistakes, which stemmed from confusion due to a reduction of grammar clarity. This aspect remains to be examined more closely.

Our results have implications for both practice and academia. For practice, the results showed that for business users a higher usability of BPMN compared to UML AD cannot be empirically supported. Although, in literature BPMN is currently often claimed to be more useable (Nysetvold & Krogstie 2005, Weske 2007, White 2004) and even standardization organizations such as the OMG seem to have followed that conclusion, there are indications that BPMN still has shortcomings, which are likely to hinder its efficient adoption by business users in practice. And where business users are unable to use a modeling language adequately, the communication between the various stakeholders is compromised. Taking into account that our results also revealed that the use of BPMN implied a decrease of process flexibility, a misfit with the basic idea of optimizing business processes becomes obvious. Therefore, especially in environments where the technical advantages of BPMN are not needed, or not usable because of technical limitations, a possible switch to BPMN should be carefully deliberated.

Future research should concentrate on further examining the empirical indications identified in this paper. In order to definitely judge the usability of BPMN and UML AD for business users, a deeper understanding of their individual strengths and weaknesses has to be gained. Following the framework of Gemino and Wand (2004), additional empirical studies of model creation, as well as model interpretation tasks, should be conducted. The consolidated findings of such studies could provide a basis for merging BPMN and AD, which is eventually being planned in future to form a truly unified process modeling notation that combines the

strengths of both languages (White 2004). Therefore, it seems reasonable that efforts towards a version 2.0 of BPMN should consider already gained experiences and proven findings to support the primary goal of BPMN: "to provide a notation that is readily understandable by all business users" (OMG 2006).

References

Batra, D., J.A. Hoffler and R.P. Bostrom (1990). Comparing representations with relational and EER models. Communications of the ACM, 33 (2), 126-139.

Becker, J. and R. Schütte (2004). Handelsinformationssysteme. 2. Ed. Verlag Moderne Industrie, Frankfurt.

Bevan, N. (1995). Measuring usability as quality of use. Software Quality Journal, 4, 115-150.

Bodart, F., A. Patel, M. Sim and R. Weber (2001). Should Optional Properties Be Used in Conceptual Modelling: A Theory and Three Empirical Tests. Information Systems Research, 12 (4), 384-405.

Boneau, C.A. (1960). The Effects of Violations of Assumptions Underlying the t Test. Psychological Bulletin, 57 (1), 49-64.

Cobb, G.W. (1998). Introduction to Design and Analysis of Experiments. Springer, New York, NY.

Cook, D., D.F. Swayne and A. Buja (2007). Interactive and Dynamic Graphics for Data Analysis: With R and GGobi. Springer, New York, NY.

Crawley, M.J. (2007). The R Book. Wiley, Hoboken, NJ.

Creswell, J.W. (2008). Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. 3. Ed. Sage Publications, Thousand Oaks, CA.

Dean, A. and D. Voss (1999). Design and Analysis of Experiments. Springer, New York, NY.

Dumas, M., W.M.P.v.d. Aalst and A.H.M.t. Hofstede (2005). Process-Aware Information Systems. Bridging People and Software Through Process Technology. Wiley, Hoboken, NJ.

Gemino, A. and Y. Wand (2003). Evaluating Modeling Techniques based on Models of Learning. Communications of the ACM, 46 (10), 79-84.

Gemino, A. and Y. Wand (2004). A framework for empirical evaluation of conceptual modeling techniques. Requirements Engineering, 9 (4), 248-260.

Gemino, A. and Y. Wand (2005). Complexity and clarity in conceptual modeling: Comparison of mandatory and optional properties. Data & Knowledge Engineering, 55, 301–326.

Hadar, I. and P. Soffer (2006). Variations in Conceptual Modeling: Classification and Ontological Analysis. Journal of the Association for Information Systems, 7 (8), 569-593.

ISO (1998). Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11 : Guidance on usability. ISO Standard, ISO 9241-11:1998(E), International Standard Organization.

Kutner, M.H., C.J. Nachtsheim, J. Neter and W. Li (2005). Applied Linear Statistical Models. 5. Ed. McGraw-Hill/Irwin, Boston, MA.

Nielsen, J. (1994). Usability Engineering. Morgan Kaufmann, San Francisco.

Norusis, M.J. (2008). SPSS 16.0 Guide to Data Analysis. 2. Ed. Prentice Hall, Upper Saddle River.

Nysetvold, A.G. and J. Krogstie (2005). Assessing Business Processing Modeling Languages Using a Generic Quality Framework. In CAiSE'05 Workshops, pp. 545-556.

OMG (2006). Business Process Modeling Notation Specification. Final Adopted Specification, dtc/06-02-01, Object Management Group.

OMG (2007). Business Process Model and Notation (BPMN) 2.0. Request for Proposal, BMI/2007-06-05, Object Management Group.

Popper, K.R. (1980). The Logic of Scientific Discovery. Hutchinson, London.

Recker, J. (2008). BPMN Modeling - Who, Where, How and Why. BPTrends, 5 (3), 1-8.

Recker, J. and A. Dreiling (2007). Does It Matter Which Process Modelling Language We Teach or Use? An Experimental Study on Understanding Process Modelling Languages without Formal Education. In Proceedings of the ACIS 2007.

Recker, J., M. Indulska, M. Rosemann and P. Green (2009). Business Process Modeling - A Comparative Analysis. Journal of the Association for Information Systems, 10 (4), 333-363.

Recker, J., M. zur Muehlen, K. Siau, J. Erickson and M. Indulska (2009). Measuring Method Complexity: UML versus BPMN. In Proceedings of the AMCIS 2009.

Schuette, R. and T. Rotthowe (1998). The Guidelines of Modeling - an approach to enhance the quality in information models. In ER '98, pp. 240-254, LNCS 1507, Springer-Verlag.

Silverstein, M. (1972). Linguistic Theory: Syntax, Semantics, Pragmatics. Annual Review of Anthropology, 1, 349-382.

Soffer, P. and I. Hadar (2003). Reusability of Conceptual Models: The Problem of Model Variations. In EMMSAD, Proceedings of the Eighth CAiSE/IFIP8.

Topi, H. and V. Ramesh (2002). Human factors research on data modeling: a review of prior research, an extended framework and future research directions. Advanced Topics in Database Research, 3, 188-217.

van der Aalst, W.M.P., A.H.M. ter Hofstede, B. Kiepuszewski and A.P. Barros (2003). Workflow Patterns. Distributed and Parallel Databases, 14 (1), 5-51.

Wahl, T. and G. Sindre (2005). An Analytical Evaluation of BPMN Using a Semiotic Quality Framework. In CAiSE'05 Workshops, pp. 533-544.

Wand, Y. and R. Weber (1993). On the Ontological Expressiveness of Information Systems Analysis and Design Grammars. Journal of Information Systems, 3 (4), 217-237.

Weske, M. (2007). Business Process Management: Concepts, Languages, Architectures. Springer, Berlin, Heidelberg.

White, S.A. (2004). Process Modeling Notations and Workflow Patterns. In The Workflow Handbook 2004 (Fischer, L. Ed., pp. 265-294, Future Strategies Inc., Lighthouse Point, FL.

Wohed, P., W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede and N. Russell (2006). On the Suitability of BPMN for Business Process Modelling. In BPM 2006, pp. 161-176, LNCS 4102, Springer.

II.2 Beitrag: „Experience Report: Appropriateness of the BCI-Method for Identifying Business Components in large-scale Information Systems“

Autoren: Bernhard Selk, Sebastian Klöckner, Bettina Bazijanec, Antonia Albani

Bernhard Selk, Sebastian Klöckner, Bettina Bazijanec

Lehrstuhl WI-SE, Universität Augsburg,

Universitätsstraße 16, D-86135 Augsburg,

Email: bernhard.selk@wiwi.uni-augsburg.de

sebastian.kloeckner@wiwi.uni-augsburg.de

bettina.bazijanec@wiwi.uni-augsburg.de

Antonia Albani

Chair of Software Engineering

Delft University of Technology

PO Box 5031, 2600 GA Delft, The Netherlands

Email: a.albani@ewi.tudelft.nl

Erschienen in: Turowski, K.; Zaha, J. M. (Hrsg.): Component-Oriented Enterprise Applications. Proceedings of the Conference on Component-Oriented Enterprise Applications (COEA 2005). Erfurt, 20. September 2005. Lecture Notes in Informatics (LNI) P-70. Gesellschaft für Informatik, Bonn 2005, S.87-92 (2005).

Der Einsatz von Fachkomponenten in großen Unternehmensinformationssystemen bietet ein enormes Potenzial. Dennoch ist sowohl der Findungsprozess als auch die Bestimmung der richtigen Fachkomponenten eine nicht zu unterschätzende Herausforderung. Während die Bestimmung von Fachkomponenten in kleinen Geschäftsdomänen relativ einfach ist, wird dies in größeren Domänen außerst komplex. Dieser Beitrag illustriert die Erfahrungen, die während des Modellierungsprozesses einer integrierten Informationssystemachitektur im Bereich des Customer Relation Management (CRM) und des Supply Chain Management (SCM) mit mehr als 500 Funktionen und 1000 Informationsobjekten und unter Verwendung der Business Component Identification (BCI)-Methode gemacht wurden. Auf diesen Erfahrungen aufbauend werden mögliche Verbesserungen sowie notwendige Erweiterungen vorgeschlagen und erläutert.

1 Introduction

Innovations in information and communication technologies, primarily the emergence of the Internet, shifted managerial attention towards the use of information technologies to increase flexibility of the business system and to improve intercompany collaboration in value networks, often referred to as inter-organizational systems (IOS), e-collaboration and collaborative commerce [6, 9]. In order to enable such an inter-organizational information exchange an integration of all involved parts of the value chain is necessary. This paper shows how an integrative information system architecture (ISA) for Customer Relation Management (CRM) and Supply Chain Management (SCM) was developed and which experiences were made during the modeling process of a component-oriented ISA under usage of the Business Component Identification (BCI)-Method [1, 4].

Therefore, in section 2 the need for a component-oriented and integrated information system architecture for CRM and SCM is illustrated. Section 3 shows how the BCI-Method was used in order to develop an integrated ISA and which experiences were made during the application of this method. Concluding remarks and an outlook to future work can be found in section 4.

2 The need for a component-based integration of CRM and SCM

Actual research efforts show that information systems, which are interconnected along the value chain, offer great potential for risk and cost reduction [8]. This implies that every organization within the value chain has to connect its own IT-application systems with the ones of its suppliers and customers. The goal is to establish an inter-organizational system that allows automated data interchange and integrated information processing. In most cases IT-systems for Customer Relation Management (CRM) and Supply Chain Management (SCM) represent the communication endpoints of an enterprise with regard to inter-organizational information exchange. These can therefore act as integration points for the above mentioned interconnection task. However, integration of these systems is also necessary within the enterprise in order to make value networks work.

While inter-organizational aspects of integration are subject of current research and standardization efforts [5], the internal interconnection of CRM-, ERP- and SCM-systems is usually solved by adapters or standard interfaces. But this kind of integration causes data redundancy and actuality problems as well as a loss of functionality as the adapters and standard interfaces frequently do not provide access to all functions of the respective systems. In order to improve this situation in terms of inter-organizational interconnection, an integration of CRM- and SCM-systems into one integrative information system architecture (ISA) seems to be advisable. This ISA would allow an efficient connection of CRM- and SCM-related functionality through the deployment of an integrated database which would

also improve the consistency and up-to-dateness of data. However, an implementation of the ISA into a monolithic architecture would not be favorable with regard to flexibility and future developments. It would rather be reasonable to develop an integrated but also component-oriented architecture, so that the inter-organizational system can be reconfigured along the value chain over time, e.g. by replacement of single business components if newer versions or extended functionality is available. In this work, we therefore followed a component-oriented approach to derive an integrative information system architecture.

3 Evaluation of the BCI-Method

After a detailed analysis of contemplable methods for the development of the integrative information system architecture based on business components, the decision came to the Business Component Identification (BCI)-Method [1, 4], because it considers function-information-relationships to identify suitable components and therefore uses a bottom-up approach to identify suitable components. This is in line with the goal to avoid a monolithic architecture while trying to integrate functionality belonging together.

At first, the development of an integrative information system architecture for CRM and SCM required a detailed analysis of both application domains. Therefore, a broad literature research as well as an analysis of a variety of CRM- and SCMproducts (e.g. SAP, Siebel, I2) has been conducted. This analysis primarily focused on the identification and decomposition of relevant functions and information objects as well as the determination of relationships between them. This was achieved by functional decomposition diagrams [7], information object decomposition diagrams, and input-output-tables. The latter were used to illustrate relations between functions and information objects. For checking the consistency of the above mentioned models, further models and diagrams have been used, e.g. semantic data models [10] and information flow diagrams [11]. Figure 1 shows an excerpt of the information flow diagram for the process create customer order.

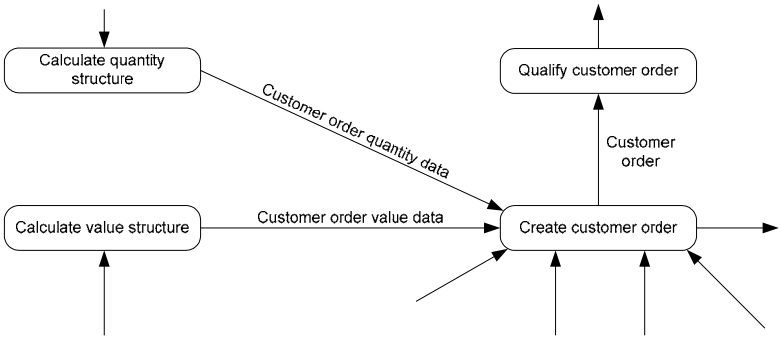

Figure 1: Information flow diagram

For creating the customer order, value and quantity data amongst others is necessary. This data is created by functions calculate quantity structure and calculate value structure and

sent to the function create customer order. This diagram illustrates the coherence between different functions by information objects. Table 1 shows the way how relations between information objects and functions have been illustrated. An Input-Information Object is necessary to execute a function whereas an Output-Information Object is the result of an executed function.

Input-Information Object	Function	Output- Information Object
…	Calculate quantity structure	Customer order quantity structure
…	Calculate value structure	Customer order value structure
Customer order quantity structure	Create customer order	Customer order
Customer order value structure	Create customer order	…
…	Create customer order	

Table 1: Input-output-table

Using above mentioned diagrams, models and tables permitted a broad description of CRM and SCM with more than 500 functions and more than 1000 information objects. This collection of models was then used as input for the BCI-Method, which is explained in the following.

The BCI-Method takes business tasks of a specific domain as input, as e.g. defined in the functional-decomposition diagram, and the domain based data model, both obtained from the domain analysis. In a first step a matrix is built defining the relationships between the single business tasks and the information objects. The relationships are visualized inserting “C” and “U” in the matrix. “C” denotes that the data is created by the specific business task, and “U” indicates the usage of informational data by a given task. In our case especially the input-output-tables have been helpful to build this matrix. A definition of relationships between functions and information objects after having put them into the rows and columns of the matrix would not have been possible due to the size of this model. In changing the order of data and of business tasks according to some metrics defined, e.g. minimal communication between and maximal compactness of components, groups of relationships can be recognized [4]. These groups identify potential business components. If some “U”s are outside the groups, arrows are used to identify the data flow from one group to the other. The result of the BCI is an abstract business component model with defined dependencies between the components [1, 2, 3]. Figure 2 shows an excerpt of the matrix with two business components.

As the matrix is quite large a test of all possible combinations in order to find the most suitable component model is impossible, even if tool support is available. Though, a repeated execution of the BCI-Method with random initial configurations showed varying results. After a close examination of the initial configuration’s impact on the outcome of the algorithm, we noticed that specific knowledge about component characteristics has to be included into the initial configuration. Such knowledge includes for example information about functional

relationships or dependencies between single information objects. For the development of the integrative information architecture information about functional relationships, taken from the functional decomposition diagrams, was inserted into the initial configuration. However, the resulting abstract component model is not a copy of the functional diagrams. It contains 48 business components. These components were identified under the objective of minimizing the communication between the components.

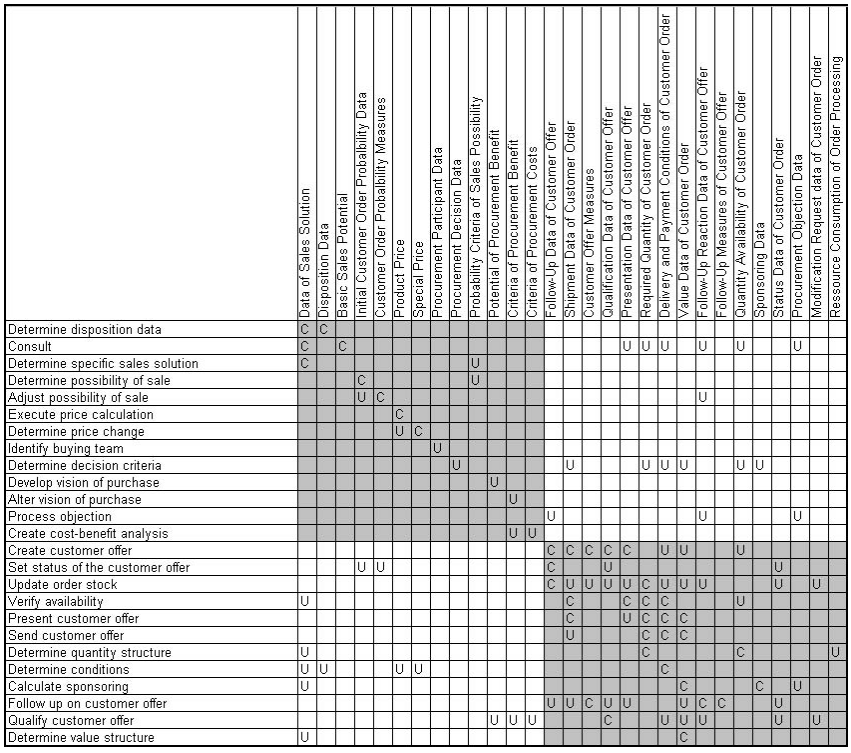

Figure 2: Excerpt of the matrix

There is still a potential to extend and further optimize the BCI-Method as well as the obtained component model. Besides the inserting of knowledge about functional dependencies, which has already led to promising results, we suggest following extensions:

- At this time, the implemented functional coherence follows the functional decomposition diagrams. It should be examined if this restricts the range of possible solutions.
- Additional consideration of relationships between information objects.
- By using information flow diagram the relations between functions and information flows are illustrated. These information flows are similar to process steps, but they do not include real business processes and relationships of all functions. Thus, processes have to be considered within the BCI-Method.
- The communication intensity of functions is not considered in the BCI-Method, although it is relevant for mapping a function to a business component.

4 Conclusion

In this experience report, we addressed the problem of identifying business components in large-scaled information systems by the BCI-Method. Although the BCI-Method is specialized for the design of component-orientated information systems, it has to be modified in order to be applicable for voluminous input models. In these cases tool support, which relies on an explicit formulation of algorithms and their constraints, is necessary. After having implemented the first enhancement to our BCI-Toolset, future work will consider the suggestions made in chapter 3. With these extensions, an application of the BCI-Method within large-scaled information systems is expected to lead to better model quality. Hence, test series using the extensive input models are planned to allow a comparison of different BCI-Method modifications.

References

[1] Albani, A., et al. Domain Based Identification and Modelling of Business Component Applications. In 7th East-European Conference on Advances in Databases and Informations Systems (ADBIS-03), LNCS 2798. 2003. Dresden, Deutschland: Springer Verlag.

[2] Albani, A., et al. Component Framework for Strategic Supply Network Development. In 8th East-European Conference on Advances in Databases and Information Systems (ADBIS-04), LNCS 3255. 2004. Budapest, Hungary: Springer Verlag.

[3] Albani, A., et al. Komponentenmodell für die Strategische Lieferkettenentwicklung. In 6. Internationale Tagung Wirtschaftsinformatik (WI-03) Medien - Märkte - Mobilität. 2003. Dresden, Deutschland: Physica-Verlag.

[4] Albani, A., J.L.G. Dietz, and J.M. Zaha. Identifying Business Components on the basis of an Enterprise Ontology. In Interop-Esa 2005 - First International Conference on Interoperability of Enterprise Software and Applications. 2005. Geneva, Switzerland.

[5] Bussler, C. (2003), B2B Integration, Springer, Berlin.

[6] Kopanaki, E., et al. The Impact of Inter-organizational Information Systems on the Flexibility of Organizations. In Proceedings of the Sixth Americas Conference on Information Systems (AMCIS). 2000. Long Beach, CA.

[7] Mertens, P.; Bodendorf, F.; König, W.; Picot, A.; Schumann, M.: Grundzüge der Wirtschaftsinformatik. 5.Aufl., Springer, Berlin et al., 1998.

[8] Mertens, P.: Integrierte Informationsverarbeitung 1. Operative Systeme in der Industrie. 14. Aufl., Gabler Verlag, Wiesbaden 2004.

[9] Picot, A., R. Reichwald, and R. Wiegand, Die grenzenlose Unternehmung - Information, Organisation und Management. 4. Auflage ed. 2001, Wiesbaden. 2-6.

[10] Scheer, A.-W.: Architektur integrierter Informationssysteme, Grundlagen der Unternehmensmodellierung. Springer, Berlin et al., 1991.

[11] Thaler, K.: Supply Chain Management – Prozessoptimierung in der logistischen Kette. 3. Auflage. Fortis, Köln/Wien, 2001.

II.3 Beitrag: „Zur systematischen Identifikation von Services: Kriterien, aktuelle Ansätze und Klassifikation“

Autoren: Dominik Birkmeier, Sebastian Klöckner, Sven Overhage
Alle Lehrstuhl WI-SE, Universität Augsburg,
Universitätsstraße 16, D-86135 Augsburg,
Email: dominik.birkmeier@wiwi.uni-augsburg.de
sebastian.kloeckner@wiwi.uni-augsburg.de
sven.overhage@wiwi.uni-augsburg.de

Erschienen in: Loos, P.; Nüttgens, M.; Turowski, K.; Werth, D. (Hrsg.): Modellierung betrieblicher Informationssysteme (MobIS 2008). Modellierung zwischen SOA und Compliance Management. Lecture Notes in Informatics (LNI) P-141. Gesellschaft für Informatik, Bonn 2008, S.255-272 (2008).

Die Einführung serviceorientierter Architekturen verspricht mit ihrem modularen Ansatz eine Vielzahl von Vorteilen für die betriebliche Anwendungsentwicklung. Eine erfolgreiche Einführung dieses Architekturkonzepts hängt jedoch entscheidend von einer ausreichenden methodischen Unterstützung des serviceorientierten Paradigmas ab. Die Vielzahl veröffentlichter und einschlägiger Ansätze verdeutlicht, dass derzeit insbesondere die Entwicklung systematischer Methoden für diese Identifikation von Services im Mittelpunkt des wissenschaftlichen Interesses steht. Die in der Literatur vorhandenen Ansätze weisen hinsichtlich ihrer Konzeption und Vorgehensweise allerdings eine starke Heterogenität auf. In diesem Beitrag werden daher grundlegende Kriterien zur Einordnung der verschiedenen Ansätze entwickelt und vorhandene Ansätze anhand eines detaillierten Klassifikationsschemas einander gegenübergestellt. Neben dem aktuellen Stand der Technik und den Unterschieden der einzelnen Ansätze werden dabei vor allem noch vorhandene Schwächen identifiziert, aus denen sich der weitere Forschungsbedarf ableiten lässt.

1 Motivation

An die betriebliche Anwendungsentwicklung wird heute eine Vielzahl von Ansprüchen gestellt: So zählen zur Zeit vor allem die Beherrschung der hohen Anwendungskomplexität, die Möglichkeit zur flexiblen Anpassung von Informationssystemen an Änderungen im Geschäftsumfeld, sowie die schnelle Realisierung neuer Funktionalität auf Basis der bestehenden Anwendungssysteme zu den zentralen Herausforderungen, die es mit geeigneten Konzepten zu bewältigen gilt [Bro00, CD01, PTD+06]. Als Architekturansatz, der mit seinem modularen Paradigma einen wichtigen Beitrag zur Lösung der genannten Herausforderungen verspricht, wird derzeit vor allem die Einführung serviceorientierter Architekturen diskutiert [KL04, PTD+06].

Die erfolgreiche Anwendung des serviceorientierten Paradigmas in der Praxis hängt allerdings entscheidend davon ab, dass der Entwicklungsprozess ausreichend methodisch unterstützt wird [MDW02, PTD+06]. Neben der Frage, wie Services zu beschreiben, in Katalogen aufzufinden und (nach den Vorgaben eines Geschäftsprozesses) zu kombinieren sind, steht dabei vor allem die Entwicklung von Methoden und Praktiken, mit denen sich informationstechnisch umzusetzende Services systematisch identifizieren lassen, im Mittelpunkt des wissenschaftlichen Interesses [Ars04, EAK06].

Die Identifikation geeigneter Services steht am Beginn des serviceorientierten Entwicklungsprozesses und stellt die Grundlage für alle weiteren Entwicklungsschritte sowie die anschließende Phase der Komposition bzw. Nutzung dar [Erl05]. Ihr kommt deshalb eine zentrale Bedeutung für den gesamten Prozess zu, die sich auch in einer Vielzahl bereits publizierter Ansätze hinsichtlich der dabei einzuschlagenden Vorgehensweise widerspiegelt. Die in der Literatur zu findenden Ansätze weisen bislang allerdings eine starke Heterogenität auf. So reichen sie bspw. von allgemeinen Hinweisen, die bei der Identifikation von Services zu berücksichtigen sind, über umgangssprachlich dargestellte bewährte Praktiken bis hin zu algorithmisch spezifizierten Vorgehensweisen. Ferner liegen den verschiedenen Ansätzen abweichende Servicedefinitionen und Vorgehensweisen zugrunde, wodurch sie zu deutlich abweichenden Ergebnissen gelangen können. Ein bestimmter Ansatz hat sich dabei schon aufgrund der relativ kurzen Zeit, seit der die Identifikation von Services untersucht wird, noch nicht durchsetzen können. Zudem fehlen vergleichende Betrachtungen, die die entstandenen Ansätze gegenüberstellen und das Forschungsgebiet auf diese Weise strukturieren. Durch eine Kategorisierung lassen sich zum einen komplementäre, einander ergänzende Ansätze identifizieren oder bislang noch unbehandelte Forschungsfragen ableiten, aus denen sich weiterer Forschungsbedarf ergibt. Zum anderen lässt eine solche Kategorisierung Rückschlüsse auf die Anwendbarkeit einzelner Ansätze (z.B. in verschiedenen Entwicklungskontexten) zu.

In diesem Beitrag werden vorhandene Ansätze, die zu einer systematischen Identifikation von Services beitragen sollen, gegenübergestellt und anhand eines differenzierten Schemas eingeordnet. Dazu werden in Kapitel 2 zunächst grundlegende Kriterien benannt, die für Ansätze zur Identifikation von Services charakteristisch sind. Als Forschungsmethode liegt diesem Teil des Beitrags eine Literaturauswertung zugrunde. Die dabei zusammengestellten Kriterien bilden den Klassifikationsrahmen für die Gegenüberstellung und Bewertung der einzelnen Ansätze. In Kapitel 3 werden dann verschiedene Ansätze aus der Literatur vorgestellt und in das Klassifikationsschema eingeordnet. Dabei werden die Unterschiede zwischen den einzelnen Ansätzen verdeutlicht und vorhandene Schwächen argumentativ-deduktiv abgeleitet. Nachdem in Kapitel 4 kurz auf verwandte Arbeiten eingegangen wird, schließt Kapitel 5 mit einem Ausblick auf den noch verbleibenden Forschungsbedarf, der sich aus der Diskussion der hier vorgestellten Ansätze ergibt.

2 Kriterien für die Identifikation von Services

Die in der Literatur vorgeschlagenen Ansätze zur Identifikation von Services unterscheiden sich zum Teil deutlich hinsichtlich ihrer jeweiligen Konzeption und Vorgehensweise voneinander. Zum Vergleich und zur Einordnung der verschiedenen Ansätze lässt sich eine Reihe grundlegender Kriterien benennen, die einen detaillierten Aufschluss über dessen jeweilige Gestaltung geben. Die grundlegenden Kriterien wurden in Anlehnung an die Methodenforschung der Ingenieursdisziplinen bzw. der (Wirtschafts-) Informatik aufgestellt [PBFG03, BS04], die sich mit der Konzeption von systematischen Vorgehensweisen für die Entwicklung bzw. Konstruktion beschäftigt. Sie beschreiben

- die konzeptionellen Grundlagen, von denen der jeweilige Ansatz Gebrauch macht,
- das allgemeine Vorgehen, das mit dem Ansatz vorgeschlagen wird,
- die Modellbildung, die zur Identifikation von Services vorgenommen wird sowie
- Maßnahmen, die die Anwendung des Ansatzes unterstützen.

Die im Folgenden näher beschriebenen Kriterien sollen dabei vor allem Auskunft darüber geben, inwiefern ein Ansatz tatsächlich geeignet ist, zur gewünschten systematischen Identifikation von Services beizutragen und wo ggf. Schwächen bestehen.

2.1 Konzeptionelle Grundlagen

Die konzeptionellen Grundlagen beschreiben das begriffliche Verständnis, das dem jeweiligen Ansatz zur Serviceidentifikation zugrunde liegt. Dabei ist zunächst zu betrachten, welche Servicedefinition zur Identifikation verwendet wird. Ferner ist zu unterscheiden, wie exakt die jeweilige Vorgehensweise beschrieben wird und ob sie ggf. in ein übergeordnetes Vorgehensmodell mit Bezug zu den angrenzenden Entwicklungsphasen eingebettet ist. Im Einzelnen ergeben sich somit folgende Kriterien:

Servicedefinition. Betrachtet man die in der Literatur vorhandenen Servicedefinitionen, zeigt sich, dass die Autoren den Begriff Service sehr unterschiedlich fassen. Die verschiedenen Fassungen reichen dabei von technisch geprägten Auslegungen des Begriffs Service im Sinne von Web-Services als Software-Komponenten [SGM02, Nat03, Ars04, MSJL06] bis hin zu rein betriebswirtschaftlich geprägten Definitionen im Sinne von Dienstleistungen [BS06]. Technisch orientierte Definitionen legen den Fokus zumeist auf die technische Spezifikation und Implementierung von Services als softwaretechnisch umgesetzte Artefakte, die eine festgelegte Funktionalität anbieten. Im Mittelpunkt dieser Definitionen steht dabei die lose Kopplung von wiederverwendbaren und plattformunabhängigen, durch wohldefinierte Schnittstellen beschriebenen Services.

Stärker fachlich ausgerichtete Autoren beschreiben Services als eine zusammengehörende Menge von vermarktbaren Diensten, welche über in einer standardisierten Sprache verfasste Schnittstellenbeschreibungen verfügen [OT07]. Ein Dienst wird hier als eine Aktion eines (betrieblichen) Anwendungssystems [verstanden], welche die Bewältigung einer bestimmten Menge von (betrieblichen) Aufgaben unterstützt [Tur03]. Daneben hat mittlerweile auch eine betriebswirtschaftliche Interpretation von Services als Dienstleistung in die Literatur Eingang gefunden [BS06]. Diese neue Definition der Service Science, die deutlich über den Fokus der hier diskutierten Serviceorientierten Architekturen hinausgeht, wird allerdings nicht weiter betrachtet. Das divergierende Verständnis des Begriffs Service spiegelt sich, schon wegen der unterschiedlichen Ausrichtung und den sich daraus ergebenden unterschiedlichen Zielsetzungen, auch in den darauf aufbauenden Methoden zur Identifikation von Services wider.

