Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Dieter Masak

SOA?

Serviceorientierung in Business und Software

Mit 82 Abbildungen und 39 Tabellen

Springer

Dieter Masak
plenum Management Consulting
Hagenauer Str. 53
65203 Wiesbaden
dieter.masak@plenum.de

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

ISSN 1439-5428
ISBN 978-3-540-71871-0 Springer Berlin Heidelberg New York

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

Satz: Druckfertige Daten des Autors
Herstellung: LE-TEX, Jelonek, Schmidt & Vöckler GbR, Leipzig
Umschlaggestaltung: KünkelLopka Werbeagentur, Heidelberg
Gedruckt auf säurefreiem Papier 33/3180 YL – 5 4 3 2 1 0

Danksagung

. . . für Christiane. . .

Dr. Dieter Masak

Inhaltsverzeichnis

1

Prolog

Think what you will, we seize into our hands
His plate, his goods, his money and his lands.

King Richard II
William Shakespeare
1564 – 1616

Die Themen Serviceorientierung und Service Oriented Architecture sind ein weites Feld; das vorliegende Buch erhebt keinen Anspruch darauf, in die letzten technischen Feinheiten vorzudringen, sondern will einen Überblick zeigen und – vor allen Dingen – dem Leser[1] Denkanstöße vermitteln. Die hier vermittelten Details sind exemplarischer Natur und werden verwendet, damit dem Leser ein Einstieg in die Materie der Serviceorientierung ermöglicht wird, nicht um alle Antworten auf alle Fragen zu liefern; insofern wird auch nicht der Versuch unternommen, Serviceorientierung allumfassend zu beschreiben. Ziel des Buches ist es, dem Leser ein Verständnis für die Anforderungen und die möglichen Konsequenzen der Serviceorientierung zu vermitteln, hierzu ist aber auch ein rudimentäres Verständnis der technologischen Grundlagen nötig. Aber, wie bei allen neuen Technologien, sind die langfristigen Auswirkungen auf die Organisationen und den einzelnen Menschen viel grundlegender als man es am Anfang vermutete. Diese Aussage trifft de facto auf jede Technologie zu und führt zu dem sogenannten Hypecycle: Wir überschätzen die kurzfristigen Erfolge und Einsatzgebiete einer Technologie und unterschätzen die langfristigen Auswirkungen kleinerer Änderungen. Besonders die Kapitel 10-12 beschäftigen sich mit den langfristigen Veränderungen aus dem Blickwinkel der Systemtheorie und Kap. 4 mit den Auswirkungen von, und den Voraussetzungen, für Serviceorientierung in Organisationen.

Das in diesem Buch vorzufindende „Denglisch“ mag die Puristen der deutschen Sprache befremden, aber es passt sich dem in der IT-Welt in Deutsch-

[1] Die maskuline Form Leser, Softwareentwickler, Manager usw. wird in diesem Buch als Rollenbezeichnung benutzt; hierbei kann es sich im konkreten Fall auch stets um eine weibliche Person handeln.

land vorherrschenden Sprachgebrauch an.[2,3] Diese Veränderung der deutschen Sprache durch Reinterpretation bestehender oder die Einbeziehung neuer Wörter ist kein Defizit, sondern geradezu ein Beweis für die Vitalität der deutschen Sprache.

In letzter Zeit ist das Thema SOA als ein Ausschnitt des Problemkreises Serviceorientierung immer stärker in den Mittelpunkt des öffentlichen Interesses gerückt. Der Hauptgrund für dieses Interesse liegt nicht darin begründet, dass die Unternehmen einen unbedingten Handlungsbedarf im Umfeld von SOA haben, sondern daran, dass man sich von SOA ein Milliardengeschäft verspricht. Die Förderer der SOA-Hype sind drei Gruppen: Softwarehersteller, Consultingunternehmen und Fachzeitschriften. Diese haben zusammen ein reges Interesse daran das Thema Serviceorientierung im Markt präsent zu halten und – mit zum Teil absurden – positiven Eigenschaften zu belegen. Ziel dieses Buches ist es dem Leser die Fähigkeit zu vermitteln, selbst zu entscheiden wo Serviceorientierung Sinn macht und wo nicht.

[2] Interessanterweise haben die selbsternannten Wächter der deutschen Sprache nur mit angelsächsischen Ausdrücken Schwierigkeiten, Fachausdrücke griechischer oder lateinischer Herkunft werden sofort akzeptiert.

[3] Die meisten der Deutschpuristen benutzen bestimmt nicht solche schönen Ausdrücke wie Meuchelpuffer für Pistole, Dörrleiche für Mumie, Schweißloch für Pore oder Geistesanbau für Kultur...

2

Einleitung

Kühner als das Unbekannte zu erforschen
kann es sein, Bekanntes zu bezweifeln.

Alexander von Humboldt
1769 – 1859

Heute ist die Software zu dem dominanten Träger von Information geworden. Diese Dominanz führt zu einer immer stärkeren Durchdringung der gesamten Lebens- und Arbeitswelt mit Software und damit zu immer größeren und komplexeren Systemen. Es gibt keine Unternehmen mehr, welche keine Software einsetzen, in der Praxis gilt die Faustformel: Je größer das Unternehmen, desto größer die Softwaresysteme. Diese großen Systeme sind fast immer dadurch gekennzeichnet, dass sie:

- sehr komplex sind,
- sich nichtdeterministisch verhalten,
- eine Mischung aus menschlichen Aktionen und Software darstellen.

Folglich wird es in der Zukunft nicht mehr ausreichen, allein die Software zu betrachten. Auch andere Wissenszweige wie Psychologie, Soziologie und Systemwissenschaften müssen bei der Beurteilung und Entwicklung solcher großen Softwaresysteme zu Rate gezogen werden. Ein Mittel zur Strukturierung großer Systeme ist die Idee der Services.

Die Serviceorientierung ist im Grunde keine wirklich neue Idee. Gesellschaften haben schon sehr früh eine Blüte entwickelt, in dem sie Spezialisierung von Aktivitäten und Fertigkeiten als Dienstleistung (Service) vorangetrieben haben; daraus entstanden im Laufe der Zeit einzelne Berufszweige wie Schmied, Müller oder Arzt.

Techniker und IT-Fachleute sind stets versucht, sich auf „ihre“ Technik zurückzuziehen und dementsprechend IT-Technologie, ihren Einsatz und ihre Auswirkung nicht in einem globaleren Kontext zu betrachten. Diese Herangehensweise führt oft zu einer Art „Blindheit“, denn Technologie hat immer auch Folgen für den einzelnen Menschen und die Organisationen, in denen er agiert. Speziell die Serviceorientierung ist ohne eine entsprechende Ausrichtung der gesamten Organisation und damit implizit auch des einzelnen Mitarbeiters nicht möglich.

