Serviceorientierte Architekturen: Einordnung im Business Engineering

Jan W. Schemm, Roger Heutschi, Tobias Vogel, Kristin Wende, Christine Legner

Bericht Nr.: BE HSG / CC BN3 / 1
Lehrstuhl: Prof. Dr. H. Österle
Version: 1.5.1
Datum: 03.07.2006

Universität St. Gallen - Hochschule für Wirtschafts-, Rechts- und Sozialwissenschaften (HSG)

Institut für Wirtschaftsinformatik
Müller-Friedberg-Strasse 8
CH-9000 St. Gallen
Tel.: ++41 / 71 / 224 2420
Fax: ++41 / 71 / 224 2777

Prof. Dr. A. Back
Prof. Dr. W. Brenner (geschäftsführend)
Prof. Dr. H. Österle
Prof. Dr. R. Winter

Inhaltsverzeichnis

1 Einleitung 6

1.1 Serviceorientierung als Enabler der Vernetzung 6

1.2 Zielsetzung und Aufbau des Arbeitsberichts 9

2 Grundlagen 10

2.1 Gestaltungsebenen des Business Engineering 10

2.2 Architekturen im Business Engineering 10

2.3 Strukturierung der Integrationsarchitektur 11

2.4 Anwendungsbezogene Sicht (Anwendungslogik) 13

2.4.1 Ebene Anwendungssystem 13

2.4.2 Ebene Workflowintegration 15

2.4.3 Ebene Desktopintegration 16

2.5 Anwendungsneutralen Sicht (Integrationsmechanismen) 17

2.5.1 Ebene Anwendungssystem 17

2.5.2 Ebene Workflowintegration 20

2.5.3 Ebene Desktopintegration 22

3 Serviceorientierte Architekturen: Einordnung und Framework 23

3.1 Begriffsbildung 23

3.2 Metamodell 24

3.3 Designprinzipen 31

3.4 Services als Bestandteil einer Integrationsarchitektur 34

3.5 Umsetzung serviceorientierter Architekturen 35

© HSG / IWI / CC BN3 / 1

3.5.1 Web Services 35

3.5.2 Weitere Umsetzungsalternativen 39

3.6 Qualitative Anforderungen an Services in SOAs 39

3.7 Nutzenpotenziale und Herausforderungen 40

4 Ausprägungen des Frameworks in konkreten Fallbeispielen 43

4.1 Serviceorientierte Architektur bei der Deutschen Post Brief 43

4.2 Serviceorientierte Architektur bei Credit Suisse 47

4.3 Serviceorientierte Architektur für einen integrierten Beraterarbeitsplatz bei einer Regionalbank 50

4.4 Web Service Schnittstelle für die zwischenbetriebliche Auftragsabwicklung in der Schweizer Haustechnikbranche 53

5 Zusammenfassung und Ausblick 56

Literatur 57

© HSG / IWI / CC BN3 / 1

Abkürzungsverzeichnis

BAP	Beraterarbeitsplatz
BAPI	Business Programming Application Interfaces
BPM	Business Process Management
CA	Composite Application
CC BN3	Kompetenzzentrum Business Networking 3
CORBA	Common Object Request Broker Architecture
CRM	Customer Relationship Management
DCOM	Distributed Component Object Model
EAI	Enterprise Application Integration
EJB	Enterprise Java Beans
ERP	Enterprise Resource Planning
ETL	Extraction-Transformation-Load
HTTP	HyperText Transfer Protocol
IDL	Interface Definition Language
IS	Informationssystem
IWI	Institut für Wirtschaftsinformatik
J2EE	Java 2 Platform, Enterprise Edition
JDBC	Java DataBase Connectivity
OCL	Object Constraint Language
ODBC	Open DataBase Connectivity
RPC	Remote Procedure Call

© HSG / IWI / CC BN3 / 1

SOA	Serviceorientierte Architektur
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SRM	Supplier Relationship Management
TP	Transaction Processing
UDDI	Universal Description, Discovery and Integration of Web Services
URI	Unified Resource Identifier
W3C	Worldwide Web Consortium
WFM	Workflow Management
WSA	Web Service Architecture
WSDL	Web Services Description Language
XML	Extensible Markup Language

© HSG / IWI / CC BN3 / 1

1 Einleitung

1.1 Serviceorientierung als Enabler der Vernetzung

Mit zunehmender Geschwindigkeit sind Unternehmen heute gezwungen, Anpassungen am Geschäftsmodell vorzunehmen [vgl. Eisenhardt/Brown 1998] und diese organisatorisch umzusetzen. Eine integrierte Systemunterstützung ist häufig notwendig, um in Echtzeit auf Kundenwünsche einzugehen, massgeschneiderte Lösungen anzubieten und diese in enger Kooperation mit den Partnern in der Wertschöpfungskette zu wettbewerbsfähigen Kosten herzustellen. Hier zeigen sich die Grenzen bestehender Architekturen, deren Anpassung oft sehr aufwändig ist. Die Standardisierung und Harmonisierung von Prozessen und Systemen wird zur Voraussetzung, um Komplexität und Kosten beherrschbar zu halten oder gar zu reduzieren. Das Konzept serviceorientierter Architekturen (SOA) sieht vor, Anwendungen oder Anwendungsbestandteile als Dienste zu konzipieren, die sich beliebig verteilen und dynamisch zu Geschäftsprozessen verknüpfen lassen. Prinzipiell spielt es dabei keine Rolle, ob Services von internen Anwendungen bereitgestellt oder extern bezogen werden.

Trotz zahlreicher Publikationen in Wissenschaft und Praxis bleibt das Konzept der Serviceorientierung schwer zu fassen und befindet sich momentan an der Spitze des Hype Cycles im Bereich der Anwendungsintegration [s. McCoy et al. 2004]. Als umfassendes Paradigma wird es momentan aus unterschiedlichen Richtungen diskutiert:

- *Serviceorientierung als Verbindung zwischen Softwareentwicklung und Anwendungsintegration:* Angesichts bestehender Applikationslandschaften ist es kaum denkbar, eine neue Applikation zu entwickeln, ohne Funktionalität aus bestehenden Fachkomponenten einzubinden. Gleichzeitig muss bei der Neuentwicklung von Software bereits an Schnittstellen gedacht werden, die eine zukünftige Nutzung von anderen Anwendungen aus erlaubt. Die Entwicklung neuer Fachfunktionalität (*Programming in the Small*) und deren Integration als Service entlang betrieblicher Prozesse (*Programming in the Large*) liefern ein zweistufiges Programmierungsmodell (*Metaprogrammierung*) mit dem Ziel, dass die IT der Dynamik der Geschäftsprozesse zukünftig besser folgen kann [Kossmann/Leymann 2004]. Wurden bisher die Disziplinen Softwareentwicklung und Anwendungsintegration isoliert betrachtet, rücken diese durch das SOA-Konzept näher zusammen.

© HSG / IWI / CC BN3 / 1

- *Serviceorientierung als umfassender Middlewareansatz:* Unter Middleware verstehen wir nach [Österle 1996b, S. 18] „eine Softwareschicht, welche auf Basis standardisierter Schnittstellen und Protokolle Dienste für eine transparente Kommunikation verteilter Anwendungen bereitstellt." *Klassische Middlewarekonzepte* erleichtert die Entwicklungen von verteilten Applikationen durch Bereitstellung generischer, eher technisch geprägter Funktionalität. Sie verbirgt deren Komplexität durch geeignete Abstraktionskonzepte vor dem Anwendungsentwickler, so dass er sich auf fachliche Aspekte konzentrieren kann [Alonso et al. 2003]. Steht die unternehmensinterne Integration von bestehenden, zum Teil sehr heterogenen Applikationslandschaften im Vordergrund, so kommen umfangreichere Middlewarelösungen, die man als *Enterprise Application Integration* Middleware bezeichnet, zum Einsatz, die auf Nachrichtenaustausch, d.h. (Messaging) Konzepten, basieren. Zunehmend müssen Middleware-Konzepte auch die unternehmensübergreifende Vernetzung abdecken. Die Anforderung an eine *„Business-to-Business" (B2B) Middleware* liegt vor allem in der Implementierung von Businessvokabularen und –protokollen, wie etwa ebXML und RosettaNet.

  Für die Umsetzung einer SOA bedarf es einer ausgewiesenen Middlewareplattform, wenngleich keine grundsätzlich andere Funktionalität und Mechanismen erforderlich sind, wie sie in klassischer, EAI- und B2B-Middleware realisiert sind. So sind es gerade EAI- resp. Workflowkonzepte, die im grossen Umfang Eingang in das SOA-Paradigma gefunden haben. Mit einem auf Standardisierung und Interoperabilität ausgerichteten Technologieportfolio, allen voran die XML- und Web Service Technologien, konvergieren bestehende Middlewarekonzepte und -lösungen im Rahmen einer SOA-Plattform zu einem universellen Middlewareansatz.

- *SOA als Architekturparadigma:* Beim SOA-Konzept handelt es sich um eine (horizontale) Schichtenarchitektur, die Fachfunktionalität von unterschiedlichen Applikationen in einer Serviceschicht kapselt, um diese anschliessend im Rahmen einer Prozessintegration entlang der betrieblichen Abläufe zu verschalten. Etwaige Benutzerinteraktion wird vorzugsweise über eine Portalschnittstelle abgewickelt. Mit Hilfe einer SOA bietet sich ferner eine vertikale Segmentierung von Applikationen und Services an – die Zusammenfassung zu Domänen entlang der fachlichen Zuge-

© HSG / IWI / CC BN3 / 1

hörigkeit, wie etwa die Kundenverwaltung. Services dienen dabei in erster Linie der dömanenübergreifenden Kommunikation. Sowohl die eher technisch motivierte horizontale Schichtenarchitektur als auch die vertikale Segmentierung von Applikationen und Services dient dem Zweck einer Komplexitätsreduzierung in Applikationslandschaften. Insofern lässt sich SOA auch als umfassender Architekturansatz verstehen, der bestehende Architekturmodelle (z.B. das Zachman-Framework oder das Business Engineering) weiterentwickelt und den Übergang von der Prozess- zur Systemarchitektur stärkt.

Für die nachfolgende Diskussion und Einordnung serviceorientierter Ansätze im Business Engineering wird die Architektur- und die Integrationsperspektive im Vordergrund stehen. Das Kompetenzzentrum Business Networking 3 (CC BN3) identifizierte gemeinsam mit Praxispartnern wichtige Fragen der Umsetzung und Gestaltung serviceorientierter Architekturen (vgl. Abbildung 1-1):

- *Künftiges Architektur- bzw. Serviceframework.* Welche Bedeutung hat die Serviceorientierung in künftigen Geschäfts-, Prozess- und Systemarchitekturen? Wie sieht ein künftiges Architektur-Framework aus, das eine Dienste- bzw. Serviceorientierung umsetzt?

- *Neue Geschäftslösungen und Nutzen.* Welche innovativen Lösungen werden durch Serviceorientierung möglich? Was sind gute Beispiele für die Umsetzung Serviceorientierter Architekturen? Welche (Shared) Services entstehen für organisations- und applikationsübergreifende Geschäftsprozesse?

- *Umsetzung einer serviceorientierten Architektur.* Nach welchen Kriterien müssen Services gebildet werden, um Services in unterschiedlichen Kontexten wieder verwenden und zu durchgängigen Lösungen integrieren zu können? Welche Service-Granularität ist sinnvoll? Wie viele Funktionen bzw. Daten muss ein Service enthalten, um eine Wiederverwendung zu ermöglichen und gleichzeitig die Komplexitätsfalle zu umgehen? Wie erfolgt die Migration zu einer serviceorientierten Architektur?

© HSG / IWI / CC BN3 / 1

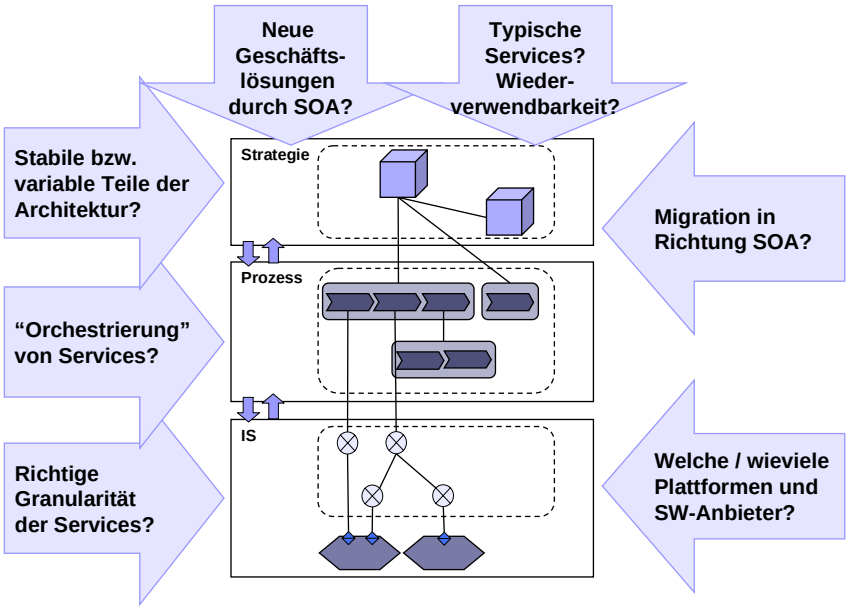


Abbildung 1-1: Fragestellungen rund um serviceorientierte Architekturen

1.2 Zielsetzung und Aufbau des Arbeitsberichts

Ziel des vorliegenden Arbeitsberichts ist es, Hilfestellung bei der Einordnung und Bewertung serviceorientierter Architekturen zu bieten. Er ordnet diese im Kontext des Business Engineering ein und erweitert die Architektur des Echtzeitunternehmens um den Aspekt der Serviceorientierung. Damit bildet er ein Rahmenwerk für die Beschreibung service-orientierter Architekturen und zeigt dessen Anwendung an drei Beispielen.

Kapitel 2 gibt einen Überblick über bestehende Architekturkonzepte im Business Engineering. Als Grundlage für die spätere Einordnung einer SOA auf Systemebene be-schreibt dieses Kapitel Integrationsarchitekturen anhand verschiedener Ebenen und Sichten. Kapitel 3 entwickelt darauf aufbauend ein Rahmenwerk zur Beschreibung serviceorientierter Architekturen. Kapitel 4 schliesst mit der Ausprägung des Rahmenwerks in konkreten Fällen aus der betrieblichen Praxis. Anhang 0 fasst wichtige Begriffsdefinitionen in Form eines Glossars zusammen.

© HSG / IWI / CC BN3 / 1

2 Grundlagen

2.1 Gestaltungsebenen des Business Engineering

Business Engineering bezeichnet eine methoden- und modellbasierte Konstruktionslehre zur Gestaltung von Unternehmen im Informationszeitalter [s. Österle 1995, 13 ff., Österle/Winter 2003, 7 ff.]. Als ganzheitlicher Ansatz betrachtet Business Engineering die Ebenen Strategie, Prozesse und Systeme bei der Ableitung und Realisierung von Geschäftslösungen.

- Die *Geschäftsstrategie* trifft Entscheidungen zur mittel- bis langfristigen Entwicklung eines Unternehmens. Hierzu zählen insbesondere die Positionierung im Wertschöpfungsnetzwerk und die Festlegung der Marktleistungen unter Berücksichtigung von Kundenbedürfnissen und Wettbewerb auf der einen Seite („market based view“) sowie den eigenen Kernkompetenzen („resource based view“) auf der anderen Seite.

- *Geschäftsprozesse* setzen die strategischen Entscheidungen um. Ein Prozess erbringt die Leistungen des Unternehmens und beinhaltet eine Abfolge von Aufgaben. Zentrale Entscheidungen auf der Prozessebene betreffen die Prozessleistungen, die effiziente Abfolge von Prozessschritten sowie Aspekte der Prozessführung.

- *Informationssysteme* konkretisieren Geschäftsprozesse in Form einer detaillierten Spezifikation der automatisierten Informationsverarbeitung. Zur Abwicklung von Geschäftsprozessen kommen Anwendungen bzw. Applikationen zum Einsatz, die eine Menge von Funktionen unter Rückgriff auf Datensammlungen zur Verfügung stellen. Basis für die Implementierung von Applikationen bilden Informationstechnologie-Komponenten in Form von Hardware, Netzwerken oder Systemsoftware.

2.2 Architekturen im Business Engineering

Unter dem Begriff (Unternehmens-)Architekturen werden im Rahmen dieser Arbeit Modelle zur Beschreibung der Gestaltungselemente von Geschäftslösungen und deren Beziehungen zueinander sowie die den Modellen zugrunde liegenden Gestaltungsprinzipien zusammengefasst [vgl. Sinz 1999, 1035, Winter 2004, 315].

© HSG / IWI / CC BN3 / 1

- Die *Geschäftsarchitektur* beschreibt auf der strategischen Ebene den Gesamtzusammenhang der Leistungsverflechtung in einem Wertschöpfungsnetzwerk [s. Winter 2004, 318]. Dazu zählen das Zusammenwirken von Unternehmen bzw. Geschäftsbereichen in Wertschöpfungsnetzwerken sowie das Geschäftsmodell inklusive der Kundenprozesse und -segmente [vgl. Winter 2003, 96 ff., Alt et al. 2004, 24 ff.].

- Eine *Prozessarchitektur* beschreibt den Gesamtzusammenhang der Leistungsentwicklung, -erstellung und des Leistungsvertriebs innerhalb einer Organisation [s. Winter 2004, 318]. Sie umfasst u.a. die Verknüpfung der Prozesse auf Unternehmens- bzw. Geschäftsbereichsebene, Beschreibungen der Leistungen, Führung, Abläufe und Verantwortlichkeiten einzelner Prozesse sowie ein Modell der von den Prozessen benutzten bzw. erzeugten Informationen [vgl. Winter 2003, 101 ff.].

- Auf der Systemebene lassen sich die *Applikations- und Integrationsarchitektur* unterscheiden [vgl. Alt et al. 2004, 40 ff.]. Die Applikationsarchitektur stellt eine logische, funktionale Sicht auf die Anwendungen dar [s. Winter 2003, 107 ff., Alt et al. 2004, 43], während die Integrationsarchitektur die softwaretechnische Umsetzung der Integration ursprünglich separat entwickelter, heterogener Anwendungssysteme beschreibt.

2.3 Strukturierung der Integrationsarchitektur

Für die spätere Betrachtung serviceorientierter Architekturen ist die Integrationsarchitektur von besonderer Bedeutung, so dass wir an dieser Stelle auf deren Aufbau näher eingehen. Die Integration von Anwendungen - seien es alte oder neue, Standardsoftware oder Eigenentwicklungen, vorgefertigte Komponenten verschiedener Hersteller oder ganze Informationssysteme zuvor konkurrierender Unternehmen - stellt einen Erfolgsfaktor für die Anpassung der Unternehmensabläufe an Geschäftsanforderungen dar. Gleichzeitig sind in den letzten Jahren unter den Bezeichnungen Middleware, Enterprise Application Integration (EAI), Workflow Management und Business Process Management verschiedenste Integrationskonzepte diskutiert und in entsprechende Produkte umgesetzt worden.

