Roger Zacharias

Management- und Web Services-Architekturen: Konzeption und Realisierung eines Überwachungssystems für Bankperipheriegeräte

Diplomarbeit

GRIN

Bibliografische Information der Deutschen Nationalbibliothek:

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

ISBN: 9783656982517

Roger Zacharias

Management- und Web Services-Architekturen: Konzeption und Realisierung eines Überwachungssystems für Bankperipheriegeräte

Examicus Verlag

Examicus - Verlag für akademische Texte

Der Examicus Verlag mit Sitz in München hat sich auf die Veröffentlichung akademischer Texte spezialisiert.

Die Verlagswebseite www.examicus.de ist für Studenten, Hochschullehrer und andere Akademiker die ideale Plattform, ihre Fachtexte, Studienarbeiten, Abschlussarbeiten oder Dissertationen einem breiten Publikum zu präsentieren.

FACHHOCHSCHULE
GIESSEN
FRIEDBERG

Diplomarbeit

zum Thema

Management- und Web Services- Architekturen

Konzeption und Realisierung eines Überwachungssystems für Bankperipheriegeräte

Zur Erlangung des akademischen Grades
Diplom-Informatiker (FH)

vorgelegt am
Fachbereich Mathematik, Naturwissenschaften und Informatik der Fachhochschule Gießen-Friedberg

von
Roger Zacharias
Im Februar 2002

Referent: **Prof. Dr. A. H. Kaufmann**
Koreferent: **Prof. Dr. M. Scheer**

II

Vorwort

Technology of a sufficiently advanced level
will be indistinguishable from magic. (Arthur C. Clarke)

Die vorliegende Diplomarbeit entstand im Zeitraum 10/2001 – 02/2002 an der Fachhochschule Gießen-Friedberg in Zusammenarbeit mit der Wincor Nixdorf GmbH & Co. KG unter der Betreuung von Herrn Prof. Dr. Kaufmann und Herrn Prof. Dr. Scheer. Die Aufgabenstellung ergab sich aus einem Projekt der Wincor Nixdorf GmbH & Co. KG, in welchem eine umfangreiche Überwachungslösung im Bankenbereich entwickelt wird. Dazu sollen die zurzeit neu aufkommenden Architekturen und Paradigmen untersucht und ein möglicher Prototyp erstellt werden und bei entsprechender Eignung gegebenenfalls in diese Überwachungslösung mit einfließen.

Die besondere Herausforderung dieser Diplomarbeit lag in der Aktualität der beschriebenen und verwendeten Architekturen und Technologien, welche einen Mangel an fundierter Literatur und stabilen Laufzeitumgebungen sowie einen stetigen Weiterentwicklungs- und Änderungsprozess implizierten. Teilweise musste, um die volle Funktionalität der Technologien auszunutzen, auf Alpha-Versionen zurückgegriffen werden, was die Arbeit nicht immer erleichterte, aber natürlich sehr interessant machte.

Zur Konzeption und Realisierung des Prototyps bediente sich der Autor einer Kombination aus dem ´Rational Unified Process´ und ausgewählten Elementen des ´Extreme Programming´. Diese Vorgehensweise kann, nach Meinung des Verfassers, für Projekte in diesem Rahmen sehr empfohlen werden, da sie sich insbesondere durch schnelle Reaktionsmöglichkeiten bei ständig wechselnden Anforderungen auszeichnet.

DANKSAGUNGEN

An dieser Stelle möchte ich mich bei meinen Betreuern für die geleistete Unterstützung bedanken, insbesondere bei Herrn Prof. Dr. Kaufmann, der in einigen Gesprächen und vielen E-Mails durch zahlreiche Anregungen zum Gelingen dieser Arbeit beigetragen hat.

Mein Dank geht auch an meine Kollegen, die durch anregende Diskussionen und konstruktive Kritik sowie aufmunternde Worte einen wichtigen Teil beitrugen, insbesondere an die, die Angst hatten an dieser Stelle vergessen zu werden, Herrn Dipl.-Ing. Peter Diel und Herrn Dipl.-Math. Gerhard Schneider.