Formalisierungsgrad. Auch der Formalisierungsgrad der verschiedenen Ansätze zur Serviceidentifikation variiert bisweilen deutlich. Er reicht prinzipiell von sog. ad-hoc Strategien, die Services allerdings nur mit einer unscharf beschriebenen opportunistischen Vorgehensweise identifizieren, über kodifizierte allgemeine Ratschläge bis hin zu strukturierten Vorgehensweisen und algorithmisch beschriebenen Methoden. Während die den ad-hoc Strategien zuzurechnenden Ansätze allenfalls sehr grob dargestellte Verfahren zur Serviceidentifikation skizzieren, geben allgemeine Ratschläge zumindest generalisierte Hinweise darauf, was bei der Identifikation von Services zu berücksichtigen ist. Strukturierte Vorgehensweisen beinhalten hingegen sowohl detailliert vorgegebene und beschriebene Arbeitsschritte, die für eine systematische Identifikation von Services zu durchlaufen sind, sowie klar spezifizierte Identifikations- und Selektionskriterien. Algorithmisch beschriebene Methoden verketten diese Arbeitsschritte zu einem klar vorgegebenen Identifikationsprozess, der ggf. von Werkzeugen unterstützt oder sogar automatisiert werden kann.

Übergeordnetes Vorgehensmodell. Ansätze für die Identifikation von Services werden üblicherweise im Rahmen einer umfassenden Entwicklungsmethodik eingesetzt, die den

Entwicklungsprozess insgesamt steuert und begleitet. Die Integration der einzelnen Arbeitsschritte erfolgt dabei üblicherweise durch ein übergeordnetes Vorgehensmodell, das den Entwicklungsprozess strukturiert und die Weiterverwendung der in den einzelnen Phasen erreichten Teilergebnisse in einer abgestimmten Weise regelt [CD01, DW99, Sch98]. Verzichten einzelne Ansätze auf die Integration in ein übergeordnetes Vorgehensmodell und die Abstimmung mit den Erfordernissen späterer Entwicklungsphasen, sind deren Ergebnisse dort unter Umständen nur eingeschränkt weiterverwendbar.

2.2 Allgemeines Vorgehen

Das allgemeine Vorgehen beschreibt die grundsätzliche Strategie, die von einem Ansatz zur Identifikation von Services verfolgt wird. Dabei sind folgende Aspekte zu betrachten:

Ableitungsrichtung. Die Ableitungsrichtung beschreibt die Richtung der Analyse bei der Identifikation. Sog. Top-Down-Ansätze [Mil71] zur Serviceidentifikation betrachten dabei zunächst die Anwendungsdomäne und leiten Services aus den erstellten fachkonzeptionellen Modellen ab. In einem zweiten Schritt werden diese informationstechnisch spezifiziert und dann ggf. auf die vorhandene Anwendungslandschaft abgebildet. Bottom-Up-Ansätze gehen dagegen von der existierenden Anwendungslandschaft aus und modularisieren diese zunächst nach technischen Gesichtspunkten. In einem zweiten Schritt werden die identifizierten Module dann mit einer fachkonzeptionellen Bedeutung versehen und als Services bereitgestellt.

Da sowohl Top-Down- als auch Bottom-Up-Ansätze in einer jeweils idealtypischen Form die Gefahr von Fehlentwicklungen bergen (bspw. die redundante Realisierung von Funktionalität als Bestandteil verschiedener Services oder die Realisierung von aus fachlicher Sicht nicht eigenständigen Services), kombinieren einige Ansätze beide Analyserichtungen. Diese werden im Folgenden als Meet-In-The-Middle-Ansätze bezeichnet.

Optimierendes Verfahren. Die Identifikation von Services kann zugleich als mathematisches Optimierungsproblem aufgefasst werden. Wird bspw. nach dem etablierten Ansatz verfahren, zusammengehörige Funktionalität möglichst zu bündeln und so die Kommunikation zwischen Services zu minimieren [Par72], entsteht eine Aufgabenstellung, die etwa durch Clustering-Verfahren gelöst werden kann. Von den nicht-optimierenden sind deshalb optimierende Vorgehensweisen abzugrenzen, die ihrerseits wiederum in exakte und heuristische Verfahren unterteilt werden können.

2.3 Modellbildung

Zur Identifikation von Services werden jeweils konzeptionelle Modelle als Abbild der Realität herangezogen. Die verschiedenen Ansätze unterscheiden sich vor allem im Hinblick auf die Komplexität und den Inhalt der herangezogenen Modelle. Theoretische Grundlage eines methodischen Vorgehens bei der Serviceidentifikation sollten jedoch zumindest implizit die

Konzepte der Systemtheorie, die auch auf (informations-) technische Systeme anzuwenden sind [Rop99], sowie die der Konstruktionslehre [PBFG03] sein. Dementsprechend sind folgende Kriterien zu betrachten:

Einbindung verschiedener Modellsichten. Die Systemtheorie schlägt zur Konzeption und Beschreibung (informations-) technischer Systeme drei Modellsichten vor, die auch in Modellierungsansätzen der (Wirtschafts-) Informatik verbreitet sind [Rop99, Sch98, DW99, Dav93]. Die statische Sicht (auch Daten-, Typ- oder Informationssicht) beschreibt die Beschaffenheit wesentlicher Systemattribute und deren Struktur. Die operationale Sicht (auch Funktionssicht) beschreibt das beobachtbare Systemverhalten und verknüpft dabei Systemattribute zu Ein- und Ausgaben. Darüber hinaus wird eine funktionale Dekomposition durchgeführt, die den Zusammenhang zwischen komplexen Funktionen und ihren Teilfunktionen beschreibt. Die dynamische Sicht (auch Prozesssicht) beschreibt schließlich die zeitliche Abfolge von Systemfunktionen.

Für die Identifikation von Services sind grundsätzlich alle Modellsichten heranzuziehen, schon da sie erst in der Zusammenschau eine umfassende Systemsicht ergeben [Rop99]. Von den verschiedenen Ansätzen werden jedoch verschiedene Teilmengen der hier genannten Modellsichten genutzt, woraus sich spezifische Vor- und Nachteile ergeben.

Berücksichtigung existierender Systemstrukturen. Die Identifikation von Services erfolgt häufig in einer bereits existierenden Systemumgebung, die Einfluss auf die einzuschlagende Vorgehensweise haben kann. So sind ggf. bereits existierende Services oder zu modularisierende Legacy-Anwendungen einzubinden [DW99]. In der Literatur wird dieser Umstand bislang nicht einheitlich behandelt.

Berücksichtigung von Abhängigkeiten zum Umsystem. Services ergeben in der Regel erst im Zusammenspiel mit anderen Services die benötigte Funktionalität. Sie werden also entwickelt, um mit anderen Services verbunden zu werden [SGM02]. Häufig wird deshalb zugleich eine Identifikation mehrerer, aufeinander abgestimmter Services in einem umfassenderen Ansatz durchgeführt, der bspw. danach strebt, zusammengehörige Teilfunktionen möglichst einem Service zuzuordnen und so bspw. Abhängigkeiten zwischen verschiedenen Services zu minimieren. Weiterhin bestehende Abhängigkeiten eines Services zum Umsystem lassen sich mit einem solchen Ansatz zudem exakt spezifizieren. Andere Ansätze betrachten dagegen jeweils nur einen Service und lassen bestehende Abhängigkeiten ggf. vollständig außer Acht.

Unterscheidung von Servicehierarchien. Bei der Identifikation lassen sich prinzipiell zusammengesetzte Services, die aus der Komposition anderer, ggf. bereits bestehender Services entstehen, und elementare Services, die nicht weiter in Services unterteilt werden, unterscheiden. Ansätze, die eine solche Unterscheidung ermöglichen, berücksichtigen das sog. hierarchische Systemkonzept der Systemtheorie, das eine schrittweise Verfeinerung

identifizierter Services bis hin zu elementaren Einheiten erlaubt [Rop99]. Andere Ansätze gehen dagegen davon aus, dass eine solche schrittweise Verfeinerung (bspw. schon wegen der grundsätzlich zu verbergenden Innensicht von Services) nicht notwendig ist und verzichten auf eine entsprechende Differenzierung. Zur weiteren Verfeinerung von Services sind solche Ansätze ggf. wiederholt anzuwenden.

Unterscheidung von Servicetypen. Schließlich können bei Serviceorientierten Architekturen grundsätzlich Teile unterschieden werden, deren primäre Aufgabe entweder die Verwaltung von Informationen (Entity Service) oder die Bearbeitung von Aufgaben (Task Service) ist [CD01, HS00]. Einige Ansätze berücksichtigen dies schon bei der Identifikation von Services und erreichen auf diese Weise eine Trennung zwischen daten- und funktionsbezogenen Diensten. Es ist jedoch umstritten, ob eine solche Trennung zu einem optimalen Ergebnis führt. So verlangt Parnas bspw. eine Gruppierung von Daten und darauf arbeitenden Funktionen in einem Modul [Par72]. In verschiedenen Ansätzen wird die Unterscheidung von Entity und Task Services deshalb nicht explizit getroffen.

2.4 Anwendung

Die Maßnahmen zur Unterstützung der Anwendung eines Ansatzes geben Aufschluss darüber, wie effizient sich dieser in der Praxis einsetzen lässt. Neben einer nachvollziehbar beschriebenen Vorgehensweise und einer adäquaten Modellbildung sind dafür vor allem folgende Kriterien ausschlaggebend:

Werkzeugunterstützung. Die Anwendbarkeit der verschiedenen Ansätze in der Praxis wird durch die Verfügbarkeit von entsprechenden Werkzeugen erleichtert, die die vorhandene Komplexität für den Entwickler besser handhabbar machen und ihn durch die notwendigen Schritte zur Serviceidentifikation leiten. Das Fehlen entsprechender Werkzeuge behindert insbesondere eine ggf. mögliche Optimierung des Ergebnisses bei der Identifikation von Services.

Qualitätsaussage. Die Qualität des Ergebnisses einer durchgeführten Serviceidentifikation ist für die praktische Anwendbarkeit eines propagierten Ansatzes von grundlegender Bedeutung, da die Realisierung der geplanten SOA mit nicht zu vernachlässigenden strategischen, aber auch finanziellen Konsequenzen verbunden ist. Im Idealfall kann die verwendete Identifikationsmethode die Korrektheit eines Ergebnisses, insbesondere die Vermeidung lokaler Optima bei einer ggf. durchgeführten Optimierung, garantieren. Dies ist vor allem bei der Verwendung algorithmisch beschriebener Methoden zu fordern. Kommt ein heuristisches Verfahren zum Einsatz, das aufgrund methodeninhärenter Einschränkungen eine solche Garantie nicht geben kann, bieten sich andere Möglichkeiten die Lösungsqualität zu analysieren. Oftmals ist es möglich die maximale Abweichung von einem globalen Optimum theoretisch zu bestimmen (vgl. hierzu bspw. [Jun07]). Weiterhin bietet eine Sensitivitätsanalyse dem Anwender die Möglichkeit zur Überprüfung der Empfindlichkeit

einer erreichten Lösung hinsichtlich der Variation von Inputparametern. Zahlreiche Ansätze verzichten jedoch gänzlich auf eine Evaluation des Identifikationsergebnisses.

Validierung. Die Validierung eines Serviceidentifikationsansatzes erlaubt weitgehende Rückschlüsse auf die Korrektheit, Verwendbarkeit und Erprobung der dargestellten Vorgehensweisen. Während eine Plausibilitätsprüfung zwar die grundsätzliche Korrektheit eines Vorgehensmodells demonstriert, zeigt erst die Betrachtung von Anwendungsfällen die notwendigen Nutzungsbedingungen sowie potentielle Einsatzmöglichkeiten und –grenzen auf. Im Idealfall kann der präsentierte Identifikationsansatz bereits durch vielfache praktische Anwendung und daraus resultierende „Best Practices“, die u.a. den Einsatz weiter operationalisieren, bestätigt werden.

2.5 Klassifikationsschema

Die zuvor beschriebenen Kriterien lassen sich zu einem Klassifikationsschema zusammenfassen, dass in Tabelle 1 dargestellt ist. Die Ausprägungen der einzelnen Kriterien wurden dabei zu einem morphologischen Kasten zusammengefasst. Das Klassifikationsschema dient als Grundlage zur Einordnung der verschiedenen Ansätze zur Identifikation von Services, die im nächsten Kapitel vorgestellt werden.

Klassifikationsrahmen für Methoden zur Serviceidentifikation				
Zugrundeliegende Servicedefinition	Keine	Technisch	Fachlich	
Formalisierungsgrad	Ad-Hoc	Allgemeine Ratschläge	Strukturiert	Algorithmus
Übergeordnetes Vorgehensmodell	Ja	Nein		
Ableitungsrichtung	Top Down	Bottom Up	Meet In The Middle	
Optimierendes Verfahren	Ja (Exakt)	Ja (Heuristisch)	Nein	
Modellsichten	Daten	Funktionen	Prozesse	
Berücksichtigung existierender Systemstrukturen	Gegeben	Nicht gegeben		
Berücksichtigung von Umsystemabhängigkeiten	Ja	Nein		
Unterscheidung von Servicehierarchien	Ja	Nein		
Unterscheidung von Servicetypen	Ja	Nein		
Werkzeugunterstützung	Gegeben	Nicht gegeben		
Qualitätsaussage	Keine	Sensitivitätsanalyse	Qualitätsgarantie	
Validierung	Keine	Plausibilitäts-prüfung	Anwendungsfall	Best Practices

Tabelle 1: Klassifikationsrahmen

3 Klassifikation von Ansätzen zur Serviceidentifikation

Um serviceorientierte Architekturen erfolgreich einsetzen zu können, bedarf es vor allem eines Ansatzes zur systematischen Identifikation von Services [Ars04], der auf das Konzept einer SOA abgestimmt ist. Ziel dieses Kapitels ist es daher, die in der Literatur publizierten und häufig referenzierten Ansätze zur Identifikation von Services nach den in Abschnitt 2 identifizierten Kriterien zu klassifizieren und im Hinblick auf die geforderte systematische Identifikation zu bewerten.

In Abschnitt 3.1 werden dabei zunächst Ansätze vorgestellt, die speziell auf die Identifikation von Services bei der Gestaltung serviceorientierter Architekturen ausgelegt wurden. Diese werden in Abschnitt 3.2 um weitere Ansätze ergänzt, die zur Identifikation von Informationssystemteilen allgemein bzw. von Software-Komponenten entwickelt wurden, sich prinzipiell jedoch auch zur Identifikation von Services verwenden lassen. Abschnitt 3.3 schließt mit der zusammenfassenden Gegenüberstellung der vorgestellten Ansätze.

3.1 Spezialisierte Ansätze zur Identifikation von Services

Service-Oriented Analysis and Design (Zimmermann et al. und Arsanjani). Die Eignung und Übertragung bewährter Methoden der Softwareentwicklung bei der Einführung von SOA untersuchen Zimmermann et al. [ZKG04]. Elemente aus Object-Oriented Analysis and Design (OOAD), Enterprise Architecture (EA) Frameworks und Business Process Modeling (BPM) werden zu einem Service-Oriented Analysis and Design (SOAD) Ansatz kombiniert und erweitert. Qualitätsfaktoren zu SOAD werden definiert und allgemeine Ratschläge zu allen Phasen des Einführungszyklus gegeben. Ein ganzheitliches Vorgehensmodell wird nicht beschrieben. Die meisten der in Kapitel 2.3 angeführten Kriterien zur Modellbildung werden angesprochen, jedoch bleiben konkrete Empfehlungen ebenso aus, wie eine Definition des Servicebegriffs. Im Hinblick auf eine Serviceidentifikation wird darauf hingewiesen, dass SOA meist nicht auf der grünen Wiese, sondern auf Grundlage bereits existierender Strukturen eingeführt wird, weshalb ein reiner Top-Down-Ansatz nicht ausreichend wäre. Die mangelnde Anwendbarkeit klassischer Entwicklungsmethoden auf die Servicefindung kommentieren die Autoren mit ”there is room for additional creative thinking.“ [ZKG04], ohne jedoch Alternativen darzustellen. Das zur Veranschaulichung angeführte, theoretische Fallbeispiel unterstreicht den reinen Empfehlungscharakter des Beitrags, der als Grundlage für weitere Forschung dienen kann.

Aufbauend auf dem von Zimmermann et al. propagierten SOAD Ansatz formuliert Arsanjani [Ars04] konkretere Empfehlungen zur Identifikation, Spezifikation und Realisierung von Services. Im Sinne einer eher technischen Sichtweise von Services wird insbesondere die Identifikationsphase konkretisiert. Die Durchführung von Top-Down- und Bottom-Up-Analysen wird ergänzt durch ein goal-service modeling, um nicht identifizierte aber benötigte Services aufzudecken. Auch dieser weitestgehend theoretische Ansatz ohne

Referenzbeispiel kommt nicht über den Status einer losen Sammlung von Ratschlägen hinaus. Ein strukturiertes Vorgehen zur Servicefindung bleibt weiterhin offen.

Service-Oriented Analysis (Erl). Erl beschreibt in seinen Büchern zu Konzeption und Design von serviceorientierten Architekturen einen Ansatz zur Identifikation von Services im Rahmen eines ganzheitlichen Entwicklungsmodells [Erl05, Erl07]. Basierend auf einer Analyse von Geschäftsprozessen sowie existierender Systemstrukturen werden Servicekandidaten bestimmt, aus denen wiederum die Services herauskristallisiert werden können. Erl unterscheidet in dem Meet-In-The-Middle-Ansatz elf Servicehierarchien und -klassen, die sich teils auch von seiner allgemeinen, eher technischen Servicedefinition entfernen. Abhängigkeiten zwischen Services werden nur am Rande berücksichtigt. Eine mögliche Unterstützung durch Werkzeuge wird nicht in Betracht gezogen, ebenso wenig eine Zusicherung von Qualitätsgarantien. Das Vorgehen ist im Rahmen des übergeordneten Vorgehensmodells strukturiert, jedoch wird kein explizites Verfahren angegeben, vielmehr bleibt es weitgehend bei Ratschlägen und allgemeinen Anleitungen. Die Vorgehensweise insgesamt wird durchgehend anhand von Case Studies demonstriert.

SOA Framework (Erradi et al.). Die Identifikation von Services als Teil eines architektonischen SOA Frameworks (SOAF), welches die Phasen der Spezifikation und Realisierung umfasst, wird von Erradi et. al. präsentiert [EAK06]. Der Ansatz kommt ohne die Formulierung einer konkreten Definition von Services aus. Auf Basis von Geschäftsmodellen werden zu erfüllende Services in einem Top-Down-Ansatz identifiziert. Bestehende Services werden bottom-up aus dem vorhandenen Code und der zugehörigen Datenstruktur herausgearbeitet. Noch zu realisierende Services werden dann durch einen Abgleich von bestehenden und zu erfüllenden Services bestimmt. Ein sog. Tool-based Mining unterstützt hierbei die Bottom-Up-Analyse der bestehenden Code- und Datenfragmente. Die Top-Down-Analyse der Geschäftsprozesse erfolgt durch eine Kombination aus Interviews und Tools. Abgesehen von Hinweisen auf eine mögliche Werkzeugunterstützung, werden keine Werkzeuge erwähnt und auch das Vorgehen für einen Abgleich der vorhandenen sowie benötigten Services nicht näher beschrieben. Ausschließlich das Ergebnis eines Fallbeispiels, nicht jedoch die Anwendung der präsentierten Methode, wird dargestellt.

BPM and SOA Handshake (Inaganti und Behara). Ein strukturiertes Vorgehen zur Identifikation von Services beschreiben Inaganti und Behara [IB07]. In vier Schritten werden potentielle Services sowohl top-down, als auch bottom-up ermittelt und gegenübergestellt. Darüber hinaus notwendige Services werden auf unbestimmte Art und Weise ergänzt. Obwohl der Beitrag die Identifikation detaillierter und strukturierter beschreibt als bspw. Zimmermann et al. [ZKG04] und Arsanjani [Ars04], reicht er selten über eine Auflistung von Möglichkeiten und Ratschlägen hinaus. Die Entwicklung von Optimierungsverfahren und

deren Unterstützung durch Werkzeuge wird nicht betrachtet. Ein Referenzbeispiel findet keine Erwähnung.

Identifikation und Gestaltung von Services (Winkler). Winkler stellt einen Ansatz vor, der neben der Serviceidentifikation auch die Phasen der Gestaltung und Umsetzung von Services abdeckt [Win07]. In der Identifikationsphase werden Services auf Basis von UML Aktivitätsdiagrammen so bestimmt, dass drei zuvor definierte Anforderungen an Services erfüllt werden. Die Anforderungen umfassen die Wiederverwendbarkeit von Services, eine Vermeidung redundanter Implementierungen in verschiedenen Services, sowie eine lose Kopplung von Services über klar definierte, einfache Schnittstellen. Die Servicefindung durchläuft die vier aufeinander folgenden Schritte der ”Erstellung von Aktivitätsdiagrammen“, der ”Aufbereitung der Aktivitätsdiagramme“ sowie der ”Identifikation potentieller Services“ und zuletzt einer ”Analyse der Häufigkeit der Verwendung“. Auf der Grundlage einer impliziten, fachlichen Servicedefinition, werden Services in einem semistrukturierten Top-Down-Ansatz bestimmt. Eine Optimierung der Servicestruktur findet im Rahmen der Identifikationsphase nicht statt. Auch die Einhaltung der definierten Anforderungen wird während der Servicefindung nicht garantiert. Die Unterscheidung von Servicehierarchien wird ebenso wie die Berücksichtigung von Abhängigkeiten und bestehenden Strukturen erwähnt, jedoch bleibt unklar inwiefern dies Auswirkungen auf die Servicefindung hat. Der komplette Prozess wird anhand eines kurzen Beispiels aus dem Finanzdienstleistungsbereich beschrieben. Eine Unterstützung durch Werkzeuge findet keine explizite Erwähnung.

Konzeptionsmethode zu SOA (Beverungen et al.). Beverungen et al. vergleichen unterschiedliche Ansätze zur Entwicklung serviceorientierter Architekturen und stellen als Resultat einen eigenen Ansatz vor, der die Phasen der Identifikation und Spezifikation von Services abdeckt und in ein übergeordnetes Vorgehensmodell integriert ist [BKM08]. Services werden hierbei top-down durch eine Dekomposition von Geschäftsprozessen identifiziert. Besonderes Augenmerk wird auf die Analyse der sog. Übernahme- und Sichtbarkeitspotenziale einzelner Prozessschritte für Geschäftspartner gelegt. Existierende Strukturen und Services finden in der Identifikationsphase Berücksichtigung. Abhängigkeiten zwischen Services werden dagegen erst in der Spezifikationsphase identifiziert. Es wird zwischen den zwei Servicehierarchien Process und Basic unterschieden. Eine Unterscheidung von Servicetypen wird zwar angesprochen, ist aber nicht integraler Bestandteil der Methode. Ein strukturiertes Vorgehen zur Identifikation wird propagiert, allerdings bleiben Details unerwähnt. Die Möglichkeiten der Implementierung einer Optimierung, sowie die Unterstützung des Identifikationsprozesses durch Werkzeuge wird ebenfalls nicht angesprochen. Ein realer Anwendungsfall zeigt die Praktikabilität der Methode, wobei Teilschritte jedoch nicht expliziert werden.

3.2 Allgemeine Ansätze zur Identifikation von Modulen

Business Systems Planning (IBM). IBM beschreibt einen Top-Down-Ansatz, mit dem sich betriebliche Informationssysteme auf der Basis erhobener Geschäftsprozess- und Datenmodelle systematisch modularisieren lassen [IBM84]. Dabei werden detaillierte Arbeitsschritte und Vorgehensweisen angegeben. Für die Identifikation von Systemteilen (die sich ggf. als Services nutzen lassen) wird eine graphische Methode beschrieben, die Aktivitäten eines Geschäftsprozesses und die von ihnen jeweils verarbeiteten Daten gegen überstellt. Auf heuristischer Basis kann dabei eine Gruppierung zusammengehöriger Aktivitäten solange optimiert werden, bis bei der Verarbeitung möglichst wenige Daten aus anderen Informationssystemteilen bezogen werden (also bestehende Datenabhängigkeiten zwischen Aktivitätsbündeln minimiert sind). Die vorgestellte Methode macht allerdings weder eine Qualitätsaussage bezüglich der erreichten Optimierung, noch lassen sich dabei bereits bestehende modulare Strukturen berücksichtigen.

CompMaker (Jain et al.). Einen aus der komponentenorientierten Anwendungsentwicklung stammenden Ansatz beschreiben Jain et al. [JCIR01]. Durch Einbeziehung von Informationen eines Domänenmodells in UML Notation sowie existierender objektorientierter Klassen, werden auf Wiederverwendung ausgelegte Komponenten in einem Bottom-Up-Ansatz identifiziert. Eine durch das Gruppieren von Klassen erreichte Startlösung wird mittels Add-, Move- und Exchange-Heuristiken, sowie manuell verbessert. Klassen spielen also als Bausteine der Komponenten eine hohe Rolle. Das Ergebnis wird weiterhin stark durch die vom Designer zu definierenden Präferenzen beeinflusst. Services werden in diesem sehr strukturierten, komponentenorientierten Ansatz nicht explizit fokussiert, können jedoch abgeleitet werden. Der Identifikationsprozess wird durch das Werkzeug CompMaker unterstützt und auf ein Fallbeispiel aus dem Autoversicherungsbereich angewandt. Eine Validierung der Ergebnisse findet keine Erwähnung.

Modularitätsanforderungen (Szyperski et al.). In seinem Buch zur komponenten-orientierten Anwendungsentwicklung beschreibt Szyperski 15 Modularitätsanforderungen, die von identifizierten Software-Komponenten und Services idealerweise zu erfüllen sind [SGM02]. So sollten identifizierte Services nicht nur aus fachlicher und technischer Sicht eigenständig, sondern bspw. auch unabhängig von anderen weiter zu entwickeln, auszuliefern, zu installieren und zu warten sein. Daneben sollten sie auch im Rahmen der Abrechnung und bei der Abwicklung von Haftungsfragen möglichst unabhängig voneinander betrachtet werden können. Die genannten Kriterien sind zwar detailliert formuliert, gehen jedoch nicht über das Niveau allgemeiner Ratschläge hinaus. Szyperski gibt darüber hinaus keine Hinweise, wie die Einhaltung der Kriterien bei der Identifikation ggf. durch eine bestimmte Vorgehensweise gewährleistet werden kann oder welche Modellbildung dafür zu

erfolgen hat. So ergibt sich aus den Kriterien allenfalls ein konzeptioneller Rahmen, anhand dessen sich identifizierte Services validieren lassen.

BCI und BCI-3D (Albani et al.). Albani et al. beschreiben in [ADZ05, AD06] die Business Component Identification (BCI) Methode zur Identifikation von Komponenten, sowie deren Weiterentwicklung zu BCI-3D. In beiden Versionen werden Komponenten auf Basis der Prozess- und Datenstruktur eines fachlichen Domänenmodells mit Hilfe von Heuristiken identifiziert. Die Komponentenidentifikation erfolgt in einem algorithmischen Top-Down-Ansatz, der durch ein Werkzeug unterstützt wird und sich in den Business Component Modeling Process (BCMP) integriert. Der erste Ansatz lehnt sich hierbei an [IBM84] an und betrachtet lediglich Abhängigkeiten zwischen einzelnen Funktionen und Datenobjekten. Die Weiterentwicklung verwendet graphentheoretische Clustering-Methoden, die mit Hilfe von Start- und Verbesserungsheuristiken aus Prozessschritten und Datenobjekten Komponenten identifizieren. Hierbei werden auch Beziehungen zwischen und innerhalb von Funktionen sowie Daten berücksichtigt. Eine Qualitätsgarantie auf die heuristisch ermittelte Lösung wird nicht gegeben. Vorhandenen Strukturen sowie auftretenden Abhängigkeiten wird mittels einer Integration in das Domänenmodell nur teilweise Rechnung getragen. Fallstudien zu beiden Methoden wurden durchgeführt [SBAK05, AD06]. Durch die Festlegung von Gewichtungen während der Optimierung kann der Designer Einfluss auf das Ergebnis nehmen, was mangels klarer Empfehlungen die Anwendbarkeit der Methode erschwert.

3.3 Vergleich der Ansätze

Ein Überblick über die Ergebnisse der Klassifikation aller Ansätze ist in Tabelle 2 zusammengefasst. Hieraus wird deutlich, dass keiner der gängigen Ansätze in allen Kriterien überzeugt. So wird beispielsweise kaum einer der Ansätze durch Werkzeuge unterstützt und von keinem eine Qualitätsgarantie gegeben. Eine Validierung der Ansätze durch Best Practices ist bisher ebenfalls noch nicht erfolgt. Auffällig ist ferner, dass die aus der älteren Disziplin der komponentenorientierten Anwendungsentwicklung stammenden Identifikationsmethoden meist strukturiertere, durch Algorithmen unterstützte und besser validierte Vorgehensweisen beschreiben, als die Ansätze aus dem Gebiet der vergleichsweise jungen serviceorientierten Architekturen. Hervorzuheben ist noch, dass zahlreiche der vorgestellten Ansätze ganz offenbar ohne jede Servicedefinition auskommen.

	Zimmermann et al. [ZKG04]	Arsanjani [Ars04]	Erl [Erl05, Erl07]	SAP [SAP05]	AIER [Aie06]	Erradi et al. [EAK06]	Inaganti, Behara [IB07]	Winkler [Win07]	Beverungen et al. [BKM08]	IBM [IBM84]	Jain et al. [JCIR01]	Szyperski et al. [SGM02]	Albani et al. [ADZ05, AD06]
Zugrundeliegende Servicedefinition	Keine	Technisch	Technisch	Technisch	Keine	Fachlich	Keine	Fachlich	Keine	Keine	Keine	Keine	Keine
Formalisierungsgrad	Allgemeine Ratschläge	Allgemeine Ratschläge	Allgemeine Ratschläge	Allgemeine Ratschläge	Algorithmus	Strukturiert	Allgemeine Ratschläge	Strukturiert	Strukturiert	Strukturiert	Algorithmus	Allgemeine Ratschläge	Algorithmus
Übergeordnetes Vorgehensmodell	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✗	✓
Ableitungsrichtung	Meet In The Middle	Meet In The Middle	Meet In The Middle	Meet In The Middle	Meet In The Middle	Meet In The Middle	Meet In The Middle	Top Down	Top Down	Top Down	Bottom Up	k. A.	Top Down
Optimierendes Verfahren	✗	✗	✗	✗	✓ (Heuristisch)	✗	✗	✗	✗	✓ (Heuristisch)	✓ (Heuristisch)	✗	✓ (Heuristisch)
Modellsichten	Prozesse und Daten	Prozesse, Daten und Funktionen	Prozesse, Daten und Funktionen	Prozesse, Daten und Funktionen	Prozesse und Daten	Prozesse und Daten	Prozesse, Daten und Funktionen	Prozesse und Funktionen	Prozesse	Prozesse und Daten	Funktionen ***	k. A.	Prozesse und Daten
Berücksichtigung existierender Systemstrukturen	✓	✓	✓	✓	✗	✓	✓	✗	✓	✗	✗	✓	✓ ****
Berücksichtigung von Umsystemabhängigkeiten	✗	✓	✗	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓
Unterscheidung Servicehierarchien	✗	✓ (2)	✓ *	✓ (2)	✗	✓ (2)	✓ (2)	✗	✓ (2)	✗	✗	✗	✗
Unterscheidung von Servicetypen	✗	✗	✓ *	✓	✗	✗	✗	✗	✓ (2)	✗	✗	✗	✗
Werkzeugunterstützung	✗	✗	✗	✗	✓	✓ **	✗	✗	✗	✗	✓	✗	✓
Qualitätsaussage	Keine	Keine	Keine	Keine	Keine	Keine	Keine	Keine	Keine	Keine	Keine	Keine	Keine
Validierung	Keine	Keine	Plausibilitäts-prüfung	Keine	Anwendungsfall	Plausibilitäts-prüfung	Keine	Anwendungsfall	Anwendungsfall	Anwendungsfall	Anwendungsfall	Keine	Anwendungsfall

* 11 Servicearten, ** nur Teilschritte, *** durch Integration in Domänenmodell, **** Klassen aus der OO

Tabelle 2: Einordnung der vorgestellten Methoden in den Klassifikationsrahmen

4 Verwandte Ansätze

Die Entwicklung von Ansätzen zur systematischen Identifikation von Services ist, schon aufgrund der relativ kurzen Zeit seit der dieses Thema betrachtet wird, derzeit zum großen Teil erst noch im Gange. Dennoch existieren bereits einige Publikationen, in denen entsprechende Ansätze vorgeschlagen wurden. Die Qualität dieser Ansätze und damit ihre Eignung, die gewünschte systematische Identifikation von Services im Sinne eines methodischen Vorgehens zu unterstützen, variiert jedoch deutlich. Parallel zur Entwicklung neuer Ansätze für die Identifikation von Services empfiehlt sich deshalb auch eine vergleichende Einordnung der vorhandenen Ansätze, anhand derer sich der aktuelle Stand der Technik ermitteln lässt.