© HSG / IWI / CC BN3 / 1

Um die Komplexität einer Integrationsarchitektur beherrschbar zu machen und eine verständliche Beschreibung der Architektur zu ermöglichen, strukturiert diese Arbeit die Integrationsarchitektur in verschiedene Ebenen und Sichten. Sie orientiert sich dabei an den Vorarbeiten des Kompetenzzentrums Prozess- und Systemintegration (CC PSI [s. Vogler 2003a, 78 ff.]) und unterscheidet die in Abbildung 2-1 dargestellten Ebenen und Sichten. Eine *Architekturebene* umfasst hier eine vollständige Beschreibung der Architektur unter einem bestimmten Blickwinkel, während *Architektursichten* eine weitere Komplexitätsreduktion durch Projektionen auf Teilausschnitte der Elemente einer oder mehrerer Modellebenen erlauben [vgl. Sinz 1999, 1036 f.].

Die Ebene *Desktopintegration* bringt die zur Erfüllung von Aufgaben notwendigen betrieblichen Anwendungen an einem Arbeitsplatz zusammen. Sie stellt die Sichtweise des Benutzers und nicht so sehr die des übergreifenden Prozesses in den Vordergrund. Dagegen ist die Konzeption und Realisierung der Ablaufsteuerung eines Prozesses Gegenstand der Ebene *Workflowintegration.* Die Ebene *Anwendungssystem* schliesslich bietet mit der Beschreibung und Realisierung sämtlicher betrieblicher Funktionen und Daten in Form von Anwendungssystemen die Grundlage für die beiden anderen genannten Ebenen.

Die *anwendungsneutrale, konzeptionelle Sicht* beschreibt für jede der genannten Ebenen jeweils die grundlegende Anwendungslogik, während die *anwendungsneutrale Sicht* den entsprechenden Integrationsmechanismus erfasst, d.h. die Basiskomponenten, welche Dienste und Protokolle zur Implementierung der Systemintegration bereitstellen [vgl. Mantel et al. 2004, 4]. Ausgewählte Elemente der beiden Sichten sind in Abbildung 2-1 dargestellt. Die folgenden Abschnitte beschreiben die einzelnen Architekturelemente der verschiedenen Ebenen und Sichten detailliert.

© HSG / IWI / CC BN3 / 1

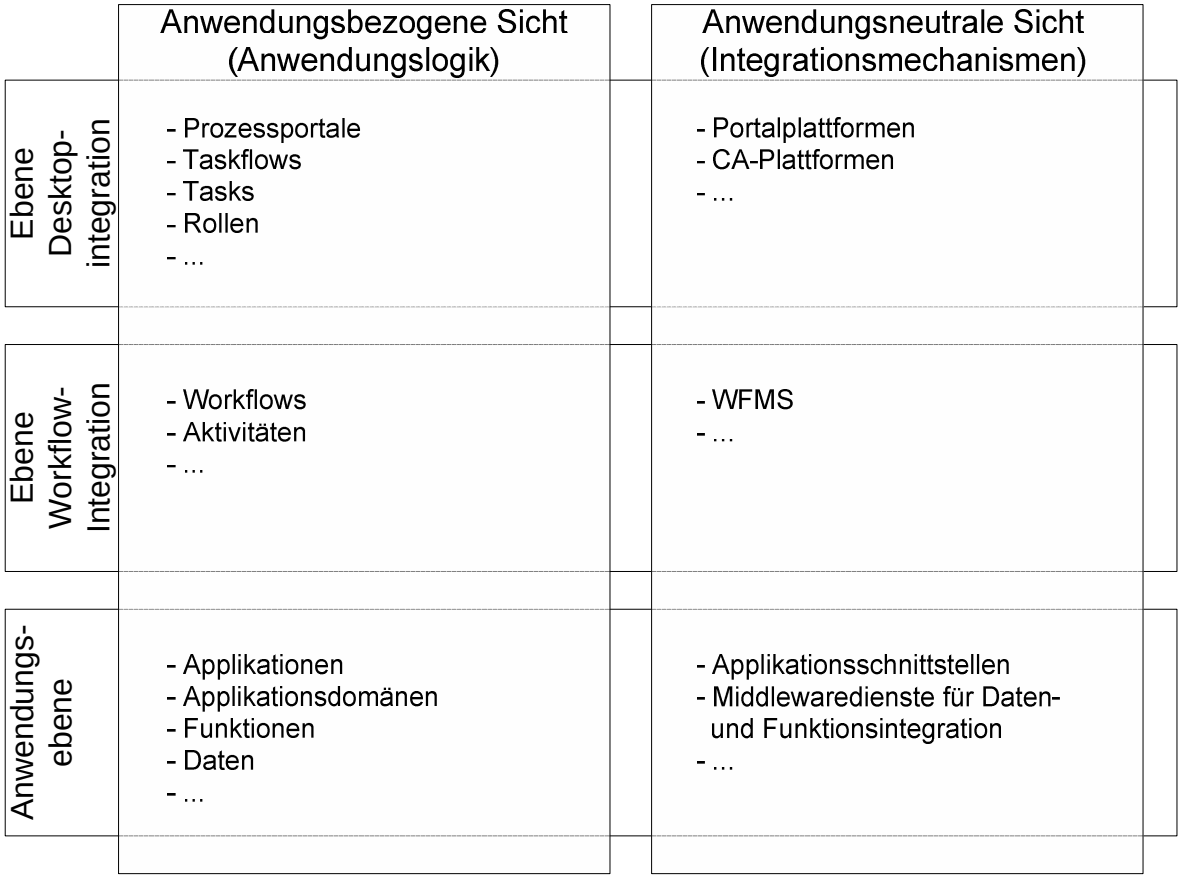


Abbildung 2-1: Ebenen und Sichten zur Strukturierung von Integrationsarchitekturen

2.4 Anwendungsbezogene Sicht (Anwendungslogik)

2.4.1 Ebene Anwendungssystem

Die Ebene Anwendungssystem umfasst betriebliche Anwendungssysteme, die auf Basis von Soft- und Hardwareplattformen Applikationsfunktionen und Datensammlungen zur automatisierten Unterstützung von Geschäftsprozessen implementieren [s. Vogler 2003b, 88f.]. Sie repräsentiert in diesem Modell die Ressourcensicht, welche die in Informations-system vorhandene, potenziell als Services kapselbare fachliche Funktionalität betrachtet, und stellt die Schnittstelle zwischen der Applikations- und Integrationsarchitektur dar.

Moderne Anwendungssysteme sind mehrschichtig aufgebaut. Typischerweise werden die logischen Schichten Präsentation, Applikationsfunktionalität und die Datenhaltung unterschieden, die sich je nach Architektur unterschiedlich in einem Netzwerk verteilen lassen (s. [Alonso et al. 2003, 4ff.], [Riehm/Vogler 1996, 29]). Die Präsentationsschicht ist für die Informationspräsentation und die Steuerung der Ein- und Ausgaben von bzw. an

© HSG / IWI / CC BN3 / 1

Nutzer des Anwendungssystems z.B. über eine grafische Benutzerschnittstelle (Graphical User Interface, GUI) zuständig. Die Applikationsfunktionalitätsschicht implementiert die fachliche Logik des Anwendungssystems in Form von Programmen bzw. Operationen. Die Datenhaltungsschicht verwaltet die vom Anwendungssystem für die Ausführung der Programme benötigten Daten. Abbildung 2-2 zeigt exemplarisch die unterschiedliche Verteilung dieser Anwendungssystemschichten für die Architekturen Mainframe und Client-Server.

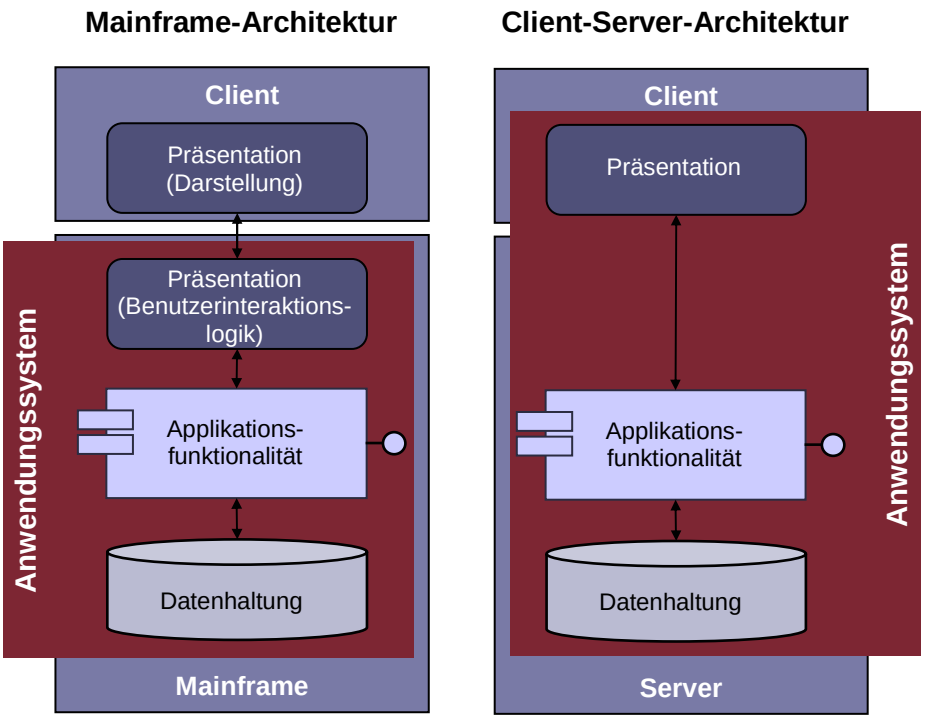


Abbildung 2-2: Verteilung von Anwendungsschichten in Anlehnung an [Alonso et al. 2003, 11f.]

Anwendungssysteme werden in der Applikationsarchitektur oft in Form von Anwendungsdomänen gruppiert [s. Winter/Schelp 2005, 49]. Indem sie die in der Applikationsarchitektur verfügbare Funktionalität anhand ihrer fachlichen Zusammengehörigkeit (z.B. fachliche Verwandtschaft von Funktionen, gemeinsame Nutzung von Informationsobjekten oder Zugehörigkeit zu einer bestimmten Organisationseinheit) gruppieren und möglichst redundanzfrei implementieren [s. Herr et al. 2004, 230], dienen Anwendungsdomänen der fachlichen Strukturierung der Applikationsarchitektur und der Zuteilung organisatorischer Verantwortung für Anwendungssysteme [s. Gallas 2004, 244f.]. Anwendungsdomänen stellen möglichst isoliert betreibbare und verantwortbare, modulare Einheiten dar, die eng

© HSG / IWI / CC BN3 / 1

voneinander abhängige Anwendungen kapseln und Anwendungen anderer Domänen im Idealfall nur über definierte Schnittstellen bzw. Services Zugriff auf die gekapselte Funktionalität bieten (vgl. [Winter/Schelp 2005, 49], [Schlamann 2004, 60]).

2.4.2 Ebene Workflowintegration

Die Workflowintegration umfasst Beschreibungen und Umsetzungen von Maßnahmen zur Steuerung einzelner applikationsübergreifender Prozesse sowie der Zusammenarbeit zwischen Prozessen [vgl. Vogler 2003a, 54]. Um Verwechslungen zwischen Konzepten auf der Prozess- und der Systemebene der Business Engineering Architektur zu vermeiden, verwendet diese Arbeit in Abweichung von [Vogler 2003b, 54] den Begriff „Workflowintegration“ anstatt „Prozessintegration“. Wesentliche Zielsetzung dieser Architekturebene ist die Durchgängigkeit der anwendungsübergreifenden Bearbeitung eines Geschäftsvorfalls, d.h. die Vermeidung von Medienbrüchen bei der Prozessabwicklung sowie die aktive Steuerung der Arbeitsabläufe im Sinne der Koordination von mehreren Benutzern für die Zusammenarbeit in einem Prozess.

Die grundlegende Philosophie der Workflowintegration besteht in der Trennung der Spezifikation von Ablauf- und Anwendungslogik. Typischerweise enthalten Anwendungssysteme neben den Umsetzungen einzelner geschäftlicher Aktivitäten in Form von Algorithmen ebenfalls Implementierungen von Kontrollflüssen, die den Ablauf der einzelnen Algorithmen zur Unterstützung eines bestimmten Prozesses festlegen [vgl. Leymann/Roller 2000, 309]. Im Rahmen einer Workflowintegration ist die Ablauflogik des Integrationssystems hingegen in Form eines operationalisierbaren Prozessmodells definiert. Dieses Prozessmodell spezifiziert die anwendungsübergreifende Abfolge von Informationsflüssen und Funktionsaufrufen zur systemtechnischen Abbildung des zu unterstützenden Geschäftsprozesses. Diese Ebene einer Integrationsarchitektur spezifiziert somit nicht die Integration von Anwendungssystemen auf einer Abstraktionsebene der Daten- oder Annwendungslogik, sondern behandelt ausschliesslich die Definition, Steuerung und Kontrolle des Vorgangs bzw. Prozesses, der die Integration verschiedener Anwendungsysteme erfordert [vgl. Linthicum 2000, 319 ff., Kaib 2002, 119 ff.].

© HSG / IWI / CC BN3 / 1

2.4.3 Ebene Desktopintegration

Die Desktopintegration stellt eine spezielle Form der Workflowintegration dar [s. Vogler 2003a, 54] mit der Zielsetzung, dem Benutzer alle zur Erfüllung einer Aktivität notwendigen Anwendungssysteme an seinem Arbeitsplatz integriert zur Verfügung zu stellen [vgl. Vogler 2003a, 84]. Jeder Benutzer verwendet an seinem Arbeitsplatz zur Erfüllung seiner Prozessaufgaben i.d.R. verschiedene Anwendungen. Eine umfassende Unterstützung des Benutzers erfordert die Integration der beteiligten Anwendungen am elektronischen Arbeitsplatz. Eine desktoporientierte Architekturebene spezifiziert hierzu die Einbindung der verschiedenen Applikationen im Arbeitsablauf, die Darstellung der erforderlichen Informationen in einer einheitlichen Benutzeroberfläche sowie ggf. den Datenaustausch zwischen den Anwendungen am Arbeitsplatz [s. Derungs 1997, 22, Vogler 2003a, 55].

Während eine prozessorientierte Integrationsebene den Ablauf der Aktivitäten innerhalb eines anwendungsübergreifenden Prozesses spezifiziert, beschreibt eine desktoporientierte Ebene die Abfolge einzelner Arbeitsschritte, die ein Benutzer im Rahmen einer Aktivität an seinem Arbeitsplatz durchführt. Für diesen Steuer- und Datenfluss zwischen Arbeitsschritten innerhalb einer Aktivität verwendet [Vogler 2003b, 55] den Begriff Taskflow.

Die Desktopintegration wird heute in der Regel über Portale (in dieser Ausprägung als Prozessportale bezeichnet) realisiert. Diese stellen ein internetbasiertes, personalisierbares und integriertes Zugangssystem zu verschiedenen unternehmensinternen und -externen Informationsquellen und Anwendungsprogrammen (Applikationen). Jeder Nutzer erhält über nur einen einzigen Anmeldeprozess (Single Sign On) und über verschiedene Kanäle (z.B. Internet-Browser oder mobile Endgeräte) Zugriff auf die für ihn autorisierten Funktionen und Inhalte. Diese werden auf einer Portalseite in sog. Portlets dargestellt und durch Personalisierung auf die Bedürfnisse des Nutzers angepasst. Der Taskflow und die Portalleistungen können auf verschiedene Nutzergruppen (Personalisierung durch Kategorisierung) oder auf den einzelnen Nutzer zugeschnitten sein (Personalisierung durch Individualisierung). Bei beiden Stufen der Personalisierung kann diese entweder vom Nutzer selbst (Pull-Personalisierung) oder vom Portalanbieter ausgehen (Push-Personalisierung).

Ein dem Prozessportal vergleichbares Konzept einer desktoporientierten Integrationsarchitektur wird aktuell unter dem Begriff der sog. *Composite Application*s (CA) diskutiert. Eine CA wird im Rahmen dieser Arbeit in Anlehnung an eine Definition der Gartner Group

© HSG / IWI / CC BN3 / 1

definiert als: „A software assembly that implements one business function (one step) and where the component parts are heterogeneous in their information architecture“ [Pezzini 2003]. Die angeführte Definition verdeutlicht, dass es sich beim Konzept der CA um eine Umsetzungsmöglichkeit der Desktopintegration handelt. Eine CA beschreibt eine integrierte Software bzw. Anwendung, die eine geschäftliche Funktion implementiert. Bezogen auf die oben eingeführte Terminologie unterstützt eine CA den Benutzer bei der Ausführung einer geschäftlichen Aktivität. Darüber hinaus besteht eine CA aus Komponenten mit heterogenen Informationsarchitekturen [vgl. Woods 2003, 3 f.], d.h. eine CA integriert heterogene Anwendungssysteme am Arbeitsplatz des Benutzers. Zwecks Zusammenführung einzelner Arbeitsschritte zu einer Aktivität verfügt eine CA ferner über eine gewisse Ablauflogik [vgl. Woods 2003, 4].

2.5 Anwendungsneutralen Sicht (Integrationsmechanismen)

Die Elemente der anwendungsneutralen Sicht setzen die Spezifikation und Realisierung der Zusammenführung heterogener Anwendungssysteme um [vgl. Vogler 2003a, 56 f.]. In Abhängigkeit von der jeweiligen Integrationsebene stehen unterschiedliche Integrationsmechanismen im Vordergrund.

2.5.1 Ebene Anwendungssystem

Basierend auf der durch das Client/Server-Modell definierten Unterscheidung der drei Anwendungsebenen Daten, Funktionalität und Präsentation lassen sich die drei Formen der Daten-, Funktions- und Präsentationsintegration differenzieren. Diese werden in den folgenden Abschnitten mit Bezug zu den jeweils vorherrschenden Integrationsmechanismen kurz dargestellt.

2.5.1.1 Datenintegration

Der zentrale Gegenstand der Datenintegration sind die den jeweiligen Anwendungssystemen zugrunde liegenden Daten. Bei der Datenintegration werden verschiedene Anwendungssysteme dadurch zusammengeführt, dass sie direkt auf die von den jeweiligen Systemen gespeicherten Daten zugreifen [vgl. Ruh et al. 2001, 19 f., Kaib 2002, 63]. Unter Umgehung der Präsentations- bzw. Anwendungslogik ermöglicht diese Form der Integration somit die gemeinsame Nutzung von Daten durch mehrere Anwendungssysteme [s. Mantel/Schissler 2002, 173, Mantel et al. 2004, 15].

© HSG / IWI / CC BN3 / 1

Die datenorientierte Integration basiert entweder auf dem Austausch von Daten bzw. der Replikation oder aber der Datenföderation [vgl. Linthicum 2000, 28 ff., Kaib 2002, 64, Linthicum 2003, 6 ff.]. Beim Ansatz der Datenreplikation werden physische Kopien von Datensätzen, die in den zu integrierenden Datenspeichern redundant vorliegen, (ggf. unter Einbezug von Transformationen) zwischen Datenspeichern ausgetauscht. Beim Ansatz der Datenföderation hingegen werden die Daten virtuell oder logisch integriert. Heterogene Datenquellen werden in diesem Fall (ggf. unter Einbezug logischer Transformationen) in ein einheitliches Datenschema eingebunden. Über diese einheitliche Sicht wird auf die physisch verteilten, nicht-redundanten Datenquellen zugegriffen.