Für ihre Unterstützung möchte ich mich bei meinen Eltern und Schwiegereltern bedanken.

Weiterhin bedanke ich mich bei meinen Kommilitonen und Freunden, die mir durch ihr großes Interesse an dieser Arbeit, durch Fragen und Diskussionen geholfen haben, dieses Gebiet in seiner Gesamtheit zu begreifen und aus verschiedenen Blickpunkten zu betrachten.

Für zahlreiche Tipps und Tricks bezüglich Design und Layout bedanke ich mich bei Sylvia Brandt.

Mein besonderer Dank geht an meine Verlobte Alexandra Georg für Ihr liebevolles Verständnis, ihre Unterstützung und ihre umfangreiche Hilfe beim Redigieren dieser Arbeit.

Darüber hinaus danke ich allen weiteren Personen, die auf irgendeine Art zum Gelingen dieser Arbeit beigetragen haben.

Februar 2002 Roger Zacharias

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Gleichzeitig versichere ich, diese Arbeit in gleicher oder ähnlicher Form weder veröffentlicht, noch einer anderen Prüfungsbehörde vorgelegt zu haben.

Dillenburg, den 15.02.2002

Inhaltsverzeichnis

KAPITEL **1**

Einführung

One does not discover new lands
without consenting to lose sight
of the shore for a very long time.
(Andrew Gide)

1 Einführung

1.1 Einleitung

Nichts ist beständiger als der Wandel! Diese Erkenntnis trifft momentan genau den Kern. In allen Bereichen der IT werden bestehende Architekturen, Produkte und Prozesse hinterfragt. Einhergehend mit dieser Entwicklung ist das Bestreben nach Standardisierung, Offenheit und die Nutzung bereits vorhandener und bewährter Implementierungen zu beobachten, ohne aber bereits getätigte Investitionen zu vernachlässigen. Im Bereich der Softwareentwicklungsprozesse konkurrieren RUP[1] und XP[2] (oder werden sinnvoll gemeinsam eingesetzt), UML[3] hat sich als Standard durchgesetzt (ist aber mittlerweile für viele schon zu umfangreich geworden), XML[4] ist kein ´Wunderding´ mehr, Begriffe wie ´Web Services´[5], ´SOAP´[6] und ´EAI´[7] kursieren (und werden überall anders definiert), mächtige Frameworks wie ´J2EE´[8] entstehen, worauf Microsoft mit ´.NET´[9] antwortet und SAP liegt mit ´mySAP.com´[10] voll im Trend. Das Internet ist unersetzbarer Bestandteil unseres Lebens geworden, der E-Commerce kommt langsam in Fahrt und es wächst der Wunsch der Unternehmen, Prozesse über das Internet mit anderen Unternehmen zu integrieren. Viele dieser Entwicklungen sind mit weitreichenden Marketing-Maßnahmen verbunden, die diesen zunächst den Status des Hype verleihen, sich aber wie XML später doch als ´ganz brauchbar´ herausstellen. Auch im Bereich der Banken- bzw. Überwachungs-Software, in dem der praktische Teil dieser Arbeit angesiedelt ist, gelten diese Regeln. Globalisierung, Vernetzung, anspruchsvollere Konsumenten und verstärkter Konkurrenzdruck sind die Herausforderungen an die Finanzwirtschaft, in der, wie in kaum einem anderen Bereich, die IT-Systeme durch

---

[1] Unter dem Rational Unified Process (RUP) versteht man ein objektorientiertes, UML-basiertes Vorgehensmodell, welches gleichzeitig kommerziell von der Fa. Rational Software vertrieben wird. Weitere Informationen unter http://www.rational.com/rup.