In der Literatur finden sich bislang allerdings nur wenige Vergleiche existierender Ansätze zur Identifikation von Services, die eine systematische Einordnung versuchen. In ihrem Beitrag zur Konzeption serviceorientierter Architekturen vergleichen Beverungen et. al unter anderem auch Ansätze zur Identifikation von Services [BKM08]. Jedoch liegt der Fokus hier eher auf der Betrachtung ganzheitlicher Ansätze, die den gesamten Entwurfsprozess einer SOA abdecken. Die zusammengestellten Ansätze sind daher nur bedingt auf die Identifikation von Services ausgelegt. Zudem erscheinen die genannten Vergleichskriterien mitunter willkürlich gewählt und sind zudem nicht aus der Literatur abgeleitet.

Einen Vergleich strukturierter Vorgehensweisen zur Identifikation von Software-Komponenten, die wichtige Hinweise auf die Gestaltung entsprechender Ansätze im Bereich der Serviceorientierung liefern können, führen Wang et. al durch [WXZ05]. Dort liegen allerdings vor allem Kriterien zugrunde, die Aufschluss über die gewählte Vorgehensweise geben. Dabei werden Ansätze mit einer sog. Domain-Engineering-Strategie, einer Gegenüberstellung von Daten und verarbeitenden Funktionen sowie solche mit einem Verfahren zur Gruppierung zusammengehöriger Funktionen unterschieden. Die Unterscheidung entsprechender Vorgehensweisen lässt sich auf die hier verglichenen Ansätze zur Serviceidentifikation jedoch nur schwer übertragen, da die meisten über kein vergleichbares Vorgehen verfügen. Gerade dieser Umstand kann ggf. als Indiz für die noch mangelnde Reife der meisten heute verfügbaren Ansätze fungieren.

5 Schlussbetrachtung

In Kapitel 3 wurden aktuelle Ansätze zur Identifikation von Services einander gegenübergestellt und ihre jeweiligen Stärken und Schwächen diskutiert. Dafür wurde zuvor ein detaillierter Katalog mit Kriterien aus der Methodenforschung der Ingenieursdisziplinen bzw. der (Wirtschafts-) Informatik aufgestellt, die Ansätze zur systematischen Serviceidentifikation idealerweise erfüllen sollten. Wesentliche Erkenntnisse, die den

Reifegrad der bislang vorgestellten Ansätze in Frage stellen, wurden dabei bereits in Abschnitt 3.3 hervorgehoben und sollen hier nicht noch einmal wiederholt werden.

Anzumerken ist darüber hinaus jedoch, dass die verschiedenen Ansätze hinsichtlich der zu berücksichtigenden Abhängigkeiten zu anderen Services oder der unterschiedenen Servicehierarchien bzw. Servicetypen voneinander abweichen. Eine ggf. durchzuführende schrittweise Verfeinerung bei der Serviceidentifikation wird deshalb nicht von allen Ansätzen automatisch unterstützt. Durch die von manchen Ansätzen vorgenommene Trennung sog. Entity und Task Services wird in der Regel ein anderes Ergebnis bei der Serviceidentifikation erreicht als bei Ansätzen, die auf diese Unterscheidung verzichten. Hierauf ist bei der Auswahl eines Ansatzes ggf. zu achten.

Erheblicher Forschungsbedarf besteht insbesondere bei der offenen Frage, wie sich die Ansätze zur Identifikation von Services hin zu ausgereiften Methoden mit einer stärker formalisierten und detaillierten Vorgehensweise weiterentwickeln lassen. Nur so wird sich eine systematische Identifikation von Services im Sinne eines ingenieurmäßigen Vorgehens realisieren lassen. In diesem Zusammenhang ist auch der Einsatz von optimierenden Verfahren und Vorgehensweisen, die zumindest eine Abschätzung der Güte von erreichten Ergebnissen zulassen, stärker voranzutreiben.

Bezeichnend ist, dass bei der Identifikation von Services nicht auf bereits bestehende Ansätze der Komponentenorientierung zurückgegriffen, sondern ganz offenbar wieder von Neuem begonnen wird. Diese Beobachtung trifft für das Vorgehen im Kontext serviceorientierter Architekturen an vielen Stellen zu und wird in der Literatur deshalb auch zu Recht kritisiert [SGM02]. Bei genauerem Hinsehen lässt sich jedoch feststellen, dass zahlreiche Methoden, die im Kontext sog. Fachkomponenten (engl. Business Components) ebenfalls mit starkem fachlichen Bezug entwickelt wurden, durchaus bei der Gestaltung serviceorientierter Architekturen eingesetzt werden können. Eine synergetische Betrachtung beider Disziplinen könnte das Entstehen ausgereifter Methoden für die Serviceidentifikation deshalb nachhaltig beschleunigen.

Literatur

[AD06] Antonia Albani und Jan L.G. Dietz. The Benefit of Enterprise Ontology in Identifying Business Components. In IFIP World Computing Conference, Santiago de Chile, Chile, August 2006.

[ADZ05] Antonia Albani, Jan L.G. Dietz und Johannes Maria Zaha. Identifying Business Components on the basis of an Enterprise Ontology. In D. Konstantas, J.-P. Bourrieres, M. Leonard und N. Boudjlida, Hrsg., Interoperability of Enterprise

Software and Applications, Seiten 335–347, Geneva, Switzerland, 2005. Springer Verlag.

[Ars04] Ali Arsanjani. Service-oriented modeling and architecture - How to identify, specify, and realize services for your SOA. IBM developerWorks Web services zone, Nov. 2004.

[BKM08] Daniel Beverungen, Ralf Knackstedt und Oliver Müller. Entwicklung Serviceorientierter Architekturen zur Integration von Produktion und Dienstleistung - Eine Konzeptionsmethode und ihre Anwendung am Beispiel des Recyclings elektronischer Geräte. Wirtschaftsinformatik, 50(3):220–234, 2008.

[Bro00] A.W. Brown. Large-Scale, Component-Based Development. Prentice Hall, Upper Saddle River, NJ, 2000.

[BS04] Jörg Becker und Reinhard Schütte. Handelsinformationssysteme – Domänenorientierte Einführung in die Wirtschaftsinformatik. Redline, Franfurt, 2004.

[BS06] Hans-Jörg Bullinger und Peter Schreiner. Service Engineering: Ein Rahmenkonzept für die systematische Entwicklung von Dienstleistungen. In Hans-Jörg Bullinger und August-Wilhelm Scheer, Hrsg., Service Engineering: Entwicklung und Gestaltung innovativer Dienstleistungen, Kapitel 1, Seiten 53–84. Springer, 2006.

[CD01] John Cheesman und John Daniels. UML Components. A Simple Process for Specifying Component-Based Software. Addison-Wesley, Upper Saddle River, NJ, 2001.

[Dav93] A. M. Davis. Software Requirements. Objects, Functions, and States. Prentice Hall, Englewood Cliffs, NJ, 1993.

[DW99] Desmond Francis D’Souza und Alan Cameron Wills. Objects, Components, and Frameworks with UML. The Catalysis Approach. Addison-Wesley, Upper Saddle River, NJ, 1999.

[EAK06] Abdelkarim Erradi, Sriram Anand und Naveen Kulkarni. SOAF: An Architectural Framework for Service Definition and Realization. Services Computing, 2006. SCC ’06. IEEE International Conference on, Seiten 151–158, Sept. 2006.

[Erl05] Thomas Erl. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[Erl07] Thomas Erl. SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl). Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.

[HS00] P. Herzum und O. Sims. Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise. John Wiley & Sons, New York, NY, 2000.

[IB07] Srikanth Inaganti und Gopala Krishna Behara. Service Identification: BPM and SOA Handshake. Bericht, BPTrends, March 2007.

[IBM84] IBM Corporation. Business Systems Planning: Information Systems Planning Guide. Technical report ge20-0527-4, International Business Machines Corporation, 1984.

[JCIR01] Hemant Jain, Naresh Chalimeda, Navin Ivaturi und Balarama Reddy. Business Component Identification - A Formal Approach. In EDOC ’01: Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing, Seite 183, Washington, DC, USA, 2001. IEEE Computer Society.

[Jun07] Dieter Jungnickel. Graphs, Networks and Algorithms. Springer, Berlin, 3. Auflage, 2007.

[KL04] Donald Kossmann und Frank Leymann. Web Services. Informatik Spektrum, 26(2):117–128, 2004.

[MDW02] Rainer Minz, Anthony Datel und Holger Wenzky. Web Services - nur eine Schimäre? Information Management and Consulting, 17(3):6–12, 2002.

[Mil71] H.D. Mills. Top-down programming in large systems. In R. Ruskin, Hrsg., Debugging Techniques in Large Systems, Seiten 41–55. Prentice Hall, 1971.

[MSJL06] James Mcgovern, Oliver Sims, Ashish Jain und Mark Little. Enterprise Service Oriented Architectures. Springer, 2006.

[Nat03] Yefim V. Natis. Service-Oriented Architecture Scenario. Bericht ID Number: AV-19-6751, Gartner Research, 2003.

[OT07] Sven Overhage und Klaus Turowski. Serviceorientierte Architekturen - Konzept und methodische Herausforderungen. In V. Nissen, M. Petsch und H. Schorcht, Hrsg., Service-orientierte Architekturen. Chancen und Herausforderungen bei der Flexibilisierung und Integration von Unternehmensprozessen, Seiten 3–17. Deutscher Universitätsverlag, 2007.

[Par72] D. L. Parnas. On the Criteria to be Used in Decomposing Systems into Modules. Communications of the ACM, 15(12):1053–1058, 1972.

[PBFG03] Gerhard Pahl, Wolfgang Beitz, Jörg Feldhusen und Karl-Heinrich Grote. Konstruktionslehre: Grundlagen erfolgreicher Produktentwicklung - Methoden und Anwendung. Springer, Berlin, Heidelberg, 2003.

[PTD+06] Mike Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann und Bernd Krämer. Service-Oriented Computing: A Research Roadmap. In Dagstuhl

Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik, 2006. Schloss Dagstuhl.

[Rop99] Günter Ropohl. Allgemeine Technologie: Eine Systemtheorie der Technik. Hanser, München, Wien, 1999.

[SBAK05] Bernhard Selk, Bettina Bazijanec, Antonia Albani und Sebastian Klöckner. Experience Report: Appropriateness of the BCI-Method for Identifying Business Components in large-scale Information Systems. In Klaus Turowski und Johannes Maria Zaha, Hrsg., Component-Oriented Enterprise Applications (COEA), LNI. Bd. P-70, Seiten 87–92, 2005.

[Sch98] A. W. Scheer. ARIS: Vom Geschäftsprozess zum Anwendungssystem. Springer, Berlin, Heidelberg, 3.. Auflage, 1998.

[SGM02] Clemens Szyperski, Dominik Gruntz und Stephan Murer. Component Software. Beyond Object-Oriented Programming. Addison-Wesley, Harlow, 2.. Auflage, 2002.

[Tur03] Klaus Turowski. Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme. Shaker Verlag, 2003.

[WH07] Thomas Wilde und Thomas Hess. Forschungsmethoden der Wirtschaftsinformatik – Eine empirische Untersuchung. Wirtschaftsinformatik, 49(4):280–287, 2007.

[Win07] Veronica Winkler. Identifikation und Gestaltung von Services - Vorgehen und beispielhafte Anwendung im Finanzdienstleistungsbereich. Wirtschaftsinformatik, 49(4):257–266, 2007.

[WXZ05] Zhongjie Wang, Xiaofei Xu und Dechen Zhan. A Survey of Business Component Identification Methods and Related Techniques. International Journal of Information Technology, 2(4):229–238, 2005.

[ZKG04] Olaf Zimmermann, Pal Krogdahl und Clive Gee. Elements of Service-Oriented Analysis and Design - An interdisciplinary modeling approach for SOA projects. IBM developer-Works Web services zone, Jun. 2004.

II.4 Beitrag: „A Survey of Service Identification Approaches - Classification Framework, State of the Art, and Comparison”

Autoren: Dominik Birkmeier, Sebastian Klöckner und Sven Overhage
Alle Lehrstuhl WI-SE, Universität Augsburg,
Universitätsstraße 16, D-86135 Augsburg,
Email: dominik.birkmeier@wiwi.uni-augsburg.de
sebastian.kloeckner@wiwi.uni-augsburg.de
sven.overhage@wiwi.uni-augsburg.de

Erschienen in: Enterprise Modelling and Information Systems Architectures – An International Journal, 4(2): 20-36 (2009).

Aufgrund ihrer modularen Natur verspricht die Verwendung serviceorientierter Architekturen bei Anwendungssystemen viele Vorteile. Die erfolgreiche Einführung von SOAs hängt jedoch in entscheidendem Maße von der effizienten methodischen Unterstützung des zugrundeliegenden neuen Entwicklungsparadigmas ab. Wie die Anzahl der momentan veröffentlichen Beiträge anschaulich illustriert, steht insbesondere die Entwicklung systematischer Methoden zur Identifikation geeigneter Services, die dann als Bausteine von Anwendungssystemen verwendet werden können, im Mittelpunkt des Interesses. Die in der Literatur vorgestellten Ansätze unterscheiden sich jedoch signifikant im Hinblick auf ihre Konzepte und Vorgehensweisen. In diesem Beitrag werden daher der aktuelle Stand der Technik im Bereich der Serviceidentifikation untersucht und die jeweiligen Unterschiede zwischen den vorgestellten Ansätzen genauer beleuchtet. Für die Bewertung der einzelnen Serviceidentifikationsansätze wird ein Klassifikationsschema mit den notwendigen Unterscheidungsmerkmalen vorgestellt. Dieses Schema wird dann für die Gegenüberstellung sowie die Analyse der verschiedenen Ansätze verwendet. Basierend auf diesem Vergleich werden anschließend die jeweiligen Stärken und Schwächen der einzelnen Ansätze herausgearbeitet und sich daraus ergebende Auswirkungen für die Praxis abgeleitet. Abschließend werden mögliche neue Forschungsgebiete, die bei einer Weiterentwicklung des aktuellen Stands der Technik im Bereich der Serviceidentifikation interessant sein könnten, aufgezeigt.

1 Motivation

Today, business application development is facing a whole set of demanding challenges. Among them, especially managing the complexity of applications, flexibly adapting applications to changes in the business environment, and extending existing applications to quickly implement new functionality are key challenges, which have to be solved by adequate development techniques ([Brow00], [ChDa01], [PaTr+06]). With their underlying modular development paradigm, service-oriented architectures (SOA) offer a promising contribution to better meet all of these challenges ([KoLe04], [PaTr+06]).

A prerequisite for the success of the new, service-oriented development paradigm in practice is however a sufficient support with adequate methods and tools [PaTr+06]. Besides the questions of how services can be described, found in catalogues, and composed (according to business requirements), especially the development of methods and practices for a systematic identification of services is in the focus of scientific as well as practical interest ([Arsa04], [ErAn+06]).

The identification of suitable services, which has to be accomplished at the beginning of the development process, provides the basis for the next design steps as well as for the service composition and usage later on [Erl05]. For this reason, it is of central importance for the service-oriented development process as a whole and has accordingly been addressed by a variety of approaches which have been published in literature. These approaches, however, show a significant heterogeneity. They range from ad-hoc findings (which have been gathered by creative thinking or charting an initial project) and general recommendations (which should be considered during the identification of services) to structured methods and algorithmic procedures. Moreover, the underlying service definitions as well as their respective strategy to identify services vary significantly. Due to the short time since a systematic identification of services is in the focus of research, none of the approaches was so far able to become broadly accepted and dominate the others. Comparative examinations, which assess the evolving approaches and help structuring the area of research, are missing as well. For academia, such an assessment helps identifying complementary approaches, which could be combined to obtain improved results, as well as uncovering unresolved issues for further research. For practice, a detailed comparison reveals consequences for the applicability of the proposed approaches in different development scenarios and contexts.

In this paper, we present a survey of service identification approaches, which we classify according to a detailed scheme with distinguishing factors. To determine relevant factors as well as the eligible service identification approaches themselves, we conducted an exhaustive literature study. Starting from a compilation of service identification approaches

already known by the authors, we systematically analyzed relevant conferences and journals (both of the information systems and software engineering disciplines) to collect a first set of approaches. We then traced the citations to search backwards and used Google scholar as well as the Web of Science to look for upcoming approaches. Our research approach is thereby based on the methodology for literature analyses as described by Webster and Watson [WeWa02].

Our survey is structured as follows: Section 2 gives an overview of existing related work to further motivate the research gap. In section 3, we determine and discuss characteristic criteria of service identification approaches. We then refine these criteria into a set of distinguishing factors to evaluate and classify service identification approaches. The distinguishing factors are therefore aggregated into a detailed classification scheme. In section 4, we present the service identification approaches identified during the literature survey and analyze them according to the classification scheme. Commonalities and differences between the presented approaches are highlighted during a comparative discussion based on an argumentative-deductive approach. After describing the current state of the art in service identification and uncovering areas requiring further research, we conclude by summarizing key findings and outlining future directions to further improve existing service identification approaches.

2 Related Work

The development of systematic service identification approaches is, especially due to the short time since this research area is under investigation, still in progress. Nonetheless, there are numerous publications which propose relevant approaches. But the quality of these contributions and therefore their suitability for the desired systematic identification of services in the sense of a methodical procedure is varying significantly. To keep track of the ongoing development, classifying existing approaches and providing an overview of the state of the art is advisable in parallel with the creation of new approaches for service identification.

However, scientific literature offers only few comparisons of service identification approaches, which present a systematic classification. While Beverungen et al. [BeKn+08] also evaluate different approaches for service identification as part of their contribution on the conception of a SOA, they mainly focus on the comparison of integrated approaches, which cover the whole development process of a SOA. The presented approaches are therefore only partly focused on the identification of services or the identification is only mentioned aside. Specialized approaches for service identification were mostly left unconsidered, as they do not have an integrated procedure. In addition, the comparison criteria stated by Beverungen et al. seemed to be arbitrarily selected and not deduced from literature.

A comparison of structured approaches for the identification of software components, which could provide important insights for the design of corresponding approaches in the area of service-orientation, has been provided by Wang et al. [WaXu+05]. This contribution only offers criteria which give insights into the used procedure, but does not mention other aspects. It primarily distinguishes approaches which use a so-called domain-engineering strategy. Their comparison, however, can hardly be mapped to the approaches presented in this paper, as most of them do not contain a comparable procedure. This might be a first indicator that many of the existing service identification approaches are still in a premature state or at least diverge from established approaches of the component-based software engineering discipline.

3 Classification Scheme for Service Identification Approaches

A thorough analysis of service identification approaches published in literature reveals different layouts, e.g. with respect to their conceptual design and the identification strategy. In order to compare and classify the different approaches, we firstly introduce a set of criteria which characterizes service identification methods and provide insights into the features of such approaches. For the deduction of characteristic criteria, we refer to research focusing on the conception of systematic design methods in the software engineering as well as in other engineering disciplines [PaBe+07]. From there, we take the following criteria as being characteristic for systematic methods in general:

- the conceptual foundations, on which the approach is built;
- the procedure that is applied by the approach;
- the underlying model used by the approach;
- the supporting measures, which improve its application in practice.

Examining this rather compact set of general criteria allows a better understanding of whether an approach is able to contribute to the aspired systematic identification of services and where deficiencies exist. While these abstract criteria might not necessarily be complete, they have been proven to adequately describe systematic development approaches in theory ([PaBe+07], [Somm06]). For this reason, we used them as a starting point for building our classification scheme and refined them as documented below to describe service identification approaches in particular.

3.1 Foundations

The conceptual design of service identification approaches is manifested in its foundations. They describe the understanding of central concepts, in particular the underlying service definition of the respective approaches. Furthermore, the different approaches have to be distinguished with respect to their degree of formalization and their integration into a comprehensive development process model. While the degree of formalization provides

information about how exact the service identification strategy is being described, the latter indicates whether the approach has been designed to work with data created during earlier development activities and provides specific results for later design steps.

3.1.1 Service definition

In accordance with the fact that a standard service terminology has not been established yet, published identification approaches make use of different service definitions which are being discussed in literature. While services can be broadly defined as “acts of performance offered by one party to another” [LoLe+99] more specific service definitions focus on a variety of additional aspects [Alte08]. Such specific service definitions generally either built upon a more technically oriented viewpoint in the sense of (web) services as being software components ([Arsa04], [McSi+06], [Nati03], [SzGr+02]) or take a domain-oriented perspective ([Alte08], [BaDu06]) to concentrate on conceptual aspects such as the actual business function performed by a service.

Technically oriented service definitions often focus on how to specify and implement services as software artefacts that provide a distinctive functionality. Properties like a loose coupling, reusability, platform-independence, or well-defined service interfaces are at the core of such definitions [SzGr+02]. By contrast, domain-oriented service definitions emphasize that services should provide self-contained sets of functionality which are meaningful from a business perspective. In such definitions, a service typically is understood as an activity of a business application system, which supports the accomplishment of a certain set of business tasks [Alte08]. Technical aspects are, accordingly, of secondary concern.

Since technically and domain-oriented service definitions diverge in central aspects, they are likely to promote different results when being taken as the basis to identify suitable services. To reflect these general differences in the following, we distinguish between approaches with a domain-oriented focus from those with a more technically-oriented service understanding. While a more detailed analysis of the underlying service definitions would also be a desirable research goal, we can only differentiate between the two mentioned archetypes of service definitions in this paper. Gathering and discussing the various service definitions had to be left as a direction of future research. We also did not examine the service definitions underlying the new service science discipline [ChSp06], which focuses on the engineering of services in general. Here, we aim at comparing service identification approaches that promote the development of a SOA for a business application system. Service definitions and approaches belonging to the service science discipline are therefore out of the scope of this particular survey.

3.1.2 Degree of formalization

The degree of formalization ranges from a presentation of so-called ad-hoc findings and general recommendations to structured methods and algorithmic procedures. Ad-hoc

findings are based on creative thinking or experiences gathered by charting an initial project. Typically, they offer only a fuzzy strategy to identify services. General recommendations are proven practices that have been repeatedly applied to identify services with certain desirable characteristics. While they are based on more thoroughly researched findings, they usually concentrate on specific aspects or best practices that should be taken into consideration, but do not combine these into a systematic procedure. Structured methods, in contrast, provide the designer with detailed work steps and arrange them into a service identification process. They also provide clearly specified identification criteria. Algorithmic procedures finally comprise a formal plan that combines individual work steps into a comprehensive work-flow.

3.1.3 Overall development process model

The identification of services is usually part of a software development project which is guided by a development process model. This process model defines the in- and output of major development phases (such as design, implementation etc.) and coordinates the usage of achieved results in subsequent phases ([ChDa01], [SoWi99]).

Service identification approaches should ideally be integrated into an overall process model. Such an integration predefines which development phase has to deliver the information taken as input and how identified services have to be described to be useful as input for subsequent phases of the development process. Renouncing an integration into an overall process model carries the danger that results of earlier phases have only limited value for subsequent phases.

3.2 Procedure

The procedure describes what kind of technique to identify services is applied by an approach. The following aspects are taken into account:

3.2.1 Direction

The analysis to identify services can generally be carried out in two directions. Top-down approaches [Mill71] use domain-specific conceptual models (like business concept and process models) to identify services, which are then specified and mapped onto a software landscape. In contrast, bottom-up approaches start by analyzing the existing software landscape and modularizing it. Identified modules of this landscape will then be equipped with meaningful domain-specific semantics after the identification.

Since unidirectional top-down as well as bottom-up approaches carry the risk of leading to undesirable results, e.g. by identifying services that might not be suitable from the opposite technical or business-oriented viewpoint, some approaches try to combine both directions and strive for a compromise between a domain- and a technology-centric view. These will be distinguished as meet-in-the-middle approaches in the following.

3.2.2 Optimization approach

An important characteristic of service identification approaches is whether they try to find an optimized solution, i.e. services with presumably optimal properties in terms of the designer’s preferences. Optimizing approaches e.g. often try to identify a set of services with maximal cohesion and minimal dependencies, following a principle already stated by Parnas [Parn72]. Such preferences can be translated into mathematical optimization problems and then be approached with appropriate techniques (e.g. clustering methods). Thereby, one has to distinguish between approaches that use exact or heuristic methods. Exact methods find the overall best solution (a global optimum), while heuristics come up with the best solution that can be found with reasonable effort (a local optimum).

3.3 Model

Generally, the identification of services is based on conceptual models which reflect reality. From a theoretical perspective, a systematic approach for the identification of services should, at least partly, use the model views introduced by general systems theory [Bert76], which can also be applied to information systems [YoCo79]. With respect to the content and complexity of the utilized models, the examined approaches diverge significantly. Differences become apparent regarding the analyzed model views, the consideration of legacy structures and system dependencies as well as a differentiation of service hierarchies and predefined service types.

3.3.1 Model views

Independent of the question whether a domain-oriented or technical perspective is being used, socio-technical systems in general and software systems in particular can always be described from three model views, which stem from general systems theory and are widely used in software design methods such as Syntropy, Catalysis, or ARIS ([CoDa94], [SoWi99], [Sche00]): The data view describes processed information objects as well as their respective structure as system attributes. The functions view documents the system behaviour and combines system attributes as inputs and outputs. In addition, a functional decomposition describes the relationship between complex functions and their sub-functions. The process view finally describes the temporal relationships between functions and combines them to workflows.

Basically, the identification of services can take all three model views into account, since only their synopsis provides a comprehensive view. Many approaches, however, only use a subset of these model views, which leads to specific advantages and drawbacks.

3.3.2 Consideration of existing structures

The identification of services is often performed in an existent software environment with legacy systems or services in place. Therefore it has to be evaluated if the respective approaches consider existing structures appropriately and weave them into their procedure.

3.3.3 Consideration of system dependencies

Services normally provide their functionality only in cooperation with other services. They are consequently developed to be interconnected with other services [SzGr+02]. Service identification approaches might therefore aim at finding sets of collaborating services with thoroughly analyzed inter-dependencies and explicitly specify the remaining interdependencies with peripheral systems. Others instead concentrate on identifying single services and disregard potential dependencies with the environment.

3.3.4 Differentiation of service hierarchies

During the identification, one can generally distinguish between complex services, which themselves are composed of services, and elementary services, which are not to be further divided into smaller services. Identification approaches which explicitly support such a distinction follow the hierarchical systems concept of general systems theory [Bert76] and implement a stepwise decomposition until no complex services are identified anymore [AtBa+01]. Others do not explicitly support a stepwise decomposition and leave the structuring of a composition into a hierarchy to the designer.

3.3.5 Differentiation of predefined service types

Some of the service identification approaches distinguish services of predefined types [BeKn+08]. Often, these approaches differentiate between services whose primary purpose is the management of data (Entity Services) and those who coordinate and execute application-specific tasks (Task Services). Such an identification procedure inherently leads to a separation of data- and task-specific services.

It is debatable, however, if such procedures deliver an optimal result, especially since many authors argue for a grouping of data and related tasks into a single part [Parn72]. Other approaches, therefore, do not build upon a distinction of predefined service types.

3.4 Supporting measures

Supporting measures enhance the applicability of service identification approaches in practice. They can be classified into tool support, quality assertions, and evaluation.

3.4.1 Tool support

The practical applicability of service identification approaches can be enhanced by providing software tools which guide the designer through the identification process and help to manage complexity. The absence of such supporting tools hampers especially a possible optimization of service identification results.

3.4.2 Quality assertions

The quality of an identification result produced by a certain approach has a significant impact, since the implementation and roll-out of new software artefacts is always associated with considerable strategic and financial risks.

Therefore, a service identification approach is ideally able to guarantee the correctness of its result. Especially for an algorithmic procedure it is important to avoid local optima. If a formal guarantee is not feasible, e.g. due to method-inherent restrictions as in the case of heuristic procedures, approaches should at least support other kinds of quality assertions. They could for example state the maximum deviation from an optimal solution through an upper and lower bound approximation [Jung08] or allow a sensitivity analysis.

3.4.3 Evaluation

The evaluation of service identification approaches ensures their correctness and proofs their applicability in practice. While a plausibility check demonstrates the principal correctness, only comprehensive use cases and best practices reveal terms of use as well as possible areas of application and limitations of a certain approach. Ideally, an identification approach is evaluated by multiple applications in practice and complemented with “Best Practices”, which help to ease its application.

3.5 Classification Scheme

The previously mentioned distinguishing factors are the basis to form a classification scheme as depicted in Table 1. Thereby, values of the identified distinguishing factors have been summarized as a morphological box and will be used to classify individual identification approaches later on.

When looking at the classification scheme as a whole, one might suspect that the depicted distinguishing factors are not independent from each other. A bottom-up approach to identify services might, e.g., probably use a technical service definition. Similarly, an approach that uses matrices to analyze relationships between design elements as part of its identification strategy might probably do this in a formalized (algorithmic) procedure.

Classification Scheme for Service Identification Approaches				
Service definition	None	Technical	Domain-oriented	
Degree of formalization	Ad-Hoc	General Recommendations	Structured	Algorithm
Development process model	No		Yes	
Direction	Top Down	Bottom Up	Meet In The Middle	
Optimizing approach	None	Heuristic	Exact	
Model views	Data	Functions	Processes	
Consideration of existing structures	No	Yes		
Consideration of system dependencies	No	Yes		
Differentiation of service hierarchies	No	Yes		
Differentiation of predefined service types	No	Yes		
Tool support	No	Yes		
Quality assertions	None	Sensitivity analysis	Quality guarantee	
Evaluation	None	Plausibility check	Use Case	Best Practices

Table 1: Classification scheme for service identification approaches

When analyzing the distinguishing factors closely, it becomes obvious that they are orthogonal to each other, however. Accordingly, the apparent coincidences described above can easily be proven to be wrong: first of all, it is quite conceivable that a bottom-up approach might also use a domain-oriented service definition. Such an approach will start to identify services by analyzing conceptual models of an existing software landscape. Services will accordingly be identified from existing systems by analyzing them from a domain-oriented perspective and mapping results back onto the existing software landscape. In the same manner, identification approaches might as well use matrices to analyze relationships between design elements, but not conduct the analysis in an algorithmic procedure.

4 Classification of Service Identification Approaches

In this section, we provide an overview of the state of the art in service identification, and elaborate on various approaches published in literature. After introducing specialized service identification approaches in section 4.1, we extend the compilation in section 4.2 with general modularization approaches. The examined approaches will then be compared and classified

in section 4.3 according to the previously defined criteria. Section 4.4 concludes with important implications that can be drawn for practice and academia.

4.1 Service identification approaches

4.1.1 Service-oriented Analysis and Design (Zimmermann et al. and Arsanjani)

Zimmermann et al. examine the applicability and transferability of established software engineering methods to the introduction of SOAs [ZiKr+04]. Elements of Object-Oriented Analysis and Design (OOAD), Enterprise Architecture (EA) Frameworks and Business Process Modeling (BPM) are combined and expanded to form a Service-Oriented Analysis and Design (SOAD) approach. Although they define quality factors for SOAD and give general recommendations for all phases of the adoption process, they do not present an overall process model. While most of the mentioned model criteria of section 3.3 are addressed, both a service definition and concrete recommendations are missing.

In regard to service identification, it is pointed out that a SOA is usually not introduced in a greenfield approach. Therefore, a pure top-down approach would not be sufficient as existing structures have to be taken into account. The poor applicability of classical development methods to identify services is commented by the authors with “there is room for additional creative thinking.” [ZiKr+04], but they do not reveal any alternatives. The presented theoretical example, which is used for demonstration purposes, underlines the recommendatory character of their contribution, which can be used as starting point for further research.