Die Serviceorientierung als ein zentrales Paradigma kann heute faktisch überall wiedergefunden werden. Auf der einen Seite kann die Serviceorientierung aus dem Blickwinkel der Organisation und auf der anderen Seite aus der Sicht der IT betrachtet werden.

Zu den Visionen über Services gehört auch die Vorstellung, dass es zu jedem gesuchten Service stets mehrere Provider für diesen Service geben wird. Eine andere Vision hinter der Serviceorientierung ist die Vorstellung, dass in einer Welt aus kooperierenden Services neue Applikationen mit sehr kleinem Aufwand aus bestehenden Services aufgebaut[1] werden können. Diese höhere Geschwindigkeit bei der Entwicklung einer Applikation führt zur echten Unterstützung einer sich rasch verändernden Organisation, die nun, softwaretechnisch gestützt, auf neue oder veränderte Marktanforderungen reagieren kann. Der heutige Mechanismus des einfachen Informationsaustausches im Rahmen der Applikationsintegration[2] kann auf Konzepte wie Zugang, Programmierung und Integration bestehender Services und Applikationen ausgedehnt werden, in dem die bestehenden Applikationen[3] wiederum zu Services werden. Ein wichtiges ökonomisches Argument ist, dass durch den Einsatz von Service Oriented System Engineering (s. Kap. 8) und Service Oriented Computing (s. Kap. 9) das dynamische Wachstum von Applikationsportfolios stark beschleunigt werden kann. Systeme, die z.Z. ein Silodasein als mehr oder minder getrennte Applikationen fristen, können durch das Serviceorientierungsparadigma zu völlig neuen Applikationen kombiniert und so vereint werden.

Aber nicht nur die Software wird durch das Serviceorientierungsparadigma berührt werden, auch unsere Organisationen und Organisationsformen werden sich durch den erfolgreichen Einsatz von Serviceorientierung verändern. Servicetechnologien werden durch unsere Gesellschaft geformt und formen gleichzeitig auch unsere Gesellschaft durch ihren Einsatz. Ohne die Idee der Dienstleistungsgesellschaft, welche von Soziologen schon lange formuliert wurde, wäre ein Serviceorientierungsparadigma in der Technik nie entstanden. Umgekehrt wird das Serviceorientierungsparadigma die Fähigkeit der Organisation zum Outsourcing (der Nutzung von externer Dienstleistung) auf globaler Ebene stärken.

Aus Sicht der Organisation resultiert Serviceorientierung aus dem Versuch heraus in der Lage zu sein, Dienstleistungen „outsourcen“[4] zu können. Sogar die IT selbst, ist aus diesem Blickwinkel betrachtet, ein Service, der im entsprechenden Sprachgebrauch einen „customized“ Service eines Serviceproviders darstellt. Dieser spezielle IT-Service ist für die Fachbereiche interessant, weil die komplexen Details einer Lösung vom Serviceprovider beherrscht[5] und

[1] Legoprinzip
[2] EAI, **E**nterprise **A**pplication **I**ntegration.
[3] Dies kann bei Legacysoftware zum Teil sehr aufwändig sein.
[4] An andere Organisationen auslagern zu können.
[5] hoffentlich...

vor dem Kunden verborgen[6] werden. Folglich muss eine serviceorientierte IT ihre Produkte den Kunden so einfach wie möglich zur Verfügung stellen, mit der Folge, dass diese IT sich immer stärker kundenorientiert aufstellen muss. Eine weitergehende Konsequenz aus dieser verstärkten Kundenorientierung ist, dass die nachfolgende Entwicklung zu immer einfacheren und vor allen Dingen flexibleren Softwarelösungen führen muss, um mit Hilfe der Software die Kundenzufriedenheit zu erhöhen.

Eine andere Sichtweise auf die Serviceorientierung ist die technische Blickrichtung. In den heutigen IT-Systemen spielt die Zielsetzung der möglichen Integration eine wichtige Rolle, da große Systeme aus immer kleineren Komponenten in beliebiger Art und Weise zusammengesetzt werden. Wenn diese Komponenten in Form von Services zur Verfügung gestellt werden, so wird auch hier die implementierte Komplexität vor den Anwendern versteckt. Diesem wird nur ein Interface für die Nutzung nicht jedoch die Implementierung zur Verfügung gestellt. Diese Zielsetzung ist nicht neu, schon die Idee der Objektorientierung als auch die Idee der Komponente haben dieselbe Zielsetzung, den Benutzer vor der Implementierungskomplexität durch ein öffentliches Interface zu schützen.

2.1 Services

Was ist ein Service? Ein Service ist eine Dienstleistung, welche einem Kunden zur Verfügung gestellt wird. Wie jede Form von Dienstleistung haben Services als Charakteristika ein hohes Maß an Kundenbeteiligung in ihrer Definition und Weiterentwicklung, sowie die Schwierigkeit der Standardisierung. Die Schwierigkeit hinter der Standardisierung liegt in dem Wunsch der Kunden begründet, ein hohes Maß an Individualisierung in den Services zu haben. Dieses hohe Maß an Individualisierung schafft umgekehrt Probleme für die Standardisierung, dem genauen Gegenteil einer Individualisierung.

Die Geschäftswelt hat auf ihrem Weg zur Dienstleistungsgesellschaft[7] ein gewisses Maß an Erfahrung über Services, deren Nutzung, Verwendung und Einsatz aufgebaut. Die IT-Welt und hierbei speziell die Softwarestruktur in Form von Services steht jedoch noch am Anfang einer solchen Entwicklung.[8] Services in einer Software müssen dem Anwender eine wohldefinierte Funktionalität in einem veränderbaren Kontext zu verifizierbaren Qualitätskriterien und ab initio festgelegten Preisen bieten können.

Eine interessante Eigenschaft von Services aus der Geschäftswelt wird sehr oft bei der Einführung von Services in der Software übersehen: Erfolgreiche

[6] Idealerweise...

[7] Aus soziologischer Sicht die Phase nach der Industrialisierung.

[8] Wenn man bedenkt, dass im Rahmen der Organisationsform einer Softwareentwicklung das Prinzip der Softwarefactory, im Sinne der Industrialisierung von Softwareentwicklung, von einigen Consultingunternehmen empfohlen wird, so lässt sich erkennen, wie weit die IT-Welt der Geschäftswelt hinterherhinkt.

Services werden stets aus der Sicht des Kunden definiert und nicht aus Sicht des Providers! Im Gegensatz dazu entstehen die meisten heutigen Services in der Software aus Sicht des Providers, der sein bestehendes System in Services zerlegt und diese anbietet. Diese Form der Serviceentwicklung führt zu großen Hindernissen bei der Nutzung und Akzeptanz.

2.2 SOC und SOSE

Das **S**ervice **O**riented **C**omputing (SOC) und das **S**ervice **O**riented **S**ystem **E**ngineering (SOSE) sind zwei Entwicklungsparadigmen, welche die Services als fundamentale Elemente zur Entwicklung von Applikationen und anderen Services einsetzen.