Aufgrund der frühen Entwicklung und weiten Verbreitung der Datenintegration haben sich eine Reihe datenorientierter Integrationsmechanismen herausgebildet, angefangen von einfachen Mechanismen bspw. in Form des Austauschs strukturierter Textdateien, über komplexere Integrationsmechanismen bspw. in Form datenorientierter Middleware wie *Open DataBase Connectivity* (ODBC) / *Java DataBase Connectivity* (JDBC) oder Replikationsmechanismen relationaler Datenbankmanagementsysteme bis hin zu umfangreichen Integrationsmechanismen in Form föderierter Datenbankmanagementsysteme oder *Extraction-Transformation-Load* (ETL)-Werkzeugen im Rahmen von Datawarehouse Lösungen.

2.5.1.2 Funktionsintegration

Im Gegensatz zur Datenintegration basiert die Integration heterogener Anwendungssysteme im Fall der Funktionsintegration auf der Ebene der Anwendungslogik. Das Grundkonzept dieser in unterschiedlichen Variationen auftretenden Integrationsform besteht darin, einzelne Teilfunktionalität von Applikationen nach aussen als Dienste anzubieten und somit die gemeinsame Nutzung von Funktionalität durch mehrere Anwendungssysteme zu ermöglichen [vgl. Ruh et al. 2001, 27 ff., Kaib 2002, 65 ff.].

Funktionsorientierte Integrationsmechanismen werden in der Literatur häufig unter dem Begriff funktionaler Middleware zusammengefasst. Den grundlegendsten Mechanismus zur funktionalen Integration von Anwendungssystemen stellen *Remote Procedure Calls* (RPC) dar. Es handelt sich hierbei um definierte Schnittstellen, die einen transparenten Zugriff auf bestimmte Teile der Anwendungslogik der zu integrierenden Systeme über ein Netzwerk ermöglichen. In der Regel sind RPCs synchron, d.h. die aufrufende Anwendung blockiert, bis die aufgerufene Prozedur beendet ist bzw. ein Ergebnis geliefert hat. Neben dieser Ein-

© HSG / IWI / CC BN3 / 1

schränkung auf synchrone Interaktion wirkt sich negativ aus, dass RPCs fest in den Programmcode der beteiligten Systeme eingefügt werden und somit der Wartungsaufwand von Integrationssystemen mit steigender Anzahl integrierter Systeme entsprechend stark ansteigt [vgl. Schelp/Winter 2002, 12]. Dennoch bilden RPCs die grundlegende konzeptionelle Basis für die im weiteren skizzierten funktionalen Integrationsmechanismen [vgl. Schelp/Winter 2002, Alonso et al. 2003, 44 f.].

Im Falle der Integration mehrerer Systeme mit einer entsprechend komplexeren Folge von Funktionsaufrufen ist eine Synchronisierung der in Form von RPCs aufgerufenen Funktionen nötig, um eine konsistene Integration zu garantieren. *Transaction Processing Monitors* (TP Monitore) stellen vereinfacht ausgedrückt einen Integrationsmechanismus dar, der RPCs um die Möglichkeit der Transaktionsverarbeitung erweitert [vgl. Schelp/Winter 2002, 13, Alonso et al. 2003, 33]. TP Monitore unterstützen die Koordination von Funktionsaufrufen über den Abstraktionsmechanismus der Transaktion und ermöglichen dadurch eine vereinfachte und konsistente funktionale Integration von Anwendungssystemen.

Zur Zeit der Entwicklung von Integrationsmechanismen wie RPC oder TP Monitoren bildeten imperative Sprachen das dominante Programmierparadigma. Mit der Durchsetzung des objektorientierten Paradigmas wurde ein entsprechender Integrationsmechanismus in Form des *Object Brokers* entwickelt. Ein Object Broker kann als funktionale Middleware zur Unterstützung der Interoperabilität verteilter Objekte aufgefasst werden [s. Alonso et al. 2003 ff.]. Object Broker bilden das objektorientierte Gegenstück zu RPCs bzw. TP Monitoren, indem sie den transparenten, ggf. synchronisierten Aufruf von Methoden verteilter Objekte ermöglichen.

Sog. *Message Broker* bieten in Ergänzung der skizzierten Integrationsmöglichkeiten weitere Funktionalität zur loseren Kopplung der integrierten Systeme. Einerseits ermöglicht diese Form des Integrationsmechanismus zusätzlich zur synchronen auch die asynchrone Kommunikation zwischen den zu integrierenden Anwendungen durch den Nachrichtenaustausch über eine Warteschlange. Die beteiligten Systeme müssen somit weder laufend aktiv sein, noch muss ein System bei Anforderung einer bestimmten Funktionalität eines anderen Systems bis zum Erhalt des Ergebnisses blockieren. Andererseits ermöglichen es Message Broker, über Routing-, Filter- und Transformationsmechanismen einen Teil der Integrationslogik zu zentralisieren und die Interaktion zwischen den Anwendungssystemen somit

© HSG / IWI / CC BN3 / 1

dynamischer zu gestalten [vgl. Alonso et al. 2003, 71 ff.]. So müssen bspw. die Empfänger bestimmter Nachrichten nicht in den Anwendungen kodiert werden und dynamische Gruppenkommunikationen sind etwa durch das Publish/Subscribe-Modell [s. Alonso et al. 2003, 75 ff.] einfach umsetzbar.

Application Server schliesslich, die auch als komponentenorientierte Middleware bezeichnet werden, umfassen i.d.R. sämtliche oder zumindest viele Funktionen der beschriebenen funktionalen Integrationsmechanismen verbunden mit einer starken Orientierung an Web-Protokollen und –Standards [vgl. Alonso et al. 2003, 102]. Sie integrieren verschiedene funktionale Integrationsmechanismen unter Rückgriff auf Spezifikationen wie Enterprise Java Beans (EJB) oder dem Distributed Component Object Model (DCOM) und ermöglichen den Zugriff auf gekapselte Anwendungslogik über standardisierte Web-Protokolle wie bspw. das *HyperText Transfer Protocol* (HTTP).

2.5.1.3 Präsentationsintegration

Unter Präsentationsintegration versteht man die Integration verschiedener Anwendungssysteme auf der Ebene der Darstellungs- bzw. Präsentationslogik. Die Zusammenführung der zu integrierenden Systeme wird i.d.R über die Simulation eines Benutzerdialogs vorgenommen [s. Kaib 2002, 62] und stellt insofern lediglich eine „Notlösung“ für die Integration von Anwendungssystemen dar. Sie kommt insbesondere dann zum Einsatz, wenn es gilt, unzureichend dokumentierte Altanwendungen mit einer hohen internen Verflechtung zu integrieren [vgl. Riehm/Vogler 1996, 47].

2.5.2 Ebene Workflowintegration

Mechanismen zur Unterstützung der Workflowintegration werden durch Systeme zur Umsetzung der Ansätze des *Workflow Management* (WFM) bzw. *Business Process Management* (BPM) umgesetzt. BPM wird häufig aufgefasst als ein auf WFM aufbauender „nächster Schritt“, der den Ansatz des WFM weiterentwickelt und insbesondere um Konzepte zur Analyse automatisierter Prozesse ergänzt [vgl. van der Aalst et al. 2003, 4 ff., Zhao/Cheng 2004, 3]. Im Rahmen dieser Arbeit werden die Ansätze WFM bzw. BPM hingegen als gleichwertig angesehen [vgl. Zhao/Cheng 2004, 3, Zur Muehlen 2004, 2]. Die folgende Charakterisierung prozessorientierter Integrationsmechanismen basiert auf der Begriffswelt des WFM, kann jedoch ohne weiteres auf die Begriffe aus dem Bereich BPM übertragen werden.

© HSG / IWI / CC BN3 / 1

Zentraler Gegenstand ist ein Workflow Management System (WFMS), welches Workflows definiert, gestaltet, steuert und automatisiert ausführt. Abbildung 2-3 stellt die Kernkomponenten eines WFMS auf einer hohen Abstraktionsebene dar und dient im Folgenden der Skizzierung des Integrationsmechanismus mittels WFM.

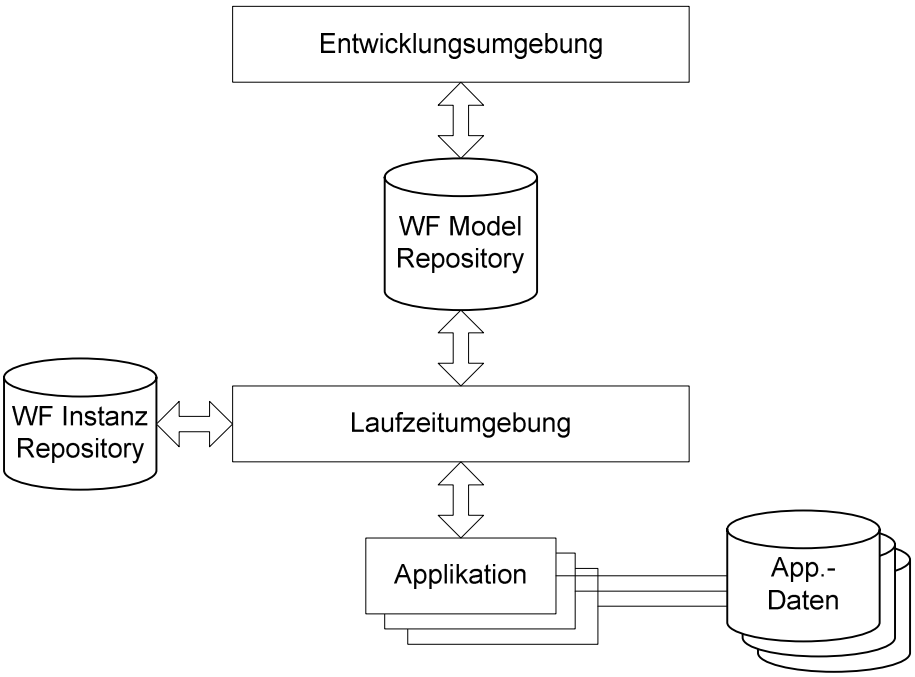


Abbildung 2-3: Komponenten eines WFMS [vgl. Derungs 1997, 40, Zur Muehlen 2004, 117]

Die Struktur eines WFMS lässt sich unterteilen in eine Entwicklungs- und eine Laufzeitumgebung. Die Entwicklungsumgebung stellt Werkzeuge zur Definition von Workflow-Modellen zur Verfügung, welche in einer Datenbank, dem sog. Workflow-Modell Repository verwaltet werden. Ein Workflow-Modell stellt eine automatisierbare Repräsentation eines Geschäftsprozesses dar [vgl. WFMC 1999, 52]. In der Laufzeitumgebung werden die Workflow-Modelle in Form konkreter Workflows instanziiert, deren Statusinformationen in dem WF Instanz Repository verwaltet werden. Ein Workflow wird im Rahmen dieser Arbeit definiert als eine Automatisierung eines (Teil-)Prozesses, während der Dokumente, Informationen oder Aufgaben anhand bestimmter Regeln von einer bearbeitenden Ressource zur nächsten weitergeleitet werden [vgl. WFMC 1999, 8]. Ein Workflow besteht aus einer Abfolge von Aktivitäten, die jeweils logische und fachlich zusammengehörende Arbeitsschritte innerhalb des automatisierten Prozesses zusammenfassen [vgl. WFMC 1999, 13, Vogler 2003a, 39 f.]. Ein Arbeitsschritt ist eine elementare, nicht weiter zerlegbare Verrichtungseinheit in einem Workflow [vgl. Vogler 2003a, 40]. Arbeitsschritte werden unter Nutzung von

© HSG / IWI / CC BN3 / 1

Ressourcen, d.h. entweder manuell durch Personen oder automatisiert durch IT-Applikationen ausgeführt. Im Kontext der Betrachtung von WFM als Mechanismus zur Integration von Anwendungssystemen sind insbesondere sog. Produktions-Workflows [vgl. Georgakopoulos/Sheth 1995, 124 f., Stohr/Zhao 2001, 286 f., Zur Muehlen 2004, 41 f.] von Interesse, deren Struktur vor der Ausführung detailliert definiert ist und die sich zum grösstenteils aus automatisierten Aktivitäten zusammensetzen. Einzelne instanziierte Aktivitäten solcher Workflows führen Anwendungen bzw. Anwendungsteile aus und integrieren somit die verschiedenen Anwendungsfunktionen über die Ablaufsteuerung des Workflows.

2.5.3 Ebene Desktopintegration

Zur Umsetzung der Desktopintegration kommen i.d.R. Portalplattformen zum Einsatz. Diese umfassen Funktionen zur Präsentation, Navigation, Interaktion, Personalisierung sowie zur Administration und Sicherheit [Puschmann 2004]. Zusätzlich bieten einige Softwarehersteller inzwischen spezielle Umgebungen zur Entwicklung von Composite Applications an. Beispielsweise hat die SAP AG eine solche Umgebung in Form ihres *Composite Application Framework* [vgl. SAP 2004, Weilbach/Herger 2004] auf dem Markt.

© HSG / IWI / CC BN3 / 1

3 Serviceorientierte Architekturen: Einordnung und Framework

3.1 Begriffsbildung

Der im Hinblick auf die Beschreibung einer serviceorientierten Architektur zentrale Begriff des *Service* wird in der Literatur unterschiedlich detailliert und mit unterschiedlichen Schwerpunkten charakterisiert. In Tabelle 3-1 sind einige Definitionen der Begriffe SOA und Service aufgeführt. Eine serviceorientierte Architektur wird im Rahmen dieser Arbeit als eine mehrschichtige, verteilte Systemarchitektur verstanden, die Teile der Applikationsarchitektur als Services kapselt und dabei eine Reihe von Designprinzipien berücksichtigt. Services stellen abstrakte Software-Elemente bzw. Schnittstellen dar, die anderen Applikationen unter Nutzung von verbreiteten Standards stabile, wieder verwendbare Software-Funktionalität auf einer anwendungsorientierten, fachlichen Granularitätsstufe anbieten.

Während in der Literatur weitgehend Einigkeit über die Gestaltungselemente service-orientierter Architekturen herrscht, gehen die Meinungen bezüglich der anzuwendenden Designprinzipien auseinander. Die Gestaltungselemente werden im Folgenden in ein Metamodell eingeordnet, das sich am Business Engineering orientiert.

© HSG / IWI / CC BN3 / 1

Quelle	Definition
[Arsanjani 2004]	A *SOA* is an enterprise-scale IT architecture for linking resources on demand. In a SOA, resources are made available to participants in a value net, enterprise, line of business (typically spanning multiple applications within an enterprise or across multiple enterprises). It consists of a set of business-aligned IT services that collectively fulfill an organization's business processes and goals. A *service* is a software resource (discoverable) with an externalized service description. This service description is available for searching, binding, and invocation by a service consumer.
[Gioldasis et al. 2003, 12f.]	*Service-Oriented Architecture (SOA)* refers to an application software topology according to which business logic of the applications is separated from its user interaction logic and encapsulated in one or multiple software components (services), exposed to programmatic access via well defined formal interfaces. Each *service* provides its functionality to the rest of the system as a well-defined interface described in a formal markup language and the communication between services is platform and language independent.
[Natis 2003]	*SOA* is a software architecture that starts with an interface definition and builds the entire application topology as a topology of interfaces, interface implementations and interface calls. […] SOA is a relationship of services and service consumers, both software modules large enough to represent a complete business function. *Services* are software modules that are accessed by name via an interface, typically in a request-reply mode.
[Oasis 2005, 30f.]	[Channabasavaiah et al.] software architecture of services, policies, practices and frameworks in which components can be reused and repurposed rapidly in order to achieve shared and new functionality. [...] SOA uses the object-oriented principle of encapsulation in which entities are accessible only through interfaces and where those entities are connected by well-defined interface agreements or contracts. [A *service* is] a behavior or set of behaviors offered by one entity for use by another according to a policy and in line with a service description.
[Sprott/Wilkes 2003, 5]	[*SOA*:] The policies, practices, frameworks that enable application functionality to be provided and consumed as sets of services published at a granularity relevant to the service consumer that can be invoked, published and discovered, which are abstracted away from the implementation using a single, standards based form of interface.
[van Zyl 2002, 249]	*Service based architecture (SBA)* is a layered architecture that separates the usage and definition of software components, from the implementation software architecture in order to define software-as-services using a common standard.
[W3C 2004a], [W3C 2004c]	A *Service Oriented Architecture (SOA)* is a form of distributed systems architecture. [It consists of] a set of components which can be invoked, and whose interface descriptions can be published and discovered. A *service* is an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities.

Tabelle 3-1: Definitionen der Begriffe SOA und Service

3.2 Metamodell

Im Business Engineering Modell lässt sich SOA als mehrschichtige Integrationsarchitektur einordnen, die die Prozess- und IS-Ebene verbindet und als elementaren Architekturbestandteil Services beinhaltet. Die Serviceorientierte Architektur legt den Schwerpunkt auf

© HSG / IWI / CC BN3 / 1

die Identifikation wieder verwendbarer Services und deren Integration entlang von automatisierbaren Workflows, nutzt aber darunter liegende Mechanismen der Funktions- und Datenintegration. Soll zusätzlich auf Desktopebene integriert werden, so geschieht dies mit Hilfe eines Portals oder eines Composite Application Frameworks.

Abbildung 3-1 gibt einen Überblick über die wichtigsten der genannten Architekturkomponenten und ihre Beziehungen in Form eines Metamodells. Die Darstellung des Modells basiert auf einer vereinfachten Entity-Relationship-Notation, wobei die modellierten Objekte durch Knoten und die Beziehungen zwischen diesen durch Kanten dargestellt sind [vgl. Österle 1995, 187 ff.]. Die Pfeile der Kanten geben die Leserichtung der Beziehung an und an den Enden der Kanten sind die Kardinalitäten angegeben, wobei die Kardinalität „1“ für genau eine, „c“ für keine oder eine, „cn“ für keine, eine oder mehrere und „n“ für mindestens eine steht. Durch den Bogen wird ein exklusives Oder in Beziehungen zwischen Objekten ausgedrückt. In der folgenden Tabelle sind die Metaentitätstypen erläutert.