Based on the SOAD-approach of Zimmermann et al. Arsanjani [Arsa04] articulates concrete recommendations for the identification, specification and realization of services. Following a mainly technical perspective, especially the identification phase is concretized. The execution of top-down and bottom-up approaches is extended with a goal-service-modeling in order to find yet unidentified, but needed services. This mainly theoretical approach without any reference examples again does not exceed the state of a loose collection of general advices.

4.1.2 Service-Oriented Analysis (Erl)

In his books about the conception and design of service-oriented architectures, Erl describes an approach for the identification of services in the context of an overall development model ([Erl05], [Erl07]). Based on an analysis of business processes and existing system structures, service candidates are identified, which can then be refined into services. In a meet-in-the-middle approach Erl differentiates eleven service hierarchies and classes, which sometimes depart from his general, rather technical definition of services. Dependencies between services are only mentioned aside. Tool support as well as quality assertions are not discussed. The approach is part of an overall development process model, but it only

offers recommendations and general instructions instead of explicit procedures. The application of the overall approach is however illustrated in case studies.

4.1.3 Enterprise Service Design Guide (SAP)

In 2005, the SAP AG published an Enterprise Service Design Guide in the context of its SAP Developer Network [SAP05]. Besides basics of enterprise services, this guide also comprises the discovery and design phases. Based on a rather technical definition of services, it describes how services can be identified on two hierarchical levels following a meet-in-the-middle approach. The approach offers 16 different indicators in order to help designers of Enterprise Services identifying potential services based on business processes and associated scenarios. The subsequent division into simple and composite services is supported by 10 guidelines. With this document, the authors offer a manual for the identification of services. The application of the approach is left to the designer, though.

4.1.4 EA Builder (Aier)

Aier presents an approach for the identification of an enterprise architecture and its related services based on business processes and IT-systems [Aier06]. Different architectural views are mapped onto graphs and then partitioned based on a clustering algorithm, which was initially developed for the identification of communities within social networks. The underlying service definition and many other details of the meet-in-the-middle approach remain unclear, however. While a differentiation of service hierarchies and types is mentioned, its realization in the supporting tool, the EA-Builder, is not further addressed. A quality assertion of the results is missing, but the approach has been tested on the basis of a use case.

4.1.5 SOA Framework (Erradi et al.)

The identification of services, as part of an architectural SOA Framework (SOAF), which covers the specification and realization phases, is presented by Erradi et al. [ErAn+06]. Based on an analysis of business processes, needed services are identified in a top-down approach. Existing services are extracted from the code base and the related data structures. By comparing needed and existing services, additionally required services are identified. A so-called tool-based mining supports the bottom-up analysis of code and data fragments. The top-down analysis of the business processes is realized by a combination of interviews and tools. The approach does not present a concrete definition of services and, besides notes about possible tool support, no tools are mentioned nor does it offer a procedure for matching needed with existing services. While the authors present a case study, their explanations only cover central results and do not document the practical application of the approach at all.

4.1.6 BPM and SOA Handshake (Inaganti and Behara)

A structured approach for the identification of services is offered by Inaganti and Behara [InBe07]. Potential services are identified and opposed to each other in four steps using a top-down as well as a bottom-up approach. Additionally needed services are added in an undefined way. While this contribution describes the identification in a more structured and detailed manner than Zimmerman et al. [ZiKr+04] and Arsanjani [Arsa04], it rarely exceeds the level of a listing of possibilities and recommendations. Optimization methods as well as tool support are not considered. A reference example is not given either.

4.1.7 Identification and design of services (Winkler)

Winkler presents an approach which covers service identification as well as the design and realization of services [Wink07]. During the identification phase services are defined based on UML activity diagrams. These services have to comply with three previously defined criteria, namely reusability of services, avoidance of redundant implementation of different services as well as loose coupling of services based on well-defined and simple interfaces. The service identification itself has four subsequent steps: creation of activity diagrams, rework of activity diagrams, identification of services, and analysis of usage frequency. On the basis of an implicit business-oriented service definition, the service identification follows a semi-structured top-down approach. An optimization of determined services is not part of the identification phase and the compliance of the identified services with the previously defined criteria is not validated. While service hierarchies, service dependencies and structures are mentioned, it stays unclear how these aspects affect the service identification. The whole process is described on the basis of an example from the financial service sector. A supporting tool is not mentioned.

4.1.8 Method for the conception of SOA (Beverungen et al.)

Beverungen et al. [BeKn+08] compare different approaches for the development of SOAs. As a result, they offer an own approach, which covers the phases of service identification and specification and is integrated into an overall development process model. Services are identified through a top-down decomposition of business processes. Special attention is placed on an analysis of so-called transfer and visibility potentials of single process steps for business partners. While existing structures and services are taken into account during the identification phase, the dependencies between services are only considered in the specification phase. Service hierarchies are divided into two types, Process and Basic. A differentiation of service types is mentioned, but not integral part of the approach. Although a structured identification process is propagated, further details about such a process are missing. Moreover, neither a possible optimization nor a supporting tool is mentioned. A use-case demonstrates the practicability of the approach, but does not describe any details of the sub-steps.

4.2 General modularization approaches

4.2.1 Business systems planning (IBM)

IBM describes an approach to systematically decompose business information systems on the basis of business process and data models [IBM84]. The approach offers detailed steps and procedures to identify system modules by examining the relations between process activities and data objects in a matrix analysis. Based on a heuristic optimization procedure, the grouping of process activities is rearranged to minimize the number of shared data objects between the groups of activities. A quality assertion for the results of this optimization is not given and existing system structures cannot be incorporated into the presented approach. Furthermore, it remains unclear how the rearrangement of process activities should be conducted.

4.2.2 Modularity criteria (Szyperski et al.)

In his book about component-based software engineering, Szyperski introduces 15 modularity criteria which should ideally be satisfied by identified components and services [SzGr+02]. According to these criteria, services should not only be self-contained with respect to their functionality, but also be independently implementable, installable, and maintainable. In addition, they should be independent with respect to billing and handling of liability issues. While the criteria are formulated in detail, they do not exceed the level of general recommendations. A structured procedure with concrete work steps to guide the designer is missing completely. The approach can therefore only be characterized as a conceptual framework which might be used to validate identified services.

4.2.3 CompMaker (Jain et al.)

Jain et al. present an approach which originates from the domain of component-based application development [JaCh+01]. Based on the Analysis Level Object Model, a business domain model in UML notation, the approach identifies reusable components following a top-down approach. The domain model contains at least object-oriented class diagrams, use cases and sequence or interaction diagrams. Structural and dynamic relationships between the different objects in the domain model are used to compute the Class Relationship Strength.

In a first step these relationships are used to identify components through a grouping of classes by applying a hierarchical agglomerative clustering algorithm. This initial solution is then improved through automated add-, move- or exchange- heuristics, as well as manual interventions. While classes, as basic building blocks of components, play an important role for the approach, the final results are strongly influenced by the designer’s preferences as different measures for the best possible solution can be applied. The identification process is

supported by the CompMaker tool and illustrated on the basis of a case study from the automotive insurance sector. An evaluation of identification results is not mentioned.

4.2.4 BCI and BCI-3D (Albani et al.)

In ([AlDi06], [AlDi+05], [AlOv+08]), Albani et al. describe the Business Component Identification (BCI) method, as well as its enhancement to BCI-3D. Both versions identify components by using algorithms that work on process and data structures of a domain model. The component identification follows an algorithmic top-down approach, which is supported by specialized tools and part of the Business Component Modelling Process (BCMP). BCI in its original version [AlDi+05] is adapted from [IBM84] and considers only the dependencies between single functions and data objects. By contrast, BCI-3D ([AlDi06], [AlOv+08]) uses graph-based clustering methods, which identify components by combining an opening- and an improving-heuristic from graph theory. Information about data objects, process steps and actors, plus their relationships is mapped onto vertices and edges of a graph. Weights are then assigned to the edges depending on the relation type and the designer’s preferences. A quality assertion for the resulting solution is not given and legacy structures as well as existing dependencies can only be partly included by integrating them into the domain model. Case studies have been conducted for both methods ([AlDi06], [SeKl+05]). The BCI-3D tool supports the designer during the identification process, but missing advices for the assignment of weights hamper the application of the proposed method.

4.3 Classification and comparative discussion

The identification of services and modules has been an area of continuous research. Whereas all of the specialized service identification approaches were published between 2004 and 2008, the more general modularization approaches are mainly older and dated before 2004. An overview of the classification results of all approaches is summarized in Table 2. It shows that none of the 9 specialized and the 4 general approaches covers all aspects sufficiently, but rather all of them have their individual strengths and weaknesses. Below follows a differentiated examination and comparison of the approaches.

	Zimmermann et al. [ZiKr+04]	Arsanjani [Arsa04]	Erl ([Erl05], [Erl07])	SAP [SAP05]	Aier [Aier06]	Erradi et al. [ErAn+06]	Inaganti, Behara [InBe07]	Winkler [Wink07]	Beverungen et al. [BeKn+08]	IBM [IBM84]	Szyperski et al. [SzGr+02]	Jain et al. [JaCh+01]	Albani et al. ([AlDi+05], [AlDi06], [AlOv+08])
Year of Publication	2004	2004	2005, 2007	2005	2006	2006	2007	2007	2008	1984	1998, 2002	2001	2005, 2006, 2008
Service definition	None	Technical	Technical	Technical	None	Domain-oriented	None	Domain-oriented	None	n/a	n/a	n/a	n/a
Degree of formalization	General Recommendations	General Recommendations	General Recommendations	General Recommendations	Algorithm	Structured	General Recommendations	Structured	Structured	Structured	General Recommendations	Algorithm	Algorithm
Development process model	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✗	✓
Direction	Meet In The Middle	Meet In The Middle	Meet In The Middle	Meet In The Middle	Meet In The Middle	Meet In The Middle	Meet In The Middle	Top Down	Top Down	Top Down	n/a	Top Down	Top Down
Optimizing approach	✗	✗	✗	✗	✓ (heuristic)	✗	✗	✗	✗	✓ (heuristic)	✗	✓ (heuristic)	✓ (heuristic)
Model views	Processes and data	Processes, data and functions	Processes, data and functions	Processes, data and functions	Processes and data	Processes and data	Processes, data and functions	Processes and functions	Processes	Processes and data	n/a	Processes and functions	Processes and data
Consideration of existing structures	✓	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✗	✓
Consideration of system dependencies	✗	✓	✗	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓
Differentiation of service hierarchies	✗	✓ (2)	✓ *	✓ (2)	✗	✓ (2)	✓ (2)	✗	✓ (2)	n/a	n/a	n/a	n/a
Differentiation of predefined service types	✗	✗	✓ *	✓	✗	✗	✗	✗	✓ (2)	n/a	n/a	n/a	n/a
Tool support	✗	✗	✗	✗	✓	✓ **	✗	✗	✗	✗	✗	✓	✓
Quality assertions	None	None	None	None	None	None	None	None	None	None	None	None	None
Evaluation	None	None	Plausibility check	None	Use Case	Plausibility check	None	Use Case	Use Case	Use Case	None	Use Case	Use Case

* 11 Service types, ** only sub-steps

Table 2: Classification of service identification approaches.

4.3.1 Foundations

The underlying service definitions vary heavily between the approaches. More technical definitions are used in three cases ([Arsa04], [Erl05], [Erl07], [SAP05]) and two approaches are built upon domain-oriented definitions ([ErAn+06], [Wink07]). However, an interesting observation is that the authors of eight approaches identify services without any definition of what they try to find. This is understandable for the four general approaches not aiming specifically at service identification, but even four of the explicit service identification approaches do not provide any definition ([Aier06], [BeKn+08], [InBe07], [ZiKr+04]). Furthermore, even if a definition is given, it is oftentimes imprecise and therefore handicaps a comparison of the approaches.

The degree of formalization is in four cases structured ([BeKn+08], [ErAn+06], [IBM84], [Wink07]) and an algorithm is provided for three approaches ([Aier06], [AlOv+08], [JaCh+01]). In the remaining six cases general recommendations are given. It is noticeable that the older, general modularization approaches are overall more formalized than those originating from the newer SOA discipline. On the other hand, most of the newer approaches are embedded in a development process model and thus better support an integrated design than the older ones.

Generally, the amount of information and sub-steps explained varies noticeably in the evaluated literature, ranging from detailed step-by-step instructions to rather coarse-grained explanations. A low level of detail especially hampers the applicability of an approach. Interestingly, the level of detail is not correlated with the degree of formalization, which one might have expected.

4.3.2 Procedures

The direction of the analysis to identify services is meet-in-the-middle for most (seven) cases, whereas only two of the specialized approaches ([BeKn+08], [Wink07]) and three of the general approaches ([AlOv+08], [IBM84], [JaCh+01]) follow a top-down course. Also, not a single one of the approaches uses a bottom-up strategy. Generally, it seems like the specific service identification approaches tend to consider technical information more often in combination with business domain information than the more general modularization techniques which solely rely on the business domain.

One of the explicit service identification methods uses a heuristic to optimize the results [Aier06], whereas the other eight do not implement any optimization. The general approaches on module identification are more advanced in this category, as three of the four methods try to optimize the identified service structure using a heuristic ([AlOv+08], [IBM84], [JaCh+01]).

4.3.3 Model

All of the strategies are based on a process view of the model. Oftentimes this is completed by additionally considering information on functions and/or data. Existing structures and system dependencies are each taken into account by nine of the approaches. This is usually granted for the meet-in-the-middle approaches, but for the top-down approaches it can only be achieved through an integration of the information into the domain model. Two third of the proposed techniques for service identification care for service hierarchies and include corresponding arrangements in their strategy. The differentiation of predefined service types is covered in one third of the papers. Service hierarchies and service types are not applicable for the non-service-specific approaches.

4.3.4 Supporting measures

A support of the proposed service identification approaches through corresponding tools is only mentioned in two of the service-specific ([Aier06], [ErAn+06]) and two of the general modularization approaches ([AlOv+08], [JaCh+01]). Quality assertions are so far completely missing for all of the approaches. For the evaluation of the approaches, use cases and plausibility checks are provided in eight cases, but none is evaluated through best practices. Overall, this does not go far enough to ensure a high quality solution. However, this would be crucial for the further development of the procedures.

4.3.5 Implications

Several implications can be derived from the identified state of the art for researchers in the field of service identification methods, as well as for software engineers of service-oriented software systems. The former ones can use the results from Table 2 to identify areas requiring further research, improve their own approaches and fill out the blanks. To improve the usability of several methods, the analysis of the supporting measures shows that software tools are needed, especially in the case of optimizing approaches. Furthermore, the given quality assertions and evaluations are mostly quite rudimental. Additionally, researchers might be able to use combinations of existing service-specific and general approaches for further improvements of the state of the art in service identification. For example, we see a good chance that the BCI approach from Albani et al. [AlOv+08] might be successfully combined with the technique from Aier [Aier06], since the former one identifies components from a domain model through graph-based clustering methods and the latter one elaborates on algorithms derived from social-networks.

A software engineer can use the provided comparison of approaches to select one that is most appropriate for his/her particular development scenario. Above all, the utilized service definition, the direction of the approach as well as the required model views provide insights whether an approach is suited to support a particular development scenario or not. In a greenfield software development project, where services can be identified during the early

design phases and without taking existing software systems into account, top-down approaches which use a domain-oriented service definition should be chosen. But in general the decision for a specific approach should be depending upon whether the overall goal during the identification is to identify reusable services or to primarily create a modular system design.

In a scenario where existing software systems have to be modularized or at least to be integrated into the identification of services, meet-in-the-middle approaches should be preferred. These approaches either start from existing software systems and aggregate implementation classes to form services or at least take existing software structures into account during the identification of services. As the classification shows, an integrated service identification approach, which combines the strengths of the mentioned approaches and is able to cover all depicted scenarios, is not available so far. Therefore, it currently depends on the knowledge of the designer, if a suitable approach is chosen and useful results can be achieved. Our paper thus provides useful insights by identifying and detailing on the state of the art.

By analyzing the classified approaches from a chronological perspective, it becomes furthermore obvious that the approaches related to the older discipline of component-based application development usually possess more structured, algorithm-based and better evaluated procedures compared to the approaches that specifically support the identification of services. It also has to be highlighted that most of the service-specific identification approaches come without an explicit service definition. For the designer, it therefore often remains unclear which criteria of services are assumed and guaranteed by the approaches during the identification process. In conclusion, service-specific approaches hence appear to be in a comparatively premature state. Additional research effort is therefore required to further advance the state of the art and support an engineering approach to identify services.

5 Conclusions and future directions

In this paper, we compared several approaches for the identification of services, which have been published in literature, and discussed their individual strength and weaknesses. The discussion was based on a classification scheme that contains various characteristics of service identification approaches as distinguishing factors and has been specifically developed to compare existent as well as future developments. The assembled characteristics were initially based on results from research focusing on the conception of systematic design methods in general and then refined to characterize service identification approaches. We used the resulting classification scheme to compare various service identification approaches and to reveal differences in their conceptual design as discussed in section 4.3.

It has to be mentioned that the different approaches vary in respect to their consideration of dependencies between services as well as different service hierarchies and types. A stepwise refinement, which might become necessary during the service identification process, is therefore not supported by all approaches. Approaches distinguishing so-called Entity (Data) and Task (Process) Services are likely to promote different solutions than those which do not build upon such a predefined distinction of service types. This distinction has to be taken into account when a certain approach is selected.

Significant further research is necessary to answer the question if the existing approaches for service identification can be further developed into mature methods with more formalized and detailed procedures. To realize the aspired systematic service identification as part of an engineering process, existing approaches will eventually have to be enhanced in various aspects. In this context, especially the usage of optimization methods and procedures, which allow at least an estimation of the solution quality, has to be considered.

It appears to be characteristic that the identification of services does not build upon existing, more mature approaches from the closely related (and older) component-based software engineering discipline. Instead, it seems as if research has started anew with the introduction of SOAs. In fact, this course of action can be observed in many areas of the SOA discipline and sometimes is therefore rightfully criticized in literature [SzGr+02]. A more detailed examination reveals that many modularization approaches, which were developed to identify business components, as defined by Cheesman and Daniels [ChDa01], could well be used for the design of service-oriented architectures. A synergetic examination of these two disciplines could hence significantly accelerate the development of mature methods for service identification.

References

[Aier06] Aier, Stephan: How Clustering Enterprise Architectures helps to Design Service Oriented Architectures. Proceedings of IEEE International Conference on Services Computing (SCC 2006). 2006, pp. 269–272.

[AlDi06] Albani, Antonia; Dietz, Jan Leonardus Gerardus: The Benefit of Enterprise Ontology in Identifying Business Components. Proceedings of IFIP World Computing Conference. 2006, pp. 243-254.

[AlDi+05] Albani, Antonia; Dietz, Jan Leonardus Gerardus; Zaha, Johannes Maria: Identifying Business Components on the basis of an Enterprise Ontology. In: Konstantas, D.; Bourrieres, J.-P.; Leonard, M.; Boudjlida, N. (Eds.): Interoperability of Enterprise Software and Applications. Springer Verlag, Geneva, Switzerland, 2005, pp. 335–347.

[AlOv+08] Albani, Antonia; Overhage, Sven; Birkmeier, Dominik: Towards a Systematic Method for Identifying Business Components. In: Chaudron, Michel R.V.; Szyperski, Clemens A.; Reussner, Ralf (Eds.): Proceedings of Component-Based Software Engineering, 11th International Symposium, CBSE 2008. 2008, Lecture Notes in Computer Science, pp. 262-277.

[Alte08] Alter, Steven: Seeking Synergies Between Four Views of Service in the IS Field. Proceedings of 14th Americas Conference on Information Systems (AMCIS 2008). 2008.

[Arsa04] Arsanjani, Ali: Service-oriented modeling and architecture - How to identify, specify, and realize services for your SOA. IBM developerWorks Web services zone, http://www.ibm.com/developerworks/ webservices/library/ws-soa-design1/, (08.10.2008).

[AtBa+01] Atkinson, Colin; Bayer, Joachim; Bunse, Chris-tian; Kamsties, Erik; Laitenberger, Oliver; Lagua, Roland; Muthig, Dirk; Paech, Barbara; Wust, Jürgen; Zettel, Jorg: Component-Based Product Line Engineering with UML. Addison-Wesley, 2001.

[BaDu06] Barros, Alistair P.; Dumas, Marlon: The Rise of Web Service Ecosystems. In: IT Professional 8 (2006) 5, pp. 31-37.

[Bert76] Bertalanffy, Ludwig Von: General System Theory: Foundations, Development, Applications. George Braziller, New York, 1976.

[BeKn+08] Beverungen, Daniel; Knackstedt, Ralf; Müller, Oliver: Entwicklung Serviceorientierter Architekturen zur Integration von Produktion und Dienstleistung - Eine Konzeptionsmethode und ihre Anwendung am Beispiel des Recyclings elektronischer Gerate. In: Wirtschaftsinformatik 50 (2008) 3, pp. 220-234.

[Brow00] Brown, Alan W.: Large-Scale, Component-Based Development. Prentice Hall, Upper Saddle River, NJ, 2000.

[ChDa01] Cheesman, John; Daniels, John: UML Components. A Simple Process for Specifying Component-Based Software. Addison-Wesley, Upper Saddle River, NJ, 2001.

[ChSp06] Chesbrough, Henry; Spohrer, Jim: A research manifesto for services science. In: Communications of the ACM 49 (2006) 7, pp. 35 - 40.

[CoDa94] Cook, Steven; Daniels, John: Designing Object Systems. Object-Oriented Modelling with Syntropy. Prentice Hall, Englewood Cliffs, NJ, 1994.

[SoWi99] D'Souza, Desmond Francis; Wills, Alan Cameron: Objects, Components, and Frameworks with UML. The Catalysis Approach. Addison-Wesley, Upper Saddle River, NJ, 1999.

[Erl05] Erl, Thomas: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[Erl07] Erl, Thomas: SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl). Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.

[ErAn+06] Erradi, Abdelkarim; Anand, Sriram; Kulkarni, Naveen: SOAF: An Architectural Framework for Service Definition and Realization. Proceedings of IEEE International Conference on Services Computing 2006 (SCC '06). 2006, pp. 151-158.

[IBM84] IBM, Corporation: Business Systems Planning: Information Systems Planning Guide. Technical Report Nr. GE20-0527-4. International Business Machines Corporation, 1984.

[InBe07] Inaganti, Srikanth; Behara, Gopala Krishna: Service Identification: BPM and SOA Handshake. BPTrends, 2007.

[JaCh+01] Jain, Hemant; Chalimeda, Naresh; Ivaturi, Navin; Reddy, Balarama: Business Component Identification - A Formal Approach. Proceedings of 5th IEEE International Conference on Enterprise Distributed Object Computing (EDOC '01). 2001, pp. 183 -187.

[Jung08] Jungnickel, Dieter: Graphs, networks, and algorithms. 3. Ed., Springer, Berlin, 2008.

[KoLe04] Kossmann, Donald; Leymann, Frank: Web Services. In: Informatik Spektrum 26 (2004) 2, pp. 117-128.

[LoLe+99] Lovelock, Christopher H.; Lewis, Barbara; Vandermerwe, Sandra: Services Marketing: European Perspectives. Prentice Hall, London, 1999.

[McSi+06] Mcgovern, James; Sims, Oliver; Jain, Ashish; Little, Mark: Enterprise Service-Oriented Architectures: Concepts, Challenges, Recommendations. Springer, 2006.

[Mill71] Mills, Harlan D.: Top-down programming in large systems. In: Ruskin, R. (Eds.): Debugging Techniques in Large Systems. Prentice Hall, 1971, pp. 41-55.

[Nati03] Natis, Yefim V.: Service-Oriented Architecture Scenario. Research Report Nr. AV-19-6751. Gartner Research, 2003.

[PaBe+07 Pahl, Gerhard; Beitz, Wolfgang; Feldhusen, Jörg; Grote, Karl-Heinrich: Engineering Design: A Systematic Approach. Springer, Berlin, Heidelberg, 2007.

[PaTr+06] Papazoglou, Mike; Traverso, Paolo; Dustdar, Schahram; Leymann, Frank; Krämer, Bernd: Service-Oriented Computing: A Research Roadmap. Proceedings of Dagstuhl Seminar. 2006.

[Parn72] Parnas, David L.: On the Criteria to be Used in Decomposing Systems into Modules. In: Communications of the ACM 15 (1972) 12, pp. 1053-1058.

[SAP05] SAP: Enterprise Services Architecture: Enterprise Services Design Guide. SAP AG, 2005.

[Sche00] Scheer, August-Wilhelm: ARIS - Business Process Modeling. 3. Ed., Springer, Berlin, 2000.

[SeKl+05] Selk, Bernhard; Kloeckner, Sebastian; Bazijanec, Bettina; Albani, Antonia: Experience Report: Appropriateness of the BCI-Method for Identifying Business Components in large-scale Information Systems. In: Turowski, Klaus & Zaha, Johannes Maria (Eds.): Proceedings of Component-Oriented Enterprise Applications (COEA 2005). 2005, Lecture Notes in Informatics, pp. 87-92.

[Somm06] Sommerville, Ian: Software Engineering. 8. Ed., Addison-Wesley, 2006.

[SzGr+02] Szyperski, Clemens; Gruntz, Dominik; Murer, Stephan: Component Software. Beyond Object-Oriented Programming. 2. Ed., Addison-Wesley, Harlow, 2002.

[WaXu+05] Wang, Zhongjie; Xu, Xiaofei; Zhan, Dechen: A Survey of Business Component Identification Methods and Related Techniques. In: International Journal of Information Technology 2 (2005) 4, pp. 229-238.

[WeWa02] Webster, Jane; Watson, Richard T.: Analyzing the past to prepare for the future: Writing a literature review. In: MIS Quarterly 26 (2002) 2, pp. xiii-xxiii.

[Wink07] Winkler, Veronica: Identifikation und Gestaltung von Services - Vorgehen und beispielhafte Anwendung im Finanzdienstleistungsbereich. In: Wirtschaftsinformatik 49 (2007) 4, pp. 257-266.

[YoCo79] Yourdon, Edward; Constantine, Larry L.: Structured Design: Fundamentals of a Discipline of Computer Program and System Design. Prentice-Hall, Upper Saddle River, NJ, USA, 1979.

[ZiKr+04] Zimmermann, Olaf; Krogdahl, Pal; Gee, Clive: Elements of Service-Oriented Analysis and Design - An interdisciplinary modeling approach for SOA projects. IBM developerWorks Web services zone, http://www.ibm.com/developerworks /library/ws-soad1/, (15.07.2008).

III Beiträge zu Strukturen und Architekturen betrieblicher Anwendungen in agilen Umgebungen

Die Gestaltung des Untersuchungsgegenstandes ist ein zweite wichtige Hauptaufgabe der Wirtschaftsinformatik, wobei insbesondere die deutschsprachige Wirtschaftsinformatik zu konstruktionsorientierten Methoden und praxisorientierten Arbeiten zur Gewinnung und Validierung von Kenntnissen wie beispielsweise dem Erstellen und Evaluieren von Prototypen neigt.

Im Rahmen des Kapitels III sollen daher konzeptionelle als auch prototypische Gestaltungsvorschläge präsentiert werden, die eine sinnhafte Vollautomation voran treiben könnten. Unterkapitel III.1 stellt den Beitrag „Enabling Interoperability of Networked Enterprises Through an Integrative Information System Architecture for CRM and SCM" vor. Dieser präsentiert eine integrierte und komponentenbasierte Informationssysteminfrastruktur für das CRM und SCM und stellt dar, wie diese in den einzelnen Funktionsbereichen bzw. Szenarien ausgestaltet sein müsste.
Darüber hinaus wird in Unterkapitel III.2 der Beitrag „Something is Missing: Enterprise Architecture from a Systems Theory Perspective" vorgestellt. In diesem Beitrag werden die bestehenden Konzepte im Bereich der Unternehmensarchitekturen unter Verwendung der Systemtheorie der Technik genauer auf bisher möglicherweise nicht berücksichtigte Aspekte untersucht. Im Verlauf dieser Untersuchung hat sich insbesondere gezeigt, dass menschliche Handlungsträger in den bestehenden Konzepten bisher nicht ausreichend berücksichtigt werden.
Unterkapitel III.3 präsentiert mit dem Beitrag „FAST ACCESS: A system architecture for RESTful Business Data" schließlich eine neuartige Architektur auf Basis der REST-Prinzipien zur Integration verteilter Unternehmensdaten. Hierbei werden zunächst die bestehenden Probleme bei der Integration und unter Verwendung klassischer Web Service sowie die Eigenschaften REST-basierter Architekturen aufgezeigt. Auf Basis dieser Merkmale wird dann ein entsprechendes Architekturkonzept zur Integration verteilter Unternehmensdaten abgeleitet.

III.1 Beitrag: „Enabling Interoperability of Networked Enterprises Through an Integrative Information System Architecture for CRM and SCM“

Autoren: Bernhard Selk, Sebastian Klöckner, Antonia Albani

Bernhard Selk, Sebastian Klöckner

Beide Lehrstuhl WI-SE, Universität Augsburg,

Universitätsstraße 16, D-86135 Augsburg,

Email: bernhard.selk@wiwi.uni-augsburg.de

sebastian.kloeckner@wiwi.uni-augsburg.de

Antonia Albani

Chair of Software Engineering

Delft University of Technology

PO Box 5031, 2600 GA Delft, The Netherlands

Email: a.albani@ewi.tudelft.nl

Erschienen in: Bussler, C.; Haller, A. et al. (Hrsg.): Business Process Management Workshops: BPM 2005 International Workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS. Springer Lecture Notes in Computer Science (LNCS) 3812. Springer, Berlin, Heidelberg 2005, S.305-316 (2005).

Mit dem Einsatz spezialisierter Anwendungssysteme, bspw. im Customer Relationship Management (CRM) oder im Supply Chain Management (SCM), nimmt die Komplexität der Beziehungen zwischen Unternehmen kontinuierlich zu. Gleichzeitig steigt die Notwendigkeit, Informationen zwischen Unternehmen, die Teil von Wertschöpfungsnetzen sind, auszutauschen. Diese notwendige Kopplung wird jedoch häufig dadurch behindert, dass sowohl Datenstrukturen als auch Funktionen schwierig zu integrieren sind. Der Einsatz einer integrierten Informationssystemarchitektur (ISA) würde in diesem Fall die inter-organisationale Integration deutlich vereinfachen und gleichzeitig die Probleme des Datenmanagements erheblich reduzieren. Dieser Beitrag stellt daher auf Basis zweier Beispiele dar, wie eine integrierte Informationssystemarchitektur für das CRM und das SCM die interorganisationale Integration unterstützen und den kontinuierliche Austausch von Daten zwischen Unternehmen eines Wertschöpfungsnetzes ermöglichen kann.

1 Introduction

Innovations in information and communication technologies, primarily the emergence of the Internet, combined with drastic changes in the competitive landscape (e.g. globalization of sales- and sourcing-markets, shortened product lifecycles, innovative pressure on processes), shifted managerial attention towards the use of information technologies to increase flexibility of the business system and to improve intercompany collaboration in value networks, often referred to as inter-organizational systems (IOS), e-collaboration and collaborative commerce [1, 2].

In order to support not only intra- but also inter-organizational business processes along the value network, the systems used in the single network nodes need to be integrated. This implies that the IT application systems of customers, partners and suppliers need to be integrated into an inter-organizational system in order to allow automated data interchange in the value network. The integration includes different functional areas like service, marketing, sales, procurement, production, distribution and waste disposal. Only the integration of all these functional areas, which are directly involved in the value creation, allows a realization of a transparent and continuous supply network. Thanks to the deployment of enterprise resource planning (ERP) system internal integration is already on a high level and at the same time precondition for the realization of inter-organizational integration, which contains a potential by far larger than that of intra-organizational integration [3-5]. Inter-organizational integration in this context is defined as “(…) the ability of two or more systems or components to exchange information and to use the information that has been exchanged” [6].

Nonetheless, the realization of inter-organizational integration in the business world is not as far as science has explored it. This is mainly caused by the lack of standards and absence of adequate information system architectures, which contain the construction plan of the information system – in terms of a specification and documentation of its components as well as their relationships – as well as the rules of construction for the creation of such a construction plan [7].