Das Paradigma der Serviceorientierung bezieht sich nicht nur auf Organisationen und die Art und Weise, wie Software produziert wird, sondern auch auf die Software selbst bzw. auf ihre Architektur. Technisch gesehen erzeugt das Service Oriented Computing keine neuen Lösungsmöglichkeiten, da dieselben Lösungen auch mit CORBA (s. Abschn. 6.3.2) oder anderen Aufrufmechanismen[9] gebaut werden könnten. Aus diesem Grund ist die Einführung eines Serviceorientierungsparadigmas auch eine viel stärker philosophisch ausgerichtete Frage nach dem Gesamtkontext und der Veränderung aller Beteiligten denn ein originär technisches Problem.

Etwas mehr Klarheit bei der Softwareproblemstellung erhält man, wenn man sich die Unterschiede zwischen der Serviceorientierung auf der einen und der Objektorientierung auf der anderen Seite betrachtet. Obwohl beide ähnliche Zielsetzungen haben, in dem sie Funktionalität hinter einem Interface verstecken und damit funktionale Kapseln sowie eine gemeinsame Messageorientierung bilden, so zeigen sich doch Unterschiede. In der Objektorientierung wird die Funktionalität durch Objekte oder Klassen gekapselt, welche diese Funktionalität als Methoden anbieten. Im Serviceorientierungsparadigma wird die Funktionalität als Prozedur eines Services geliefert. Der größte Unterschied ist jedoch die Behandlung von Zuständen. Objekte werden gebaut, um damit Zustände in ihren Attributen halten zu können. Aus diesem Grund sind fast alle Objekte in der Objektorientierung, bis auf triviale Ausnahmen, zustandsbehaftet. Im Gegensatz dazu werden Services nicht entworfen, um Zustände zu halten, es wird stets versucht, sie zustandslos zu halten. Aber in der realen Welt macht es wenig Sinn, nur zustandslose Services zu haben, denn dies würde bedeuten, dass der Zustand jedes Mal beim Aufruf eines Services vollständig mitgeliefert werden müsste. Aus diesem Grund ist es sinnvoll, dem Service Zugang zu den Zustandsinformationen zu geben. Aber dieser Zugang sollte auch den allgemeinen Prinzipien der Serviceorientierung genügen. Diese Serviceorientierung impliziert, dass die Services nach fachlichen Gesichtspunkten, relativ grobgranular, entstehen und alle voneinander unabhängig sind. Im

[9] RPC, DCOM, RMI...

Gegensatz hierzu führt die Objektorientierung durch das Prinzip, dass alles in Form von Objekten beschrieben wird, zu komplexen Netzen aus Objekten, in denen eine recht hohe Zahl von Relationen das fachliche Wissen zu einem gewissen Grad sogar externalisiert. In dem Serviceorientierungsparadigma sind die Interfaces breiter und die ausgetauschten Messages größer, was wiederum stärker der „Lebenserfahrung“ des Menschen entspricht.

In der Praxis ist jedoch oft eine Mischung aus beiden Paradigmen, Objektorientierung und SOC zu beobachten. Oft werden die Services einer SOC, nicht aus fachlichen Gesichtpunkten konstruiert, sondern die Technik ist die treibende Kraft. Es entsteht eine Art Wrappingschema, um bestehende Objekte als Services darstellen zu können. Dieser Zugang führt nicht wirklich zu einer SOA (s. Kap. 5) oder folgt der Grundidee des SOC da er generell das Prinzip der losen Koppelung eklatant verletzt.[10]

2.3 Heutiger Zustand

Die heutigen Organisationen sind durch das Phänomen der gewachsenen Architekturen[11] geprägt. Diese Architekturen sind meist nicht planvoll, sondern durch ein zunehmendes Wachstum der Applikationen auf der einen Seite und simultan durch die Einführung immer neuer Architekturformen auf der anderen Seite entstanden. Diese Abfolge der Architekturen legt sich über die Organisation als Ganzes und erzeugt ein Muster, welches einer Abfolge von erkalteten Lavaströmen ähnelt.[12] Neben den diversen Architekturen gibt es noch eine Vielzahl von Kommunikations- und Integrationsformen, die sich über die Organisation und alle ihre Partner verteilen. Integrationsformen rangieren von Diskettenaustausch, Tapetransfer über FTP, E-Mail bis hin zu CORBA und Webservices.

Im Rahmen des Serviceorientierungsparadigmas müssen alle diese Architekturen aufgebrochen und in Services überführt werden. Ein solches Unterfangen bindet auf lange Zeit erhebliche Kräfte in den Organisationen und verlangt sehr hohe Investitionen, schließlich wird die Gesamtmenge der Software, welche sich in den letzten 30 Jahren angehäuft hat, verändert. Ob dies tatsächlich sinnvoll ist, oder ob es nicht einfacher wäre, Teile der bestehenden System einzufrieren und lange mit einer Koexistenz zwischen diversen Architekturformen zu leben ist eine Frage, die sich nur im Einzelfall klären lässt, dieser Ansatz kann betriebswirtschaftlich interessant sein, minimiert er doch die notwendigen Investitionen. Aber wie bei jeder neuen Technologie verspre-

[10] Obwohl es von Vertretern dieses Zugangs oft als Implementierung einer SOA verkauft wird. Speziell im CORBA-Umfeld ist dies technisch recht einfach möglich. Eine solche Implementierung sollte man allerdings nicht SOA, sondern „CORBA through Port 80“ nennen.

[11] Accidential Architecture

[12] Lava Flow Pattern

chen auch hier die Anhänger, dass die Einführung einer Serviceorientierung alle Probleme der Vergangenheit lösen kann.[13]

2.4 SOA

Technologien und Softwareparadigmen schlagen sich stets auch in der Architektur nieder. Die aus dem Serviceorientierungsparadigma entstehende Architektur wird als **S**ervice **O**riented **A**rchitecture (SOA) bezeichnet. Eine solche SOA ist die Folge und simultan auch die notwendige Voraussetzung für die Zerlegung bestehender und den Aufbau neuer Applikationen aus Services.

Tabelle 2.1: Evolution der Architekturen

Gebiet	**1960-70**	**1980-90**	**1990+**	**1995+**	**2000+**	**2010+**
Marktimperativ	Marktanteile	Effektivität	Dezentralisierung	Kundenbindung	Real Time Enterprise	Service Oriented Enterprise
Architektur	Mainframe	Module	Client-Server	Applikationsserver	SOA	SOA
Zielvorstellung	Skalenökonomie	Business Process Reengineering	Business Applikationen	Kundenbindung	Entflechtung	Serviceökonomie
Treiber	Status Quo, keine Skalierung	sinkende CPU-Kosten	PC und Netzwerke	WWW	Webservices	semantische Services

In Veröffentlichungen wird sehr oft der Begriff SOA identisch zum Serviceorientierungsparadigma gesehen, obwohl beide etwas völlig anderes beschreiben. Diese Vermischung verwirrt viele Leser. Selbst der Begriff SOA wird nicht einheitlich genutzt und diverse Autoren definieren ihn unterschiedlich, so z.B.:

- *Arsanjani –*

  SOA is not a product – it's about bridging the gap between business and IT through a set of business-aligned IT services using a set of design principles, patterns and techniques.