© HSG / IWI / CC BN3 / 1

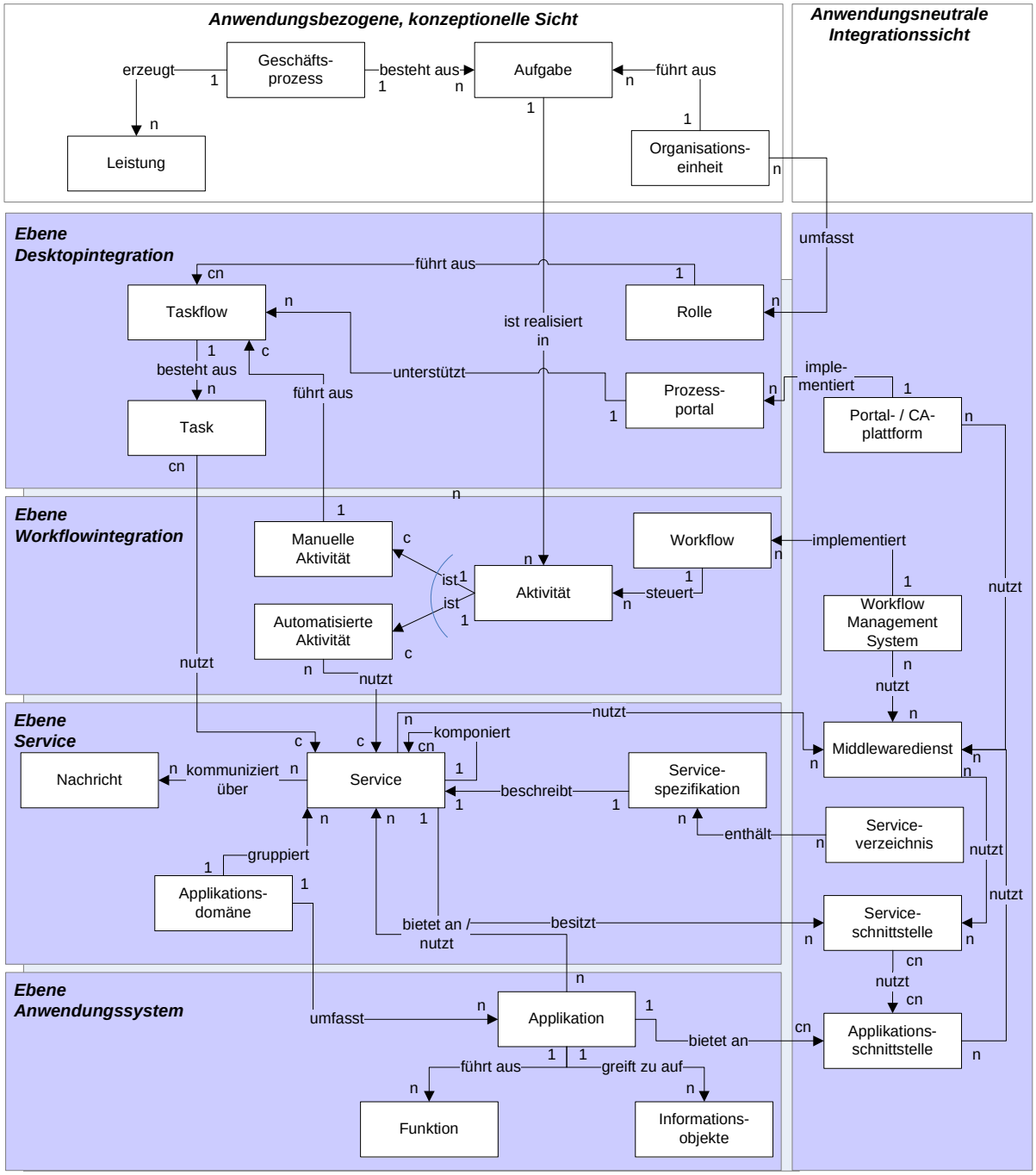


Abbildung 3-1: Metamodell für Serviceorientierte Architekturen

Aktivität

Beschreibung	Eine Aktivität ist eine in sich geschlossene Verrichtungseinheit in einem Prozess, die logisch und fachlich zusammengehörende Arbeitsschritte in einem aus Sicht des Benutzers sinnvollen Arbeitspaket zusammenfasst [vgl. WFMC 1999, 13, Vogler 2003a, 41 f.]. Aktivitäten können vollständig automatisch im Hintergrund (automatisierte Aktivität) oder im Dialog mit dem Benutzer (manuelle Aktivität) ablaufen.
Beziehungen	Ein Workflow umfasst mehrere Aktivitäten. Eine Aktivität bündelt einen oder mehrere Tasks.

Applikationsdomäne

Beschreibung	Applikationsdomänen sind ein Konzept zur Strukturierung der Applikationslandschaft. Sie kapseln eng voneinander abhängige Anwendungen oder Funktionen und Informationsobjekte aufgrund von fachlichen Kopplungsanalysen (z.B. gemeinsam genutzte Informationsobjekte, verwandte Funktionalität, gemeinsame Ownership) und bieten Applikationen anderer Domänen über definierte Schnittstellen (i.d.R. Services) Zugriff auf die gekapselte Funktionalität (vgl. [Winter/Schelp 2005, 49], [Schlamann 2004, 60]).
Beziehungen	Eine Applikationsdomäne umfasst mehrere Applikationen.

Applikationsschnittstelle

Beschreibung	Eine Applikationsschnittstelle ist Bestandteil einer Applikation und bietet Zugriff auf Teile seiner Funktionen oder Daten. Applikationsschnittstellen sind implementierungsspezifisch in der Technologie der Applikation umgesetzt und entsprechen i.d.R. nicht den anwendungsübergreifenden Standards auf Serviceebene. Beispiele sind EJB-Interfaces, Standardsoftware-APIs oder Message Queue Nachrichten (vgl. [Vogler 2003b, 91], [Cheesman/Ntinolazos 2004, 15]).
Beziehungen	Zu einer Applikation kann es keine, eine oder mehrere Schnittstellen geben.

Applikation

Beschreibung	Eine Applikation bezeichnet eine Software, die konkrete betriebliche Anwendungen unterstützt [vgl. Mertens 1997, 37-39].
Beziehungen	Eine Applikation kann eine Service-Implementierung enthalten. Eine Applikation kann Funktionalität und Daten über Integrationsmechanismen zur Verfügung stellen.

Funktion

Beschreibung	Eine Applikationsfunktion realisiert eine (Geschäfts-)Funktion und ist Bestandteil einer Applikation [s. Schwinn 2006, 34]. Applikationsfunktionen operieren auf Informationsobjekten.

© HSG / IWI / CC BN3 / 1

Beziehungen	Mehrere auf den gleichen Informationsobjekten operierende Funktionen werden zu einer Applikation zusammengefasst.

Geschäftsprozess

Beschreibung	Ein Prozess fasst eine Menge von Aufgaben, die in einer vorgegebenen Ablauffolge zu erledigen sind, und ggf. durch Applikationen unterstützt werden, als Einheit zusammen [vgl. Hess 1996, 124 ff., Vogler 2003a, 366]. Seine Wertschöpfung besteht aus Leistungen an anderem Prozess innerhalb und ausserhalb des eigenen Unternehmens.
Beziehungen	Ein Prozess wird bei der Implementierung mit einer Steuerungskomponente, wie bspw. eines Workflow-Management-Systems, in einen oder mehrere Workflows zerlegt.

Informationsobjekt

Beschreibung	Informationsobjekte stellen auf Dauer gespeicherte Informationen dar, auf die wiederholt zugegriffen wird [s. Schwinn 2006, 34]. Eine oder mehrere Applikationsfunktionen erzeugen, verändern oder löschen Informationen in Datensammlungen. Die Datensammlung entspricht im relationalen Datenmodell einer Datenbank, kann aber bspw. auch aus einer Sammlung von Dokumenten (Word-Dokumente, HTML-Pages etc.) bestehen [s. Vogler 2003b, 93].
Beziehungen	Eine Applikation operiert auf eine gemeinsame Menge von Informationsobjekten.

Nachricht

Beschreibung	Nachrichten dienen der Kommunikation zwischen Services und Servicenutzern. Sie enthalten die Daten, die Service und Servicenutzer zur Ausführung einer durch eine Serviceoperation unterstützten Arbeitseinheit austauschen müssen [s. Erl 2005, 288]. Sie werden durch den Schnittstellenkontrakt definiert und setzen sich aus einem Header- und einem Payload-Teil zusammen [s. Krafzig et al. 2004, 34].
Beziehungen	Eine Nachricht kann von mehrerern Services zur Kommunikation mit anderen Services eingesetzt werden.

Middlewaredienst

Beschreibung	Ein Middlewaredienst bzw. Integrationsmechanismus umfasst Basiskomponenten, welche Dienste und Protokolle für die transparente Kommunikation verteilter heterogener Applikationen bereitstellen (vgl. [Vogler 2003b, 95], [Mantel et al. 2004, 4]). In Abhängigkeit von einer gewählten Integrationsebene und der damit verbundenen Teilsicht der Systemintegration lassen sich verschiedene Integrationsmechanismen unterscheiden

© HSG / IWI / CC BN3 / 1

Beziehungen	Elemente der anwendungsneutralen Sicht wie Portal-/CA-Plattform und Serviceschnittstelle nutzen Middlewaredienste zur Durchführung ihrer Aufgaben. Ein Service ist ebenfalls auf Middlewaredienste angewiesen.

Service

Beschreibung	Ein Service stellt ein abstraktes Software-Element bzw. eine Schnittstelle dar, die anderen Applikationen unter Nutzung von verbreiteten Standards stabile, wieder verwendbare Software-Funktionalität auf einer anwendungsorientierten, fachlichen Granularitätsstufe anbietet.
Beziehungen	Services werden genutzt von Composite Applications, automatisierten Aktivitäten oder automatisierten Tasks.

Servicespezifikation

Beschreibung	Eine Servicespezifikation enthält sämtliche Informationen, die ein Servicenutzer benötigt, ohne die interne Realisierung des Services zu kennen [Stojanovic 2005]. Inhalte einer Servicespezifikation umfassen die Schnittstellen-, die Verhaltens, die Aufgaben-, die Qualitäts-, die Terminologie-, die Aufgaben- sowie die Vermarktungsebene eines Services [Turowski et al. 2002].
Beziehungen	Eine Servicespezifikation beschreibt einen Service. Mehrere Servicespezifikationen werden in einem Serviceverzeichnis hinterlegt.

Serviceschnittstelle

Beschreibung	Eine Serviceschnittstelle stellt die softwaretechnische Realisierung einer Servicespezifikation dar. Sie nutzt über Applikationsschnittstellen die (Fach-)Funktionalität von bestehenden Applikationen.
Beziehungen	Eine Serviceschnittstelle realisiert eine Servicespezifikation.

Serviceverzeichnis

Beschreibung	In einem Serviceverzeichnis werden die verfügbaren Services registriert und ihre Servicespezifikation abgelegt. Das ermöglicht dem Servicenutzer einen geeigneten Service zu identifizieren und aufzurufen.
Beziehungen	Ein Serviceverzeichnis speichert mehrere Servicespezifikationen.

Prozessportal

Beschreibung	Ein Portal ist eine web-basierte Anwendung, die Inhalte, Dienste und Funktionen benutzerspezifisch integriert [s. Schelp/Winter 2002].
Beziehungen	Zur Umsetzung eines Prozessportals wird eine Portalplattform benötigt. Über ein Prozessportal greift eine Rolle auf die von ihm

© HSG / IWI / CC BN3 / 1

auszuführenden Taskflows zu.

Portal- / CA-Plattform

Beschreibung	Portal- oder Composite Application- (CA) Plattformen bieten Entwicklungs- bzw. Integrations- und Laufzeitdienste für die Realisierung von desktopintegrierten Applikationen (s. [Vogler 2003b, 55], [Puschmann 2003, 74ff.], [Woods 2003, 4]).
Beziehungen	Eine Portal- /CA-Plattform ist für die Implementierung von Prozessportalen zuständig. Dabei ist der Zugriff auf Middlewaredienste notwendig.

Rolle

Beschreibung	Rollen sind Klassifikationen von Mitarbeitern, die sich aus den Fähigkeiten, Aufgaben und Verantwortungsbereich ableiten lassen. [Puschmann 2003]
Beziehungen	Im hier gegebenen Betrachtungsfokus wird ein Taskflow von genau einer Rolle ausgeführt.

Task

Beschreibung	Ein Task bzw. ein Arbeitsschritt ist eine atomare Verrichtungseinheit in einem Prozess, d.h. er lässt sich nicht weiter zerlegen [vgl.Vogler 2003a, 40]. Ein Task ist entweder manuell, d.h. er wird von einem Benutzer ausgeführt oder automatisiert, d.h. er läuft IT-unterstützt ab.
Beziehungen	Eine Aktivität bündelt einen oder mehrere Tasks. Ein automatisierter Task nutzt einen oder mehrere Services.

Workflow

Beschreibung	Ein Workflow ist die Automatisierung eines (Teil-)Prozesses. Er steuert die koordinierte Ausführung von Aktivitäten durch Applikationen bzw. Benutzer unter Rückgriff auf ein Workflow-Management-System [vgl. WFMC 1999, 8, Zur Muehlen 2004, 39].
Beziehungen	Ein Workflow implementiert einen Ausschnitt aus einem Prozess. Ein Workflow besteht aus einer oder mehreren Aktivitäten.

Workflow-Management System

Beschreibung	Ein Workflow Management System bietet Entwicklungs- bzw. Integrations- und Laufzeitdienste für die Realisierung automatisierter Abläufe (Workflows) zur anwendungsübergreifenden Integration von Geschäftsprozessen.
Beziehungen	Zur Steuerung der Workflowausführung wird ein Workflow-Management-System benötigt. Es greift selbst wiederum auf Middlewaredienste zurück.

© HSG / IWI / CC BN3 / 1

3.3 Designprinzipen

Ziel dieses Abschnitts ist es, die grundlegenden Designprinzipien zur Gestaltung von SOA und Services herauszuarbeiten. Grundlage dazu bilden eine Reihe von SOA-Charakterisierungen durch Standardisierungsgremien, Wissenschaftler, Analysten und Softwareanbieter. Dabei besteht eine Herausforderung darin, dass

- unterschiedliche Autoren verschiedene Designprinzipien betonen und
- die Designprinzipien verschieden detailliert definieren,
- sich mit unterschiedlichen Begriffen bezeichnete Prinzipien inhaltlich überlappen oder
- inhaltlich gleiche Prinzipien unterschiedlich benannt werden.

Bspw. existieren divergierende Meinungen darüber, ob der Begriff SOA an die Verwendung von Web Service Standards gebunden ist (vgl. [Zimmermann et al. 2003, S. 161] und [Dostal et al.] oder ob Services nur einen synchronen „Request/Reply"-Interaktionsstil oder auch eine asynchrone Kommunikation unterstützen (vgl. z.B. [Natis/Schulte 2003] und [Bloomberg 2004]). Viele Autoren bezeichnen die „lose Kopplung" als eine Kerneigenschaft von SOA. Während aber [Brown et al. 2002] darunter die Verwendung bestimmter auf Standards basierender Kommunikationsmechanismen versteht, bezieht sich der Begriff bei [Fritz 2004] auf eine logische Unabhängigkeit zwischen Architekturkomponenten, die eine voneinander entkoppelte Evolution erlaubt. [Papazoglou 2003] verbindet damit das Verbergen interner Strukturen, was andere Autoren eher unter der Eigenschaft Abstraktion einordnen (vgl. [Fritz 2004], [W3C 2004c]).

Die folgenden Abschnitte konsolidieren die in der Literatur identifizierten Designprinzipien von SOA und Services in vier Klassen:

Schnittstellenorientierung

Designprinzipien der Klasse Schnittstellenorientierung behandeln die explizite und umfassende Beschreibung von Serviceschnittstellen und die Abstraktion von Implementierungsdetails aus Sicht eines Servicenutzers. Ein Service stellt eine stabile Schnittstelle im Sinne eines Vertrags dar und ist über Metadaten technisch und fachlich vollständig spezifiziert.

© HSG / IWI / CC BN3 / 1

Interoperabilität

Im Rahmen einer SOA werden auf Ebene Gesamtarchitektur verschiedene technische (z.B. zu verwendende Schnittstellentechnologien oder Datentypen) und fachliche (z.B. standardisierte Datenmodelle für wichtige Informationsobjekte) Standardisierungsentscheidungen getroffen, die auf Ebene Einzelservices verbindliche Vorgaben für Designentscheidungen definieren. Erst die Abstützung auf offene, plattformunabhängige und verbreitete Standards ermöglicht die Interoperabilität eines Dienstes und seine Verwendung in unterschiedlichem Kontext.

Autonomie und Modularität

Eine SOA strukturiert die Applikationsarchitektur in eine überschaubare Menge teilautonomer Subsysteme (Domänen und Services). Nach bekannten Prinzipien der Modul- oder Komponentenbildung werden dabei Funktionalität oder Ressourcen mit grosser Abhängigkeit (Kohäsion) untereinander so zusammengefasst, dass sie gegenüber den anderen Subsystemen eine möglichst geringe Abhängigkeit (lose Kopplung) aufweisen.

Bedarfsorientierung

Services orientieren sich bezüglich ihres Leistungsumfangs an geschäftlichen Objekten und Prozessaktivitäten. Sie bieten grob granulare, fachlich abgrenzbare und aus geschäftlicher Sicht sinnvolle und wertschöpfende Leistungen.

© HSG / IWI / CC BN3 / 1

Kategorie	Designprinzip	[Baskerville et al. 2005]	[Brown et al. 2002, 4f.]	[Erl 2005, 291f.]	[Fritz 2004]	[Klesse et al. 2005, 262ff]	[McGovern et al. 2003, 40f]	[Newcomer/Lomow 2004, 72f]	[Papazoglou 2003, 3]	[W3C 2004c]
Schnittstellenorientierung	Abstraktion von der Serviceimplementierung	●	●	●	●		●	●	●	●
	Umfassende Servicespezifikation	●	●	●		●	●	●		●
	Stabile, gemanagte Servicekontrakte			●	●	●		●		
Interoperabilität	Technische Standardisierung	●	●		●		●	●	●	●
	Fachliche Standardisierung		●					●		
	Verwendung offener und verbreiteter Industriestandards	●			●			●	●	
Autonomie und Modularität	Hohe Servicekohäsion und schwache logische Kopplung	●		●	●	●	●	●		
	Lose gekoppelte Kommunikation		●	●			●	●	●	
Bedarfsorientierung	An Geschäftskonzepten orientierte Servicegranularität		●				●	●		●
	Generalisierung der Serviceleistung			●			●	●		
	Fachlich wertschöpfende Serviceleistung									

Tabelle 3-2 SOA-Designprinzipien

© HSG / IWI / CC BN3 / 1

3.4 Services als Bestandteil einer Integrationsarchitektur

Dieser Abschnitt stellt die Rolle von Services im Rahmen des in Abschnitt 2.3 skizzierten Rahmenwerks für Integrationsarchitekturen dar. Services bilden eine neue Ebene in der Integrationsarchitektur, auf der fachlich abgrenzbare, grob granulare Leistungen in Form standardisierter Schnittstellen angeboten werden. Sie werden auf der Service-Ebene definiert und auf der darunterliegenden Ebene der Anwendungssysteme implementiert. Die Implementierung nutzt die beschriebenen funktions- oder datenorientierten Integrationsmechanismen. Service-Nutzer sind bspw. Workflows auf der Ebene der Workflowintegration oder desktoporientierte, Portal- oder CA-basierte Anwendungen auf der Ebene der Desktopintegration. Durch die Serviceebene wird so eine zunehmende Entkopplung und damit Flexibilisierung zwischen den Ebenen der Desktop- und Workflowintegration einerseits und der Anwendungssystem-Ebene andererseits möglich.