As an inter-organizational integration has an enormous influence on the design of information systems and as preliminary and downstream enterprises in the supply network have to be integrated, the creation of an inter-organizational information system architecture is necessary in order to ensure a continuous exchange of information between the involved partners throughout the whole value network. But before such an inter-organizational integration becomes possible, a corresponding intra-organizational integration has to be achieved.

Inter-organizational integration is supported by systems like Customer Relationship Management (CRM) and Supply Chain Management (SCM) [8]. These two concepts cover most of the functional areas involved directly in value creation and are therefore adequate as basis for the development of an integrative information system architecture [9]. Additionally, SCM and CRM systems are usually the endpoints of the internal value chain. CRM creates the connection with the customer within the functional areas of service, sales and marketing, while SCM creates the connection with the suppliers, partners and customers through the functional areas procurement, production, logistics and waste disposal.

This paper aims to illustrate the adequacy of an integrative Information Systems Architecture (ISA) for CRM and SCM. In doing so the following research question will be answered: Why is an intra-organizational integration necessary in order to achieve inter-organizational integration? In what way does a business components-based information system architecture contribute to the interoperability in a value network? What types of business components are relevant for the interoperability and how do the components need to be composed in an inter-organizational system?

Therefore, in section 2 the state of the art of inter-organizational systems is illustrated, showing the necessity of integrating CRM and SCM in an enterprise in order to provide an adequate basis for the development of inter-organizational systems. In section 3 an integrative information systems architecture for CRM and SCM is derived in order to ensure interoperability. Section 4 explains the integrative architecture by means of two examples. Concluding remarks and future work can be found in section 5.

2 Inter-organizational information systems

By interconnecting the CRM-systems of the suppliers with the SCM-systems of the customers – allowing business partners to streamline and optimize for example their production processes, inventory management as well as their customer service – the collaboration between enterprises can be improved and the exchange of information between customer and supplier accelerated. As far as only the relationship between the supplier and the customer is taken into account this kind of e-collaboration is adequate. But as actual real-world examples show the limitation of focus to only one relationship in the whole supply network as area of interest is not sufficient. While the considered relationship between supplier and customer performs well, problems at preliminary stages of the supply network might cause extensive interruptions of production processes downstream.

Demand driven value networks [10, 11] do not just only take the relationship from the OEM to the subsequent tier into account, but the whole supply network with several tiers. As shown in Fig. 1, each supplier in the supply network is viewed as a node which knows its preliminary supplier and where each node checks his own preliminary supplier for its ability to deliver in

case that a downstream customer request for quotation arrives. The resulting information allows each supplier to judge if he and his preliminary suppliers are able to accomplish the potential customer order. But the assumption that every node is able to process an incoming request for quotation and convert it into outgoing requests for quotation for his own preliminary suppliers is not valid in most cases.

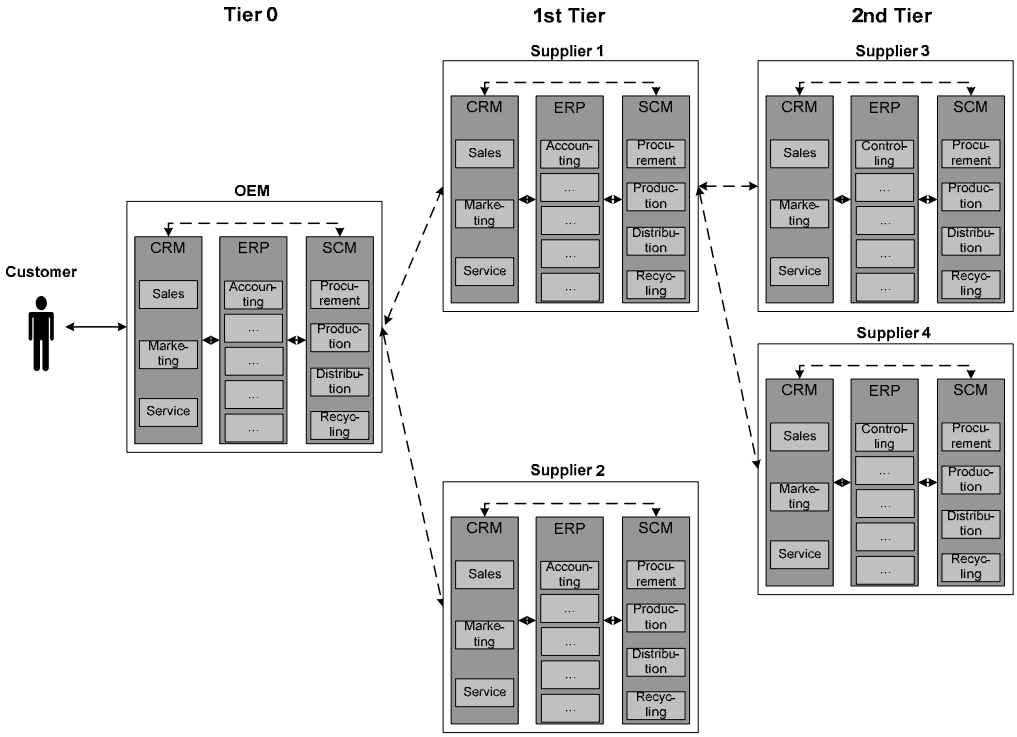

Fig. 1: Actual state of interconnection of internal and external systems

In fact, most often the internal systems of a company are coupled by standard interfaces or customized adapters. As the interfaces and adapters do not fit exactly with the coupled systems, the systems cannot communicate with each other correctly resulting in a loss of information or functionality. The loss of information can be caused by several reasons: first, as two or more independent systems exist, data is stored redundantly and concurrent updates are not guaranteed. As a result, the reliability and actuality of data cannot be assured making it impossible to determine if an order or request can be fulfilled or not. Second, as structures and semantics of data can vary between the systems, a matching of the ontologies can be difficult or sometimes even impossible [12-15]. The loss of functionality can also be caused by several reasons: first, a system does not offer a public interface or API for a certain function. Consequently, this function cannot be called by an external system. Second, even if a public interface or API is offered, the signature of the interface may not be suitable for another system as additional information would be needed. Accordingly, even if the function could be called by an external system, it would not operate correctly as not all input parameters may be available [16, 17].

The problems mentioned become even worth as some systems – especially SCM and CRM systems – are interconnected only through a third system – usually the ERP system – multiplying the problems of lost information and functionality and, in the worst case scenario, not allowing any exchange of information or utilization of functionality of the connected systems at all [18-20]. While a direct connection between a CRM system and a SCM system would be favorable, in most cases such a direct interconnection does not exist. Therefore, the data exchange between those systems inside an enterprise is very limited, while it would be the precondition for successful interconnection of the whole supply network. In order to solve this problem, direct connections between these systems have to be established. Direct interconnection does allow to reciprocally access the data and to use the functionality of the independent systems.

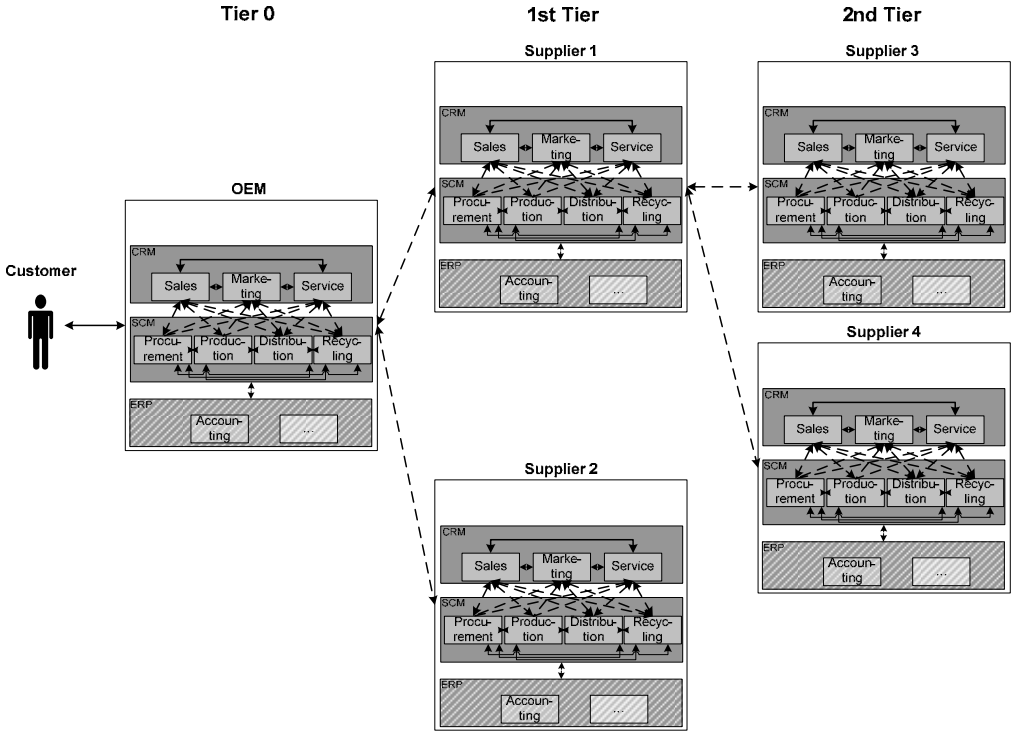

Fig. 2: Direct interconnection of CRM and SCM systems

As shown in Fig. 2, the precondition for such a direct interconnection is the availability of all important interfaces of an independent system to external systems.

The publication of interfaces of subsystems of a concerned system allows additional enhancement of the interconnection of the systems. Consequently, each subsystem could be interconnected making it possible to make use of further functionality. But even if all interfaces are published and available to external systems, the problem of redundant data, concurrent updates and different data structures still exists as both systems contain independent data management functionality. For solving this problem an integrated

information systems architecture for CRM and SCM, as the endpoints of the internal value chain, has to be developed [21].

3 Integrative information system architecture for CRM and SCM for ensuring interoperability

One possibility to ensure inter-organizational integration is, as described in section 2, the usage of an integrated information system architecture for CRM and SCM. Fig. 3 shows the information relationships between seven functional areas in terms of an integrated ISA. Subsequent to Fig. 3 the methodology used for the development of the integrated ISA will be illustrated.

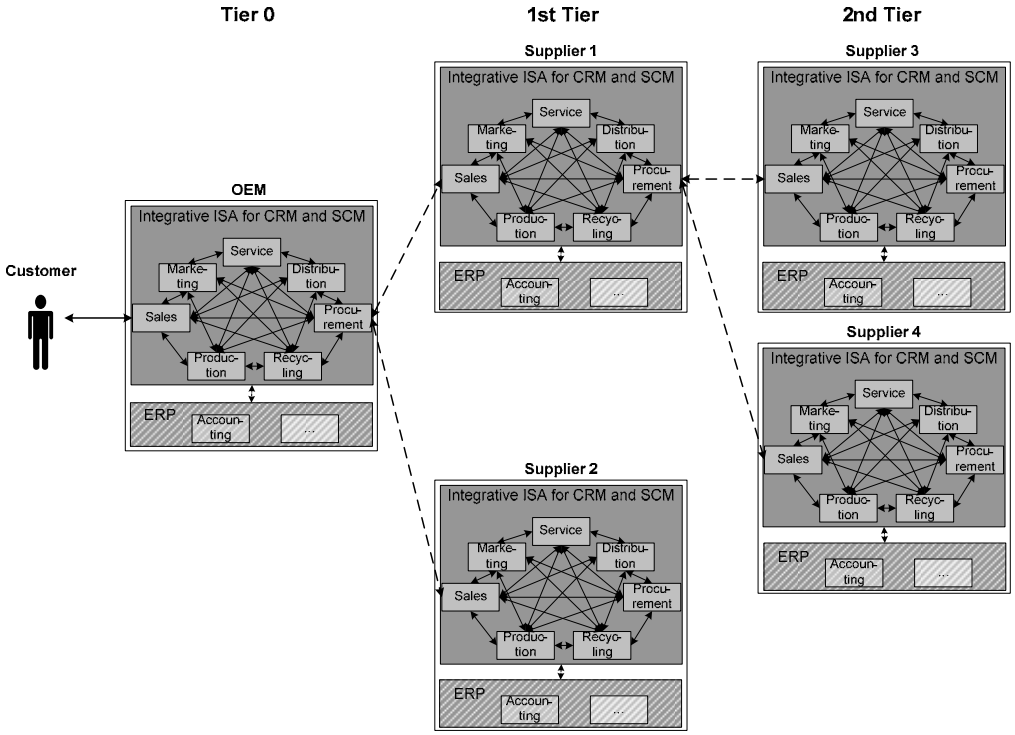

Fig. 3: Integrative information systems architecture for CRM and SCM

The integration of SCM and CRM systems, as shown in Fig. 3, allows direct interaction of the functional areas and usage of a coordinated data management, reducing the described problems of data consistency. Within the integration context some functions of one business area are assigned to a business component [22, 23] containing additional functions of different business areas and providing the functionality to the outside world over well defined services [23] following the idea of a service oriented architecture. Under an integrative perspective business components are no longer strictly predetermined by membership in a certain business area, but rather composed in a way that relationships of information objects are optimized. The goal of the composition is to maximize the exchange of information within a business component and minimize the exchange of information between the business components while simultaneously avoiding that the technical purpose of the business

component is affected negatively. The rearrangement of functionality does not only allow direct access to functions and information objects of other business areas. In addition, due to the incorporation in one integrative ISA, other problems like data redundancy or data matching issues are solved, because of a coordinated data management. Consequently, this integrative information system architecture allows the inter-organizational integration of organizations involved in the supply network and therefore a continuous exchange of information throughout the whole supply network.

Due to complexity reduction reasons the interconnection of the integrated ISA with other application systems, like an ERP-System as shown in Fig. 3, will not be explained here in depth. Nonetheless, an interconnection or integration of such a system can be achieved in the same way as it was shown for SCM and CRM. The ERP-Systems illustrated in Fig. 3 are rather for pointing out that additional interfaces for other systems are necessary.

For the development of the integrative information system architecture the Business Component Identification (BCI)-Method [24, 25] has been used, which is based upon the Business System Planning (BSP) [26] method and which has been modified for the field of business components identification. The basis for the BCI method is a well elaborated domain analysis. The BCI method takes as input the business tasks of a specific domain, as e.g. defined in the functional decomposition diagram, and the domain based data model, both obtained from the domain analysis. In a first step a matrix is built defining the relationships between the single business tasks and the informational data. The relationships are visualized inserting “C” and “U” in the matrix. “C” denotes that the data is created by the specific business task, and “U” denotes the usage of informational data by a given task. In changing the order of data and of business tasks according to some metrics defined – e.g. minimal communication between and maximal compactness of components – groups of relationships can be recognized [25]. These groups identify potential business components. If some “U”’s are outside of the groups, arrows are used to identify the data flow from one group to the other. The result of the BCI is an abstract business component model with some already defined dependencies between components [24, 27, 28].

Fig. 4 illustrates a subset of the identified business components of the integrated ISA for SCM and CRM. The components shown are all member of the “customer request” process. Due to display reasons an illustration of all identified business components is not feasible, whereas the relevant components for the examples illustrated in the next section are described in detail in section 4. Additionally, one further component is included: the orchestration component. Its purpose is to coordinate the communicating between the other components. This allows simple adjustments of the communication channels if the services of one or more components are altered.

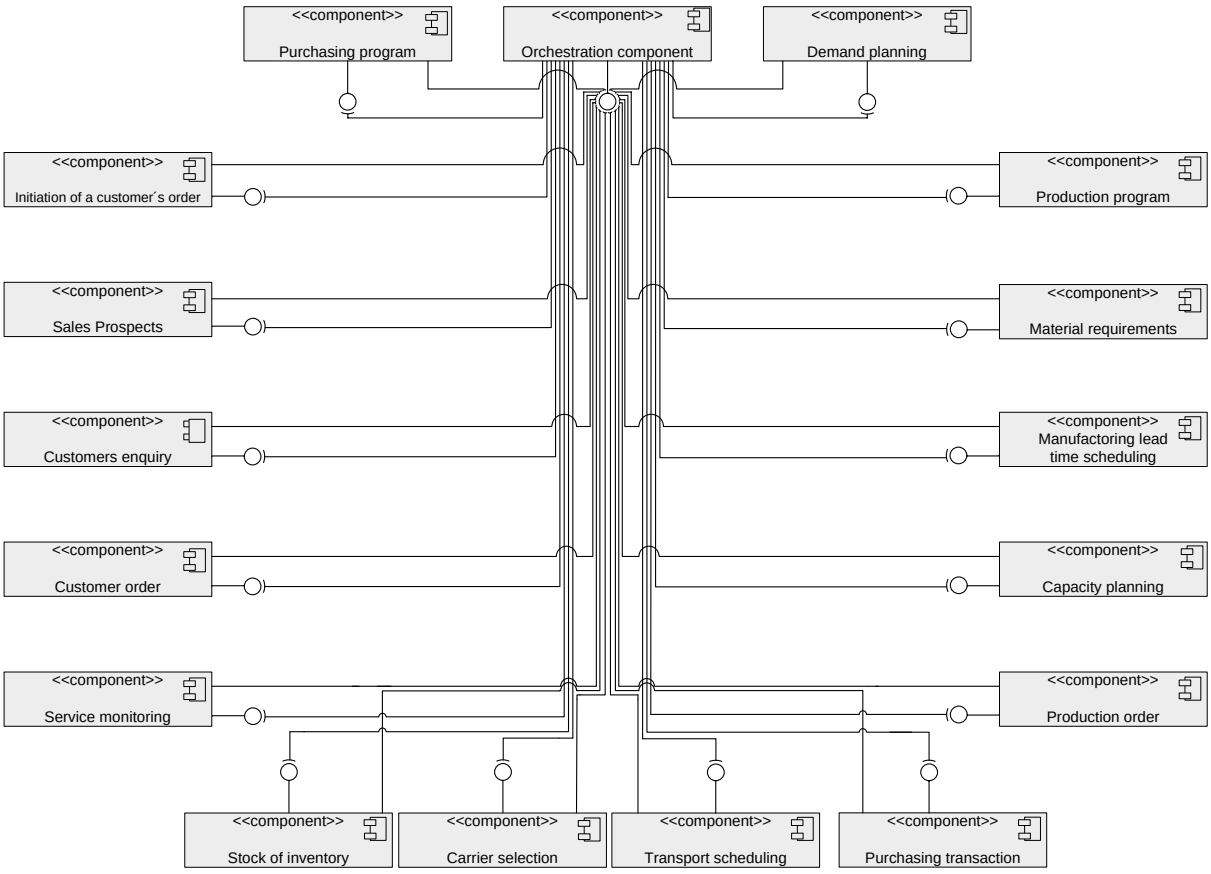

Fig. 4: Identified business components for the integrative ISA for CRM and SCM for the customer process

4 “Request for Quotation” processes within the integrative ISA

In order to illustrate the advantages of the integrated information system architecture for CRM and SCM, which has been introduced in section 3, two examples of the “request for quotation” process are described in this section. The first example process shows a request for quotation, which can be satisfied without additional requests to preliminary suppliers. The second example extends the first one by assuming that the organization does not have all required parts on stock. Consequently, preliminary suppliers have to be involved in order to answer the request of the customer correctly. The preliminary suppliers themselves also have to contact their preliminary suppliers for evaluating the availability of the required parts. Due to complexity reasons only three of the seven examined functional areas of the integrated ISA are included. Both examples show the informational relationships of the business components involved. As already mentioned above, the orchestration component is also included. Additionally, a communication component, which is responsible for the communication with external systems, is incorporated for completeness reasons. Its functionality is not explained further in this paper as it does not have a direct influence on the ISA, but on the inter-organizational communication.

Already the simple request for quotation illustrates the need for an integration of SCM and CRM and demonstrates that company-internal integration is a precondition for an inter-organizational integration.

Prior to the illustration of the example processes, the business components of the example processes are explained. From all the business components of the component model presented in section 3 only the 6 components, which are relevant for the example processes, are described next.

Customer Order

The business component customer order contains all necessary functionality for further processing of a customer quote. With regard to the example process the order execution planning and the delivery date confirmation are of particular importance. Additionally, all basic agreements of the customer order are arranged and submitted to the customer.

Material Requirements

The business component material requirements prepares the on-time allocation of all required materials in regard to type, quality and amount for the production process.

Capacity Planning

The functions of the business component capacity planning include design of capacity capability, determination of the capacity demand, deployment of staff, comparison of capacity demand and availability, scheduling of capacity and planning of machine allocation sequence.

Stock of Inventory

The business component stock of inventory contains the functionality which is associated with stock movements. Within others, this affects the inventory management, which is the link between demand and order planning.

Purchasing Transaction

The business component purchasing transaction includes all functions which are needed for the transaction-oriented part of the procurement initiation. This involves acceptance and transmission of requirement requests, requests for offers, which precede the registration and evaluation of requirements.

Manufacturing lead time scheduling

The business component manufacturing lead time scheduling contains all relevant functionality for successful scheduling like determination of process steps and operations or definitions of process and administrative times. By creation of task schedules, determination of lead times and evaluation of possible lead time reductions preliminary starting and ending times for the different process can be generated.

Scenario for the “Request for Quotation” examples

Both examples are based on the following assumptions: A customer asks the OEM for a customized product. Besides the price and the configuration of the product, the sales employee wants to tell the customer the delivery date as it is one of the crucial factors for his purchase decision. The determination of the delivery date requires access to all functional areas involved. For example the capacity of production, the delivery dates of required materials in procurement as well as the shipment slots in the distribution department. The resulting delivery date should already be available during the customer meeting since alternatives might be necessary. This information would allow discussing possible alternative product configurations fitting with the requirements of the customer in regard to the delivery date.

While example one assumes that all required material is available from stock, example two requires communication with the preliminary suppliers.

Example 1: “Request for Quotation” process within a company

After the definition of the basic agreements and the planning of the order processing, a date of delivery for confirming the customer order is required. The data of the expected customer order is passed to the lead time determination (manufacturing lead time scheduling component) in order to execute the required scheduling. For this determination, information of the business component material requirement, like net primary demand, is needed (see Fig. 5). The check if the required materials are available from stock is only feasible if the stock of inventory component delivers the required information. The business component purchasing transaction is not needed in this scenario, as it is assumed that all required materials are available from stock.

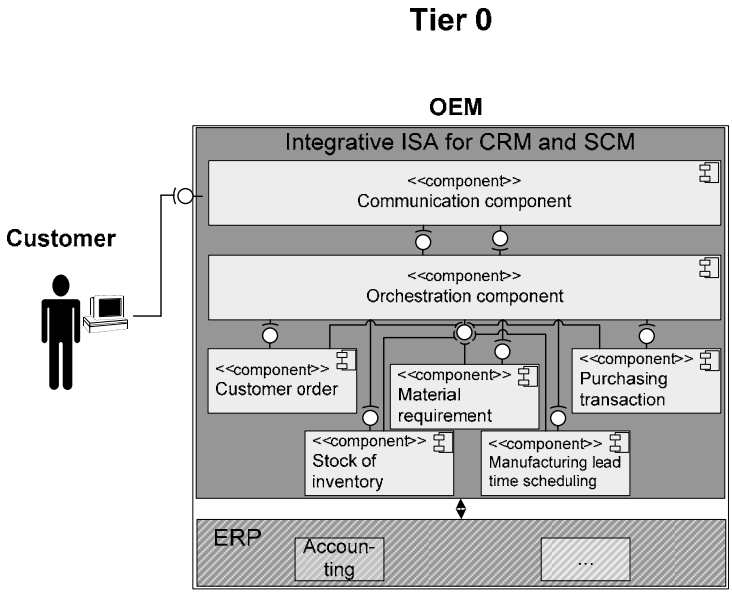

Fig. 5: Information relationships within the OEM for the request for quotation process

Example 2: “Request for Quotation” process within the supply network

In contrast to the first example, the OEM now does not have all required parts on stock. Consequently, he has to obtain the parts needed from his suppliers. Supplier 2 can fulfill the order of the OEM from stock, while supplier 1 has to obtain the parts for his (sub-) product from supplier 3 and 4 in order to satisfy the order of the OEM. Supplier 4 can serve the request of supplier 1 out of his inventory, while supplier 3 again has to contact his preliminary suppliers. Fig. 6 shows the relationships of information within as well as between the concerned companies due to the fact that a customer’s request not only affects the OEM, but also the preliminary suppliers in the supply network. These preliminary suppliers of supplier 3, the business components which are not part of the process shown as well as the unimportant relationships of the included business components are not illustrated in Fig. 6 due complexity and display reasons.

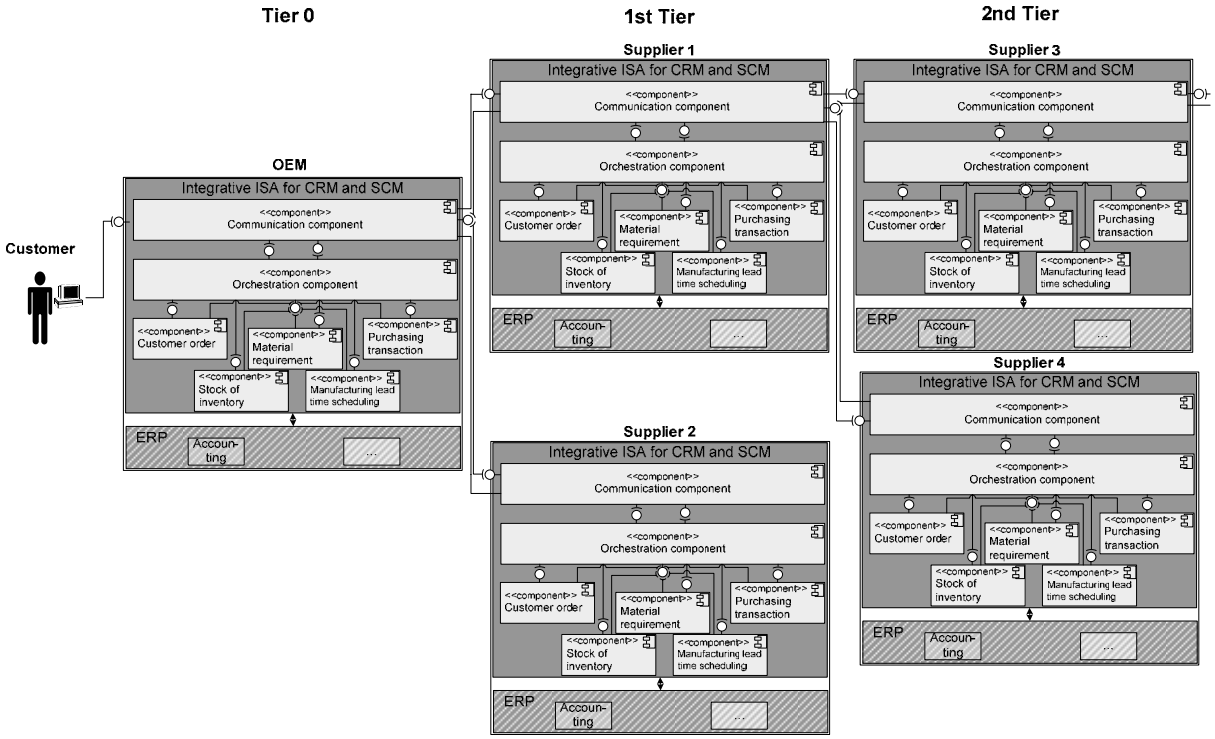

Fig. 6: Information relationships within the supply network for the request for quotation process

As it is assumed that required parts are not available from stock, a procurement transaction has to be invoked. After arrival of the estimated delivery date of the required parts, this information is passed back to the business component stock of inventory. From there the information is given to the material requirement component and further to the manufacturing lead time scheduling component. Thereby all information is available for determining the delivery date for the customer. This delivery date is then passed back to the customer order component, which transfers the information to the customer.

The request of the OEM to his suppliers invokes the same process also at supplier 1 and then at supplier 3. As supplier 2 and 4 do have the required material available from stock no further procurement transaction has to be executed.

These two suppliers can determine the date of delivery without using preliminary suppliers. The information about the date of delivery is then, as described above, passed back to the business component customer order in order to inform the customer, the OEM in the case of supplier 2 and supplier 1 in case of supplier 4.

5 Conclusion

This paper showed that enterprises have to implement inter-organizational integration due to increasing competition and globalization. But an inter-organizational integration is not feasible without prior intra-organizational integration. The presented integrated information system architecture (ISA) for CRM and SCM allows the coordination of data management and functionality of all functional areas (sales, service, marketing, procurement, production, logistics and waste disposal), which are directly involved in the value creation process. Consequently, the problems which arise if application systems are only interconnected – e.g. redundant data or different data structures – do not occur anymore. This integration is then used as basis for a transparent inter-organizational integration of all members of the whole supply network allowing a continuous exchange of information between the members. Additional investigations are needed in improving and refining the presented architecture for CRM and SCM and in integrating it with existing ERP systems.

References

1. Kopanaki, E., et al. The Impact of Inter-organizational Information Systems on the Flexibility of Organizations. In Proceedings of the Sixth Americas Conference on Information Systems (AMCIS). 2000. Long Beach, CA.
2. Picot, A., R. Reichwald, and R. Wiegand, Die grenzenlose Unternehmung - Information, Organisation und Management. 4. Auflage ed. 2001, Wiesbaden. 2-6.
3. Warnecke, H.-J., Vom Fraktal zum Produktionsnetzwerk. Unternehmenskooperationen erfolgreich gestalten, ed. J.H. Braun. 1999, Berlin.
4. Sawhney, M. and J. Zabin, The Seven Steps to Nirvana: Strategic Insights into eBusiness Transformation. 2001, New York.
5. Malone, T.W. and R.J. Lautbacher, The Dawn of the E-Lance Economy. Harvard Business Review, 1998(September-October): p. 145 - 152.
6. Engineers, I.o.E.a.E., IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. 1990.

7. Sinz, E.J., Architektur von Informationssystemen, in Informatikhandbuch, P. Rechenberger and G. Pomberger, Editors. 1999, Hanser Verlag: München/Wien.
8. Fleisch, E., Das Netzwerkunternehmen: Strategien und Prozesse zur Steigerung der Wettbewerbsfähigkeit in der "Networked economy". 2001, Berlin et. al.: Springer.
9. Selk, B., K. Turowski, and C. Winnewisser. Information System Design for Demand-Driven Supply Networks - Integrating CRM & SCM. In Fourth International ICSC Symposium on Engineering of Intelligent Systems EIS 2004, University of Madeira, Funchal, Portugal, February 29th - March 2, 2004. 2004. University of Madeira, Funchal: ICSC Academic Press.
10. Albani, A., et al. Dynamic Modelling of Strategic Supply Chains. In E-Commerce and Web Technologies: 4th International Conference, EC-Web 2003 Prague, Czech Republic, September 2003, LNCS 2738. 2003. Prague, Czech Republic: Springer-Verlag.
11. Albani, A., C. Winnewisser, and K. Turowski. Dynamic Modelling of Demand Driven Value Networks. In On The Move to Meaningful Internet Systems and Ubiquitous Computing 2004: CoopIS, DOA and ODBASE, LNCS. 2004. Larnaca, Cyprus: Springer Verlag. 12. Fingar, P., H. Kumar, and T. Sharma, Enterprise E-Commerce - The Software Component Breakthrough for Business-to-Business Commerce. 2000, Tampa: Meghan-Kiffer Press.
13. Kalakota, R. and M. Robinson, e-Business 2.0 Roadmap for Success. 2. ed. 2001, Boston: Addison Wesley Verlag.
14. Hagel, J., Out of the Box: Strategies for Achieving Profits Today and Growth Tomorrow through Web Services. 2002, Boston: Harvard Business School Press.
15. Puschmann, T., Collaboration Portale. Architektur, Integration, Umsetzung und Beispiele. 2003, Universität St. Gallen: St. Gallen.
16. Bazijanec, B., et al. Component-based Architecture for Protocol Vector Conversion in Inter-organizational Systems. In International Workshop on Modeling Inter-Organizational Systems (MIOS'04). 2004. Larnaca, Cyprus: Springer, LNCS 3292.
17. Bazijanec, B., et al. Establishing Interoperability of Coordination Protocols in ad-hoc Inter-Organizational Collaborations. In Interop-Esa 2005 - First International Conference on Interoperability of Enterprise Software and Applications. 2005. Geneva, Switzerland: Springer, science + business media.
18. Vandermerve, S., How Increasing Value to Customer Improves Business Results. MIT Sloan Management Review, 2000. 42: p. 27-37.