[13] Die Geschichte der Softwareindustrie ist voll von diesen Versprechungen, von denen in der Vergangenheit keines eingehalten wurde:
Durch Objektorientierung wird alles billiger und schneller...
Durch CASE-Tools wird alles billiger und schneller...
Durch UML wird alles billiger und schneller...
Mit MDA wird es billiger und schneller...
Mit Java wird es einfacher...

Tabelle 2.2: Paradigmenwechsel

Zeitraum	Revolution	Computing Paradigma	Architektur
1970-80	Mainframe	Monolithisch	Single Tier
1980-90	Midrange	Abteilungsorientiert	Single Tier
1990-95	Client/Server	Power für den Desktop	2 Tier
1995-2000	Web	Portale und Backendsysteme	3 Tier
2000+	SOA	servicebasiert	servicebasiert

- *Sprott und Wilkes* –

  Service Oriented Architecture (SOA) is the policies, practices and frameworks that enable application functionality to be provided and requested as sets of services published at a granularity relevant to the service requestor, which are abstracted away from the implementation using a single, standards based form of interface.
- *Erl* –

  SOA is a form of technology architecture that adheres to the principles of service orientation. When realized through the Web services technology platform, SOA establishes the potential to support and promote these principles throughout the business process and automation domains of an enterprise.
- *Gartner Group* –

  Essentially, SOA is a software architecture that builds 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 interface, typically in request-reply mode. Service consumers are software that embeds a service interface proxy (the client representation of the interface).
- *van Zyl* –

  Service based architecture [SOA] 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* –

  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.

- *Gioldasis* –
  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.
- *plenum* –
  Eine Anwendungsarchitektur, in der die Funktionalitäten als unabhängige Services mit wohldefinierten aufrufbaren Interfaces vorliegen, so dass sie – in einer sinnvollen Reihenfolge aufgerufen – einen Geschäftsprozess abdecken.

Diese verschiedenen Definitionen einer SOA zeigen die unterschiedlich gesetzten Schwerpunkte des Serviceorientierungsparadigmas auf. Je nach dem zu untersuchendem Aspekt ist es besser, folgende Begriffe zu unterscheiden:

- **S**ervice **O**riented **C**omputing (SOC) (s. Kap. 9) – Der Bau von Services.
- **S**ervice **O**riented **A**rchitecture (SOA) (s. Kap. 5) – Die aus der Nutzung von Services resultierende Architektur.
- **S**ervice **O**riented **P**latform (SOP) (s. Kap. 6) – Die Infrastruktur für den Einsatz und die Entwicklung von Services.
- **S**ervice **O**riented **S**ystem **E**ngineering (SOSE) (s. Kap. 8) – Die Vorgehensweisen zum Bau und Einsatz von Services.
- **S**ervice **O**riented **E**nterprise (SOE) (s. Kap. 4) – Die Struktur der servicenutzenden und -erzeugenden Organisation.

Die Modewelle SOA und Serviceorientierung wird jedoch nicht nur auf technischer, sondern auch auf persönlicher Ebene Auswirkungen auf Einzelne haben:

> *More CIO's*[14] *will lose their job over SOA implementations than lost their job over ERP*[15] *implementations.*
>
> Jeff Schneider

Die Versprechungen von Unternehmen in Bezug auf SOA und Serviceorientierung sind immens, besonders gefördert wird dies durch die implizite Allianz von Werkzeugherstellern und Consultingunternehmen; beide haben ein großes Interesse daran, an dem SOA-Hype zu partizipieren. Wie auch bei vergangenen „IT-Hypes“ üblich wird von den Herstellern und Consultingunternehmen

[14] **C**hief **I**nformation **O**fficer. Manche sind der Ansicht, es wäre die Abkürzung für ***C**areer **i**s **o**ver...*
[15] **E**nterprise **R**esource **P**laning

stets versprochen, dass mit einer SOA[16] alles besser wird. Es wird eine Revolution mit immensen Einsparungen prognostiziert. Die Liste der Versprechungen liest sich wie ein Wunschzettel jeder Organisation, die massiv IT einsetzt:

- *SOA wird es ermöglichen, schnell und einfach auf Veränderungen zu reagieren, nicht nur funktional, sondern auch geographisch und in Bezug auf die gewählte Plattform oder den Lieferant.* – Funktionale Veränderungen werden über fachliche Anforderungen spezifiziert und führen zu einer Evolution der Software. Die Zeiten sind hier meist durch die Domäne und nicht durch die Software bestimmt. Eventuelle Veränderungen in der Geographie sind immer Veränderungen in der Organisation. Organisatorischer Wandel hat seine eigenen Gesetze und Geschwindigkeiten, die in aller Regel länger dauern als die Software. Der Know-how-Aufbau für eine neue Plattform dauert relativ lange, da Lernkurven in neuen Technologien sehr flach sind. Außerdem verfolgen alle Lieferanten als Ziel ihrer Kundenbeziehungen niedrige Einstiegs- und hohe Ausstiegshürden damit der Kunde nicht wechselt.
- *Einfache Integration mit internen und externen Partnern.* – Die Hauptaufwände bei der Integration liegen nicht im Bereich der technischen, sondern bei der semantischen Integration.
- *Wiederverwendung wird vereinfacht.* – Dieses Versprechen wurde schon bei der Einführung der objektorientierten Sprachen strapaziert und trat nicht ein. Wiederverwendung muss antizipiert, d.h. aktiv angegangen werden. Reaktive Versuche zur Wiederverwendung sind mehr eine Art „Softwaresalvaging“.
- *Unterstützung für kurze Produktlebenszyklen.* – Für den Fall des massiven Einsetzens von Wiederverwendung ist dies theoretisch möglich. Praktische Erfahrungen zeigen jedoch, dass Produktzyklen stärker vom Markt, dem Marketing und einer organisatorischen Fähigkeit zum Wandel beeinflusst werden als durch Software.
- *Verbesserung des Return on Investment. Durch den Einsatz von austauschbaren Webservices soll sich der ROI verbessern.* – Die Kosten für einen ESB (s. Abschn. 6.3) sind allerdings immens.
- *Geschäftsprozesse werden direkt in die IT abgebildet.* – Das Alignment ist etwas komplexer als gemeinhin angenommen wird.[17]
- *Reduktion von Entwicklungskosten.* – Da es heute keine expliziten Modelle für eine Serviceentwicklung gibt, scheint es sich hierbei um Wunschdenken zu handeln.
- *Die Informationsarchitektur wird transparent.* – Im Gegenteil, die zunehmende Verknüpfung der Services führt zu einem komplexen Gesamtsystem.