Diese Definition von Services ist noch sehr generisch und lässt unterschiedliche Ausprägungen von Services zu. In der Literatur finden sich unterschiedliche Ansätze zur weitergehenden Detaillierung des Begriffs in Form von Klassifizierungen bzw. Typisierungen, die Tabelle 3-3 in Form eines morphologischen Kastens zusammenfasst.

Services lassen sich bspw. anhand der Art der bereitgestellten Funktionalität dahingehend unterscheiden, ob sie eine geschäftsorientierte oder eher eine technische, unterstützende Funktionalität anbieten. Anhand dieser Ausprägungen erfolgt in [vgl. Channabasavaiah et al. 2003] eine Klassifizierung in *Business Services* bzw. *System Services.* Anhand des fachlichen Leistungsumfangs lassen sich Services weiterhin danach unterschieden, ob sie einen kompletten Geschäftsprozess, eine Prozessaktivität, ein Geschäftsobjekt oder eine Querschnittsfunktion repräsentieren / unterstützen [vgl. Sims 2003, van de Loo 2004]. Bei SAP findet eine Unterscheidung zwischen Business resp. Enterprise Services auf der einen Seite und Application Services auf der anderen Seite statt. Bei Application Services handelt es sich um Services, die bestehende Applikationsfunktionalität als Adapter kapseln und auf der Serviceebene verfügbar machen. Zwischen Softwarekomponente und kapselnden Adapterservices besteht eine 1-zu-1 Beziehung. Der Entwurf von Application Services ist eher bottom-up motiviert. Im Gegensatz dazu erfolgt die Identifikation von Business Services aus den Anforderungen des Workflows. Business Services werden hierarchisch aus anderen Business und Application Services zusammengesetzt und können etwa geschäftsorientierte

© HSG / IWI / CC BN3 / 1

und technische Services, etwa die Kopplung eines Services zur Abfrage von Informationen mit einem Zugriffskontrollservice, kombinieren.

Aus einer eher technischen Sichtweise heraus werden Services häufig anhand des verwendeten Interaktionsstils sowie der Zustandsspeicherung unterschieden [vgl. Wilkes/Harby 2004]. Schliesslich spielen häufig die Aspekte der Verteilung sowie der Standardisierungsreichweite [vgl. Lublinsky 2004] bei Klassifizierungen eine Rolle.

	Ausprägungen					
Attribute	**Art der Funktionalität**	Geschäftsfunktionalität		Technische Funktionalität		
	Fachlicher Leistungsumfang	Geschäfts-prozess	Prozess-aktivität	Entität / Geschäftsobjekt	Commodity / Querschnitts-funktionen	
	Fachgranularität	Enterprise Service (komponiert)		Application Service (atomar)		
	Interaktionsstil	Synchron	Asynchron	Request / Response	Notification	
	Zustands-speicherung	zustandslos		zustandsbehaftet		
	Verteilung	Organisatorisch Intern	Organisatorisch Extern	Technisch		
	Standardisierungs-reichweite	Syntax		Semantik		

Tabelle 3-3: Verschiedene Attribute und Ausprägungen von Services

3.5 Umsetzung serviceorientierter Architekturen

3.5.1 Web Services

In der Web Service Technologie sehen viele einen prädestinierten Mechanismus zur Umsetzung einer serviceorientierten Architektur. Das Aufkommen der Web Services hat maßgeblich zur Verbreitung des SOA Konzepts beigetragen. Ein Web Service wird in Anlehnung an die Definition des W3C beschrieben als ein netzwerkfähiges Softwaresystem, welches unter Berücksichtigung einer Menge bestimmter, auf der *Extensible Markup Language* (XML) basierender Standards entwickelt ist [vgl. W3C 2004b]. In Bezug zur oben angeführten Definition des Servicebegriffs wird im Rahmen dieser Arbeit unter einem Web Service nicht die eigentliche Software-Implementierung, sondern vielmehr die Beschreibung

© HSG / IWI / CC BN3 / 1

eines Softwareelements in Form einer standardisierten Schnittstelle verstanden. Web Services basieren auf drei grundlegenden XML-basierten Standards:

- Die *Web Service Description Language* (WSDL) dient zur Beschreibung des Web Services, d.h. mittels dieser Sprache wird die implementierte Software-Funktionalität spezifiziert.
- Das *Simple Object Access Protocol* (SOAP) legt die Grobstruktur und die Verarbeitungsvorschriften für Nachrichten fest, d.h. mittels dieser Sprache wird die Interaktion mit Services standardisiert.
- Die Spezifikation *Universal Description Discovery and Integration* (UDDI) schliesslich definiert einen Verzeichnisdienst zur Katalogisierung von Web Services.

Eine genauere Beschreibung der genannten Web Service Standards findet sich bspw. in [vgl. Tsalgatidou/Pilioura 2002, 139 ff., Alonso et al. 2003, 151 ff.].

Web Services unterstützen zumindest zwei der vier Kernprinzipien einer SOA: die Schnittstellenorientierung sowie die Standardisierung. Das Prinzip der Anwendungsorientierung wird durch Web Services nicht erzwungen. Werden Web Services nicht zur Kapselung grob granularer, fachlich abgrenzbarer Leistungen verwendet, so ist dieser Integrationsmechanismus als weitere Möglichkeit der Umsetzung einer funktionsorientierten Integration einzuordnen [vgl. Samtani/Sadhwani 2001, Alonso et al. 2003, 131 ff.]. Erst die Orientierung an betriebswirtschaftlichen Leistungen ermöglicht die direkte Verbindung zur Workflow- bzw. Desktopintegration und macht Web Services zu einem serviceorientierten Integrationsmechanismus.

Als Beispiel einer Web Service-spezifische Ausprägung einer serviceorientierten Architektur dient die die Web Service Architecture (WSA) [WSA04] des Worldwide Web Consortiums (W3C). Ohne in die konkrete Implementierung zu gehen, liefert die WSA eine theoretische Grundlage für das Verständnis von Mechanismen und Technologien einer SOA, indem die WSA ein abstarktes Begriffsystem aus Konzepten und Beziehungen (Ontologie, Glossar) definiert, das - einmal verstanden - den Zugang und das Verständnis zu konkreten SOA-Produktplattformen erleichtert. Der WSA liegt das folgende schematische Interaktionsschema zugrunde:

© HSG / IWI / CC BN3 / 1

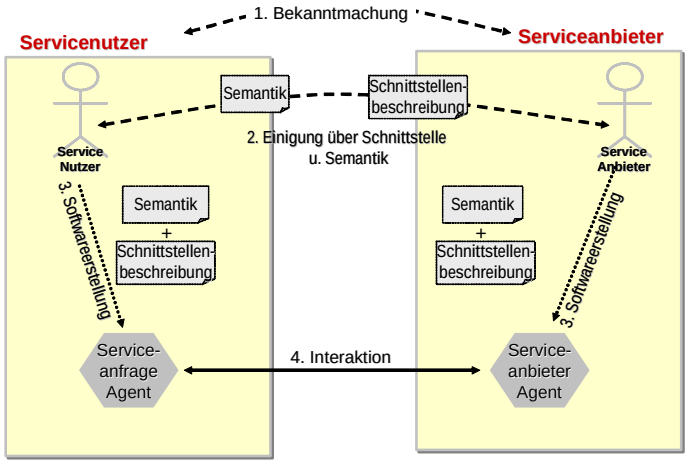


Abbildung 3-2: Interaktionsschema nach WSA

Zum prinzipiellen Ablauf der Interaktion nach der WSA lässt sich folgendes ausführen: Vor der Interaktion muss zunächst der Webserviceanbieter gefunden werden, danach wird die Schnittstellenbeschreibung und Semantik ausgetauscht. Aus dieser kann der Nutzer ein softwaretechnisches Proxykonstrukt erstellen, das die programmatische Nut-zung des Webservice erlaubt. Die WSA betont, dass die einzelnen Schritte entweder manuell oder automatisch durchgeführt werden können.

Die WSA bietet vier unterschiedliche Sichten auf die Architektur an. Im Zentrum steht das Service Oriented Model, das durch drei weitere spezifische Modelle - Policy, Message und Resource Oriented Model - ergänzt wird, wobei zwischen den einzelnen Architektursichten Querbezüge existieren. Der folgende Abschnitt gibt ausgehend von dem in nachfolgender Abbildung dargestellten Service Oriented Model eine kompakte Erläuterung.

© HSG / IWI / CC BN3 / 1

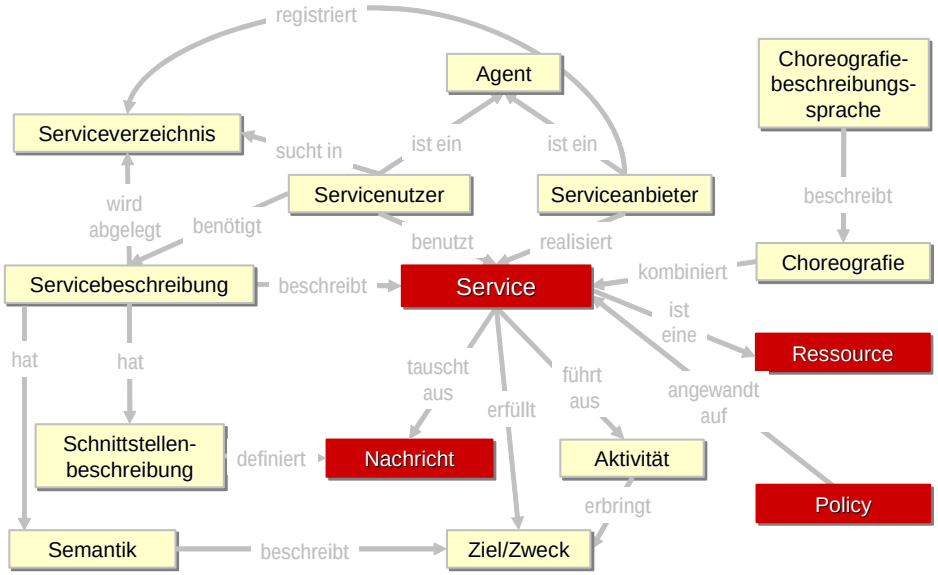


Abbildung 3-3: Semantisches Beziehungsmodell „Service Oriented Model“ der WSA

Die Interaktion mit einem Web Service Anbieter geschieht über den Austausch von Nachrichten, deren Struktur im Message Model näher ausgeführt wird. Wie bereits oben angedeutet, muss ein Web Service hinreichend beschrieben werden, um mit ihm in Interaktion treten zu können, wobei sowohl technische als auch semantische Aspekte zu berücksichtigen sind. Während die technischen Beschreibung Eigenschaften wie Nachrichtenformate, Operatorensignaturen, Nachrichteninteraktionsmuster und eindeutige Adresse festlegt, umfasst die Semantikbeschreibung die Bedeutung, das Verhalten und damit die Auswirkung eines Webserviceaufrufs.

Richtlinien zum Gebrauch eines Web Services werden im Policy Model formuliert; dies betrifft vor allem die Aspekte Sicherheit, QoS, und Management. Im Resource Model der W3C-Nomenklatur wird der Web Service in verallgemeinerter Form als Web Ressource bezeichnet, weshalb er auch eine entsprechende Unified Resource Identifier (URI) besitzen muss, um eindeutig referenziert werden zu können. Im Rahmen einer Choreografie werden Reihenfolge und Bedingungen festgelegt, unter denen mehrere voneinander unabhängige Web Services zu einem übergeordneten Geschäftszweck verschaltet werden. Dabei ermöglicht eine Choreografiebeschreibungssprache das Verschalten von Web Services in einer formalen, maschinell verarbeitbaren Weise.

Insgesamt liefert die WSA wichtige Hinweise für die notwendigen Bestandteile einer SOA und den zielgerichteten Einsatz weiterführender WS-Spezifikationen. Gleichwohl ist die

© HSG / IWI / CC BN3 / 1

Realisierung einer serviceorientierten Architektur nicht ausschliesslich auf Web Services angewiesen ist, wie der nächste Abschnitt zeigt.

3.5.2 Weitere Umsetzungsalternativen

Neben Web Services gibt es weitere Umsetzungsalternativen für Services. Hier sind insbesondere die objektorientierten Komponentenplattformen CORBA, DCOM oder mittlerweile auch Microsoft .Net und J2EE zu nennen. Diesen werden bezüglich der Implementierung einer SOA verschiedentlich Unzulänglichkeiten, insbesondere in Bezug auf ihre Plattformabhängigkeit und Interoperabilität, vorgeworfen (vgl. [Raptis et al. 2001, 167], [Baker 2002, 627], [Gisolfi 2001]): Sie basieren entweder auf proprietären Plattformen eines Herstellers (DCOM, .Net), setzen bestimmte Programmiersprachen (J2EE) voraus oder weisen je nach Spezifikationen der Hersteller unterschiedliche und teilweise untereinander inkompatible Implementierugen auf (CORBA). Die Interaktion von Softwarebausteinen zwischen verschiedenen Komponentensystemen wird u.a. wegen unterschiedlichen Techniken für die Schnittstellenbeschreibung oder proprietären Kommunikationsprotokollen beschränkt. Dies macht spezielle Adapter erforderlich und erschwert die gemeinsame Nutzung von Komponenten verschiedener Komponentensysteme. Weiter ist die in Form von Komponenten oder Objekten implementierte Anwendungsfunktionalität oft zu feingranular und nicht ausreichend an den fachlichen Anforderungen der Geschäftsprozesse orientiert [vgl. Aier/Schönherr 2004, 24].

3.6 Qualitative Anforderungen an Services in SOAs

Für den qualitätsgesicherten Betrieb in geschäftskritischen Anwendungen ergibt sich eine Reihe von Qualitätsanforderungen. Für den Informationsaustausch zwischen Unternehmen sind Sicherheitsaspekte unverzichtbar, weil davon auszugehen ist, dass es sich beim Nachrichtenaustausch um sensible, geschäftskritische Daten handelt. Im Rahmen der umfassenden Sicherheitsspezifikation WS-Security finden sich deshalb nicht nur Verschlüsselungsmechanismen, sondern auch Konzepte zur Authentifizierung, Autorisierung, Nachrichtenintegrität und Nichtabstreitbarkeit. Einen guten Einstieg in die Sicherheitsthematik und beabsichtigte Standards für Web Services bietet [IBM02].

Bezüglich Transaktionskonzepte ist zu sagen, dass neben den klassischen Ansätzen von ACID-Transaktionen und 2-Phasen-Commit verstärkt so genannte Kompensations-

© HSG / IWI / CC BN3 / 1

transaktionen bei der Serviceinteraktion eine wichtige Rolle spielen werden, weil bei so genannten long-running Transaktionen Sperrkonzepte als nicht durchsetzbar gelten [BEA03].

Unter Zuverlässigkeit ist in diesem Kontext vor allem die garantierte Übermittlung der Nachrichten bei der Service Interaktion gemeint. Die WS-ReliableMessaging [WSRM04] Spezifikation von IBM und Microsoft stellt einen wichtigen Schritt in diese Richtung dar. Es bleibt allerdings abzuwarten, ob und wie schnell sich die Konzepte durchsetzen werden.

Abschließend kann gesagt werden, dass gerade im Bereich Qualität die Web Service Technologie noch Unzulänglichkeiten aufweist, die für den geschäftskritischen Einsatz zu beheben sind. Die Bemühungen im Rahmen der Fortentwicklung von WS-Spezifikationen indes zeigen, dass dieses Problem erkannt ist.

3.7 Nutzenpotenziale und Herausforderungen

Unternehmen stehen bei der Gestaltung ihrer IS-Architekturen häufig vor einem Zielkonflikt: Sie müssen integrierte, automatisierte Abläufe so realisieren, dass die Komplexität der IS-Architektur beherrschbar und gleichzeitig die Flexibilität für Anpassungen an neue Anforderungen erhalten bleibt [vgl. Hagen 2004, 73]. Neue Anforderungen an das IS können dabei sowohl geschäftlich (z.B. Realisierung neuer kunden- bzw. partnerspezifischer Zugangskanäle auf interne Daten und Funktionen, Anpassungen von Prozessabläufen, Auslagerung von Aktivitäten an Dienstleister) als auch technisch (z.B. Harmonisierung von Systemplattformen, Releasewechsel von Anwendungssystemen) getrieben sein. Das grosse Nutzenpotenzial einer SOA liegt in der Bildung einer Abstraktionsschicht, welche die fachliche Spezifikation von Geschäftsprozessen stärker von ihrer technischen Implementierung entkoppelt. Eine SOA kann dadurch die genannten Architekturziele in verschiedener Hinsicht unterstützen:

Eine SOA ermöglicht die Trennung generischer Fachlogik auf Serviceebene, geschäftsfallspezifischer Ablauflogik auf Ebene Prozessintegration und anwenderspezifischer Benutzerführungslogik auf Ebene Desktopintegration. Die dadurch realisierte Schichtenbildung erlaubt es, bei neuen Geschäftsanforderungen Anpassungen isolierter und gezielter vorzunehmen: Änderungen im Benutzerprozess oder im Ablauf eines Auftragsabwicklungsprozesses (z.B. neue Reihenfolge einzelner Arbeitsschritte, Ergänzung des Prozesses um zusätzliche Aktivitäten) führen im Idealfall hauptsächlich zur Neudefinition von Ablauflogik und nicht zu einer Neuentwicklung von Geschäftsfunktionalität oder Schnittstellen auf Ebene der An-

© HSG / IWI / CC BN3 / 1

wendungssysteme. Ein reduzierter Anpassungsaufwand des IS entsteht insbesondere dann, wenn die in Services gekapselte Geschäftslogik eine höhere zeitliche Stabilität aufweist als die benutzer- oder geschäftsfallsspezifische Prozesslogik. Weitere Nutzenpotenziale bergen auch die mit der fachlichen und technischen Standardisierung erreichte Interoperabilität auf Serviceebene und die Abstraktion von der Art und Weise, wie Services ihre Leistungen erbringen. Dadurch muss der Servicenutzer (Composite Applications, Workflows) nicht wissen, an welchem Ort der Service physisch implementiert ist (bereichs- bzw. unternehmensintern oder –extern), auf welcher technischen Basis (in welchem Anwendungssystem, auf welcher technischen Plattform) seine Logik umgesetzt wird und welche interne Logik (interne Datenmodelle und Algorithmen, Zustände etc.) er im Detail besitzt. Solange sich die Schnittstelle nicht verändert und die vereinbarten Standards eingehalten werden, können Services folglich beliebig physisch verteilt (z.B. an einen externen Dienstleister ausgelagert) und die zugrunde liegenden Anwendungssysteme und technischen Plattformen ausgetauscht werden. Eine Einigung auf z.B. die Verwendung von Web Service-Standards erlaubt es damit, dass sich das Hauptaugenmerk bei der Integration von technischen Aspekten (z.B. Schnittstellenbeschreibungs- und Nachrichtenformate, Kommunikationsprotokolle etc.) auf die semantischen Belange der Schnittstellen verlagert. Eine an betrieblichen Prozessaktivitäten und Arbeitsschritten orientierte, grobe Granularität von Services reduziert die Komplexität der Entwicklung neuer Lösung auf Basis existierender Services weiter. Services überbrücken die konzeptionelle Lücke zwischen der fachlichen Modellierung Geschäfts- bzw. Benutzerprozessen und der über Objekt-, Funktions- oder Datenschnittstellen in einzelnen Anwendungssystemen verfügbaren Funktionalität. Services erleichtern es Entwicklern damit, bestehende Funktionalität ausserhalb ihres ursprünglich beabsichtigten Kontexts wieder zu verwenden und führen zu geringerer funktionaler Redundanz im IS. Gleichzeitig sorgt eine anwendungs- und geschäftsprozessorientierte Abgrenzung von Services dafür, dass die Kapselung bestehender Funktionalität – und damit die Implementierung von Schnittstellen auf Ebene Anwendungssystem – nur dort wo und soweit wie nötig geschieht: Die Granularität eines Services sollte so grob sein, dass er eine fachlich sinnvolle Aufgabe (z.B. Verfügbarkeitsprüfung) erfüllt, aber gleichzeitig fein genug, um in den aus Geschäftssicht erforderlichen unterschiedlichen Kontexten (z.B. Verfügbarkeitsprüfung in unterschiedlichen Varianten der Auftragsabwicklung) mehrfach verwendbar zu sein.