19. Harmon, P., M. Rosen, and M. Guttman, Developing E-Business Systems and Architectures: A Manager´s Guide. 2001, San Francisco: Morgan Kaufmann.
20. Österle, H., Geschäftsmodell des Informationszeitalters, in Business Networking in der Praxis: Beispiele und Strategien zur Vernetzung mit Kunden und Lieferanten, H. Österle, E. Fleisch, and R. Alt, Editors. 2002, Springer: Berlin et al. p. 17-38.
21. Mertens, P., Integrierte Informationsverarbeitung 1 - Operative Systeme in der Industrie. 14. ed. 2004, Wiesbaden: Gabler Verlag.
22. Turowski, K., Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme. 2003, Aachen: Shaker Verlag.
23. Turowski, K. and J.M. Zaha, Methodological Standard for Service Specification. International Journal of Services and Standards, 2004.
24. Albani, A., et al. Domain Based Identification and Modelling of Business Component Applications. In 7th East-European Conference on Advances in Databases and Informations Systems (ADBIS-03), LNCS 2798. 2003. Dresden, Deutschland: Springer Verlag.
25. Albani, A., J.L.G. Dietz, and J.M. Zaha. Identifying Business Components on the basis of an Enterprise Ontology. In Interop-Esa 2005 - First International Conference on Interoperability of Enterprise Software and Applications. 2005. Geneva, Switzerland.
26. IBM, Business Systems Planning-Information Systems Planning Guide, ed. I.D. (Ed.). 1984, Atlanta: International Business Machines.
27. Albani, A., et al. Component Framework for Strategic Supply Network Development. In 8th East-European Conference on Advances in Databases and Information Systems (ADBIS-04), LNCS 3255. 2004. Budapest, Hungary: Springer Verlag.
28. Albani, A., et al. Komponentenmodell für die Strategische Lieferkettenentwicklung. In 6. Internationale Tagung Wirtschaftsinformatik (WI-03) Medien - Märkte - Mobilität. 2003. Dresden, Deutschland: Physica-Verlag.

III.2 Beitrag: „Something is Missing: Enterprise Architecture from a Systems Theory Perspective”

Autoren: Sebastian Klöckner, Dominik Birkmeier
Beide Lehrstuhl WI-SE, Universität Augsburg,
Universitätsstraße 16, D-86135 Augsburg,
Email: sebastian.kloeckner@wiwi.uni-augsburg.de
dominik.birkmeier@wiwi.uni-augsburg.de

Erscheint in: Proceedings of 4th Workshop on Trends in Enterprise Architecture Research (TEAR) @ ICSOC2009, Springer Lecture Notes in Computer Science, 23. November 2009, Stockholm, Schweden (2010)

Die Unternehmensmodellierung hat in der letzten Dekade hohe Aufmerksamkeit durch Wissenschaft und Praxis erfahren. Insbesondere Praktiker haben dabei die Idee der Unternehmensarchitektur (Enterprise Architecture, EA) entwickelt, die zwischenzeitlich zu einem vielversprechenden und umfassenden Ansatz zur Modellierung des aktuellen (IST) oder gewünschten (SOLL) Zustands eines Unternehmens wurde. Die bestehenden Ansätze werden jedoch häufig dafür kritisiert, dass sie den geschäftlichen Nebenbedingungen, Interessen und Zielen eines Unternehmens noch immer zu wenig Aufmerksamkeit schenken. In diesem Beitrag wird ein Unternehmen als sozio-technisches System interpretiert und aus systemtheoretischer Perspektive analysiert, welche zusätzlichen Eigenschaften notwendig wären, um einen wirklich umfassenden Ansatz zu entwickeln. Davon ausgehend wird abgeleitet, ob, warum und wie weitere Aspekte der Unternehmensumwelt in das Konzept der EA integriert werden können. Unter anderem wird dabei offensichtlich, dass insbesondere der menschliche Faktor, als flexibelstes und agilstes Element von Unternehmen, nicht ausreichend in den aktuellen EA-Ansätzen berücksichtigt wird. Dementsprechend werden abschließend erste Ideen für die Einbeziehung dieses wichtigen Faktors, sowie daraus resultierenden Auswirkungen für die Praxis und neue Forschungsrichtungen für die Wissenschaft, präsentiert.

1 Introduction

Enterprise modeling in general and enterprise architectures (EA) in particular have gained significant momentum in research (e.g. [1-3]) and practice (e.g. [4, 5]) during the last years. Especially the promise to make the important elements of an enterprise and their relations visible makes it an interesting concept for the analysis and design of complex business systems. A comprehensive enterprise architecture therefore specifies, amongst others, the goals and strategies of an enterprise, its business processes as well as the associated resources like production systems, information systems and humans [6]. While the former aspects are often included in current concepts of EA, especially humans, as integral parts of enterprises, are often not taken into consideration. But only such a complete picture would essentially support necessary transformations of organizations in a flexible and agile way.

First being discussed in the 1970s, several enterprise architecture concepts, often under different names, were proposed over the time (e.g. [3, 7-10]). Today the Zachmann-Framework [11], The Open Group Architecture Framework [12] and the Federal Enterprise Architecture [13] can be seen as the most widespread frameworks for enterprise architectures. But as most of these concepts are rooted in the IT-departments of today’s enterprises and organizations, they are oftentimes strongly focused on IT-related aspects.

Although business issues are slowly moving into research focus [14, 15], few attentions have so far been paid by the information systems science community to organizational aspects. When looking at the complete picture, enterprises are socio-technical systems and therefore do not only contain technical components, but also humans and the organizational context [16]. Especially this organizational context can have a significant impact on the overall success and is one of the main reasons of budget overruns or even complete failings, as stakeholders are resistant to change or do not adopt new technologies. Dietz and Hoogervorst [17] consider the traditional black-box thinking based knowledge, i.e., knowledge about the function and the behavior of enterprises, as one of the reasons for such problems. While being sufficient for managing an enterprise within its current range of control, this kind of thinking is inadequate when an enterprise has to change. For such cases, a white-box approach, describing the construction and operation of enterprises, is needed.

In this paper, we therefore take an argumentative-deductive approach to analyze from the holistic perspective of systems theory, in particular Ropohl’s Theory of General Technology [18], where certain aspects are missing in the current concept of enterprise architectures. Furthermore, we present first ideas how these missing parts can be integrated and which benefits could be realized by such an integration.

In the remainder of the paper we will proceed as follows: after presenting related work and motivating the research gap in the next section, we introduce relevant theories and concepts. In detail, we present the common understanding of enterprise in the field of enterprise architectures, as well as the basic concepts of systems theory in general and the Theory of General Technology in particular. By interpreting enterprises as socio-technical systems, different aspects in regard to including human factors into enterprise architecture are deduced in the synthesis. Taking these aspects into account we draw the conclusion that especially an integration of human beings into the lower layers of EA is necessary. In the following section on implications for practice and academia we present that this will further disclose new optimization approaches and cost-saving opportunities. Finally, we sum up our findings, provide an outlook and raise several possible trends for EA.

2 Related Work

The evaluation and comparison of EA frameworks in general and their completeness in particular has frequently been addressed in literature. Besides publications focusing on a single framework (e.g. [19]) there are also several extensive comparisons between existing frameworks such as the ones from Leist et al. [20], Bernus et al. [21], Schekkerman [22] and Schönherr [23]. All of them elaborate on the individual strengths and weaknesses of the common frameworks and try to identify their gaps as well as possibilities for further improvement. Noran [24] furthermore examines the mapping of classical frameworks onto the Generalized Reference Architecture and Methodology (GERAM) Framework [15], which tries to provide a common, but rather abstract, regulation framework for the former ones.

In addition to those comparisons between frameworks, Aier et al. [25] provide a literature survey on established contributions from academia and practice, as well as an empirical examination on comprehension, model building and utilization of enterprise architectures. They develop a systematic overview on the current understanding and the state of the art of EA. From there, they identify discrepancies between research and practice and discuss corresponding implications for both. Through an empirical survey among practitioners and researchers in the area of EA, they identify, amongst others, which design artifacts have to be considered in an EA and the degree of their realization in practice. From this survey, it can be seen that the interaction with customers and suppliers, roles and responsibilities, as well as organizational units are all considered mainly important, but their degree of realization is generally lower than the average of all design artifacts. On the other side, many of the aspects considered as the most important ones, with the highest degree of realization, are purely technical. Those are, amongst others, interfaces, applications, data structures, software-, hardware- and network-components, etc. It can be seen from the study that, while

organizational aspects of enterprises slowly come into focus, enterprise architectures are predominantly shaped by IT-departments and their view of the enterprise.

Furthermore, an analysis of the “impact of service-oriented architecture on enterprise systems” was carried out by Bieberstein et al. [26]. Based on it, Schroth [27] presented his view of a Service-Oriented Enterprise. Both elaborate on the alignment of service-oriented architectures (SOA) with first aspects of existing organizational theories. Bieberstein et al. state that “the SOA paradigm also needs to be extended to transmute organizational structures and behavioral practices” [26] and they thus “propose the Human Services Bus (HSB), a new organizational structure that optimizes the workflow and streamlines cross-unit processes to leverage the new IT systems” [26]. Schroth [27] proposes an approach of mapping the major underlying principles of SOA, namely decentralization, agility, as well as composition and coordination of building blocks, to upcoming forms of organizations. He uses the HSB to allow advertising human services within an enterprise and provide a means for workflow monitoring. While both works consider aspects from organizational theory, they origin from a rather technical SOA context and focus on changes that an integration of SOA implicates for enterprises and so do not extend their findings to a general EA.

In summary it can be said that the issue of evaluating EA approaches has been faced from different perspectives. An extensive analysis and comparison of current frameworks, as well as a critical reflection of differences in research and practice has been utilized to identify goals for the future development. On the other hand, the idea of including aspects from organizational theory has been raised, but was limited to the field of service-oriented architectures. Though, to the best knowledge of the authors there is so far no comprehensive analysis of enterprise architecture in regard to systems theory from a scientific perspective.

3 Conceptual Foundations

As the review of the related work has illustrated, current concepts of EA are often focused on IT-related aspects. While understandable as most concepts have their origins in IT-departments, the call for a scientific backing of these concepts and an inclusion of organizational aspects gets louder. In order to get an insight into existing concepts and possible scientific theories, we firstly present the current understanding of enterprise architecture and the basic elements of systems theory.

3.1 Enterprise Architecture

As shown before, enterprise modeling is a field of significant research and is widely accepted in science and practice [6, 14, 28]. Over time several different names, perspectives and definitions for enterprise architectures were proposed (e.g. [3, 7-10]). A comprehensive overview of different enterprise modeling approaches with sometimes different perspectives

can be found in [25]. For a common understanding of the used terms, we will rely on the following definitions:

In the context of EA, The Open Group defines an enterprise as “Any collection of organizations that has a common set of goals and/or a single bottom line. In that sense, an enterprise can be a government agency, a whole corporation, a division of a corporation, a single department, or a chain of geographically distant organizations linked together by common ownership” [12]. On the other hand, the ANSI/IEEE Standard 1471-2000 defines architecture as the “fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution” [29].

While most of the available modeling approaches conform to these definitions, they diverge in certain aspects due to their respective goals and complexity. Winter and Fischer criticize that “Traditionally, architecture in the information systems is focusing on IT related artifacts […]” and suggest that “Only when ‘purely’ business related artifacts are covered by EA, important management activities like business continuity planning, change impact analysis, risk analysis and compliance can be supported.“ [14]. Based on a literature analysis, they deduce five essential layers and related artifacts of enterprise architectures, which were adapted by Aier, Riege and Winter [25] and are shown in figure 1. These proposed layers are a first step to incorporate organizational aspects into the concept of enterprise architecture, as they include artifacts like organizational goals, organizational units and responsibilities. However, they still do not consider these aspects on the lower layers. When taking the history of enterprise architecture and its roots in the IT-departments of enterprises into account, it becomes understandable that it is, mainly on the lower layers, still strongly focused on IT-related issues. But, this perspective only examines one side of the coin, as it ignores the human contribution to the overall performance.

Going further back into history, it becomes obvious that enterprises and their architectures existed before IT came into play. At this time information systems, in their classical definition as systems for the communication and processing of information, without any or only little technology (letter shoot, dockets, etc.) were already in place. The most important components of such systems and the corresponding architectures were, and probably are, human beings. While literature often states flexibility and agility as primary goals of enterprise architectures (e.g. [30]), many approaches do not take these extraordinary flexible and agile “components” of an enterprise into account. Again, this might be owed to the fact that, until recently, practitioners of IT-departments have been leading in the development of the EA discipline. It therefore seems to be advisable to examine from a scientific perspective if, why and how these important providers of services and functionality should be included into enterprise architecture in general. And as enterprises are often described as systems,

the holistic perspective of systems theory in combination with the concept of socio-technical systems seems to be a good starting point for such an examination.

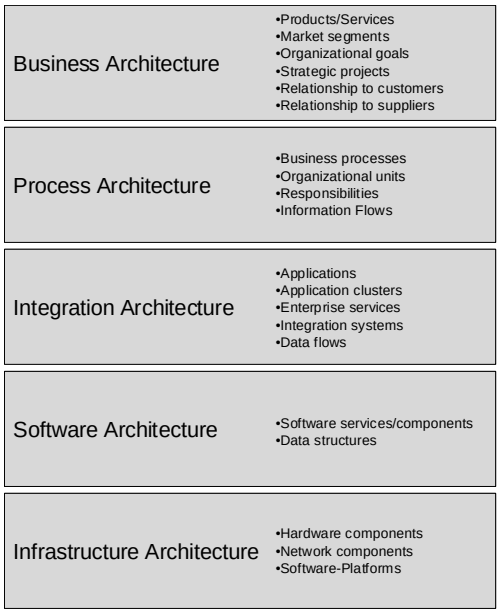

Fig. 1. Proposed layers and artifacts of Enterprise Architecture [25]

3.2 Socio-technical Systems

Systems theory is an interdisciplinary theory, which focuses on the description and analysis of any system as an integral whole that consists of interconnected elements (parts) and their relations. The roots of modern systems theory can be traced to Bertalanffy’s General Systems Theory [31] and Wiener’s Cybernetics [32]. Mainly based on these works, Ropohl developed a Theory of General Technology [18] for explaining and analyzing elements and relationships of socio-technical systems [33, 34]. Socio-technical in this context refers to the interrelatedness of social and technical aspects of a system, e.g. human beings and technical artifacts in an organization.

In his depictions, Ropohl first distinguishes between three different system perspectives: the functional, the structural and the hierarchical view. The functional system concept regards the system as a black-box, which is characterized by certain relations between in- and output properties that can be observed from outside. The structural system concept views the system as a whole, consisting of interrelated elements. And the hierarchical system concept finally emphasizes the fact that parts of a system can be considered as systems themselves and that the system itself is part of a larger system.

Based on these definitions, Ropohl develops a general model of an action system, which is characterized by three sub systems: the goal setting system (GS), the information system (IS) and the execution system (ES). The execution system obtains material and energetic attributes and performs the basic work. In the context of business information systems, the mentioned material has to be interpreted as data and information, which have to be

processed. The information system handles informational attributes. It absorbs, processes, and passes information and deduces instructions for the execution system from this information. The goal setting system creates the system's internal goals as maxim of action.
By introducing the concept of division of labor, Ropohl then further decomposes the initially "monolithic" system from a different perspective. As each action usually consists of more than one sub-action, united in one virtual or real system, these sub-actions can also be disassembled into own systems. These new subsystems then have to be linked and coordinated in order to fulfill the formerly united action. The combination of the functional decomposition of a single action and the division of labor (chain of action, respectively workflow) is shown in figure 2.

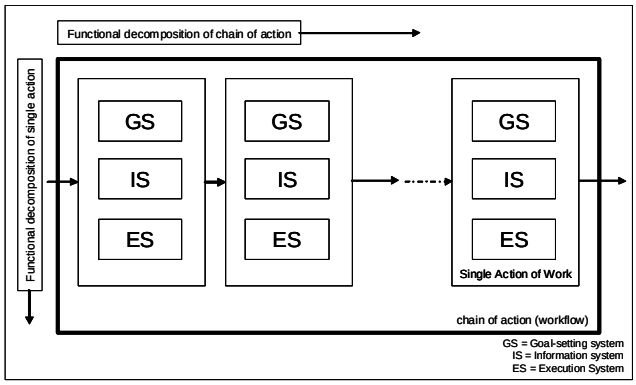

Fig. 2. Functional decomposition of single action and chain of action

Based on these two perspectives of decomposition, Ropohl then develops the concept of socio-technical division of work, shown in figure 3. Ropohl considers Adam Smith [35] to be the first, who noticed the concept of socio-technical division of work. Smith is said to have stated, in a different context, that machines are similar to humans in many ways and that machines are small systems, created to cause certain movement effects.

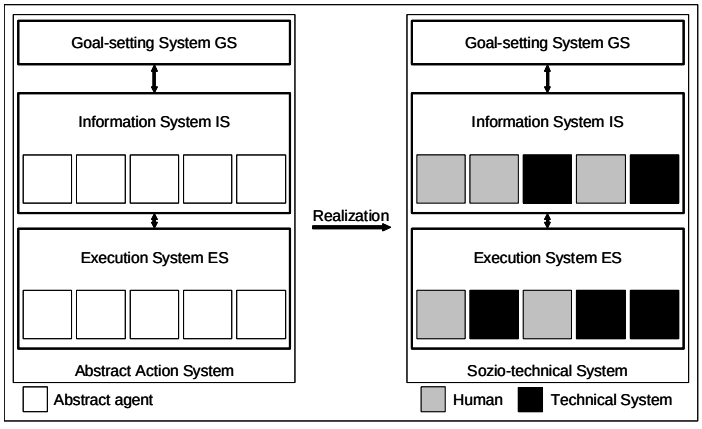

Fig. 3. Socio-technical division of work (adapted from [18])

The abstract action system, depicted on the left side of figure 3, contains an abstract agent as the agent of action. This system can be interpreted as a human being which incorporates

all subsystems or as an organization, where several actors are in charge of certain sub-actions. When interpreted as an organization and transferred into reality the single sub-actions of the respective subsystems can be either assigned to human actors or to technical artifacts. Therefore, a socio-technical system is defined as an action or work system, which is composed of human beings and technical systems (artifacts). But while technical artifacts have clearly defined system boundaries, human individuals have additional properties, exceeding their predefined roles in the system, due to their social and cognitive capabilities (e.g. social networks, creativity, problem solving, etc.).

4 Synthesis

A comparison of the basic structures of enterprises and the theory of socio-technical systems shows that many similarities exist. An organization composed of humans and artifacts, like an enterprise, must therefore be interpreted as a socio-technical system, whose structure, as stated in ANSI/IEEE Standard 1471-2000 [29], is described by its architecture. Its actions are realized by humans and/or technical artifacts and from an abstract point of view, the different management and execution layers of enterprises can directly be mapped to the different subsystems of the functional decomposition: the goal-setting system is realized by the executive managers, the information system by the middle managers and/or information systems and the execution systems is materialized by the workers and/or production systems. Due to the recursive structure of systems theory, each of these three subsystems can then again be interpreted as system and therefore consist of the three subsystems themselves, which are realized by humans and/or artifacts.

Looking at the previously described layers and artifacts of EA, many of the proposed elements also fit with the concept of socio-technical systems: The business architecture with its goals, strategic projects, desired products, targeted market segments, etc., models the goal-setting system of the enterprise. The process architecture, with its business processes, information flows and responsibilities, etc., as well as the integration architecture, containing applications, enterprise services, integration systems, data flows, etc., mirrors the idea of the information system. The software and infrastructure architectures can finally be mapped onto the execution system.

As shown, there are obvious similarities between EA and socio-technical systems and as enterprises are often characterized as socio-technical systems, it seems to be reasonable to further analyze the concept of enterprise architecture under the perspective of the Theory of General Technology. Especially, as it also becomes evident that the respectively considered elements, subsystems and relations seem to be different.

Considering the literature on EA, it first becomes apparent that EA models are used for two different purposes: many publications about enterprise architectures differentiate between

descriptive and design models (e.g. [4, 6, 19]). Descriptive models, often called “As-Is-Models”, illustrate the state of an enterprise in its current situation and how business is executed. Design models, regularly labeled as “To-Be-Models”, envision a future state of the enterprise and how it should be shaped in the future. As the purpose of these models significantly differs, it becomes obvious that they are in fact the outcome of two different types of systems. The first system, which has the descriptive model as result, can be named run-time system, the second, producing the design model, should be called design system. These two systems are not independent of each other, though. The design system pictures a future state, which shall be implemented in the run-time system. The existing run-time system on the other side is usually the basis for a new design, as complete green field approaches are typically not possible in practice. These two systems themselves contain socio-technical sub-systems. Each of these subsystems realizes certain actions, which can or cannot be fulfilled by humans and/or technical artifacts as shown in table 1. In addition, each type of agent (human being and technical artifact) has distinct properties, which make them better suitable for a certain task.

	Design System		**Run-Time System**	
	Action	**Agent**	**Action**	**Agent**
GS	(Re-)Define targets of system and actions	Humans only	Control and/or set goal of executed action	Humans and technical artifacts (goals are implicitly set)
IS	(Re-)Define necessary flows and tasks to achieve goals	Mostly humans, sometimes technical artifacts	Execute control flow or function templates, call necessary functions	Humans and technical artifacts
ES	Create and acquire necessary flow and function templates	Humans and technical artifacts	Realize and execute certain function	Humans and technical artifacts

Table 1. Different actions of sub-systems at design and execution time

During design time, the goal-setting-subsystem can only be realized by humans, since the creation of goals, targets, decisions and strategies needs creativity, which cannot be realized in technical artifacts. While a support by technical artifacts is possible, e.g. decision support systems, a replacement of humans by technical artifacts is not possible. The information subsystem of the design system has to define the necessary flows and tasks in order to make a realization of the defined goals and targets possible. While first approaches are made to at least partly automate the action of this subsystem (e.g. [36]), it is unlikely that it can be fully automated in the near future. Finally, the execution subsystem of the design system creates (make) or acquires (buy) the necessary resources to fulfill the tasks defined by the information subsystem. While this action is usually realized by humans, several approaches, like code generators or automated market places, try to computerize it. Of special interest in this area are currently approaches, which optimize the grouping of the

required functionality and corresponding data in order to find the optimal granularity of subsystems for reuse (e.g. [37, 38]).

The goal-setting subsystem of the run-time system can be realized by humans or by technical artifacts. In the case of humans, they will follow the plans, which were defined during design time. If certain situations were not considered during design time, they are able to (re-)define what has to be done and how to react to certain influences from the environments in a flexible as well as agile way. For technical artifacts the goal-setting subsystem exists only implicitly. By defining its functionality during design time, its goals are irrevocably set. If certain system-external or -internal influences or states were not considered at design time, e.g. by catching errors, a technical artifact will malfunction, e.g. an error occurs. An independent redefinition of goals and a corresponding reaction is, while being researched, not possible yet. According to the defined goals, the information subsystem executes the created or acquired flow and function templates. It controls, depending on the input and based on these templates, which agents, humans or technical artifacts, have to be called to fulfill a certain action. While humans can independently switch to comparable functions if a needed subsystem is not available (e.g. different supplier of comparable pre-products), this is, due to the normally missing capability of comparing unspecified possibilities, usually not possible for technical artifacts. The best chance for technical artifacts, if considered during design time, is to keep state until the required subsystem becomes available. The execution subsystem finally executes the defined tasks. It takes the input and transfers it into the output. The actions of the execution subsystem can, due to the hierarchical concept of systems theory, themselves be complex systems. They can be realized by an enterprise, like a sub-contractor, by technical artifacts, like web services or by human beings.

Layers of EA	**Design System**	**Run-Time System**
Business Architecture	(Re-)Define targets of system and actions	---
Process Architecture	(Re-)Define necessary flows and tasks to achieve goals	Control and/or set goal of executed action
Integration Architecture	Create and acquire necessary flow and function templates	Execute control flow or function templates, call necessary functions
Software Architecture	---	Realize and execute certain function
Infrastructure Architecture	---	---

Table 2. Mapping of socio-technical subsystems to proposed layers of enterprise architecture

Now, taking into account that the results of the information subsystem of the design system are used by the goal-setting subsystem of the run-time system and that the outcomes of the

execution subsystem at design time are utilized by the information subsystem at run time, the different subsystems can be connected as shown in table 2. In addition, it can again be shown that the different subsystems can also be mapped onto the proposed layers of EA.

5 Consequences of the Synthesis

Taking this into account, several consequences for and gaps of enterprise architecture can be deduced. While the higher layers still contain the human aspect of socio-technical systems, they disappear on the lower layers like integration, software and infrastructure architecture. Although it is understandable for the infrastructure level, where offices and buildings, walks and streets would correspond to hardware and network components, it has to be questioned, if this is comprehensible for the integration and software layer. When looking into today’s enterprises, human beings regularly execute functions of these layers: For example, whenever a direct information system integration is not possible, due to technical restrictions, human beings or teams connect the two distinct application systems over their graphical user interfaces and act as human connectors between these systems. But, as these human connectors are usually very costly compared to technical interfaces, information about them and their integration into the enterprise architecture would help to optimize the enterprise. In this context, teams of human beings could be interpreted as enterprise services which offer certain business and integration functionality to other elements of the system.

When looking at the layer of software architecture with its software components and data structures, again, the general tasks are sometimes carried out by human beings. E.g., humans often carry specialized data about certain customer preferences or realize tasks which require a certain portion of creativity. Both are characteristics which are important for successful enterprises and cannot easily be realized by technical artifacts. When these qualities are overseen, just because they were not part of the enterprise architecture, for example when replacing sales clerks by web front ends or call centers, the overall performance of the enterprise can take severe damage. In this context, human actors could be understood as software or elementary services which carry data or realize certain specialized functionality. The breakdown or loss of such system elements and the corresponding implications can be compared with those of software services, although the latter are often easier to fix.

It therefore seems to be more than reasonable to integrate human beings or aggregations of those (e.g. teams) as components or service providers with certain interfaces into the lower layers of enterprise architecture, as shown in figure 4. Only if human beings, as integral part of the socio-technical system “enterprise” are included, profound management and alignment decisions become possible.

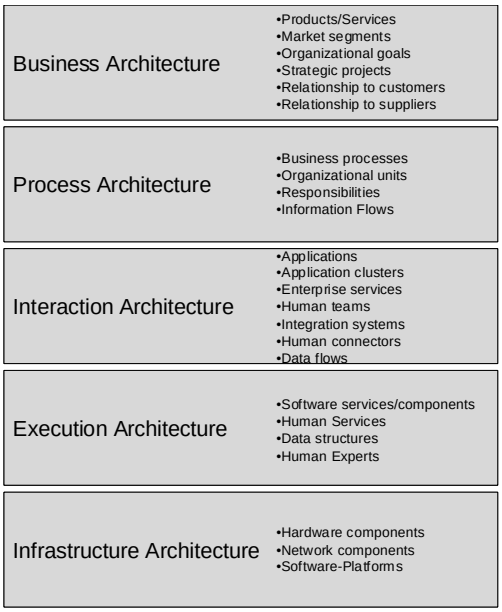

Fig. 4. Integration of human factors into proposed layers and artifacts of EA

In consequence and in order to mirror the inclusion of human beings and technical artifacts properly, the name of the integration and software architecture should be changed: Instead of using the word integration, the word interaction would better reflect the interplay between humans and technical artifacts. And execution architecture better characterizes the realization of basic services, regardless of whether they are provided by humans or technical artifacts.

6 Implications for EA in Practice and Research

The integration of the human or social aspect into enterprise architecture has several implications for the successful management of enterprises and the focus of future research. First, it would allow the concept of enterprise architecture to become a comprehensive model of all relevant aspects and elements of the enterprise. The white-box perspective of systems theory has shown that each subsystem of the supersystem enterprise is usually a socio-technical system and omitting one important element of this system would be negligent and carries the risk of wrong decisions.

Second, as an increase of agility (adapt to unexpected changes) and flexibility (adapt to expected changes) is one of the primary objectives of the concept of enterprise architecture [30], it seems to be reasonable to include the most agile and flexible components of the socio-technical system enterprise, human beings and organizational groups, into the concept. Especially due to their flexible goal-setting and information system, human beings can quickly adapt to unexpected as well as to expected changes of the environment. A comparably quick adaption of technical artifacts has to be viewed as unlikely. Furthermore, an a priori integration of solutions for unexpected changes is impossible, as they can, by

definition, not be foreseen. Therefore, the inclusion of human beings into the concept of enterprise architecture would significantly increase the level of usable resources.

Third, the inclusion of human aspects into enterprise architecture would be the first step to the often demanded integration of concepts from organizational science (e.g. [16]). In addition, an integration of human and organizational aspects would reveal areas, where organizational science or information systems science tries to solve comparable questions, which were already answered by the neighbor discipline. For example, while the component and service identification community (e.g. [37, 38]) still tries to find the best-suited solution for the grouping of technical artifacts (granularity), the organizational science already gave answers to these questions in the early 1970s (e.g. [39, 40]). Under the name “departmentalization” the authors utilize affinity analysis to create units with maximal cohesion and minimal dependencies.

Forth, as Trist and Emery [33, 34] already mentioned, the optimization of only one aspect of socio-technical systems, social or technical, carries the danger of unpredictable, undesigned relationships between elements of the system, which have significant negative influence on the overall performance, e.g. in case of premeditated misuse of technical artifacts. Therefore, only a joint optimization of the social and the technical aspect seems to be advisable. Otherwise, when only optimizing technical aspect, the danger exists that, as often the case, new technology fails to meet the expectations of designers and users alike. When omitting the human factor from enterprise architecture, not even the possibility of such an influence can be taken into account.

Fifth, if humans are considered as part of the system, the role of information system technology can change: Instead of information technology being part of human-controlled systems, humans can become part of technical-controlled systems, e.g. in the case where the process flow (information system of the run-time system) is controlled by the IT-system and humans fulfill certain tasks as part of the execution system (at execution time). While this will be bought at the cost of an asynchronous execution, as humans are slower than usual time-outs of IT-systems, it will offer new functionality and realization potentials. And perhaps this shift in perspective could even result in a change of existing programming paradigms.

Finally, only the integration of humans into the concept of enterprise architecture will shed light on spots, where new technical artifacts would allow significant cost reductions. Based on activity-based costing [41] for formerly manual tasks, the amount of cost-savings can be exactly calculated and used as argument for additional investments in information technology. This would solve one of the worst problems of IT-managers, namely to argue the return on investment of information technology. On the other side, there may also occur situations, where technical artifacts should be replaced by humans; for example due to their higher agility and flexibility or when the total costs of ownership (TCO) exceed the costs of

labor. In order to make such decisions, a new measure like Total cost of IT usage, as an aggregate of TCO and costs of labor might become reasonable.

7 Conclusions and Further Research

In this paper we analyzed the concepts of enterprise architecture from a systems theory perspective. By interpreting enterprises as socio-technical systems and under usage of the Theory of General Technology, we deduced that the concept and use of enterprise architectures implicitly includes two different systems: The design system and the run-time system. These two systems fulfill different purposes and therefore require different capabilities. Due to these requirements, it becomes obvious that humans and groupings of them play an important role in today’s enterprises and the negligence of their actions and properties results in incomplete models. Especially, as humans are the only components in the system “enterprise”, which are able to act in an agile way. While we did not explicitly name service-oriented architectures, the interpretation of humans as service providers would also perfectly integrate into the paradigm of SOA and e.g. in fact supports the realization of a Human Service Bus [26]. But the inclusion of the human factor into the concept of EA also creates several open questions:

- What shift in perspective is necessary, if humans, as active and passive elements of enterprises, in opposition to technical artifacts, as solely passive components, are included into enterprise architectures?
- How can humans and technical artifacts be included into identification approaches?
- How can additional capabilities of human individuals (e.g. social networks), not directly connected with their role in the organization, be reflected in models?
- What side-effects are created by the inclusion of the human factor?
- What measures are necessary to decide if the replacement of humans by technical artifacts or vice versa is reasonable?
- Do situations exist, where the replacement of technical artifacts by humans is reasonable?
- How can the proportion between humans and technical artifacts be optimized?