[16] Oft mutiert dabei die eigentliche Architektur (SOA) zum Überbegriff für Serviceorientierung.

[17] *Masak, D.:* 2006, IT-Alignment, Springer.

- *Die Benutzung von Standards sichert Interoperabilität.* – Die technische Interoperabilität ist ein relativ kleines Problem. Interoperabilität zeigt sich heute primär auf der semantischen Ebene (s. Abschn. 9.19).
- *Durch die Einführung von SOA und Services wird sich die Datenqualität erhöhen.* – Ein Trugschluss; Datenqualität ist eine Folge von hoher Qualität bei der Entstehung von Daten. Erst wenn der Erzeuger von Daten einen echten Anreiz hat, hohe Qualität zu liefern, entsteht Datenqualität.
- *Die IT-Governance wird einfacher.* – Durch die zunehmende Komplexität wird dies ad absurdum geführt. Außerdem können Ultra Large Scale Systeme entstehen, die sich der Governance verweigern (s. Kap. 10).
- *Services, speziell Webservices*[18] *benötigen keine Programmierung.* – Diese Verheißung wird von Werkzeugherstellern genutzt, um ihre Generatoren zu verkaufen, die automatisch generierten Services führen fast immer zu einem chaotischen System.

Diese ganzen Versprechungen sind nicht zu halten, viel sinnvoller ist es, sich mit den Möglichkeiten, Chancen und Risiken des Serviceorientierungsparadigmas auseinander zu setzen und dann selbst zu urteilen. Der Trend zur Serviceorientierung ist nicht aufzuhalten und wird in den nächsten Jahren unsere Umgebung bestimmen. In gewisser Weise ist dies auch die Folge der zunehmenden Informationsmenge. Schon in den siebziger Jahren wurde die Folge der Informationszunahme vorhergesagt:

> *What information consumes is rather obvious: it consumes attention of its recipients. Hence a wealth of information creates a poverty of attention and a need to allocate that attention efficiently among the overabundance of information sources that might consume it.*[19]

Daher muss die zunehmende Informationsmenge durch immer weniger Aufmerksamkeit bearbeitet werden, mit der Folge, dass Information zunehmend automatisiert verarbeitet wird. Einer der Gründe für die öffentliche Aufmerksamkeit für Serviceorientierung und speziell für SOA ist die Tatsache, dass die Serviceorientierung eine hervorragende Metapher für nicht technisch orientierte Menschen ist. Diese können nun auch den Wert einer Architektur verstehen und den Herausforderungen der Veränderung und Anpassung in Organisationen und Technik begegnen. Der wirkliche Mehrwert einer SOA ist, dass es die erste Architektur ist, die Software und Organisation transzendiert. Langfristig gesehen muss die Serviceorientierung sowohl Auswirkungen auf die Betriebssysteme als auch die zur Verfügung stehende Hardware haben. Heutige Ansätze eines Hardware Abstraction Layers[20] bauen auf der Idee einer generischen Hardware auf, es ist z.Z. noch unklar, wie eine serviceorientierte Hardware oder ein serviceorientiertes Betriebssystem konzipiert sein könnte.

---

[18] s. Abschn. 5.4

[19] *Simon, H.*: 1971, Designing Organizations for an Information-rich World, The John Hopkins Press.

[20] So z.B. in Windows und Linux.

3

Serviceorientierungsparadigma

I have travelled the length
and breadth of this country
and talked with the best people
and I can assure you
that data processing is a fad
that won't last out the year.

Editor Business-Books
Prentice-Hall
1957

Der Begriff „Service" wird in vielfältiger Weise im IT-Bereich verwendet. IBM z.B. bietet seinen Kunden einen „Service on Demand". Webservices wurden in letzter Zeit zumindest in der Presse sehr bekannt. Provider, welche Applikationen über das Netz zur Verfügung stellen, werden auch als Application Service Provider bezeichnet (ASP). Es gibt also in jeder Organisation eine ganze Reihe verschiedener Bereiche, in denen Services heute vorkommen. Der Erfolg des **W**orld **W**ide **W**eb (WWW) hängt unter anderem davon ab, dass es serviceorientiert aufgebaut ist. Wenn man es grob vereinfacht, so existiert im WWW hauptsächlich der Service „Seite abrufen". Dieser Service kann von einer großen Nutzergruppe verwendet werden, er wird durch den Aufruf der entsprechenden Internetseite angefordert. Es spielt keine Rolle, welchen Browser oder welches Betriebssystem der Nutzer verwendet[1], um eine Seite abzurufen. Ebenso wenig ist die verwendete Webserversoftware oder das verwendete Betriebssystem des Webservers von Bedeutung.

Ein Service in der Software zu definieren ist nicht ganz einfach, obwohl es einem intuitiv klar erscheint, daher betrachten wir zunächst einmal, was ein Service nicht ist. Ein Service ist nicht das selbe wie eine Komponente, wie man sie aus der komponentenbasierten Softwareentwicklung her kennt, wenngleich viele Konzepte auch auf die serviceorientierte Softwareentwicklung übertragen werden können. Eine andere Vorstellung ist, dass ein Service

[1] Zumindest in der Theorie. Es hat immer wieder Versuche von Microsoft gegeben ihr eigenes Webserverprodukt IIS nur durch den Internet Explorer „vernünftig" darstellbar zu machen oder durch die Einbettung von ActiveX-Elementen diese Seiten nur aktiv nutzbar zu machen, wenn der Nutzer den Internet Explorer von Microsoft einsetzt.

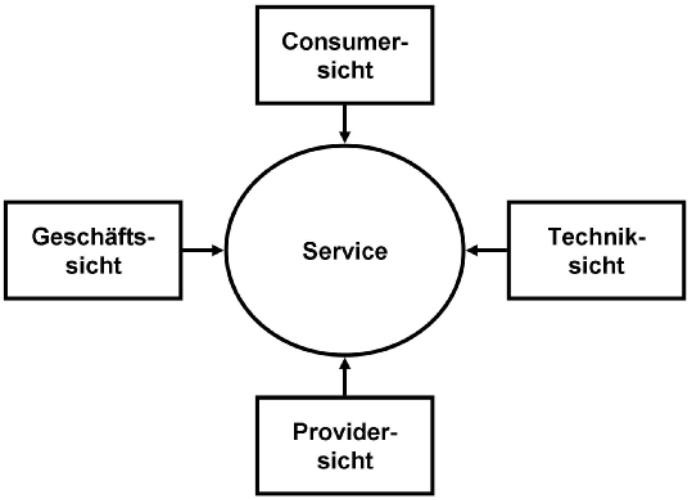

Abb. 3.1: Die unterschiedlichen Sichten auf einen Service

von einer Komponente implementiert wird und der Service damit nur das Interface der Komponente darstellt. Diese Betrachtungsweise ist etwas zu eng. Ein Service beinhaltet zwar auch ein Interface, zusätzlich dazu existiert aber ein Vertrag, welcher Eigenschaften enthält, welche die Semantik betreffen und die nicht über ein Interface definiert werden können. Dazu gehören Qualitätseigenschaften wie Verfügbarkeit oder Performance, die auch im Sinne eines juristischen Vertrages vereinbart werden können. Dafür reicht dann die Beschreibung eines Interface nicht mehr aus, sondern es bedarf eines komplexen **S**ervice **L**evel **A**greements (SLA), wie man es aus der **IT**-**I**nfrastructure-**L**ibrary (ITIL) kennt, in dem Rechte und Pflichten von Serviceprovider und Serviceconsumer klar geregelt sind.