[2] Unter Extreme Programming (XP) versteht man einen leichtgewichtigen Entwicklungsprozess für kleine Teams, der aus 12 Richtlinien besteht und hierbei auch bestimmte soziale Aspekte des Software-Engineerings berücksichtigt. Weitere Informationen unter http://www.extremeprogramming.org.

[3] Die Unified Modelling Language (UML) ist eine grafische Notation zur Erstellung objektorientierter Modelle für die Analyse und den Entwurf objektorientierter Software. Weitere Informationen unter http://www.uml.org.

[4] Die Extensible Markup Language (XML) ist eine Metasprache zur Definition von Markup-Sprachen. Eine Einführung in XML bietet *Kapitel 2.1.2 – XML (Extensible Markup Language)*.

[5, 6, 7, 8, 9, 10] Auf diese Begriffe wird in *Kapitel 2 – Architekturen, Technologien und Paradigmen* näher eingegangen.

beträchtliche Kosten, enorme Komplexität und hohe Anforderungen an die Funktionalität gekennzeichnet sind.

Die Lösung sind moderne Architekturen, die leicht in bereits bestehende heterogene Strukturen integrierbar, beliebig erweiterbar und adaptierbar sind und dabei im Idealfall auf offenen Standards basieren. Web Services versprechen diese Anforderungen zu erfüllen und sollen die nächste Generation des Internets - das ´service web´ - ermöglichen. Gleichzeitig steigt der Bedarf an Managementsystemen, welche in der Lage sind diese komplexen Systeme, welche manuell nicht mehr beherrschbar sind, zu überwachen und zu administrieren. Auch hier existieren eine Reihe neuer Technologien und Architekturen, die dies leisten wollen.

1.2 Zielsetzung

Diese Diplomarbeit verfolgt zwei Ziele: Zum einen soll sie in die Gebiete der Management- und Web Services-Architekturen einführen, deren Vor- und Nachteile, Einsatzgebiete und Entwicklung diskutieren und sie, sowie entsprechende ´state of the art´-Lösungen, damit dem Leser näher bringen. Zum anderen soll sie, basierend auf diesen Erkenntnissen, durch die prototypische Konzeption und Realisierung eines Überwachungssystem für Bankperipheriegeräte die enormen Möglichkeiten und Potentiale aufzeigen, wie durch die Kombination von Management- und Web Services-Architekturen lose-gekoppelte, plattformunabhängige, leicht integrierbare und adaptierbare Management-Systeme erstellt werden können – ein Aspekt, der nach Wissen des Autors zur Zeit noch nicht in der Literatur zu finden ist. Die Funktionalität des erstellten Prototyps umfasst nur die absolut notwendigen Aufgaben von Management-Systemen. Dieser Prototyp kann und wird entsprechend erweitert werden, um in eine zurzeit bei der Wincor Nixdorf GmbH & Co. KG entwickelte moderne Multichannel-Architektur einzufließen.

1.3 Vorgehensweise und Aufbau der Arbeit

Kapitel 2 – Architekturen, Technologien und Paradigmen – führt in das Umfeld der Management- und Web Services-Architekturen sowie in die für die Konzeption und Realisierung des Überwachungssystems notwendigen Technologien im Banken-Umfeld ein.

Kapitel 3 – Ein Überwachungssystem für Bankperipheriegeräte – beschreibt die Konzeption und Realisierung eines Überwachungssystems für Bankperipheriegeräte unter Verwendung der im zweiten Kapitel beschriebenen Konzepte.

Kapitel 4 – Bewertung – beurteilt die Ergebnisse, zu denen der Autor durch seine wissenschaftliche Auseinandersetzung mit diesen Themen gekommen ist.

Kapitel 5 – Zusammenfassung – fasst die wichtigsten Punkte dieser Arbeit zusammen und geht noch einmal auf die enormen Möglichkeiten ein, die sich aus der Verwendung dieser Architekturen bzw. Technologien ergeben.