© HSG / IWI / CC BN3 / 1

Eine SOA verspricht damit verschiedene positive Effekte sowohl bei der Entwicklung neuer wie auch der der Anpassung bestehender Anwendungssysteme: Dazu zählen positive Kosteneffekte (z.B. verringerte Entwicklungskosten und Investitionsschutz bei stärkerer Wiederverwendung von Softwarebestandteilen), Zeiteffekte (z.B. kürzere Entwicklungszeiten und „Time-to-Market“ für neue Integrationslösungen) und Qualitätseffekte (z.B. lokale Problembegrenzung innerhalb eines Services) (vgl. [Stiemerling 2002, 436], [Kaib 2002, 30], [Korthaus 2001, 24]). Sie erhöhen damit letztlich die Agilität und Wettbewerbsfähigkeit von Unternehmen.

Die hier angesprochenen Potenziale von SOA sind allerdings zu relativieren und lassen sich nur durch sorgfältige und systematische Gestaltung der Serviceebene realisieren: Der Senkung von Entwicklungs- bzw. Anpassungskosten von Anwendungssystemen stehen z.B. oft hohe Infrastrukturkosten und höhere Kosten für die Entwicklung generischer, wieder verwendbarer Services gegenüber [vgl. Baldwin/Clark 1997, 86]. Hohe Einarbeitungszeiten bei zu vielen hochgranularen und spezialisierten Services können die Zeiteinsparungen in der Anwendungsentwicklung zunichte machen [vgl. Stiemerling 2002, 437]. Schliesslich erhöhen zu viele Schnittstellen die Komplexität der IS-Architektur – und damit z.B. den Aufwand für die Fehlersuche –anstatt sie zu senken [vgl. Aier/Schönherr 2004, 40].

© HSG / IWI / CC BN3 / 1

4 Ausprägungen des Frameworks in konkreten Fallbeispielen

4.1 Serviceorientierte Architektur bei der Deutschen Post Brief

Auslöser für die Implementierung einer SOA im Unternehmensbereich Brief der Deutschen Post war die Erkenntnis, dass die existierende IT-Architektur den heutigen Agilitätsanforderungen aus dem Geschäft nicht mehr gerecht wurde. Der Unternehmensbereich besass bis dahin keine fachbereichsübergreifende, zentral geplante IT-Architektur: eine projektübergreifende Koordination und Planung von IT-Projekten fehlte. Die geringe Verfügbarkeit übergreifender Kerninformationen (Kunden, Kosten, Wettbewerberinformationen etc.) und heterogene, für Einzelprojekte implementierte Systemschnittstellen mit vielen undokumentierten Abhängigkeiten führten bei neuen Projekten zu einem hohem Anpassungsaufwand sowie mangelhafter Wiederverwendung bestehender Funktionen. IT-Projekte wurden zunehmend risikoreicher und konnten neue Geschäftsanforderungen nicht mehr in der geforderten Zeit umsetzen. Darüber hinaus flossen immer grössere Anteile des IT-Budgets in den Betrieb und die Wartung von Systemen und nur ein geringer Teil stand für Neu- bzw. Weiterentwicklungen, z.B. zur Unterstützung neuer Prozesse oder Kundekanäle, zur Verfügung [Herr/Brombach 2004].

Als Reaktion auf diese Herausforderungen beschloss Deutsche Post Brief 1999, eine IT-Architektur nach den Prinzipien serviceorientierter Architekturen zu entwickeln. Die zentrale Zielsetzung bestand darin, unternehmerische Agilität und technische Flexibilität durch eine bessere Ausrichtung der IT auf Geschäftsanforderungen (Business / IT Alignment) und eine stärkere Entkopplung von Geschäft bzw. Prozessen und unterstützender Technologie zu erreichen [Helbig 2005]. Die Umsetzung erfolgte in mehreren Teilprojekten:

- ein *strategisches Projekt* definierte voneinander abgegrenzte Anwendungsdomänen und Serviceschnittstellen für den Zugriff auf Funktionen und Daten in diesen Domänen.
- verschiedene *Fachprojekte* implementierten einzelne Domänen und Services (z.B. durch Aufbau einer zentralen Kundendatenbank).

© HSG / IWI / CC BN3 / 1

- die IT-Abteilung implementierte Middlewaredienste in Form des sog. *Service Backbone (SBB),* einer Integrationsplattform für die Kopplung von Service-Anbietern und –Nutzern.

Abbildung 4-2 verdeutlicht die Kerngestaltungsbereiche der Serviceorientierten Architektur im Fall der Deutschen Post Brief unter Nutzung des entwickelten Metamodells.

© HSG / IWI / CC BN3 / 1

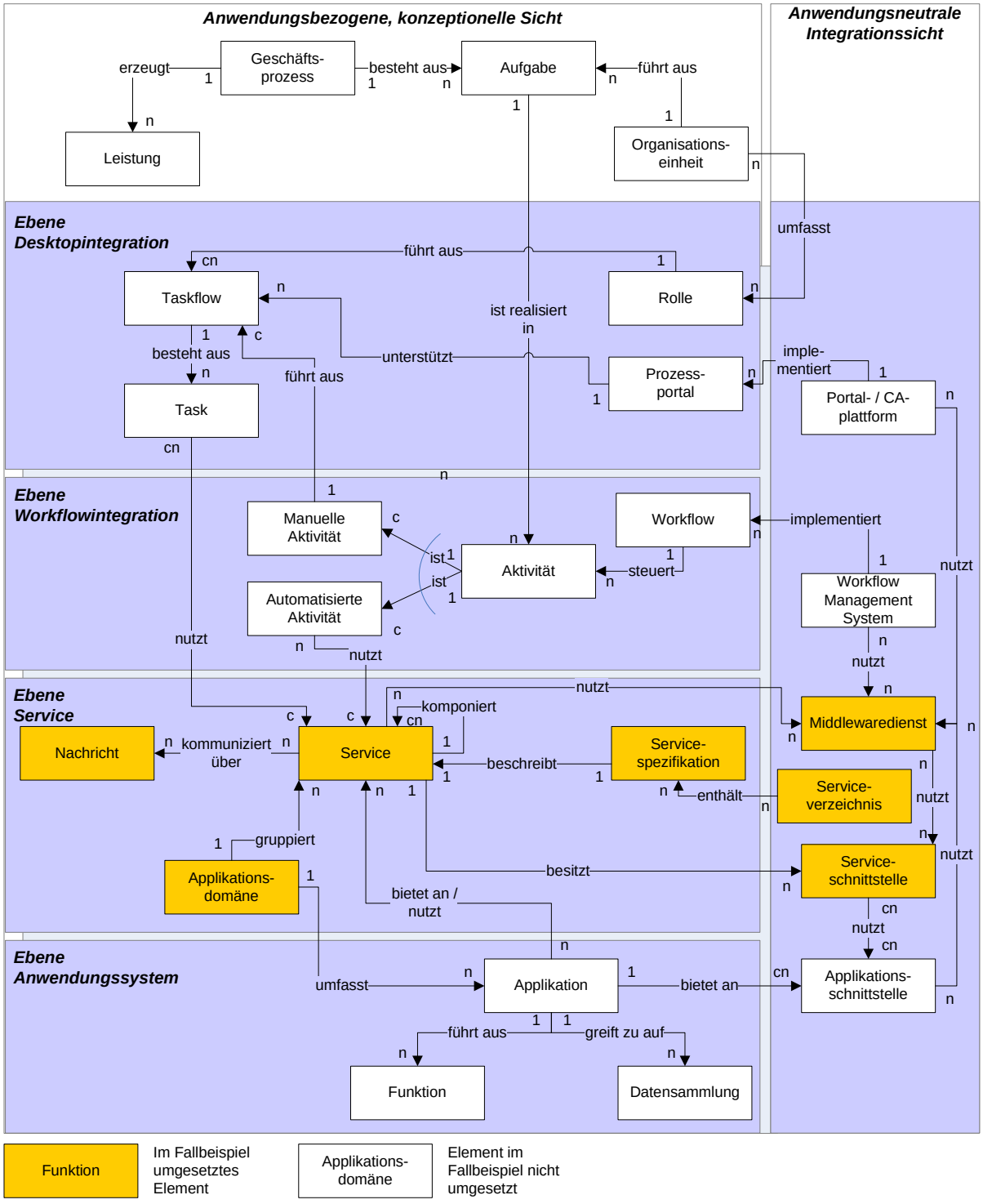


Abbildung 4-1: Hauptgestaltungsbereiche der SOA im Fall Deutsche Post Brief

Basierend auf einer Analyse der Kerngeschäftsprozesse identifizierte die Deutsche Post Brief zunächst Funktionen und Daten, die in mehreren Prozessen verwendet werden und ordnete diese fachlichen Applikationsdomänen zu. Als Cluster-Kriterium zur Abgrenzung der Domänen dienten Datenströme (Input-Output-Beziehungen) zwischen den einzelnen

© HSG / IWI / CC BN3 / 1

Geschäftsfunktionen: Gruppen von Funktionen mit vielen und komplexen Austauschbeziehungen untereinander wurden jeweils in eigenen Domänen zusammengefasst, wobei die Domänen untereinander möglichst wenige Schnittstellen aufweisen sollten. Für den Domänen-externen Zugriff auf Funktionen und Daten definierte das Projektteam Serviceschnittstellen.

Als Basis für die Serviceidentifikation diente, abgeleitet von den zu Anwendungsdomänen zusammengefassten Geschäftsfunktionen und Geschäftsobjekten, eine Sammlung von Fachklassen. Diese Klassen stellten Kandidaten für mögliche Services wie z.B. „CustomerManagement“ dar. Anhand der Basisoperationen Create, Read, Update, Delete (CRUD) auf den Fachklassen und deren wichtigsten Attributen wurden Serviceoperationen wie bspw. „createCustomer“ oder „readCustomer“ definiert. In einem nächsten Schritt fand eine Eliminierung überflüssiger Services durch eine Analyse von Use Cases und Domänenbeziehungen statt: In den Use Cases wurden nicht benötigte Operationen sowie Services und Operationen herausgefiltert, die kein Wiederverwendungspotenzial haben oder nur innerhalb einer Domäne Verwendung finden. Die Use Case Analyse gab ausserdem Hinweise auf fehlende Services und Operationen.

Der sog. *Service Backbone (SBB)* stellt den Kernbestandteil der technischen Umsetzung der SOA bei Deutsche Post Brief dar und unterstützt die Kommunikation zwischen Serviceanbietern und –nutzern durch verschiedene Middlewaredienste:

- Das *Java Messagig Service* (JMS)–basierte *Message Queuing* stellt die reibungslosen Kommunikation zwischen Serviceanbieter und -nutzer sicher.
- Das LDAP- und HTTP-basierte *Service Directory* dient als zentrales Verzeichnis für Servicespezifikationen und ermöglicht in erster Linie ein dynamisches Binden von Services zur Laufzeit sowie die Authentisierung und Authorisierung der Servicenutzung.
- Die *technischen Serviceteilnehmer* bieten klar definierte Integrationsfunktionen über SBB-Schnittstellen an. Beispiele sind die Komponenten *Security and User Management* (stellt Funktionalität zur zentralen Administration von Benutzerdaten und Zugriffsberechtigungen zur Verfügung), *Transformation Service* (stellt Routinen

© HSG / IWI / CC BN3 / 1

für die Transformation von Nachrichten und Daten bereit) oder *Parsing and Validation* (prüft die Gültigkeit der über den SBB ausgetauschten Nachrichten.

4.2 Serviceorientierte Architektur bei Credit Suisse

Ende der 90er Jahre stand die Credit Suisse vor der Herausforderung einer historisch gewachsenen, komplexen IT-Architektur, die Anpassungen nur mit unverhältnismässig hohem Aufwand erlaubte ([vgl. Hagen 2004, 68], [Heutschi/Schemm 2005]): Alleine für das Schweizer Bankgeschäft befanden sich rund 600 in verschiedenen Technologien entwickelte Applikationen im Einsatz, die unterschiedliche Lebenszyklen aufwiesen, Funktionen und Daten redundant implementierten und viele unkontrollierte Abhängigkeiten bzw. spezialisierte Punkt-zu-Punkt Schnittstellen untereinander besassen. Vor dem Hintergrund der damals geplanten umfangreichen Einführung Intranet- und Internet-basierter Applikationen untersuchte die zentrale IT-Architektur-Abteilung unter der Verantwortung des CIO der Credit Suisse verschiedene Optionen zur Vermeidung eines weiteren Komplexitätswachstums und zur Entwicklung einer beherrschbaren, zunkunftsfähigen IT-Architektur.

Unter dem Begriff „Managed Evolution“ entschied sich die Credit Suisse 1998 zur Etablierung einer zentral gesteuerten Integrationsarchitektur nach den Prinzipien der Serviceorientierten Architektur. Die Umsetzung basiert auf drei Hauptaktivitäten:

1. Der Entwicklung *zentraler Architekturrichtlinien und Organisationsstrukturen* als Basis für das Architekturmanagement.
2. Der fachlichen Gruppierung von Applikationen in *Applikationsdomänen* als Basis für Integrationsentscheidungen.
3. Der Entwicklung einer *zentralen Integrationsinfrastruktur* als technische Basis für die domänenübergreifende, servicebasierte Applikationsintegration.

Abbildung 4-2 verdeutlicht die Kerngestaltungsbereiche der Serviceorientierten Architektur im Fall der Credit Suisse unter Nutzung des entwickelten Metamodells.

© HSG / IWI / CC BN3 / 1

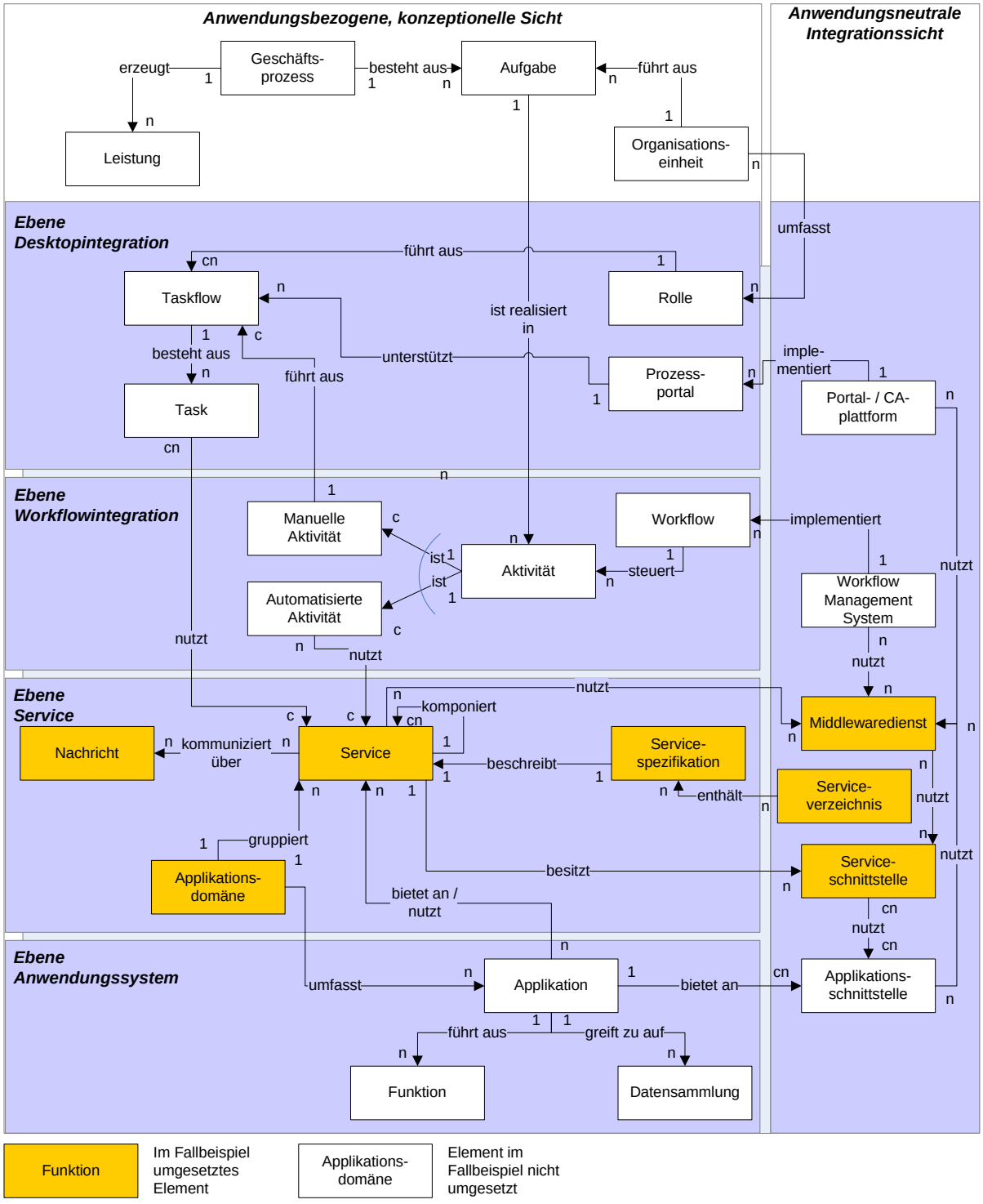


Abbildung 4-2: Hauptgestaltungsbereiche der SOA im Fall von Credit Suisse

Applikationsdomänen fassen bei der Credit Suisse alle Applikationen und Daten zusammen, die zu einem bestimmten fachlichen Bereich gehören und bilden wichtige Anhaltspunkte für die zu wählenden Kopplungsgrade in Integrationsprojekten. Während innerhalb von Domänen enge Kopplungsbeziehungen zwischen Applikationen und

Applikationsbestandteilen zulässig sind (z.B. durch Verwendung einer gemeinsamen Datenbank), wird zwischen Domänen eine konsequente Kapselung über wohldefinierte und stabile Services, die alle Details ihrer Implementierung verbergen, umgesetzt. Zusätzlich stellen die Applikationsdomänen für Credit Suisse auch ein wichtiges organisatorisches Konzept dar: jede Domäne besitzt einen verantwortlichen Architekten, Architektur-Assessments werden pro Domäne durchgeführt und auch das IT-Budget wird auf Applikationsdomänen aufgeteilt.