Die Nutzung unterschiedlicher Perspektiven ist wichtig für das Verständnis eines Services. Hier gibt es zwei wesentliche Sichtweisen:

- Geschäftssicht – Die Geschäftssicht betrachtet einen Service als Teil einer Geschäftstransaktion, die in einem Vertrag beschrieben und die durch die Geschäftsinfrastruktur durchgeführt wird. Was ein Service leistet, ist eng verknüpft mit den Erfahrungen des Geschäftsbereichs. Einen Service auf dieser Seite bezeichnet man mit dem Begriff Geschäftsservice (genau wie eine Dienstleistung). Typische Eigenschaften eines solchen Geschäftsservices sind:
  - Geschäftsvisibilität – Ein Service muss etwas sein, wofür Kunden bereit sind, etwas zu bezahlen. Die Kunden ihrerseits erhalten etwas, das für sie von Wert oder von Nutzen ist. In diesem Zusammenhang muss der Begriff „Kunde“ weiter gefasst werden als im Bereich des Produktkaufs.

Als Kunden werden nicht nur externe, sondern auch interne Consumer des Services bezeichnet.
  - Technologie – Hierbei steht das, was implementiert werden soll im Fokus, nicht wie etwas implementiert werden soll.
  - Kontext – Die Services werden durch ihren jeweiligen fachlichen Kontext definiert.
- Techniksicht – Ein Service stellt eine mehr oder minder abgeschlossene Funktionalität bereit, wobei die Semantik eines Services in Form eines Interfaces beschrieben wird. Eigenschaften technischer Services sind:
  - Technologie – Bei technischen Services steht die Technologie, also wie etwas implementiert werden soll, im Vordergrund.
  - Kontext – Ein Service ist eine vom Kontext gekapselte und abstrahierte Funktionalität.

Es existieren einige Unterschiede zwischen den Sichten und damit auch zwischen den daraus abgeleiteten unterschiedlichen Servicedefinitionen. Der wesentliche Unterschied beider Sichten liegt in der Betrachtung des Kontexts begründet. Für Geschäftsservices ist der Kontext von großer Wichtigkeit; alle Organisationen arbeiten schließlich in einem dynamischen Marktumfeld. Werden Marktchancen ergriffen, so wird von dieser Seite erwartet, dass neue Services erstellt oder bestehende Services verändert werden. Die technische Seite ist bestrebt, den Kontext möglichst statisch zu halten, da nur so die Forderung nach Effizienz und Wiederverwendbarkeit erreicht werden kann. Diese Differenzen lassen sich nicht wirklich überbrücken. Die negativen Effekte lassen sich aber abmildern, indem kommuniziert wird, welche Bedeutung ein Service etwa auf organisatorischer Ebene hat oder wie wichtig Wiederverwendung für einen implementierten Service ist.

Zusätzlich zu den beiden Perspektiven Geschäfts- und Techniksicht existieren auch eine Consumer- und eine Providerperspektive. Die Implementierung ist Teil der Providersicht und muss für den Consumer von geringem Interesse sein. Die Providersicht überschneidet sich mit der Geschäftssicht[2] und der technischen Sicht[3]. Die Consumersicht ist mehr auf die Geschäftssicht fokussiert. Wenn ein Consumer einen Service in Anspruch nimmt, muss für ihn der Wert des angebotenen Services schon vorab erkennbar sein. Ist dies der Fall, stellt der Service eine Schnittstelle zwischen Geschäftswert und Implementierung dar.

[2] Was wird implementiert und warum?
[3] Wie wird es implementiert?

3.1 Paradigma

Die Verwendung von Services in Organisationen und in der Software folgen einem gemeinsamen Grundsatz, dem „Paradigma[4] der Serviceorientierung“:[5]

> *Alle Funktionen in einem realen System, seien es Abläufe in Organisationen, Prozesse, Aktivitäten, Funktionen in Softwaresystemen, Applikationen, Teile von Applikationen oder Softwarefunktionen lassen sich als Services darstellen und aus Services aufbauen!*

Oder kürzer formuliert:

> *Alles, was aus- oder durchgeführt werden kann ist ein Service!*

Hierbei hat jeder Service mindestens einen Provider (den Lieferanten) und einen Consumer (den Kunden oder Nutzer). Außerdem ist jeder Service in seiner Funktionalität und seinen Randbedingungen vorab definiert und diese Bedingungen sind sowohl dem Provider als auch dem Consumer bekannt.

3.2 Service

Der Ursprung der Idee des Services liegt in den Dienstleistungen, dies hat zur Folge, dass es Unterschiede zwischen Produkten und Services gibt. Unabhängig von einer exakten Definition des Begriffs Service teilen sich in der Praxis alle Services eine Reihe von Eigenschaften:

- Services stellen Fähigkeiten oder Funktionen zur Verfügung.
- Services sind sofort nutzbar.
- Services haben ein wohldefiniertes Verhalten.
- Services haben definierte Ein- und Ausgaben.
- Services werden „gemanaged“, um nichtfunktionale Ziele zu erfüllen.
- Services werden aufgebaut und eingesetzt, damit ein organisatorisches Ziel erreicht werden kann.
- Services sind modellierbar.
- Services sind zusammenbaubar[6], um damit neue Services zu erzeugen.

Ein Service unterscheidet sich von klassischen Produkten[7] durch folgende Merkmale:

---

[4] Paradigma aus dem Griechischen $\pi\alpha\rho\alpha\delta\varepsilon\iota\gamma\mu\alpha$ aus $\pi\alpha\rho\alpha$ (neben) und $\delta\varepsilon\iota\gamma\nu\nu\alpha\iota$ (zeigen). Spötter behaupten ein Paradigma sei die Summe aller Vorurteile, die über etwas existiere.

[5] s. auch Servicedefinition, S. 18.

[6] Servicekomposition, s. Abschn. 9.18.

[7] Die Debatte über den Unterschied zwischen Produkten und Services (Dienstleistungen) geht auf Adam Smiths *The Wealth of Nations* zurück, welcher als erster einen Unterschied postulierte.