1.4 Beschreibungsmethoden

Zur Darstellung der entwickelten Konzepte wird in dieser Arbeit, neben der verbalen Beschreibung, auch mit der zur Zeit am weitesten verbreiteten grafischen Notationssprache, der Unified Modelling Language (UML), gearbeitet. Die UML ist die Vereinigung und Weiterentwicklung verschiedener Analyse- und Design-Notationen, wurde 1997 von der OMG[1] als Notation für Softwarearchitekturen standardisiert und liegt zum Zeitpunkt dieser Arbeit in der Version 1.4 vor.

[1] Die Object Management Group (OMG) ist eine internationale, nicht profit-orientierte, herstellerübergreifende Organisation, die von einigen großen IT-Firmen mit dem Ziel gegründet wurde, die Entwicklung von objektorientierter Software und Technologie zu unterstützen. Dies geschieht hauptsächlich durch die Erstellung von Richtlinien und Standards.

KAPITEL **2**

Architekturen, Technologien und Paradigmen

Computers are useless – they can only give you answers.
(Pablo Picasso)

2 Architekturen, Technologien und Paradigmen

Dieses Kapitel soll in die Web Services-, Management- und Bankperipheriezugriffs-Architekturen und Technologien einführen und damit verknüpfte Paradigmen aufzeigen. Das Verständnis dieser Punkte ist Voraussetzung für die in *Kapitel 3 – Ein Überwachungssystem für Bankperipheriegeräte* vorgestellte Konzeption und Realisierung eines Überwachungssystem-Prototyps. Dieser baut auf diesen Technologien auf und versucht ihre Vorteile zu nutzen, indem er sie sinnvoll verknüpft und anwendet. Da sowohl XML als auch SNMP mittlerweile weit verbreitetet und bekannt sind, können Leser, die in diesen Bereichen entsprechende Kenntnisse besitzen, *Kapitel 2.1.2 – XML (Extensible Markup Language)* und *Kapitel 2.2.4 – Internet Management / SNMP* ohne Beeinträchtigung auslassen. Die Aufteilung dieses Kapitels stellt sich folgendermaßen dar:

- Web Services Architekturen
- Management Architekturen
- Architekturen zum Zugriff auf Bankperipheriegeräte

Zu jedem Themengebiet werden zunächst Historie und Notwendigkeit beschrieben, anschließend Begrifflichkeiten geklärt und tiefer in die Technologie eingestiegen. Am Ende jedes Themengebiets werden verschiedene Anwendungsbeispiele dargestellt, wie diese Architekturen heute in der IT verwendet werden. Aufgrund der Komplexität jeder dieser drei Bereiche soll jeweils nur auf die wichtigsten übergreifenden Punkte und die für den Prototyp notwendigen Details eingegangen werden. Für denjenigen Leser, welcher tiefer in die jeweilige Thematik einsteigen möchte, bietet das Literaturverzeichnis eine Reihe weiterführender Literatur.

2.1 Web Service Architekturen

Firma X möchte im Internet die Ware W bei Firma Y bestellen, wobei Y diejenige Firma sein soll, welche W am kostengünstigsten anbietet. Um den günstigsten Anbieter zu finden, muss ein Mitarbeiter der Firma X mit Hilfe von Suchmaschinen und Katalogen verschiedene Anbieter auswählen und auf deren Webseiten nach den Konditionen recherchieren. Günstiger wäre es, die Anforderungsparameter, also Produkt, Kosten, Lieferbarkeit, Zahlungsbedingungen usw. über eine geeignete Schnittstelle zu definieren und damit einen Software-Agenten zu veranlassen, alle Angebote zu durchsuchen um den optimalen Anbieter zu finden. An dieser Stelle setzt die Vision des dynamischen E-Business unter Nutzung der ´Web Services´ Technologie an. In diesem Kapitel sollen Web Services, ihre Visionen, die zugrunde liegenden Technologien sowie Auswirkung auf E-Business und Geschäftsmodelle diskutiert werden.