Neben der Domänenzugehörigkeit von Funktions- und Datenanforderungen berücksichtigt ein mehrstufiger Serviceentwicklungs- und reviewprozess die technischen Plattformen der potenziellen Serviceanbieter und -nutzer sowie nichtfunktionale Eigenschaften des zu unterstützenden Prozesses bzw. der Interaktion bei der Anpassung bestehender oder der Entwicklung neuer Services.

Im Anschluss an die Entwicklung werden die Public Services im zentralen Serviceverzeichnis erfasst. Die in einer SQL-Datenbank enthaltenen Servicespezifikationen sind nach einheitlichen Vorgaben strukturiert und enthalten folgende Elemente:

- Eine formale, maschinenlesbare Schnittstellenspezifikation mit Operationen und Input- / Outputparametern (in IDL oder XML),
- eine knappe natürlichsprachliche Beschreibung der Servicefunktion, Operationen, Parameter und Effekte auf Daten,
- Ausnahmen und Vor- / Nachbedingungen (natürlichsprachlich, da eine formale Beschreibung in OCL zu kompliziert ist),
- angestrebte Servicelevels (Antwortzeitverhalten, Durchsatz, Reaktionszeiten im Fehlerfall),
- Planungsangaben (Status, Daten für durchgeführte oder geplante Tests, Datum der Produktivsetzung),
- Kontaktinformationen (Serviceentwickler, -betreiber und -nutzer),
- Implementierungsinformationen (Plattformen, Applikationen, Datenbanken).

© HSG / IWI / CC BN3 / 1

Die Serviceimplementierungen greifen auf Middlewaredienste einer zentralen Integrationsinfrastruktur zurück. Credit Suisse hat diese Middlewaredienste phasenweise entwickelt und ausgebaut, angefangen von einem Integrationsbus für synchrone Serviceaufrufe, über die Entwicklung eines asynchronen Event-Busses bis zur Ergänzung um einen Filetransfer-Bus für grosse Datenmengen.

4.3 Serviceorientierte Architektur für einen integrierten Beraterarbeitsplatz bei einer Regionalbank

Die regionale Universalbank steht vor der Herausforderung, steigender Kundenvolatilität und zunehmend heterogenen Kundenanforderungen gerecht zu werden, um im regionalen, nationalen und internationalen Wettbewerb zu bestehen. Gleichzeitig zwingen eine steigende Regulierungsdichte und anhaltender Druck auf Margen zu einer Effizienzsteigerung im Geschäftsbetrieb. Wichtige durch IT zu unterstützende Geschäftsziele sind daher u.a. die Steigerung der Geschwindigkeit und Qualität der Beratung von Kunden mit heterogenen Wünschen sowie eine höhere Flexibilität bei der Neugestaltung von Prozessen.

Die für die Kundenberater eingesetzte SAP Standardlösung wies für die meisten Frontoffice-Beratungsaktivitäten eine unnötig komplizierte Benutzeroberfläche und Prozessführung auf: Ca. 80% der Kundenberatungen betreffen einfache Aktivitäten wie das Anlegen von Kunden, Konten oder Dienstleistungen und nur etwa 20% der Geschäftsfälle benötigen die vollständige SAP-Funktionalität. Gleichzeitig deckte SAP Banking nicht alle im Beratungsprozess benötigten Funktionen und Informationen ab, welche in separaten Anwendungssystemen realisiert wurden. Die Entwicklung eines integrierten Beraterarbeitsplatzes (BAP) verfolgte daher das Ziel, den Kundenberatern eine auf ihre Bedürfnisse zugeschnittene, vereinfachte Benutzeroberfläche bei gleichzeitiger Integration aller benötigten Funktionalität zu bieten um so den administrativen Aufwand zu reduzieren und Medienbrüche zu vermeiden.

Die zwischen September 2003 und Oktober 2004 entwickelte Lösung des BAP integriert verschiedene Bankanwendungen und bringt Expertenfunktionen zu den Kundenbetreuern, um die Frontoffice-Prozesse im Bereich Konto- und Kartenverwaltung für Geschäfts- und Privatkunden zu unterstützen. Im Leistungsumfang enthalten sind z.B. geführte Prozessablaufsteuerungen für die Neuanlage von Partnern, Konten und Dienstleistungen. Der BAP besteht im Kern aus einer Java-basierten Frontoffice-Applikation, welche eine eigene,

© HSG / IWI / CC BN3 / 1

aus SAP Banking herausgelöste Benutzeroberfläche und Prozesssteuerung bietet und Funktionen von SAP Banking, internen nicht-SAP Systemen sowie externen Anwendungssystemen von Partnern unter einer Oberfläche integriert.

Der BAP stellt auf *Ebene der Desktopintegration* eine Composite Application dar, welche hauptsächlich Darstellungs- und (Benutzer-) Steuerungslogik in Form von Guided Procedures umfasst und selber keine persistenten Fachdaten hält. Er greift über Remote Function Calls (RFCs) auf in der *Service-Ebene* gekapselte und in (Backend-) Anwendungssystemen implementierte SAP-Funktionalität zu. Services realisieren Funktionsschnittstellen mit dem Leistungsumfang bzw. in der Granularität, in der sie von den nutzenden Prozessen benötigt werden und harmonisieren allfällige heterogene technische Schnittstellenbeschreibungen und Kommunikationsprotokolle. Momentan nicht als Service gekapselte Funktionalität (alle nicht-SAP Funktionen und Daten) werden über alternative Technologien (z.B. das Java Messaging Service (JMS) Protokoll und einen MQ Series-Adapter) direkt in den BAP integriert. Die Regionalbank verzichtet bei dieser Lösung auf eine zusätzliche Middleware - stattdessen wurden die vorhandenen SAP-Schnittstellen (Business Programming Application Interfaces, BAPIs) direkt auf dem Application Server von SAP Banking in Form von ABAP-Objects als Services gekapselt. Auf *Ebene Workflowintegration* wurde eine Business Process Management (BPM) Lösung umgesetzt, durch welche Frontoffice-Mitarbeiter aus dem BAP heraus einen einfachen Workflow zur Kommunikation mit dem Backoffice anstossen können. Die *Anwendungssystem-Ebene* enthält die eigentlichen datenhaltenden und Fachlogik implementierenden Applikationen, die ihre Funktionen über plattformspezifische Schnittstellen (im Fall von SAP über BAPIs) zur Verfügung stellen.

© HSG / IWI / CC BN3 / 1

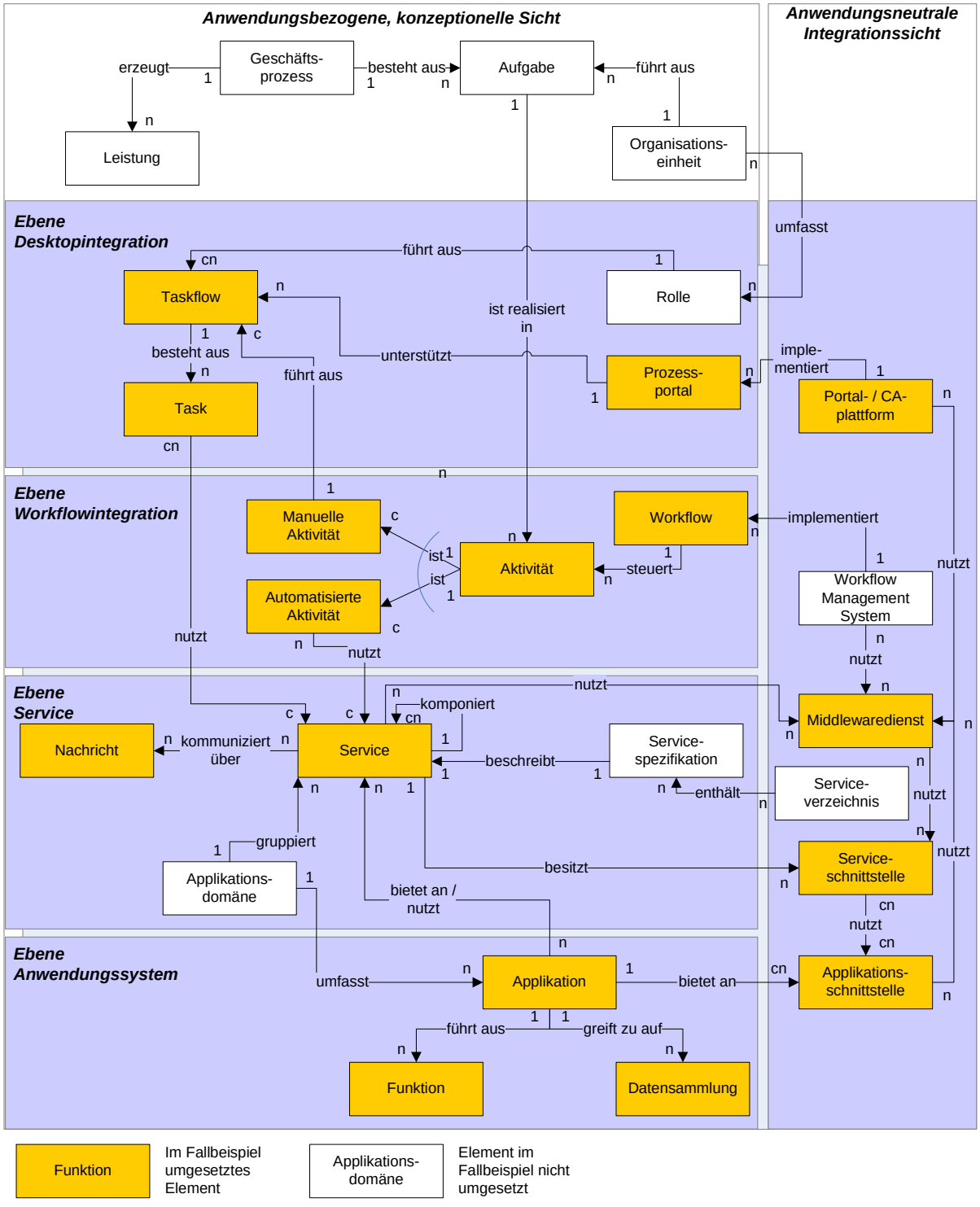


Abbildung 4-3: Die Elemente des integrierten Beraterarbeitsplatzes

© HSG / IWI / CC BN3 / 1

4.4 Web Service Schnittstelle für die zwischenbetriebliche Auftragsabwicklung in der Schweizer Haustechnikbranche

Die Interessengemeinschaft Datenverbund für die Haustechnik (IGH) ist ein Verein von Herstellern und Händlern der Bereiche Heizung, Lüftung und Sanitär in der Schweiz. Zweck der IGH ist es, Know-how, Technik und die Koordination für einen standardisierten Datenaustausch zwischen den Geschäftspartnern in der Haustechnikbranche zur Verfügung zu stellen [Heutschi 2003].

Mit dem Standard und der Software Win_Expert entwickelte die IGH in der Vergangenheit eine Lösung für den Katalogaustausch zwischen Anbietern (Herstellern und Händlern) und Installateuren. Diese erlaubt es den Installateuren direkt aus ihren Branchenanwendungen auf die elektronische Produktkataloge ihrer Händler zuzugreifen und die hinterlegten Informationen für die Zusammenstellung einer Grobofferte für Bauherren sowie für Angebotsanfragen oder Bestellungen bei Händlern zu nutzen. Im gesamten Auftragsabwicklungsprozess existierten aber mangels Möglichkeiten eines elektronischen Dokumentenaustauschs noch viele Medienbrüche. Die Notwendigkeit, per Fax oder Telefon übermittelte Anfragen, Offerten, Lieferscheine etc. manuell zu erfassen, verursachte bei allen beteiligten Parteien hohen administrativen Aufwand und machte die Offertenstellung und Auftragsabwicklung fehleranfällig.

Von April 2001 bis Oktober 2002 entwickelte die IGH ein XML-basiertes Katalog- und Dokumentenformat und den Web Service DataExpert, um einen standardisierten Austausch von Geschäftsdokumenten zu unterstützen und so die Wiederverwendung einmal erfasster Daten zu erhöhen. Die realisierte Lösung bildet hauptsächlich die *Ebene Service* und die *anwendungsneutrale Integrationssicht* aus. Keine Rolle spielen die Bildung von Anwendungsdomänen und Serviceverzeichnissen auf diesen Ebenen, da nur ein Service realisiert wurde. Die IGH stellt Anbietern (Herstellern bzw. Händlern) eine auf .Net basierende Server-Komponente des Services und Kunden (Installateuren bzw. Händlern) eine Client-Komponente zur Verfügung. Das Szenario sieht vor, dass jeder Anbieter den DataExpert-Service (Serverkomponente) bei sich installiert und die Hersteller der Branchensoftware auf Installateurseite die Client-Komponente in ihre Produkte einbauen. So wird die elektronische Übermittlung und Verarbeitung der Dokumente ‚Anfrage', ‚Offerte', ‚Bestellung', ‚Auftragsbestätigung', ‚Abruf', ‚Lieferschein' und ‚Rechnung' direkt aus den jeweiligen Anwendungssystemen heraus möglich. DataExpert übernimmt dabei

© HSG / IWI / CC BN3 / 1

ausschliesslich die Rolle einer Nachrichtenvalidierungs- und -übertragungsplattform. Die Steuerung des Geschäftsprozesses über Task- und Workflows wird nach wie vor von den Branchenanwendungen des Installateurs übernommen, ebenso wie die Abläufe (Funktionsaufrufe, Datenzugriff) in den ERP-Systemen der Lieferanten. Die Integration des Web Services in die Backendsysteme erfolgt über File-Import/Export oder über COM- bzw. Net-Schnittstellen (Middlewaredienste). Um Dokumente elektronisch empfangen und versenden zu können, müssen Installateure bei jedem ihrer Lieferanten ein Benutzerkonto eröffnen und die Zugangsinformationen in einer lokalen Konfigurationsdatei verwalten. Der Web Service stellt somit aus logischer Sicht (einheitliche Funktionalität, Semantik) einen zentralen Dienst für den Nachrichtenaustausch dar, wird aber physisch von jedem Anbieter von Sanitär-, Lüftungs- oder Heizungsprodukten in einer eigenen Instanz mit eigener Adresse (URI) betrieben, wobei die Branchenanwendung des Installateurs die richtige Serviceinstanz auswählt. Die verschiedenen DataExpert-Instanzen und die Schnittstellen in den Branchenanwendungen bilden so eine technisch und semantisch standardisierte Infrastruktur, die eine m:n-Verbindung zwischen Anbietern (Hersteller und Händler) und Kunden (Installateure) erlaubt.

© HSG / IWI / CC BN3 / 1

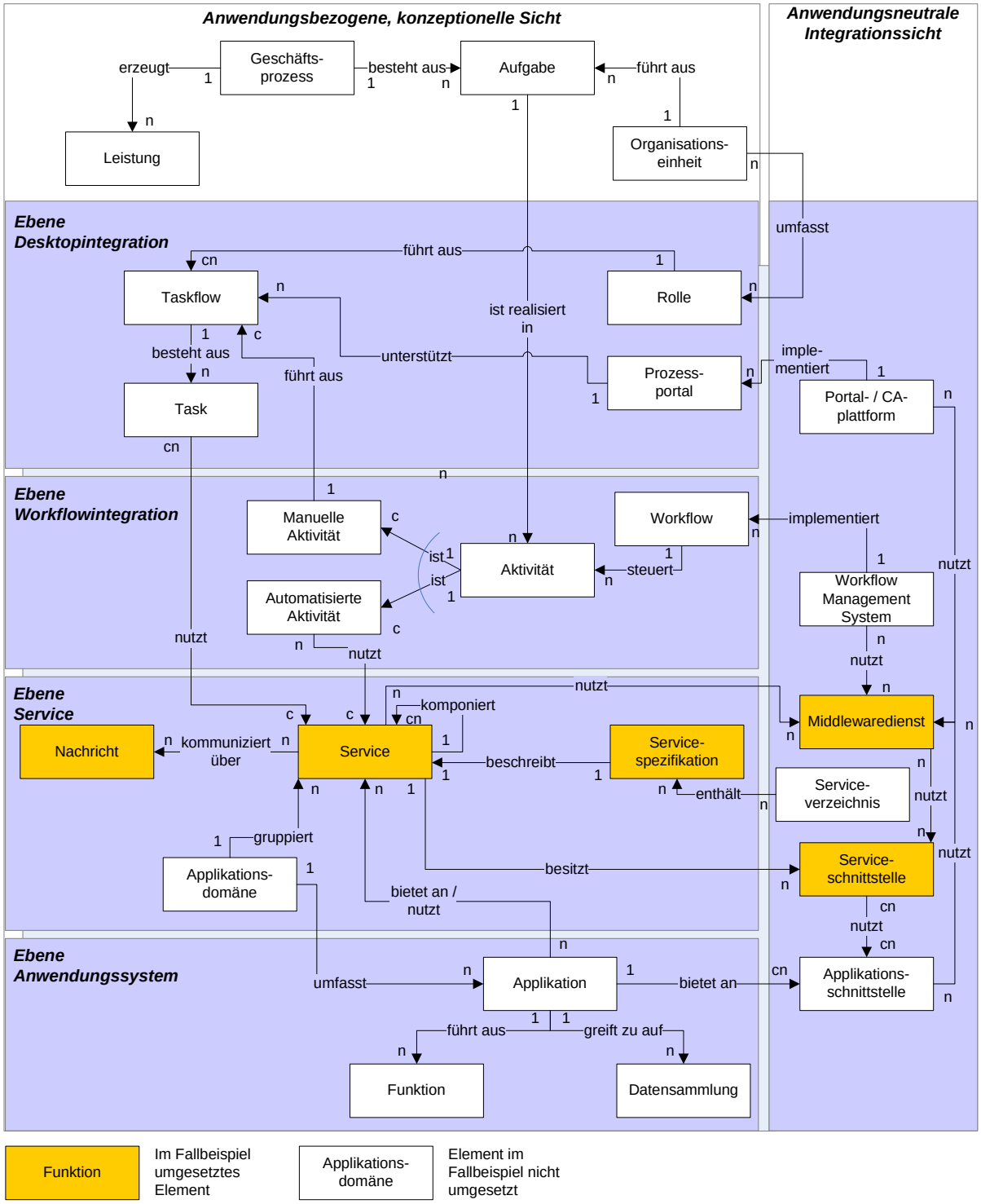


Abbildung 4-4: Hauptgestaltungselemente des Web Services für die Lieferanten-Kunden-Integration

5 Zusammenfassung und Ausblick

Der vorliegende Arbeitsbericht ordnet SOA im Rahmen des Business Engineering ein und nimmt eine Abgrenzung des Konzepts zu bestehenden Integrationsansätzen, d.h. zur Desktop-, Prozess- und Systemintegration vor. Er entwickelt ein Rahmenwerk für die Beschreibung einer SOA und führt dazu eine zusätzliche Serviceebene ein. Diese zusätzliche Schicht entkoppelt die Ebene der Desktop- und Prozessintegration von der Anwendungssystem-Ebene. Den Nutzenpotenzialen, die dies im Hinblick auf die Anpassungsfähigkeit von Informationssystemen und deren Flexibilität bietet, steht jedoch der Aufwand für den Aufbau einer entsprechenden Infrastruktur entgegen. In keinem der drei geschilderten Praxisbeispielen ist daher eine SOA vollständig umgesetzt. Vielmehr konzentrieren sich die Implementierungen auf einzelne Aspekte, in denen SOA konkreten Nutzen stiftet. Dies sind die Definition intern bzw. überbetrieblich wieder verwendbarer Services sowie deren Bündelung für den Benutzer in einer Composite Application.