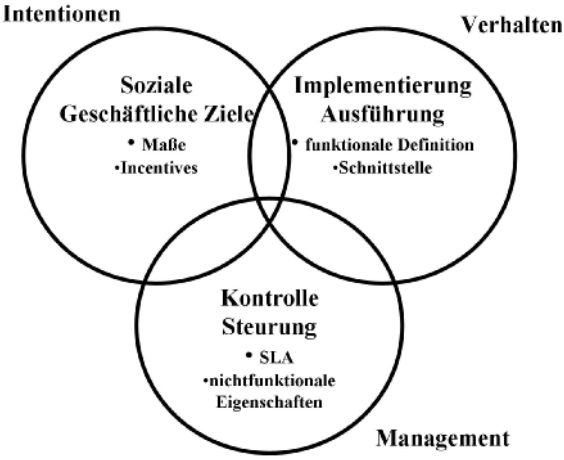

Abb. 3.2: Unterschiedliche Sichten auf Services

- Services sind nicht direkt greifbar[8]. Folglich können sie weder inventarisiert noch patentiert werden und ihr Wert ist nur schwer quantifizierbar, wobei die Nichtgreifbarkeit sich in zwei Dimensionen vollzieht. Zum einen sind Services physisch ungreifbar, d.h. sie können nicht berührt oder betastet werden, zum anderen sind sie mental ungreifbar, d.h. sie können nicht mental als ein Gegenstand[9] verstanden werden.
- Services sind heterogener und vielgestaltiger als Produkte, da sie oft von Menschen direkt erzeugt werden. So sind Kategorien wie Wiederholbarkeit und Vorhersagbarkeit oft nur schwer anwendbar.
- Services werden fast gleichzeitig produziert und verbraucht, im Gegensatz zu Produkten, welche meistens eine Lagerhaltung haben. Zentralisierung und Massenproduktion sind für Services problematisch. Dieses Kriterium ist nicht besonders trennscharf, da es viele Services gibt, die separat vom Consumer durchgeführt werden.[10]
- Services sind „verderblich“[11], nicht so Produkte, d.h. Services können nicht gespeichert werden, aber aus Sicht des Consumers kann das Ergebnis des Services durchaus unverderblich sein.
- Die Qualität eines Services hängt von vielen sehr schwer kontrollierbaren Faktoren ab. Da ein Schwerpunkt der Services im direkten Kundenkontakt

[8] intangible
[9] Ein Konto auf der Bank oder ein Lebensversicherungsvertrag sind zwar nicht physisch greifbar, stellen jedoch mental Gegenstände dar und sind somit Produkte.
[10] Paketdienste oder Reinigungen führen einen Großteil ihrer Tätigkeiten vom Consumer separat durch.
[11] perishable

besteht, hat dies hat zur Konsequenz, dass die Qualität des Services von der Fähigkeit des Kunden determiniert wird, sich zu artikulieren bzw. der Provider in der Lage ist dem Kunden zuzuhören.
- On-demand-Delivery – Die Services werden meistens direkt durch die Nachfrage nach ihnen ausgelöst.
- Im Gegensatz zu Produkten ist die Preisbildung bei Services oft schwer.

Damit diesen unterschiedlichen Charakteristika Rechnung getragen werden kann, muss eine umfassende Servicedefinition sehr abstrakt sein, die in diesem Buch verwandte Definition von Service ist daher:

> *Ein Service ist die Summe des beobachtbaren Verhaltens eines Systems (genannt Serviceprovider), gegeben durch die Menge aller möglichen Interaktionen und deren Relationen zwischen dem System und seiner Umgebung.*

Diese Definition ermöglicht es, Services als systemtheoretische Gebilde (s. Kap. 11) zu fassen und dem Serviceparadigma folgend alles als Service aufzufassen. Dieses Servicekonzept ist das Resultat der Trennung zwischen dem internen (Implementation) und externen (Interface) Verhalten eines Systems. Für den Consumer (als Teil der Umgebung) ist nur der externe Teil interessant. Implizit lässt sich aus dieser Definition eine Reihe von Eigenschaften für die Services ableiten:

- Ein Service hat einen Grad an Autonomie, da ohne Autonomie ein System überhaupt nicht identifizierbar ist.
- Services besitzen ein Interface (eine Schnittstelle), dieses wird durch die Grenze zwischen System und Umgebung gebildet. Da der Consumer ein Teil der Umgebung ist, bildet seine Umgebungsteilmenge das für ihn wahrnehmbare Interface. Der andere Teil der Grenze zur Umgebung bildet für den Service den Kontext.
- Da Systeme zu größeren Systemen zusammengesetzt werden können, gibt es auch die Möglichkeit zur Servicekomposition.
- Die funktionalen Eigenschaften sind die erwarteten Verhaltensmuster des Systems und ergeben sich aus dem Interface.
- Die nichtfunktionalen Eigenschaften sind die durch den Kontext (ohne den Consumer) ausgelösten Verhaltensmuster des Systems.

3.2.1 Funktionale Eigenschaften

Ein Service muss hinreichend gut beschrieben werden. Speziell dann, wenn der Consumer den Provider nicht kennt, ist eine semantisch reichhaltige und strukturell gute Beschreibung notwendig, damit ein Service auch entdeckt und nachfolgend eingesetzt werden kann.

Die funktionalen Eigenschaften eines Services beschreiben die fachlich gewünschten Funktionen, die der Service erfüllen soll; das, was er tatsächlich

für seinen Kunden durchführt.[12] Die funktionalen Eigenschaften beschreiben die Wirkung, nicht die Implementierung des Services, folglich wird hier die Grenze zur Umgebung und nicht die Substruktur des Systems beschrieben. Neben der Festlegung von dem, was durchgeführt werden soll, wird auch der Informationsfluss für den Service beschrieben. Services, die durch Menschen durchgeführt werden, wie z.B. Reinigung, Outsourcing oder Reparaturen, werden im Allgemeinen auch als manuelle Services oder Dienstleistungen bezeichnet. Für die manuellen Services sind wir in den meisten Fällen gewohnt, aus kultureller Erfahrung die Funktion implizit zu definieren. Oft werden Service und Berufsbezeichnung[13] austauschbar genutzt, und andere Services und deren Berufsbezeichnungen sind schon fast vollständig verschwunden[14]. Bei diesen Services herrscht eine kulturelle Übereinkunft von dem, was auszuführen und zu liefern ist. Eventuelle Details über die funktionale Definition dieser Services aus juristischer Sicht erfahren wir erst im Konfliktfall mit dem jeweiligen Serviceprovider (Handwerker) oder beim Lesen der allgemeinen Geschäftsbedingungen.

Bei manuellen Services gestaltet sich der Informationsfluss in Form eines Dialogs zwischen dem Kunden (Serviceconsumer) und dem „Handwerker“ (Serviceprovider). Die starke menschliche Verflechtung und das iterative Verhalten aller Beteiligten ermöglicht es, dass sich diese Services auf diverse Kontexte einstellen können.[15]

Im Fall von Software werden die funktionalen Eigenschaften von Services durch Interfaces (Schnittstellen), sowie den Vor- und Nachbedingungen spezifiziert. Services in der Software erzeugen ein eindeutiges syntaktisches Verhalten auf Grund der Tatsache, dass die genutzten Programmiersprachen Anforderungen an den Typ der Information in Form von Datentypen stellen, im Gegensatz zu manuellen Services.