When further applying the white-box perspective of systems theory, many other questions could also be taken into consideration. However, already the inclusion of human actors into EA models will shed light onto those actually unobserved areas of the socio-technical system enterprise. In order to answer these questions, further research, especially from a holistic, scientific perspective is needed. Only if the needs of today’s practice and the systematic procedures of science are combined, really comprehensive models and methods can be created.

References

1. Lankhorst, M.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, Berlin (2005)
2. Arbab, F., Boer, F.d., Bonsangue, M., Lankhorst, M., Proper, E., Torre, L.v.d.: Integrating Architectural Models - Symbolic, Semantic and Subjective Models in Enterprise Architecture. Enterprise Modelling and Information Systems Architectures 2 (2007)
3. Frank, U.: Multi-perspective Enterprise Modeling (MEMO) - Conceptual Framework and Modeling Languages. 35th Annual Hawaii International Conference on System Sciences (HICSS'02), Vol. 3. IEEE Computer Society, Hawaii, USA (2002) 72
4. Gaertner, W.: Ansatz für eine erfolgreiche Enterprise Architecture im Bereich Global Banking Division/Global Transaction Banking IT and Operations der Deutschen Bank. WIRTSCHAFTSINFORMATIK 46 (2004) 311-313
5. Jung, E.: Ein unternehmensweites IT-Architekturmodell als erfolgreiches Bindeglied zwischen der Unternehmensstrategie und dem operativen Bankgeschäft. WIRTSCHAFTSINFORMATIK 46 (2004) 313-315
6. Sinz, E.: Unternehmensarchitekturen in der Praxis – Architekturdesign am Reißbrett vs. situationsbedingte Realisierung von Informationssystemen. WIRTSCHAFTSINFORMATIK 46 (2004) 315-316
7. Scheer, A.-W., Schneider, K.: ARIS — Architecture of Integrated Information Systems. In: Bernus, P., Mertins, K., Schmidt, G. (eds.): Handbook on Architectures of Information Systems. Springer, Berlin (2006) 605-623
8. Johnson, P., Ekstedt, M.: Enterprise Architecture: Models and Analyses for Information Systems Decision Making. Studentlitteratur, Pozkal (2007)
9. Ferstl, O.K., Sinz, E.J.: Modeling of Business Systems Using SOM. In: Bernus, P., Mertins, K., Schmidt, G. (eds.): Handbook on Architectures of Information Systems. Springer, Berlin, Heidelberg (2006) 347-367
10. Fischer, R., Winter, R.: Ein hierarchischer, architekturbasierter Ansatz zur Unterstützung des IT-Business-Alignment. In: Oberweis, A., Weinhardt, C., Gimpel, H., Koschmider, A., Pankratius, V., Schnizler, B. (eds.): eOrganisation: Service-, Prozess-, Market-Engineering: 8. Internationale Tagung Wirtschaftsinformatik 2007, Vol. 2. Universitaetsverlag Karlsruhe, Karlsruhe, Germany (2007) 163-180
11. Zachman, J.A.: A framework for information systems architecture. IBM Systems Journal 26 (1987) 277-293
12. The Open Group: TOGAF "Enterprise Edition" Version 8.1. (2002)

13. United States Office of Management and Budget: FEA Consolidated Reference Model Document Version 2.3. (2007)
14. Winter, R., Fischer, R.: Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. Journal of Enterprise Architecture (2007)
15. IFIPIFAC Task Force: GERAM: Generalised Enterprise Reference Architecture and Methodology Version 1.6.3. (1999)
16. Picot, A., Baumann, O.: The Relevance of Organisation Theory to the Field of Business and Information Systems Engineering. Business & Information Systems Engineering 1 (2009)
17. Dietz, J.L.G., Hoogervorst, J.A.P.: Enterprise ontology in enterprise engineering. ACM Symposium on Applied Computing. ACM, Fortaleza, Ceara, Brazil (2008)
18. Ropohl, G.: Allgemeine Technologie. Eine Systemtheorie der Technik. Hanser, München/Wien (1999)
19. Noran, O.: An analysis of the Zachman framework for enterprise architecture from the GERAM perspective. Annual Reviews in Control 27 (2003) 163-183
20. Leist, S., Zellner, G.: Evaluation of current architecture frameworks. 2006 ACM symposium on Applied computing, Dijon, France (2006) 1546-1553
21. Bernus, P., Nemes, L., Schmidt, G. (eds.): Handbook on Enterprise Architecture. Springer, Berlin, Heidelberg, New York (2003)
22. Schekkerman, J.: How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework. Trafford Publishing, Victoria, British Columbia (2004)
23. Schönherr, M.: Enterprise Architecture Frameworks. In: Aier, S., Schönherr, M. (eds.): Enterprise Application Integration – Serviceorientierung und nachhaltige Architekturen. Gito, Berlin (2004) 3-48
24. Noran, O.: A Meta-Methodology for Collaborative Networked Organisations. Faculty of Engineering and Information Technology, School of Computing and Information Technology. Griffith University, Brisbane (2004)
25. Aier, S., Riege, C., Winter, R.: Unternehmensarchitektur – Literaturüberblick und Stand der Praxis. WIRTSCHAFTSINFORMATIK 50 (2008) 292-304
26. Bieberstein, N., Bose, S., Walker, L., Lynch, A.: Impact of service-oriented architecture on enterprise systems, organizational structures, and individuals. IBM Systems Journal 44 (2005) 691-708
27. Schroth, C.: The Service-Oriented Enterprise. 2nd Workshop on Trends in Enterprise Architecture Research (TEAR 2007), in conjunction with the The 15th European Conference on Information Systems (ECIS 2007), St. Gallen, Switzerland (2007)

28. Mertens, P.: Diskussionsrunde zum Thema „Unternehmensarchitekturen in der Praxis“. WIRTSCHAFTSINFORMATIK 46 (2004) 315
29. IEEE Computer Society: Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Std 1471-2000, New York (2000)
30. Schelp, J., Aier, S.: SOA and EA - Sustainable Contributions for Increasing Corporate Agility. In: Sprague, R.H. (ed.): Forty-Second Annual Hawaii International Conference on System Sciences (HICSS-42). IEEE Computer Society, Waikoloa, Big Island, Hawaii (2009)
31. Bertalanffy, K.L.v.: General System theory: Foundations, Development, Applications. George Braziller, New York (1976)
32. Wiener, N.: Cybernetics or Control and Communication in the Animal and the Machine, Vol. 2. MIT Press (1965)
33. Emery, F.E., Trist, E.L.: Socio-technical systems. In: Churchman, C.W., Verhulst, M. (eds.): Management Science: Models and Techniques, Vol. 2. Pergamon, Oxford (1960) 83-97
34. Emery, F.E., Trist, E.L.: The causal texture of organizational environments. Human Relations 18 (1965) 21-32
35. Smith, A.: An inquiry into the nature and causes of the wealth of nations (1776)
36. Heinrich, B., Bewernik, M.-A., Henneberger, M., Krammer, A., Lautenbacher, F.: SEMPA – Ein Ansatz des Semantischen Prozessmanagements zur Planung von Prozessmodellen. WIRTSCHAFTSINFORMATIK 50 (2008) 445-460
37. Albani, A., Overhage, S., Birkmeier, D.: Towards a Systematic Method for Identifying Business Components. In: Chaudron, M.R.V., Szyperski, C., Reussner, R. (eds.): 11th International Symposium on Component-Based Software Engineering, Vol. 5282. LNCS, Springer Verlag, Karlsruhe (2008) 262-277
38. Aier, S., Winter, R.: Virtuelle Entkopplung von fachlichen und IT-Strukturen für das IT/Business Alignment – Grundlagen, Architekturgestaltung und Umsetzung am Beispiel der Domänenbildung. WIRTSCHAFTSINFORMATIK 51 (2009)
39. Müller-Merbach, H.: OR-Ansätze zur optimalen Abteilungsgliederung in Institutionen. In: Kirsch, W. (ed.): Unternehmensführung und Organisation, Wiesbaden (1973) 93-124
40. Kieser, A.: Abteilungsbildung. In: Frese, E. (ed.): Handwörterbuch der Organisation, Vol. 3. Poeschel, Stuttgart (1992) 57-72
41. Peacock, E., Tanniru, M.: Activity-based justification of IT investments. Information and Management 42 (2005) 415 – 424

III.3 Beitrag: „FAST ACCESS: A System Architecture for RESTful Business Data”

Autor: Sebastian Klöckner
Lehrstuhl WI-SE, Universität Augsburg,
Universitätsstraße 16, D-86135 Augsburg,
Email: sebastian.kloeckner@wiwi.uni-augsburg.de

Erschienen in: Proceedings of the Fifteenth Americas Conference on Information Systems, Association for Information Systems, 6.-9. August 2009, San Francisco, USA: #748 (2009).

Unternehmenssysteme und -architekturen sind Gegenstand einer Vielzahl von Forschungsaktivitäten und insbesondere serviceorientierte Architekturen und Web Services erfahren hierbei große Aufmerksamkeit. Web Services versprechen dabei eine Vielzahl von Vorteilen und werden mitunter als das neue Paradigma für die Entwicklung verteilter Anwendungen und Systeme betrachtet. Allerdings wird ihr Aufbau zunehmend komplex und damit schwerer verständlich für menschliche Entwickler.
Gleichzeitig bevorzugen und verteidigen Entwickler aus dem Bereich der Web-Entwicklung das ressourcenorientierte Paradigma des REpresentational State Transfer (REST). Interessanterweise wurde das Konzept von REST, obgleich es einer der Hauptfaktoren für den Erfolg des World Wide Web (WWW) ist, jedoch bisher kaum für Verwendung bei Unternehmenssystemen und -architekturen in Betracht gezogen. In diesem Beitrag wird daher eine neue Systemarchitektur vorgestellt, die auf den Grundprinzipien von REST sowie dem Schichtenkonzept basiert. Diese Architektur verfügt dabei über eine vereinheitlichte Schnittstelle, kann die Datenschemata der verfügbaren Datenressourcen zur Verfügung stellen, ermöglicht alle grundlegenden CRUD-Operationen und könnte und kann für die Integration von Unternehmenssystemen verwendet werden.

1 INTRODUCTION

Service-oriented architectures (SOA) and especially their implementation with web services have gained broad attention by science and practice as they deliver extended interoperability and the possibility of service composition (Weerawarana, Curbera, Leymann, Storey and Ferguson, 2005) and they have been widely accepted in the research community as a new enterprise systems architecture paradigm (Papazoglou, Traverso, Dustdar, Leymann and Krämer, 2006). Some authors even went as far as to state: “By 2008, SOA will be a prevailing software engineering practice, ending the 40-year domination of monolithic software architecture (0.7 probability).” (Natis, 2003)

But the increasing complexity of web services, caused by the requirements of traditional enterprise computing and its expectations to quality of service (Weerawarana et al., 2005), have been object of criticism, especially from web development community (e.g. (Bosworth, 2004)), which favor a data centric composition model instead of the behavioral aggregation of services. Additionally, the architectural concept of REpresentational State Transfer (REST) had a massive impact on the way information is exchanged worldwide and the Web 2.0 has gained enormous popularity due to its simple structure of use. At the same time several integration projects at the university and in business contexts have shown that access to data of formerly unintegrated systems, even for atomic transactions only, solves a considerable number of integration issues.

But the regularly cited technologies, architectures and applications of Web 2.0, e.g. YouTube, Flickr, etc., usually use the standard functionality offered by HTTP. While adequate for these purposes, especially the type definition of transferred resource representation, which is based on the standardized MIME-definition, is inadequate for business purposes and cannot be easily extended. Therefore it seems to be advisable to develop a new system architecture, which follows the basic principles of REST and uses the concepts of HTTP, but solves the existing downfalls in order to be usable for business purposes.

This paper follows the design science approach of Hevner, March, Park and Ram (2004) and presents a system architecture, which was incrementally refined and tested during various integration projects. It is structured as follows: Section 2 presents related work, the research methodology and a short overview over the basic concepts of REST. Section 3 then illustrates the functional and structural concept of the proposed systems architecture. Section 4 provides conclusions and areas of further research.

2 RELATED WORK

Service-oriented architectures and web services have experienced high interest of industry and science during the last decade. By using standardized technologies and protocols

(UDDI, WSDL, SOAP, WS-BPEL, etc.) (Weerawarana et al., 2005), they are often seen as the new paradigm for building distributed applications and systems. But on the other side, there is also an ongoing debate about the problems of web services. Mainly two arguments are part of the criticism: Web Services and the related technologies are perceived as being very complex (Bosworth, 2004) and are said to disagree with the basic architectural principles of the web (Rosenberg, Curbera, Duftler and Khalaf, 2008).

The resource-oriented architecture of the web, proposed by the REpresentational State Transfer (REST) model and characterized by its imposed constraints (Fielding, 2000), has proven its effectiveness, as the Web works well (Vinoski, 2007). In the recent past the mashup concept, often based on the HTTP-implementation of REST and focused on information sharing and aggregation to support content publishing for a new generation of Web applications, has emerged. “Mashups” have become one of the hottest buzzwords in the Web applications area (Parameswaran and Whinston, 2007), and many companies and institutions are rushing to provide solutions (Yu, Benatallah, Casati and Daniel, 2008). Although many have adopted the (service) mashup concept and recognized its value, realizing the concept is still challenging, and much work remains before mashup applications in a mature stage will be seen (Benslimane, Dustdar and Sheth, 2008).

While there are many successful mashup applications, e.g. Yahoo Pipes, most of them have a read-only interface and are mainly about sharing and aggregation from disparate sources (Yu et al., 2008). A comprehensive overview of the different mashup and social computing applications is given by Parameswaran and Whinston (2007). Newer approaches, like BITE (Curbera, Duftler, Khalaf and Lovell, 2007; Rosenberg et al., 2008), begin to integrate workflow concepts and present a model for integrating REST principles into a simplified workflow language. But as these concepts still use the HTTP methods PUT and DELETE, which are usually not supported by modern browsers anymore, they are only applicable for machine-to-machine communication. In addition, these concepts are mostly based on the available HTTP MIME-types and typically use the generic type “text/xml” whenever “raw data”, as often the case with business data, is present. A dedicated description of transferred data types is at least not mentioned, but would be very helpful in a business environment. Furthermore, the internal structure of the proposed concepts is often not described.

To present state, at least to the knowledge of the authors, no layered architectural concept exists, which offers a unified interface, integrates type definitions of available resources, allows all CRUD-operations on disparate sources and could be and is used for enterprise application integration.

3 RESEARCH APPROACH

In this contribution an architectural system concept for integration of disparate business data is presented. The research approach follows the design science method of Hevner, et. al. (2004). Vaishnavi and Kuechler (2004) define the awareness of problem, suggestion, development, evaluation and conclusion as primary process steps and proposal, tentative design, artifact, performance measures and results as respective outputs of the steps of a design science approach. As shown in the introduction and related work, the web service concept has been criticized due to its complexity and its conflict with the basic architectural principles of the web. REST and mashups on the other side are intensively discussed as an additional concept for integration of distributed hypermedia systems. Therefore, this paper follows these concepts and proposes a system architecture, which further enhances the integration efforts for distributed business data. The presented concept has been incrementally tested, refined and validated during various integration projects and has shown applicability in many different data integration scenarios.

4 OVERVIEW OF REST

The architectural style of REpresentational State Transfer (REST) has been very successful for distributed hypermedia systems in the last years. The most famous example applying the REST architectural style is probably the Hypertext Transfer Protocol (HTTP) (Berners-Lee, Fielding and Frystyk, 1996; Fielding, Gettys, Mogul, Frystyk, Masinter, Leach and Berners-Lee, 1999). According to Fielding (2000), REST is characterized by the following properties:

- Client-Server architectural style
- Stateless communication
- Cache
- Uniform Interface
- Layered System
- Code on Demand

The Client-Server architectural style allows a separation of concern and therefore the involved components can evolve independently. The stateless communication induces better visibility (a single request datum discloses the full nature of the request), reliability (the recovering from partial failures is facilitated) and scalability (a server does not have to manage resources across requests). Caching improves network efficiency as some interactions can be completely eliminated. The uniform interface is the central feature, which distinguishes REST from other network-based architectural styles. According to Fielding (2000), “REST is defined by four interface constraints:

- identification of resources
- manipulation of resources through representation

- self-descriptive messages
- and, hypermedia as the engine of application state”

The layered system constrains the component behavior in a way that each component cannot “see” beyond the immediate layer with which they are interacting and therefore the complexity of the overall system is limited. Finally, the code on demand style, an optional constraint of REST, reduces the number of features required to be pre-implemented by the client.

5 FROM REST TO FAST ACCESS

The properties of the FAST ACCESS system architecture will be shown in the following section. The presentation is based on the three different perspectives of the general systems theory of Ropohl (Ropohl, 1999). These perspectives include the functional, the structural and the hierarchical view. The functional view presents a system as a “black box” and is characterized by certain properties, which can be observed from outside. The structural view shows a system as a whole of interlinked elements. It illustrates the relationship between the elements of the system. And finally the hierarchical view emphasizes the fact that elements of a system can be interpreted as systems themselves and that they might be part of another super-system. For the presented system architecture the hierarchical view is omitted, as it is obvious due to its recursive structure.

6 FUNCTIONAL BEHAVIOR OF THE EXTERNAL INTERFACE (FUNCTIONAL VIEW)

REST and the HTTP with its basic methods, GET, for requesting a resource, POST, for setting of resource, PUT, for updating a resource, and DELETE, for erasing a resource, have proven to work well. From an external perspective, especially the unique interface and the unique identification by URIs are less complex when compared with web services. While these methods are adequate for machine-to-machine communication, they cannot be used for direct human-to-machine interaction, in the sense of regular web users, as most of the prevailing browsers do not support the PUT and DELETE method anymore, at least not in a simple way. The standard procedure for creating or manipulating data is usually the use of an HTML-Form in combination with a HTTP-POST. In consequence, the basic methods of HTTP cannot be used directly in human-machine-interactions (Webuser to Webserver).

Instead the usage of four resources, which emulate and enhance the basic methods of HTTP and have proven to be needed and useful in several integration projects, in combination with HTTP GET and POST are proposed: GET, POST, LIST and SEARCH. These resources are server-side processing components (server-side scripts), which offer the necessary operations and can be used in machine-to-machine as well as, in combination with XSLT, human-to-machine communications. This unified interface in combination with possible

machine-to-machine communication allows a recursive interconnection with other frameworks of this type, as shown in Figure 1.

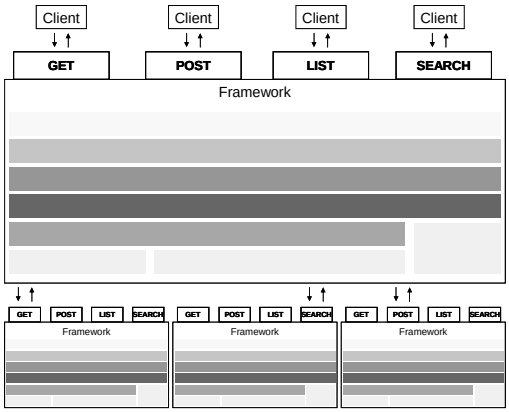

Figure 1: Generic interface of FAST ACCESS

Additionally, HTTP has one advantage, which is often overseen by other REST-based system concepts, although it allows clients type-specific processing: HTTP contains, within others, the Content-Type, based on standardized MIME-types, as header attribute for the description of the requested or delivered data. But for the exchange of business data several, sometimes conflicting, standards exist (RosettaNet, ebXML, UN/EDIFACT, SWIFT). In consequence, a system, which works with business data types, has to somehow disclose its data type structure on which the transferred data instances are based. Therefore, types and instances of these types have both to be "requestable" in the same way as representations are addressed by HTTP. This is achieved by Uniform Resource Identifiers (URI), which are realized through Globally Unique Identifiers (GUID) for type-IDs and instance-IDs. For example, requesting type-IDs results in the schema of the requested data type.

When combining the different required methods and parameters, the following function matrix is created:

FUNCTION		Headerinformation		
		None	Type-ID	Instance-ID
Body Information	GET	Find general Type Information	Find defined Type Data	Find defined Instance Data
	POST (Data)	Create Type	Update Type / Create Instance	Update Instance
	POST (No Data)	INVALID	Delete Type	Delete Instance
	LIST	Find all Types	Find all Instances	Nothing (Redirect to GET)
	SEARCH (Get)	Find Basic Types	Find type definition	Nothing (Redirect to GET)
	SEARCH (Post)	Find fitting Types	Find fitting Instance	INVALID

Figure 2: Function Matrix based on required methods and attributes

The matrix above presents the required functions, which have to be available in order to enable a simple method for exchanging data. The characteristics of each function

(method/parameter combination) will be explained in the following sections. As username and password of the respective clients are an obligatory part of the QueryString for each request, they are not mentioned again in the explanation of the behavior of the respective resources.

6.1 Behavior of the GET-Resource

The GET-Operation (GET-resource is the target of the request) can only be called with an HTTP-GET and accordingly only QueryString-Data is allowed. A HTTP-POST with data, while possible, is ignored in order to keep the principles of REST. A request to the GET-resource without any attributes does not indicate which type or instance is requested. Accordingly this operation can be used for information about type creation. Consequently, as response to a request of this kind, the general structure of types is delivered. When the GET-resource with a type-ID as QueryString is called, the system has to find the corresponding type definition (schema) and send it as response to the client. In case of a request to the GET-resource with an instance-ID as QueryString, the system has to find the matching instance data and send it together with the corresponding type-ID as response to the client. Requests for unknown types or instances are answered with an error.

6.2 Behavior of the POST-Resource

The POST-operation, where the POST-resource is the target of the request, can only be called with a HTTP-POST. The HTTP-GET is not supported and consequently responded with an error. But as HTTP-POST can contain additional data in the request body, two cases have to be differentiated, which are explained next:

6.2.1 POST-Operation with Data

When the POST-resource is requested and data is sent as part of the request body, three different cases can occur: First, if the POST-resource is requested without any type- or instance-ID, a new type has to be created as otherwise a type-ID (type-update) or an instance-ID (instance-update) is required for type or instance identification. Accordingly a new type-ID and information about the created type schema is delivered as response. Second, the POST-resource is called with a type-ID as parameter. This kind of request has two faces: It can indicate that a type has to be updated, but it can also indicate that a new instance has to be created. Which kind of operation is requested can only be decided by taking the content of the request-body into account. If the body contains type information, a type update is demanded and the following response will contain the updated type information. If instance information is sent, a new instance has to be created and the response will contain the created instance data including the new instance identifier. Finally, if the POST-resource is requested with an instance-ID as parameter an update of the defined instance has to occur. As response the updated instance is delivered.

6.2.2 POST-Operation without Data

A request to the POST-resource could also occur with no data sent in the message body. In this case also three different situations could arise: If the POST-resource is called with neither a type- or instance-ID nor any data in the request body, it cannot be decided which action to take. Therefore this kind of request is invalid and answered with an error. A request with a type-ID and an empty request-body must be interpreted as a call for a deletion of a type, as the type is specified and the content of the type shall be empty. Consequently, the response to such a request, a deletion confirmation for that type is sent. And finally, when then POST-resource is request with an instance-ID, this must be understood as a request for the deletion of the specified instance, which is answered by a confirmation, that this instance has been deleted.

6.3 Behavior of the LIST-Resource

In the HTTP-implementation of REST a LIST-operation is realized by using a HTTP-GET with a collection (folder) as URL. In response the client gets a listing of all available resources (representations) in this folder, which can contain all types of representations. A listing by representation type, as far as resources are not sorted by types, is not possible. For the purpose of business data a listing of available types and instances of a certain type is required. Therefore the following functionality is proposed: When the LIST-Resource is called with an HTTP-GET without any QueryString-arguments, the resource generates a complete list of available types as response. The client can then ask for a listing of the available instances of a certain type by calling the LIST-resource with the corresponding type-ID or request the structure of the desired type by using the GET-Resource in combination with the related type-ID. In case of a request for the LIST-resource with a type-ID as QueryString, the resource delivers all available instances of that type as response. The client can then call data of a certain instance by using the GET-resource in combination with the instance-ID of the desired instance. And finally, a call to the LIST-resource with an instance-ID as QueryString would provide the data of the specified instance. But as this functionality is already covered by a call to the GET-resource with an instance-ID as QueryString, this call is redirected to the GET-resource.

6.4 Behavior of the SEARCH-Resource

RESTful applications usually provide a way to search for specific instances. This search “function” is often realized by a special “subfolder” /search/<type> in the URI (e.g. (Mäkeläinen and Alakoski, 2008)). In order to use this functionality, the knowledge of the type schema is required in order to specify which attribute of the type should conform to a specific value during the search. Additionally, this kind of search does not allow a search for available types, as the type has to be defined for a search. FAST ACCESS takes a different approach

by differentiating between HTTP-GET and POST requests as well as the transferred values. The different combinations of HTTP-method and passed arguments are explained in the following:

6.4.1 SEARCH-Operation with Get

If the SEARCH-resource is requested with a HTTP-GET without parameters, it has to be interpreted as a demand for available attribute types in order to define the value of a certain attribute in a subsequent POST to the SEARCH-resource. Accordingly, all available basic attributes are returned as response. When the SEARCH-resource is called with a type-ID as parameter, the corresponding type schema is awaited as response. This can then be used to specify a certain value of the schema for the subsequent POST to search for the corresponding instances. And finally, the search for an instance, where instance-ID is sent with the request, would result in one specific instance. Being identical to a request to the GET-resource with an instance-ID as parameter, such a request is redirected to the GET-resource.

6.4.2 SEARCH-Operation with Post

A call to the SEARCH-resource with a HTTP-POST implies that the data transmitted to the framework has to be interpreted as values of the search parameters. As in the other cases three different types of calls to this resource are conceivable: First, when the SEARCH-resource is called with an HTTP-POST and no type- or instance-ID is transmitted, a search for all types, which match with the transmitted data, has to be executed. The response to such a request is a list of types, which fit the search criteria. When a type-ID and data is sent with an HTTP-POST to the SEARCH-resource, the attached data has to be interpreted as search criteria for instances conforming the type-ID. In response to such a request all fitting instances are listed. And, as in the case of a call to the SEARCH-resource with a HTTP-GET and an instance-ID, a POST to this resource is also redirected to the GET-resource.

The presented resources and their respective behavior cover all necessary CRUD-functions for types as well as instances and, while other behaviors are also conceivable, offer great flexibility for data exchange and integration. Furthermore, the LIST- and SEARCH-resources offer additional functionality, which is regularly needed in business environments. The following section presents the structural concept of the framework, which enables the behavior of the presented resources.

7 STRUCTURAL CONCEPT (STRUCTURAL VIEW)

The architectural structure of FAST ACCESS follows a layered architecture model in order to reduce complexity, as each layer has a different view on FAST ACCESS’s structural elements, and support uncomplicated modification and extension of certain functions if necessary. As parameters, which are passed between the layers, two standardized, but

extensible one-dimensional arrays are used. And if errors occur during request processing, category-based and partly standardized status codes and descriptions are proposed. In the following the proposed parameter arrays, the different layers of the architecture and the status code concept are presented:

7.1 Communication between layers

While the layered architectural approach offers many benefits, the communication between the layers is an important point for overall functionality and extensibility. In the proposed concept two one-dimensional arrays, the IncomingMasterArray and the OutgoingMasterArray, are passed between the layers. This approach follows the fundamental structure of HTTP-requests and -responses, where the request and the response consist of a header and a body. In the proposed framework, the mentioned arrays have two rows, containing an array with meta data in the first row and an array with the content data in the second row. The contained data is required by the different layers for correct processing, but all layers are able to handle cases of missing data.

The meta data of the IncomingMasterArray contains a type-ID or instance-ID, if they were part of the request, username and password and, in case of a post, the date of the last update of the transferred content data. Additionally it can be extended, if further data is needed by the different layers. The content part of the IncomingMasterArray contains all data that was sent with the incoming request.

The meta data of the OutgoingMasterArray includes all data that is anticipated to be useful for the requesting client. First of all a status code and description of the response is included in order to allow the client an adequate reaction to different anticipated states of responses. In addition the type-ID respectively the instance-ID of the involved type or instance as well as its creation and last alteration date are part of the meta data. The content part of the OutgoingMasterArray contains all data that is sent as response.

7.2 Layers of the architectural concept

As stated above, the structural concept of FAST ACCESS follows a layered approach. These layers include HTTP as transport layer, a request management layer, an authorization layer, a routing layer, a remote connection layer, a local processing layer, an operations management layer and a data management layer. Figure 3 presents the hierarchical concept of the layered approach and each of these layers will be described in the following.

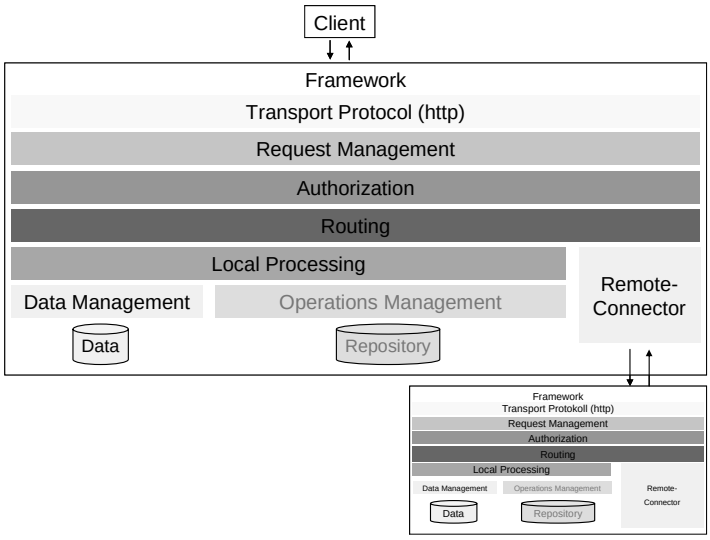

Figure 3: Layered structure of FAST ACCESS

7.2.1 HTTP-Protocol as transport layer

The HTTP-protocol is proposed as basic transport layer. But as modern browsers usually only support the GET- and POST-methods of the HTTP-protocol, only these methods of the protocol are used.

7.2.2 Request Management Layer

The request management layer is responsible for analyzing the incoming requests and routes these to the corresponding functions of the lower architectural layers. The layer itself consists of the four presented resources (GET, POST, LIST and SEARCH) and realize the logical “front-end” of the framework. In the first step the request management layer creates the IncomingMasterArray, which is used as parameter transferred between layers. The meta data of the incoming request is copied to a second array and then placed into the first row of the IncomingMasterArray. The content data of the incoming is also copied to an array and is placed into the second row of the IncomingMasterArray. Based on the incoming meta data the respective resource calls the appropriate function of the next layer with the IncomingMasterArray as parameter.

The result of the respective functions is the OutgoingMasterArray. The respective resources render the data of this array to XML and place a reference to the corresponding XSL-resource in order to make a rendering for a human user by a browser possible. The request management layer does not interpret any of the OutgoingMasterArray data.

7.2.3 Authorization Layer

The authorization layer is in charge of validating the authentication and subsequently checking the authorization of the incoming request respectively the outgoing response. Each function of the authorization layer gets the IncomingMasterArray as input from the request management layer, but depending on the requested function the authorization is verified

before or/and after calling the function of the next layer. The authentication is based on the username und password, which is passed as part of the meta data of the IncomingMasterArray. The authorization validation is then executed based on the group membership of the user and the requested operation based an access control lists (ACLs) for types and instances. The default authorization strategy is a white list, meaning that access is denied as long as a certain user is not member of the respective group.

As already mentioned above, the authorization has to be checked before or/and after calling the function of the next layer. If a certain instance is requested, the access authorization to this instance can be immediately determined. A list or search request cannot be checked a priori, as the layer does not know which types or instances are delivered as result. Therefore, the authorization inspection has to occur after the response from the next layer. In the case of list or search requests, unauthorized types or instances are deleted from the OutgoingMasterArray and the array is then passed back to the Request Management Layer.