Die vorliegende Version des Arbeitsberichts betrachtet SOA aus einer technischen Perspektive. Er klammert sowohl die Übertragung von SOA auf die Prozess-Ebene und die Nutzung von Services in Geschäftsprozessen wie auch die Rolle spezialisierter Serviceanbieter und die Entstehung neuer serviceorientierter Geschäftsmodelle aus. Diese Fragestellungen sind Inhalt der nächsten Versionen des Arbeitsberichts.

© HSG / IWI / CC BN3 / 1

Literatur

[Aier/Schönherr 2004]
Aier, S., Schönherr, M., Flexibilisierung von Organisations- und IT-Architekturen durch EAI, in: Aier, S., Schönherr, M. (Hrsg.), Enterprise Application Integration - Flexibilisierung komplexer Unternehmensarchitekturen, Aufl., GITO-Verlag, Berlin 2004, S. 1-60

[Alonso et al. 2003]
Alonso, G., Casati, F., Kuno, H., Machiraju, V., Web Services: Concepts, Architectures and Applications, Aufl., Springer, Berlin 2003

[Alt et al. 2004]
Alt, R., Cäsar, M. A., Leser, F., Österle, H., Puschmann, T., Reichmayr, C., Architektur des Echtzeit-Unternehmens, in: Alt, R., Österle, H. (Hrsg.), Real-Time Business: Lösungen, Bausteine und Potentiale des Business Networking, Aufl., Springer, Berlin etc. 2004, S. 19-52

[Arsanjani 2004]
Arsanjani, A., Service-oriented modeling and architecture - How to identify, specify, and realize services for your SOA, http://www-106.ibm.com/developerworks/webservices/library/ws-soa-design1/, 30.11.2004

[Baker 2002]
Baker, S., Web Services and CORBA, On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE Confederated International Conferences CoopIS, DOA, and ODBASE 2002, 2002, S. 618-632

[Baldwin/Clark 1997]
Baldwin, C. Y., Clark, K. B., Managing in an age of modularity, in: Harvard Business Review, 1997, Nr. September - October, S. 84-93

[Baskerville et al. 2005]
Baskerville, R., Cavallari, M., Hjort-Madsen, K., Pries-Heje, J., Sorrentino, M., Virili, F., Extensible architectures: The strategic value of service-oriented architecture in banking, Proceedings of the Thirteenth European Conference on Information Systems, 2005, S.

[Bloomberg 2004]
Bloomberg, J., Soa + Eda = Fud?, http://www.devx.com/opinion/Article/21326?trk=DXRSS_LATEST, 24.06.2004

[Brown et al. 2002]
Brown, A., Johnston, S., Kelly, K., Using service-oriented architecture and component-based development to build web service applications, Rational Software Corporation, 2002

[Channabasavaiah et al. 2003]
Channabasavaiah, K., Holley, K., Tuggle, E. M., Migrating to a service-oriented architecture, Part 1, http://www-106.ibm.com/developerworks/library/ws-migratesoa/, 26.06.2004

[Cheesman/Ntinolazos 2004]
Cheesman, J., Ntinolazos, G., The SOA Reference Model, in: CBDi Journal, 2004, Nr. März, S. 9-21

[Derungs 1997]
Derungs, M., Workflowsysteme zur Prozessumsetzung, Difo-Druck GmbH, Bamberg 1997

© HSG / IWI / CC BN3 / 1

[Dostal et al. 2005]
Dostal, W., Jeckle, M., Melzer, I., Service-orientierte Architekturen mit Web Services, 1. Aufl., Spektrum Akademischer Verlag, München 2005
[Eisenhardt/Brown 1998]
Eisenhardt, K. M., Brown, S. L., Time Pacing: Competing in markets that won't stand still, in: Harvard Bussines Review, 76, 1998, Nr. 2, S. 59-69
[Erl 2005]
Erl, T., Service-Oriented Architecture, Aufl., Prentice Hall, New York 2005
[Fritz 2004]
Fritz, F.-J., An Introduction to the Principles of Enterprise Services Architecture (ESA), http://www.sapinsideronline.com/spijsp/article.jsp?article_id=37906&volume_id=5147, 26.06.2004
[Gallas 2004]
Gallas, B. E., Der Aufbau eines Service Life Cycle Managements für eine Service Orientierte Architektur als Brücke zwischen Geschäftsprozess und IT Integration, in: Aier, S., Schönherr, M. (Hrsg.), Enterprise Application Integration - Band 2 : Serviceorientierung und nachhaltige Architektur, Aufl., GITO Verlag, Berlin 2004, S. 232-278
[Georgakopoulos/Sheth 1995]
Georgakopoulos, D., Sheth, A., An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, in: Distributed and Parellel Databases, 3, 1995, Nr. 1, S. 119-153
[Gioldasis et al. 2003]
Gioldasis, N., Moumoutzis, N., Kazasis, F., Pappas, N., Christodoulakis, S., A Service Oriented Architecture for Managing Operational Strategies, ICWS-Europe 2003, LNCS 2853, 2003, S. 11-23
[Gisolfi 2001]
Gisolfi, D., Web services architect, Part 3: Is Web services the reincarnation of CORBA?, http://www-106.ibm.com/developerworks/webservices/library/ws-arc3/,
[Hagen 2004]
Hagen, C., Integrationsarchitektur der Credit Suisse, in: Aier, S., Schönherr, M. (Hrsg.), Enterprise Application Integration - Flexibilisierung komplexer Unternehmensarchitekturen, Aufl., GITO Verlag, Berlin 2004, S. 61-83
[Helbig 2005]
Helbig, J., Welcome and Introduction, Beitrag in: Aufl., SOA Days Bonn 2005, S.
[Herr et al. 2004]
Herr, M., Bath, U., Koschel, A., Implementation of a Service Oriented Architecture at Deutsche Post MAIL, Proceedings of the European Conference on Web Services, Springer, 2004, S. 227-238
[Herr/Brombach 2004]
Herr, M., Brombach, S., SBB Management Paper: Benefits and application range, http://www.sbb.org/uploads/images/26/SOP_Management_Paper_V1_0.pdf, 09.11.2004
[Hess 1996]
Hess, T., Entwurf betrieblicher Prozesse - Grundlagen, Bestehende Methoden, Neue Ansätze, 1. Auflage. Aufl., Gabler, Wiesbaden 1996
[Heutschi 2003]

© HSG / IWI / CC BN3 / 1

Heutschi, R., Fallstudie: Elektronischer Datenaustausch in der Schweizer Haustechnikbranche, BE HSG / BECS / 17, Institut für Wirtschaftsinformatik, Universität St. Gallen, St. Gallen 2003

[Heutschi/Schemm 2005]
Heutschi, R., Schemm, J. W., Fallstudie: Serviceorientierte Architektur bei der Credit Suisse, Institut für Wirtschaftsinformatik Universität St. Gallen, St. Gallen 2005

[Kaib 2002]
Kaib, M., Enterprise Application Integration. Grundlagen, Integrationsprodukte, Anwendungsbeispiele, Aufl., DUV Wirtschaftsinformatik, Wiesbaden 2002

[Klesse et al. 2005]
Klesse, M., Wortmann, F., Schelp, J., Erfolgsfaktoren der Applikationsintegration, in: Wirtschaftsinformatik, 47, 2005, Nr. 4, S. 259-267

[Korthaus 2001]
Korthaus, A., Komponentenbasierte Entwicklung conputergestützter betrieblicher Informationssysteme, Aufl., Peter Lang, Frankfurt et al. 2001

[Kossmann/Leymann 2004]
Kossmann, D., Leymann, F., Web Services, in: Informatik-Spektrum, 27, 2004, Nr. 2, S. 117-128

[Krafzig et al. 2004]
Krafzig, D., Banke, K., Slama, D., Enterprise SOA, Aufl., Prentice Hall, 2004

[Leymann/Roller 2000]
Leymann, F., Roller, D., Production Workflow, Aufl., Prentice Hall, Upper Saddle River 2000

[Linthicum 2000]
Linthicum, D. S., Enterprise Application Integration, Aufl., Addison-Wesley, Upper Saddle River, NJ 2000

[Linthicum 2003]
Linthicum, D. S., Next Generation Apllication Integration: From Simple Information to Web Services, Aufl., Addison-Wesley, Boston 2003

[Lublinsky 2004]
Lublinsky, B., SOA Design: Meeting in the Middle, in: Java Pro, 2004, Nr. August, S. 26-31

[Mantel/Schissler 2002]
Mantel, S., Schissler, M., Application Integration, in: Wirtschaftsinformatik, 44, 2002, Nr. 2, S. 171-174

[Mantel et al. 2004]
Mantel, S., Schissler, M., Zeller, T., Überbetriebliche Integration von Anwendungssystemen: Klassifikation von Integrationsproblemen und Lösungen, in: Bartmann, D., Mertens, P., Sinz, E.J. (Hrsg.), Überbetriebliche Integration von Anwendungssystemen: FORWIN-Tagung 2004, Aufl., Shaker, Aachen 2004, S. 1-20

[McCoy et al. 2004]
McCoy, D. W., Natis, Y., Pezzini, M., Sinur, J., Schulte, R., Thompson, J., Lheureux, B., Kenney, F., Haight, C., Friedmann, T., Gassmann, B., Phifer, G., James, G. A., Blechar, M., Vecchio, D., Hype Cycle for Application Integration and Platform Middleware, 2004, G00120915, Gartner Group, 2004

[McGovern et al. 2003]

© HSG / IWI / CC BN3 / 1

McGovern, J., Tyagi, S., Stevens, M., Mathew, S., Service-Oriented Architecture, in: McGovern, J., Tyagi, S., Stevens, M., Mathew, S. (Hrsg.), Java Web Services Architecture, Aufl., Morgan Kaufmann, 2003, S. 35-63

[Mertens 1997]
Mertens, P. (Ed.), Lexikon der Wirtschaftsinformatik, 3., vollständig neu bearbeitete und erweiterte Auflage. Aufl., Berlin 1997

[Natis 2003]
Natis, Y., Service-Oriented Architecture Scenario, Gartner, Inc., Stamford 2003

[Natis/Schulte 2003]
Natis, Y., Schulte, R., Introduction to Service-Oriented Architecture, Gartner Group, 2003

[Newcomer/Lomow 2004]
Newcomer, E., Lomow, G., Understanding SOA with Web Services, Aufl., Addison-Wesley, Maryland 2004

[Oasis 2005]
Oasis, Reference Model for Service Oriented Architectures, Working Draft 09, OASIS Open, 2005

[Österle 1995]
Österle, H., Business Engineering: Prozess- und Systementwicklung, 2. Aufl., Springer, Berlin 1995

[Österle/Winter 2003]
Österle, H., Winter, R., Business Engineering, in: Österle, H., Winter, R. (Hrsg.), Business Engineering, 2. Aufl. Aufl., Springer, Berlin etc. 2003, S. 3-18

[Papazoglou 2003]
Papazoglou, M. P., Service-Oriented Computing: Concepts, Characteristics and Directions, Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE 2003), 2003, S. 3-12

[Pezzini 2003]
Pezzini, M., Composite Applications Help Turn Legacies Into Assets, TU-20-9595, Gartner Group, 2003

[Puschmann 2003]
Puschmann, T., Collaboration Portale - Architektur, Integration, Umsetzung und Beispiele, Dissertation, Dissertation, Universität St. Gallen, Difo-Druck, Bamberg 2003

[Puschmann 2004]
Puschmann, T., Prozessportale - Architektur zur Vernetzung mit Kunden und Lieferanten, Aufl., Springer, Berlin 2004

[Raptis et al. 2001]
Raptis, K., Spinellis, D., Katsikas, S., Multi-technology distributed objects and their integration, in: Computer Standards & Interfaces, 2001, Nr. 23, S. 157-168

[Riehm/Vogler 1996]
Riehm, R., Vogler, P., Middleware: Infrastruktur für die Integration, in: Österle, H., Riehm, R., Vogler, P. (Hrsg.), Middleware: Grundlagen, Produkte und Anwendungsbeispiele für die Integration heterogener Welten, Aufl., Vieweg, Braunschweig etc. 1996, S. 25-135

[Ruh et al. 2001]
Ruh, W. A., Maginnis, F. X., Brown, W. J., Enterprise Application Integration, Aufl., John Wiley & Sons, New York 2001

© HSG / IWI / CC BN3 / 1

[Samtani/Sadhwani 2001]
Samtani, G., Sadhwani, D., EAI and Web Services, http://www.webservicesarchitect.com/content/articles/samtani01.asp, 24.08.2004
[SAP 2004]
SAP, SAP Composite Application Framework, http://www.sap.com/germany/solutions/netweaver/compositeapplicationframework.aspx, 10.12.2004
[Schelp/Winter 2002]
Schelp, J., Winter, R., Enterprise Portals und Enterprise Application Integration, in: Meinhard, S., Popp, K. (Hrsg.), Enterprise-Portale & Enterprise Application Integration, Aufl., dpunkt, Heidelberg 2002, S. 6-20
[Schlamann 2004]
Schlamann, H., Service Orientation: An Evolutionary Approach, in: Cutter IT Journal, 17, 2004, Nr. 5, S. 5-13
[Schwinn 2006]
Schwinn, A., Entwurfsmusterbasierter Ansatz zur Systematisierung von Applikationsbeziehungen im Business Engineering, in: Schelp, J., Winter, R. (Hrsg.), Integrationsmanagement, 31-59. Aufl., Springer, Berlin et al. 2006, S.
[Sims 2003]
Sims, O., Service Oriented Architecture Part 3 - Federation, in: CBDi Journal, 2003, Nr. May, S. 8-15
[Sinz 1999]
Sinz, E. J., Architektur von Informationssystemen, in: Rechenberg, P., Pomberger, G. (Hrsg.), Informatik-Handbuch, 2. Aufl., Hanser, München, Wien 1999, S. 1035-1046
[Sprott/Wilkes 2003]
Sprott, D., Wilkes, L., Understanding SOA, in: CBDi Journal, 2003, Nr. September, S. 4-14
[Stiemerling 2002]
Stiemerling, O., Web-Services als Basis für evolvierbare Softwaresysteme, in: Wirtschaftsinformatik, 44, 2002, Nr. 5, S. 435-445
[Stohr/Zhao 2001]
Stohr, E. A., Zhao, J. L., Workflow Automation: Overview and Research Issues, in: Information Systems Frontiers, 3, 2001, Nr. 3, S. 281-296
[Stojanovic 2005]
Stojanovic, Z., A Method for Component-Based and Service-Oriented Software Systems Engineering, Dissertation, Delft University of Technology, 2005
[Tsalgatidou/Pilioura 2002]
Tsalgatidou, A., Pilioura, T., An Overview of Standards and Related Technology in Web Services, in: Distributed and Parellel Databases, 12, 2002, Nr. S. 135-166
[Turowski et al. 2002]
Turowski, K., Ackermann, J., Brinkop, F., Conrad, S., Fettke, P., Frick, A., Glistau, E., Jeakel, H., Kotlar, O., Loos, P., Mrech, H., Ortner, E., Raape, U., Overhage, S., Sahm, S., Schmietendorf, A., Teschke, T., Vereinheitlichte Spezifikation von Fachkomponenten, Working Paper, Gesellschaft für Informatik, Arbeitskreis 5.10.3, Augsburg 2002
[van de Loo 2004]
van de Loo, K., What is an Enterprise Service?, SAP AG, 2004
[van der Aalst et al. 2003]

© HSG / IWI / CC BN3 / 1

van der Aalst, W., Weske, M., ter Hofstede, A., Business Process Management: A Survey, Proceedings of the BPM 2003, 2003, S. 1-12

[van Zyl 2002]
van Zyl, J., A perspective on service based architecture, Proceedings of SAICSIT, 2002, S. 249

[Vogler 2003a]
Vogler, P., Prozess- und Systemintegration: Umsetzung des organisatorischen Wandels in Prozessen und Informationssystemen, Habilitationsschrift, Universität St. Gallen, 2003a

[Vogler 2003b]
Vogler, P., Prozess- und Systemintegration: Umsetzung des organisatorischen Wandels in Prozessen und Informationssystemen, Habilitationsschrift, Habilitation, Universität St. Gallen, 2003b

[W3C 2004a]
W3C, Web Services Architecture, W3C Working Group Note 11 February 2004, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, 26.06.2004

[W3C 2004b]
W3C, Web Services Glossary, http://www.w3.org/TR/ws-gloss/, 29.07.2004

[W3C 2004c]
W3C, Web Services Glossary, W3C Working Group Note 11 February 2004, http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/, 10.10.2005

[Weilbach/Herger 2004]
Weilbach, J., Herger, M., SAP xApps und das Composite Application Framework, Aufl., Galileo Press, Bonn 2004

[WFMC 1999]
WFMC, Terminology & Glossary, http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf, 03.08.2004

[Wilkes/Harby 2004]
Wilkes, S., Harby, J., SOA Blueprints Concepts, The Middleware Company, 2004

[Winter 2003]
Winter, R., Modelle, Techniken und Werkzeuge im Business Engineering, in: Österle, H., Winter, R. (Hrsg.), Business Engineering, 2. Aufl. Aufl., Springer, Berlin etc. 2003, S. 87-117

[Winter 2004]
Winter, R., Architektur braucht Management, in: Wirtschaftsinformatik, 46, 2004, Nr. 4, S. 317-319

[Winter/Schelp 2005]
Winter, R., Schelp, J., Dienste-Orientierung im Business-Engineering Framework, in: HMD-Praxis der Wirtschaftsinformatik, 2005, Nr. 241, S. 45-54

[Woods 2003]
Woods, D., Packaged Composite Applications, Aufl., O'Reilly, London 2003

[Zhao/Cheng 2004]
Zhao, J. L., Cheng, H. K., Web Services and Process Management: A Union of Convenience or a New Area of Research?, in: Decision Support Systems, in Press, 2004, Nr. S.

[Zimmermann et al. 2003]
Zimmermann, O., Tomlinson, M., Peuser, S., Perspectives on Web Services, Aufl., Springer, Berlin 2003

© HSG / IWI / CC BN3 / 1

[Zur Muehlen 2004]
Zur Muehlen, M., Workflow-based Process Controlling. Foundation, Design, and Application of workflow-driven Process Information Systems, Aufl., Logos, Berlin 2004

© HSG / IWI / CC BN3 / 1