3.2.2 Nichtfunktionale Eigenschaften

Eine Servicebeschreibung besteht neben der Funktionalität des Services auch aus der Beschreibung der nichtfunktionalen Eigenschaften. Speziell das Fehlen von Beschreibungen nichtfunktionaler Eigenschaften verhindert eine „vernünftige“ Suche und Entdeckung von Services. Sinnvolle nichtfunktionale Eigenschaften für einen Service sind:

- Name des Providers – Provider müssen eine eindeutige und verifizierbare[16] Identität haben. Die logische Folge einer Identität ist die Existenz eines eindeutigen Namens und die Zugehörigkeit zu einer Organisation.

[12] Allerdings nur aus Sicht des Kunden.
[13] Heizungsbauer, Maurer...
[14] Kesselflicken, Hochzeitsladen, Dengeln...
[15] Viable System Services sind der Versuch, diese Flexibilität auch auf Software zu übertragen, s. Abschn. 12.1.
[16] Die Verifikation kann auch über Dritte, so z.B. Trustcenter, erfolgen.

- Zeit – Angaben über die Servicezeiten oder auch Wochentage sind wichtige Größen für den Consumer.[17]
- Ort – Obwohl es in den meisten Fällen transparent sein sollte, wo der Service tatsächlich ausgeführt wird, kann der Ausführungsort bezüglich rechtlicher Belange oder Sicherheitsaspekte durchaus relevant sein. Bei nicht-IT-basierten Services kann es durchaus sein, dass der Service nur an einem bestimmten Ort durchgeführt werden kann oder darf.
  Neben den klassischen Formen der Ortsangabe können Telephonnummern oder URI-Adressen bei SLA-Verletzungen Kontakte und Möglichkeiten zum Ausweichen anbieten.
- Verfügbarkeit – Unter Verfügbarkeit versteht man die Kombination aus den zeitlichen und örtlichen Aspekten der Services.
- Obligation – Stellt die Verpflichtungen von Provider und Consumer dar.
- Preis – Neben dem Preis pro Serviceaufruf sind auch andere Formen von Preismodellen möglich:
  - Flat Rate,
  - Bulk Rate,
  - Prime Rate.
- Zahlungsmodalitäten – Analog zum Preis der Services kann auch die Art und Weise der Zahlung relevant sein.
- Strafen – Werden Zahlungsmodalitäten oder Obligationen verletzt, so treten die entsprechenden Strafen ein.
- Sprache.
- Qualität des Services (s. Abschn. 5.6).
- Sicherheit.

Für die Serviceconsumer sind die funktionalen und nichtfunktionalen Eigenschaften von Services zum Teil nicht voneinander unterscheidbar, insofern ist eine solche Unterteilung bis zu einem gewissen Grad willkürlich. Je bekannter und standardisierter die funktionalen Eigenschaften von Services sind, desto stärker rücken die nichtfunktionalen Eigenschaften bei der Entscheidung des Kunden für einen spezifischen Serviceprovider in den Vordergrund.

3.3 Enterprise Architekturen

Was bedeutet es, eine Serviceorientierung zu haben? Oder ist eine Service Oriented Architecture die Antwort auf alle Fragen? Um die Auswirkungen des Serviceorientierungsparadigmas auf die Architektur einer Organisation und einer Software beurteilen zu können sollte man versuchen, die Serviceorientierung mit Hilfe von „klassischen“ Architekturframeworks zu beschreiben.

Die Architektur ist die abstrakte Struktur einer Aktivität. Die hier genutzte Definition des Begriffs Architektur ist:

[17] Der Service Personentransport der Deutschen Bahn kennt Wochen- und Feiertagsfahrpläne.

> *Eine Architektur ist eine formale Beschreibung eines Systems, ein detaillierter Plan des Systems und seiner Komponenten, die Struktur der Komponenten, ihre Wechselwirkungen, ihre Prinzipien und Richtlinien, die ihren Entwurf, ihre Entwicklung und Implementierung steuern.*

Innerhalb von Organisationen können durchaus mehrere unterschiedliche Architekturen parallel zueinander existieren. Die diversen Sichten auf das Gesamtsystem Organisation, Prozesse und Software werden in ihrer Gesamtheit als Enterprise Architektur bezeichnet. Eine solche Enterprise Architektur überspannt auch immer mehrere technische Systeme. Eines der Ziele hinter einer Enterprise Architektur ist es, ein möglichst hohes Maß an Alignment zwischen den fachlichen Prozessen und Strukturen auf der einen Seite und der IT auf der anderen Seite zu erreichen. Eine Enterprise Architektur besteht in ihrer Gesamtheit aus vier separaten Teilbereichen:

- Geschäftsprozessarchitektur,
- Applikationsarchitektur,
- Informationsarchitektur,
- Technologiearchitektur.

Die vier Architekturkategorien werden stets gemeinsam betrachtet, da eine explizite Separation für eine übergreifende Betrachtung nicht besonders sinnvoll erscheint. Die Abhängigkeiten und Wechselwirkungen dieser verschiedenen Teile sind viel zu groß. Der Einsatz einer Enterprise Architektur unterstützt jede Organisation, welche auf Informationstechnologie angewiesen ist, in hohem Maße. Innerhalb einer Enterprise Architektur gibt es keine effektiven Grenzen bezüglich der Fähigkeit zum Informationsaustausch zwischen verschiedenen Organisationsteilen.

Nicht nur die reine Geschäftsprozessarchitektur hat Auswirkungen auf die Applikationsarchitektur, die Architektur in Organisationen ist auch immer ein soziales Phänomen, denn Konflikte in der Architektur repräsentieren oft Konflikte zwischen unterschiedlichen sozialen Gruppen. Der Konflikt ist aber nicht die Frage der reinen Zusammenarbeit auf Bitebene, sondern der Konflikt resultiert aus der unterschiedlichen Semantik, welche die Beteiligten anwenden. Die entstandenen Architekturen spiegeln die Abhängigkeiten und Konflikte der verschiedenen Beteiligten wider. Umgekehrt kann Software aber auch Kommunikationskanäle aufbauen, die vorher nicht vorhanden waren, mit der Folge, dass es zu einer wechselseitigen Beeinflussung kommt.

3.4 Zachman-Framework

Das Konzept von Enterprise Architekturen geht zurück auf die achtziger Jahre des letzten Jahrhunderts. Einer der führenden Köpfe der Architekturbewegung, John Zachman, erkannte den Wert der Nutzung einer abstrakten Architektur für die Integration von Systemen und ihrer Komponenten. Zachman