7.2.4 Routing Layer

The routing layer is responsible for directing requests and integrating local and remote data. Based on the requested type or instance of a GET- or POST-operation, the request is routed to the remote connector, if another framework is responsible for the type or instance. If the local framework is in charge for the requested operation, the request is forwarded to the local processing layer. In the case of a LIST- or SEARCH-request, the routing becomes a bit more complex. The request is sent to all, local and remote frameworks (local processing and remote connector layer), which are known for being responsible for the requested list or search criteria. The responses of the involved frameworks are then merged and then passed back to the authorization layer. In the actual version of the framework only one framework, local or remote, can be in charge of a type and the corresponding instances. This is due to the fact that the routing target of a create-instance-operation cannot be resolved if more than one framework is responsible for a certain type.

7.2.5 Remote Connection Layer

The remote connection layer establishes the connection to other known frameworks and therefore realizes the recursive structure of the overall framework concept. The layer has to fulfill the tasks of the preceding layers in a reverse order as it has to consolidate the data for making a valid request to the next framework. Therefore, it first replaces the user identification data (username and password) of the IncomingMasterArray with the username und password of the local framework. This follows the idea that each framework is an entity on its own, as it is standard in a business environment, when a company requests a delivery from its supplier. Subsequent it translates the identifier of the type or instance of the incoming request into the type or instance identifier of the remote framework. Finally, the HTTP-request to the corresponding resource is prepared and executed. The response to

GET- and POST-operations are immediately converted into the OutgoingMasterArray and passed back to the routing layer. Responses to SEARCH and LIST-requests are analyzed first, as new types or instances at the remote framework could be part of the response. If new types or instances are part of the response, they are registered to local framework as being known types or instances of the corresponding remote framework. This is required as a subsequent request of the original client could ask for one of the previously unknown types or instances. If they were not registered, the routing layer would answer such a request with a type or instance unknown error.

7.2.6 Local Processing Layer

The local processing layer handles all operations, which are determined to be requests to local resources. The layer decides if the basic CRUD-operations can be read/written directly from/to the database or if previous data processing, for example sending an email to third party in case of an instance update, is necessary. This processing can either be a local function or a known web service and therefore, the processing layer can also be seen as connector to regular web service architectures. In order to be able to determine whether additional processing is required, each defined type can have exactly one target function for each CRUD-operation. If no function is defined for a certain operation, the request is transferred to the data management layer for read or write. If a function for prior processing is set, the request is forwarded to the operations management layer.

7.2.7 Operations Management Layer

The operations management layer gets all requests, which require further processing previous to a corresponding response. As this further processing can become very complex, because it could consist of one or more activities, could be processed locally or remote and could be executed by humans or machines (e.g. regular web services), the detailed structure and functionality of the operations management layer is described in an upcoming paper. Nonetheless, even without the operations management layer, the proposed architecture can already be used, when previous processing is not necessary or this processing is executed by the database management system.

7.2.8 Data Management Layer

The data management layer is responsible for reading and storing requested and transmitted data from and to the database. In case of a read, the layer selects the requested type or instance, creates the OutgoingMasterArray and returns it as result of the call. If a write-operation, thus a create, update or delete, is requested, the data management layer first checks if the transmitted data is valid for the given type. If the passed data is valid, it is written to the database, the OutgoingMasterArray is created and returned as result.

Otherwise the data management writes an error into the OutgoingMasterArray and returns it to the caller.

As almost all layer layers need to validate, read or write data in order to fulfill their responsibilities, the overall data structure for all layers of this version of FAST ACCESS is presented in Figure 4:

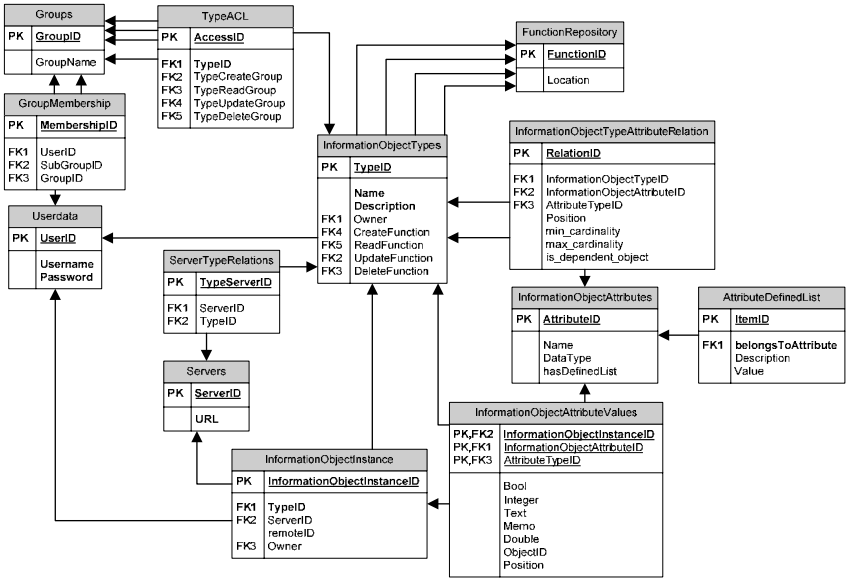

Figure 4: Data Schema of FAST ACCESS

7.3 Status Codes

Whenever functions are executed, errors are likely to occur. As in each of the presented layers errors can occur during processing, a standardized list of possible status categories and defined status codes is proposed. These status codes are sent to the requesting client as part of the response header and indicate if an error occurred during request processing. In order to avoid mistakes, which were made during the development of the HTTP-protocol, for each layer a certain category is proposed, which contains certain status codes for errors, which could be anticipated, but also additional codes for possible errors found in the future. This concept allows clients to react in a specific way, if known errors occur, while unknown errors of a certain category can be handled by default actions.

8 CONCLUSIONS AND FURTHER WORK

The presented system architecture allows the virtual integration, including all CRUD-operations, of data from various sources and has already proven its applicability in many integration projects in university as well as business contexts. Especially in cases where the existing application could not be replaced and had to stay fully functional, FAST ACCESS delivered, due to its unified interface, a comfortable and reliable way to gain access to the underlying data based on atomic transactions. A realization of the same features based on WS-technology would have been much more complex and time-consuming, as for each data

entity an own web service would have been necessary. But following Markus, Majchrzak and Gasser (2002): “Only the accumulated weight of empirical evidence will establish the validity” of the proposed system architecture.

When recursively coupled with additional frameworks, the depth is almost unlimited. Additionally, when the data management layer is replaced by specialized connectors, for example for MS Access, MySQL, SQL Server, almost all kinds of data sources, even excel files, can be made available for further use, which is one of the most frequent problems within daily business. The authorization layer, while having to be configured, protects confidential data from unauthorized access.

Nonetheless, it has to be mentioned that data integration based on atomic transactions alone is not applicable in all types of business scenarios. Therefore, additional research is necessary to present ways how the local processing layer and operation management layer have to be structured in order to allow processing before or after data storage or retrieval by the data management layer. This enhancement would also allow the integration of workflow patterns. But if these workflows contain machines and humans as executor of functionality (activity), two other components are necessary: a workflow engine for machine processing and a portal component for human interaction.

REFERENCES

1. Benslimane, D., Dustdar, S. and Sheth, A. (2008) Services Mashups: The New Generation of Web Applications, IEEE Internet Computing, 12, 5, 13-15.
2. Berners-Lee, T., Fielding, R. and Frystyk, H. (1996) "Hypertext Transfer Protocol -- HTTP/1.0", http://www.ietf.org/rfc/rfc1945.txt.
3. Bosworth, A. (2004) "ICSOC 2004 keynote talk", http://adambosworth.net /2004/11/18/iscoc04-talk/.
4. Curbera, F., Duftler, M., Khalaf, R. and Lovell, D. (2007) Bite: Workflow Composition for the Web, In: Conference Proceedings of the International Conference on Service-Oriented Computing – ICSOC 2007, 94-106.
5. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T. (1999) "Hypertext Transfer Protocol -- HTTP/1.1", http://www.w3.org/Protocols/rfc2616/rfc2616.html.
6. Fielding, R. T. (2000) Architectural Styles and the Design of Network-based Software Architectures, University of California, Irvine.
7. Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004) Design Science in Information Systems Research, MIS Quarterly, 28, 1, 75-105.

8. Mäkeläinen, S. and Alakoski, T. (2008) Fixed-Mobile Hybrid Mashups: Applying the REST Principles to Mobile-Specific Resources, In: Conference Proceedings of the Conference on Web Information Systems Engineering - WISE 2008, 172-182.
9. Markus, M. L., Majchrzak, A. and Gasser, L. (2002) A Design Theory For Systems That Support Emergent Knowledge Processes, MIS Quarterly, 26, 3, 179-212.
10. Natis, Y. V. (2003) "Service-Oriented Architecture Scenario", http://www.gartner.com/resources/114300/114358/114358.pdf.
11. Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F. and Krämer, B. J. (2006) Service-Oriented Computing Research Roadmap, In: Conference Proceedings of the Dagstuhl Seminar Proceedings 05462, Service Oriented Computing (SOC).
12. Parameswaran, M. and Whinston, A. B. (2007) Social Computing: An Overview, The Communications of the Association for Information Systems, 19, 762-780.
13. Ropohl, G. (1999) Allgemeine Technologie - Eine Systemtheorie der Technik, Hanser, München.
14. Rosenberg, F., Curbera, F., Duftler, M. J. and Khalaf, R. (2008) Composing RESTful Services and Collaborative Workflows: A Lightweight Approach, IEEE Internet Computing, 12, 5, 24-31.
15. Vaishnavi, V. K. and Kuechler, W. (2004) "Design Research in Information Systems", http://www.isworld.org/Researchdesign/drisISworld.htm.
16. Vinoski, S. (2007) REST Eye for the SOA Guy, IEEE Internet Computing, 11, 1, 82-84.
17. Weerawarana, S., Curbera, F., Leymann, F., Storey, T. and Ferguson, D. F. (2005) Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More, Prentice Hall PTR.
18. Yu, J., Benatallah, B., Casati, F. and Daniel, F. (2008) Understanding Mashup Development, IEEE Internet Computing, 12, 5, 44-52.

IV Fazit und Ausblick

IV.1 Fazit und weiterer Forschungsbedarf

Die hier vorgestellten Beiträge haben in mehrerlei Hinsicht zu einem Erkenntnisfortschritt in der Wirtschaftsinformatik, insbesondere im Bereich der komponenten- und serviceorientierten Entwicklung und Adaption betrieblicher Anwendungssysteme, beigetragen. Es konnte unter anderem gezeigt werden, dass die komponenten- und serviceorientierte Entwicklung und Adaption betrieblicher Anwendungssysteme (Turowski 2003; Weerawarana et al. 2005) durchaus das Potenzial besitzt, das von Brooks (1987) beschriebene Problem der Komplexität von Softwaresystemen zumindest teilweise zu reduzieren bzw. umzuverteilen und damit dem „Align and Enable“ (Hanschke 2009) einen Schritt näher zu kommen. Allerdings wurde gleichzeitig deutlich, dass für eine Erreichung dieses Ziels und der damit verbundenen vollständigen Praxistauglichkeit der jeweiligen Ansätze noch weiterer Forschungsbedarf besteht.

Die Kernthemen dieser Arbeit waren dabei die Evaluation von Methoden sowie die Präsentation neuer Strukturen bzw. Architekturen in den Forschungsfeldern der Unternehmensmodellierung und des Software Engineering. Die Beiträge ordnen sich damit direkt in die von der Wissenschaftliche Kommission Wirtschaftsinformatik (WKWI) beschlossenen Hauptaufgaben und Ziele (Wissenschaftliche Kommission Wirtschaftsinformatik 1994) ein.

Zwar konnte das von Parnas beschriebene Problem, dass die herkömmliche Softwareentwicklung keine zuverlässigen Programme liefern kann (Parnas 1985, S. 1329), nicht gelöst werden. Jedoch haben die Beiträge einen Anteil zur Verbindung der von Picot und Baumann (2009) geforderten Verknüpfung von Organisationswissenschaften und der Wirtschaftsinformatik, im Speziellen des „Business and Information Systems Engineering“, geliefert. Hierbei wurden sowohl interorganisationale als auch systemtheoretische Aspekte im Sinne sozio-technischer Systeme genauer beleuchtet. So konnte vielleicht ein weiteres Element zur Realisierung des von Dietz und Hoogervorst (2008) geforderten Enterprise Engineering zur Verfügung gestellt werden. Darüber hinaus könnte insbesondere der in Beitrag III.3 dargestellte, REST-basierte Architekturansatz einen möglichen Lösungsweg für das von Parnas aufgezeigte Problem darstellen.

Die ersten vier Beiträge dieser Arbeit haben sich mit der Evaluation von Methoden aus den Bereichen der Unternehmensmodellierung und des Software Engineerings beschäftigt. Hierbei hat sich gezeigt, dass die bisher zur Verfügung stehenden, und mitunter in der Praxis verwendeten Methoden nur bedingt ihren Aufgaben und Versprechen gerecht werden können und bei nahezu allen Methoden weiterer Forschungsbedarf besteht.

Beitrag II.1 hat gezeigt, dass es empirisch nicht widerlegbar ist, dass UML Aktivitätsdiagramme (UML AD) für Fachanwender nicht mindestens genauso gut verwendbar (usable) sind wie Business Process Modelling Notation (BPMN). Gleichzeitig wurde gezeigt, wie die Verwendbarkeit (Usability) von Modellierungssprachen für Fachanwender theoretisch begründet, empirisch erhoben und gemessen werden kann. Da diese Untersuchung allerdings nur für UML AD und BPMN durchgeführt wurde, besteht weiterer Forschungsbedarf im Hinblick auf andere Modellierungssprachen. Die Daten für die Modellierungssprachen UML AD, BPMN, eEPK, Petrinetze und Fachnormsprache wurden im Rahmen der Gesamtstudie bereits erhoben, jedoch steht die Auswertung dieser noch aus.

Beitrag II.2 hat gezeigt, dass die Business Component Identification (BCI) grundsätzlich für die Identifikation von Komponenten im Bereich des CRM und SCM geeignet ist, auch bei einer Vielzahl von Funktionen und Informationsobjekten. Es zeigte sich aber gleichzeitig, dass zufällige Startkonfigurationen zu unterschiedlichen Ergebnissen führen können. Eine genauere Betrachtung offenbarte, dass zusätzliches Wissen über funktionale Abhängigkeiten, abgebildet im Rahmen der Startkonfiguration, einen Einfluss auf die Ergebnisse der BCI-Methode hat. Dementsprechend besteht weiterer Forschungsbedarf im Hinblick auf den Einfluss von Beziehungen zwischen Informationsobjekten, der Kommunikationsintensität zwischen Komponenten und der Berücksichtigung realer Geschäftsprozesse sowie deren Beziehungen auf das Ergebnis der Methode. Zusätzlich sollte überprüft werden, ob die Anzahl der möglichen Lösungen durch den verwendeten Algorithmus beschränkt wird.

In Beitrag II.3 und II.4 wurden in der Literatur vorhandene Ansätze zur Identifikation von Services genauer untersucht und im Hinblick auf ihre Stärken und Schwächen analysiert. Beitrag II.3 stellt dabei den ersten Beitrag zu diesem Themengebiet dar und wurde auf der Fachtagung „Modellierung betrieblicher Informationssysteme 2008“ (MobIS 2008) präsentiert und mit einem Best Paper Award ausgezeichnet, aber mitunter auch kritisiert. In Beitrag II.4 wurde versucht, die kritisierten Schwächen von Beitrag II.3 zu beheben sowie die vergleichende Diskussion der Ansätze auszuweiten und die zu ziehenden Schlussfolgerungen expliziter darzustellen. Im Rahmen der Beiträge wurden zunächst grundlegende Kriterien zur Einordnung der verschiedenen Ansätze entwickelt und dann die vorhandenen Ansätze anhand eines detaillierten Klassifikationsschemas einander gegenübergestellt. Einerseits wurde dabei deutlich, dass die den Ansätzen zugrundeliegenden Servicedefinitionen als auch der jeweilige Formalisierungsgrad stark variieren. Andererseits hat sich gezeigt, dass erheblicher Forschungsbedarf besteht, insbesondere bei der offenen Frage, wie sich die Ansätze zur Identifikation von Services hin

zu ausgereiften Methoden mit einer stärker formalisierten und detaillierten Vorgehensweise weiterentwickeln lassen. Bezeichnend erscheint den Autoren jedoch, dass bei der Identifikation von Services nicht auf bereits bestehende Ansätze der Komponenten-orientierung zurückgegriffen, sondern ganz offenbar wieder von Neuem begonnen wird.

Die letzten drei Beiträge dieser Arbeit haben neue Strukturen bzw. Architekturkonzepte in den Forschungsbereichen der Unternehmensmodellierung und des Software Engineering präsentiert. Hierbei hat sich gezeigt, dass zu den bisher zur Verfügung stehenden, und mitunter in der Praxis eingesetzten, Architekturen durchaus Alternativen bestehen, mit denen sich die zu erfüllenden Aufgaben effizienter realisieren lassen. Da die im Rahmen dieser Arbeit vorgestellten Architekturen jedoch bisher meist nur prototypisch bzw. konzeptionell evaluiert wurden, besteht auch hier weiterer Forschungsbedarf, um diese Architekturkonzepte zu belastbaren und damit vollständig praxistauglichen Architekturen weiterzuentwickeln.

Beitrag III.1 hat dabei gezeigt, dass die Komplexität der Beziehungen, insbesondere im Bereich des Customer Relationship Management (CRM) oder im Supply Chain Management (SCM) bei Unternehmen in Wertschöpfungsnetzen, kontinuierlich zunimmt. Eine Kopplung der SCM-Systeme des Abnehmers und der CRM-Systeme des Zulieferers würde beiden Partnern eine Optimierung ihrer jeweiligen Geschäftsprozesse erlauben. Diese Kopplung wird aber häufig dadurch behindert, dass sowohl Datenstrukturen als auch Funktionen schwierig zu integrieren sind. Im Beitrag wurde daher die Verwendung einer integrierten und komponentenbasierten Informationssysteminfrastruktur für das CRM und SCM präsentiert und dargestellt, dass eine intraorganisationale Integration Voraussetzung für eine interorganisationale Integration ist. Zusätzlich wurde gezeigt, wie diese Architektur in den einzelnen Funktionsbereichen bzw. Szenarien ausgestaltet sein müsste. Die Verwendung der dargestellten Architektur kann dabei die vorhandenen Probleme bei einer interorganisationalen Integration deutlich vereinfachen und gleichzeitig die Probleme des Datenmanagements erheblich reduzieren. Allerdings sind weitere Untersuchungen notwendig, um das vorgestellte Konzept zu einer belastbaren Architektur weiterzuentwickeln und um Wege zu definieren, wie diese in existierende ERP-Systeme integriert werden kann.

Im Rahmen von Beitrag III.2 wurden die bestehenden Konzepte im Bereich der Unternehmensarchitekturen genauer auf bisher nicht berücksichtigte Aspekte untersucht. Auf Basis der Systemtheorie der Technik wurde gezeigt, dass Systeme nicht nur aus technischen Artefakten, die in den meisten Unternehmensarchitekturkonzepten berücksichtigt werden, bestehen, sondern auch menschliche Handlungsträger einen nicht

unerheblich Beitrag zur Funktionsfähigkeit dieser Systeme beitragen. Dieser Relevanz menschlicher Handlungsträger wird bei den meisten Unternehmensarchitekturkonzepten jedoch nicht ausreichend Beachtung geschenkt. Der Beitrag hat daher erste Ansatzpunkte geliefert, wie der menschliche Faktor, als äußerst flexibles und agiles Element im sozio-technischen System „Unternehmen“ und im Rahmen von Unternehmensarchitekturen überhaupt bzw. besser berücksichtigt werden kann. Zusätzlich wurde gezeigt, welche potenziellen Auswirkungen, beispielsweise auf Kennzahlen, eine solche Einbeziehung haben könnte. Gleichzeitig sind dabei allerdings auch neue Forschungsfragen, wie beispielsweise die Einbeziehung des menschlichen Faktors in Identifikationsansätze oder die Optimierung des Verhältnisses zwischen menschlichem Faktor und technischen Artefakten sowie dessen Messung, entstanden, die im Rahmen weiterer Forschungsarbeiten genauer analysiert werden sollten.

Beitrag III.3 hat schließlich eine neuartige Architektur auf Basis der REST-Prinzipien zur Integration verteilter Unternehmensdaten präsentiert. Hierbei wurden zunächst die bestehenden Probleme bei der Integration und unter Verwendung klassischer Web Service sowie die Eigenschaften REST-basierter Architekturen aufgezeigt. Auf Basis dieser charakteristischen Eigenschaften wurden dann die funktionalen als auch strukturellen Merkmale einer Architektur zur Integration von Unternehmensdaten abgeleitet und dargestellt. Aufgrund der dargestellten Architekturmerkmale können atomare Unternehmensdaten deutlich einfacher und schneller integriert werden als mit herkömmlichen WS-Technologien. Da die vorgestellte Architektur bisher allerdings nur atomare Transaktionen unterstützt, müssen in diesem Bereich weitere Lösungsansätze erarbeitet werden. Darüber hinaus sollten Konzepte entwickelt werden, wie beispielsweise Arbeitsabläufe (Workflows), die unter Umständen Menschen und technische Artefakte als Handlungsträger umfassen, in die präsentierte Architektur integriert werden können.

Abgesehen von dem direkt in den einzelnen Beiträgen aufgezeigten Forschungsbedarf, kann festgestellt werden, dass die gestalterische Aufgabe der Softwareentwicklung im komponenten- und serviceorientierten Leitbild und in Verbindung mit Aspekten der Unternehmensmodellierung bei weitem noch nicht vollständig untersucht ist. Wie auch die vorliegende Arbeit gezeigt hat, lässt aber gerade die Verbindung dieser Forschungsrichtungen auf Ergebnisse und Fortschritte hoffen, die die Wettbewerbsfähigkeit von Unternehmen stärken können. Gleichzeitig stellt sie damit aber auch zukünftig ein wissenschaftlich herausforderndes und ergiebiges Forschungsfeld innerhalb der Wirtschaftsinformatik dar.

IV.2 Ausblick

Wie die Realwelt zeigt, stößt die herkömmlicher Entwicklung von Systemen mit den herkömmlichen Methoden, selbst unter Nutzung des komponenten- und serviceorientierten Paradigmas, im Hinblick auf Komplexität, Flexibilität und Agilität scheinbar immer wieder an ihre (Komplexitäts-)Grenzen. So hat sich die Idee von Web Services zwar inzwischen zu einem sehr leistungsfähigen, flexiblen und von der Wissenschaft akzeptierten Konzept (Papazoglou et al. 2006) entwickelt, jedoch wurde dies scheinbar zum Preis einer hohen Komplexität erkauft, wenn inzwischen mehr als 150 WS-*-Spezifikationen existieren (Weerawarana et al. 2005). Ob diese für menschliche Softwareentwickler noch verständlich und nachvollziehbar sind, muss wohl bezweifelt werden.

Parnas (1985, S. 1328) hat bereits 1985 darauf hingewiesen, dass „a fundamental difference that will not disappear with improved technology“ scheinbar das ursächliche Problem für die noch immer vorhandenen Schwierigkeiten ist. Dies würde allerdings bedeuten, dass unter Umständen an einem anderen Punkt angesetzt werden müsste. Hierbei wären mehrere Ansatzpunkte vorstellbar. Einerseits könnte der Wechsel von dem bisher verfolgten Prinzip der Black-Box hin zu einer Grey-Box, beispielsweise in Bereichen, die nicht zu den Kernkompetenzen eines Unternehmens gehören, die Unsichtbarkeit von Softwarekonstrukten zumindest teilweise reduzieren. Darüber hinaus wäre in Domänen, die häufigen Veränderungen und damit Adaptionen der Software unterliegen, unter Umständen ein überdies hinausgehender Wechsel zu dem Prinzip der White Box sinnhaft. Vergleichbare Vorschläge wurden bereits auch von Dietz und Hoogervorst (2008, S. 572) im Hinblick auf das Enterprise Engineering gemacht.

Andererseits könnte unter Umständen auch ein radikaleres Umdenken im Bereich des Software Engineerings, weg von der klassischen objektorientierten, und damit zumeist imperativen, hin zur funktionalen, und damit deklarativen, Programmierung notwendig sein. Dieser Gedanke wurde auch 1978 schon von John Backus geäußert (Backus 1978). In diesem Zusammenhang könnte insbesondere die in Beitrag III.3 präsentierte, REST-basierte Architektur einen Schritt in die richtige Richtung darstellen. Der enorme Erfolg des REST-basierten Hypertext Transfer Protocols (Fielding 2000; Fielding et al. 1999) und des darauf aufbauenden World Wide Webs lassen vermuten, dass bei der Konzeption dieses Systems wohl Merkmale berücksichtigt wurden, die eine flexible und agile Entwicklung und Adaption erlauben. Gleichzeitig ergeben sich bei einem ersten Vergleich der Konzepte erstaunliche Begriffs- und Strukturübereinstimmungen zwischen diesen und den Ideen der konsensorientierten Informationsmodellierung (Becker et al. 2003), der neuen Institutionenökonomie (Picot 1991, S. 80-141), insbesondere der Agenturtheorie, der Verfügungsrechtstheorie und der Transaktionskostentheorie, sowie der Systemtheorie der

Technik (Ropohl 1999) und der funktionalen Programmierung (o.V. 2009), die vorhandene Unsicherheit über Monaden (Wadler 1995) abbildet. Wenngleich erste Übereinstimmungen von Begriffen und Strukturen nicht als wissenschaftlicher Beleg verstanden werden können, so erscheint es dem Autor dennoch ratsam, diese Konzepte und ihre mögliche Vereinbarkeit genauer zu untersuchen. Daher werden die in diesem Ausblick angeführten Konzepte und die ihnen zugrunde liegenden Ideen in einem zukünftigen und bereits in Arbeit befindlichen wissenschaftlichen Beitrag genauer dargestellt.

Literatur

Backus J (1978) Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs. Communications of the ACM 21(8):613-641

Becker J, Holten R, Knackstedt R, Niehaves B (2003) Wissenschaftstheoretische Grundlagen und ihre Rolle für eine konsensorientierte Informations-modellierung. In: Frank U (Hrsg) Tagungsband Wissenschaftstheorie in Ökonomie und Wirtschaftsinformatik 2003 – Theoriebildung und -bewertung, Ontologien, Wissensmanagement, Koblenz

Brooks FP (1987) No Silver Bullet Essence and Accidents of Software Engineering. IEEE Computer 20(4):10-19

Dietz JLG, Hoogervorst JAP (2008) Enterprise ontology in enterprise engineering. In: ACM Symposium on Applied Computing, Fortaleza, Ceara, Brazil

Fielding RT, Gettys J, Mogul J, Frystyk H, Masinter L, Leach P, Berners-Lee T (1999) Hypertext Transfer Protocol -- HTTP/1.1. http://www.w3.org/Protocols/rfc2616 /rfc2616.html. Abruf am 2009-01-31

Fielding RT (2000) Architectural Styles and the Design of Network-based Software Architectures. Dissertation, University of California

Hanschke I (2009) Strategisches Management der IT-Landschaft. Ein praktischer Leitfaden für das Enterprise Architecture Management. Hanser Fachbuch, München

o.V. (2009) Haskell. http://www.haskell.org/. Abruf am 2010-03-08

Papazoglou MP, Traverso P, Dustdar S, Leymann F, Krämer BJ (2006) Service-Oriented Computing Research Roadmap. In: Dagstuhl Seminar Proceedings 05462, Service Oriented Computing (SOC), Dagstuhl, Germany

Parnas DL (1985) Software Aspects of Strategic Defense Systems. Communications of the ACM 28(12):1326-1335

Picot A (1991) Ökonomische Theorien der Organisation. Ein Überblick über neuere Ansätze und deren betriebswirtschaftliches AnwendungsPotenzial. In: Ordelheide D, Rudolph B, Büsselmann E (Hrsg) Betriebswirtschaftslehre und ökonomische Theorie. Schäffer-Poeschel, Stuttgart, 143-170

Picot A, Baumann O (2009) The Relevance of Organisation Theory to the Field of Business and Information Systems Engineering. Business & Information Systems Engineering 1(1):62-69

Ropohl G (1999) Allgemeine Technologie – Eine Systemtheorie der Technik, 2 Aufl. Hanser, München

Turowski K (2003) Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme. Shaker Verlag, Aachen

Wadler P (1995) Monads for Functional Programming. In: Jeuring J, Meijer E (Hrsg) Advanced Functional Programming, First International Spring School on Advanced Functional Programming Techniques. Springer, London, 24-52

Weerawarana S, Curbera F, Leymann F, Storey T, Ferguson DF (2005) Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice Hall PTR, Upper Saddle River

Wissenschaftliche Kommission Wirtschaftsinformatik (1994) Profil der Wirtschaftsinformatik. WIRTSCHAFTSINFORMATIK 36(1):80-81

Publikationen des Verfassers

Wissenschaftliche begutachtete Veröffentlichungen

- „An empirical Comparison of the Usability of BPMN and UML Activity Diagrams for Business Users“, Proceedings of the 18th European Conference on Information Systems (ECIS), Association for Information Systems, 6.-9. Juni 2010, Pretoria, South Africa (2010). Mit Dominik Birkmeier und Sven Overhage.
- „Experience Report: Appropriateness of the BCI-Method for Identifying Business Components in large-scale Information Systems“, Component-Oriented Enterprise Applications: Tagungsband COEA 2005, Lecture Notes in Informatics P-70, Gesellschaft für Informatik, 20. September 2005, Erfurt: 87-92 (2005). Mit Bernhard Selk, Bettina Bazijanec und Antonia Albani.
- „Zur systematischen Identifikation von Services: Kriterien, aktuelle Ansätze und Klassifikation“, Modellierung zwischen SOA und Compliance Management: Tagungsband MobIS 2008, Lecture Notes in Informatics P-141, Gesellschaft für Informatik, 27.-28. November 2008, Saarbrücken: 255-272 (2008); Best Paper Award. Mit Dominik Birkmeier und Sven Overhage.
- „A Survey of Service Identification Approaches – Classification Framework, State of the Art, and Comparison“, Enterprise Modelling and Information Systems Architectures - An International Journal, 4(2): 20-36 (2009). Mit Dominik Birkmeier und Sven Overhage.
- „Enabling Interoperability of Networked Enterprises Through an Integrative Information System Architecture for CRM and SCM“, Business Process Management Workshops: Tagungsband BPM 2005, Springer Lecture Notes in Computer Science 3812, 5. September 2005, Nancy, France: 305-316 (2005). Mit Bernhard Selk und Antonia Albani.
- „Something is Missing: Enterprise Architecture from a Systems Theory Perspective”, Trends in Enterprise Architecture Research (TEAR) @ ICSOC2009: Tagungsband TEAR 2009, Springer Lecture Notes in Computer Science, 23. November 2009, Stockholm, Schweden (2010). Mit Dominik Birkmeier.
- „FAST ACCESS: A system architecture for RESTful Business Data“, Proceedings of the 15th American Conference on Information Systems (AMCIS), Association for Information Systems, 6.-9. August 2009, San Francisco, USA: #748 (2009).
- „Analyse von Risikofaktoren bei der Einführung, Integration und Migration von integrierten Informationssystemen an mittelgroßen deutschen Hochschulen“, Integriertes Informationsmanagement an Hochschulen – Quo Vadis Universität 2.0?: Tagungsband zum Workshop IIM, Universitätsverlag Karlsruhe, 01. März 2007, Karlsruhe: 37-53 (2007). Mit Bettina Bazijanec, Oliver Gausmann, Klaus Turowski und Oliver Beran.

Weitere Veröffentlichungen

- „Processing Fuzzy Information in Semantic Web Applications", Fuzzy Logic and the Semantic Web. Elsevier, Amsterdam: 327-339 (2006). Mit Klaus Turowski und Uwe Weng.