2.1.1 Ein neues Paradigma

Das Problem ist bekannt: Es existieren Millionen von Web-Servern weltweit, welche fast das gesamte Wissen der Menschheit jedermann zugänglich machen sollen. Die übliche Form, in der Informationen geliefert werden, ist HTML, welches – von einem Browser dargestellt – den Inhalt dem menschlichen Betrachter präsentiert. Das heißt, das World Wide Web ist zurzeit fast ausschließlich für die Maschine-Mensch-Kommunikation vorgesehen (siehe hierzu auch *Kapitel 2.1.2.1 – Warum XML?*). Web Services sind derzeit eines der am heißesten diskutierten Themen, versprechen sie doch, die Informationen auch Maschinen (Software) zugänglich zu machen, also zusätzlich die Möglichkeit der ´program-to-program interaction´ zu eröffnen. Gerade diese Möglichkeit, das volle Potential des Internets zu nutzen, ist heute für ausgefeilte B2B-, B2C- oder P2P-Lösungen von zunehmender Bedeutung.

Mit CORBA[1], RMI[2], DCOM[3] usw. existieren bereits verschiedene Protokolle für den Zugriff auf entfernte Dienste. Diese sind aber komplex, teuer und meist nicht interoperabel. Weiterhin eignen sich diese Protokolle aus dem CORBA-, Java- und

[1] Bei der Common Object Request Broker Architecture (CORBA) handelt es sich um die zurzeit bekannteste und ausgereifteste Middleware-Architektur. Diese Plattform ist sowohl programmiersprachen- als auch plattformunabhängig.

[2] RMI (Remote Method Invocation) ist das verteilte Komponentenmodell der Java-Plattform.

[3] Die Distributed Components (DCOM) Architektur ist das verteilte Komponentenmodell der Microsoft-Plattform.

Microsoft-Bereich nicht für das Internet, da sie verbindungsorientiert sind und stabile Verbindungen erfordern, die das Internet nicht bieten kann. Außerdem sind diese Protokolle für Aufrufe über das Internet ungeeignet, da sie aufgrund variabler Portzuordnungen und ähnlichem Probleme mit dem Verbindungsaufbau durch Firewalls haben. Diese ´tightly coupled systems´ werden durch ´message oriented middleware´ (MOM) ergänzt, welche die zeitliche Entkoppelung von Systemen durch asynchrone Nachrichtenverarbeitung ermöglicht, aber im Internet ähnliche Probleme wie die oben genannten besitzt. Das grundsätzliche Problem besteht darin, dass Middleware-Technologien und das Internet zwei völlig getrennte Welten darstellen. Die Idee besteht nun darin, diese beiden Welten zusammenzubringen und das Internet zu einer Art Middleware zu erweitern. Man setzt für die Kommunikation das übliche Web-Protokoll HTTP ein und benutzt für den Datentransport XML, welches selbstbeschreibend und interoperabel ist. Als Messaging-Protokoll wird die im Internet bereits existierende Messaging-Lösung eingesetzt: E-Mail über SMTP. Dieses Szenario bzw. diese Vision wird als ´Web Service Technologie´ bezeichnet.

Die Web Service Technologie ist also eher Evolution als Revolution. Bereits existierende Technologien bilden die Grundlage, auf denen Web Services aufbauen. Oder um es mit James Harts[1] Worten zu sagen: ”Web Services are, in many ways, the logical conclusion of several threads which have been developing in the IT industry for twenty years. However, this doesn’t mean we should underestimate their significance. By combining the steam engine and the wheel, George Stephenson was simply combining the logical conclusions of two established fields of engineering into their logical conclusion, but the implications of the invention of railroads can still be felt today. Web Services have something of the feel of that first steam train about them – it’s going to be an interesting ride.”[2] Folgende Abbildung verdeutlicht die Entwicklungen, welche die Basis für Web Services legten.

[1] James Hart ist einer der Autoren des bekannten Buches ´Professional Java XML´, Wrox Press.

[2] Vgl. [JHWS01], S. 2.

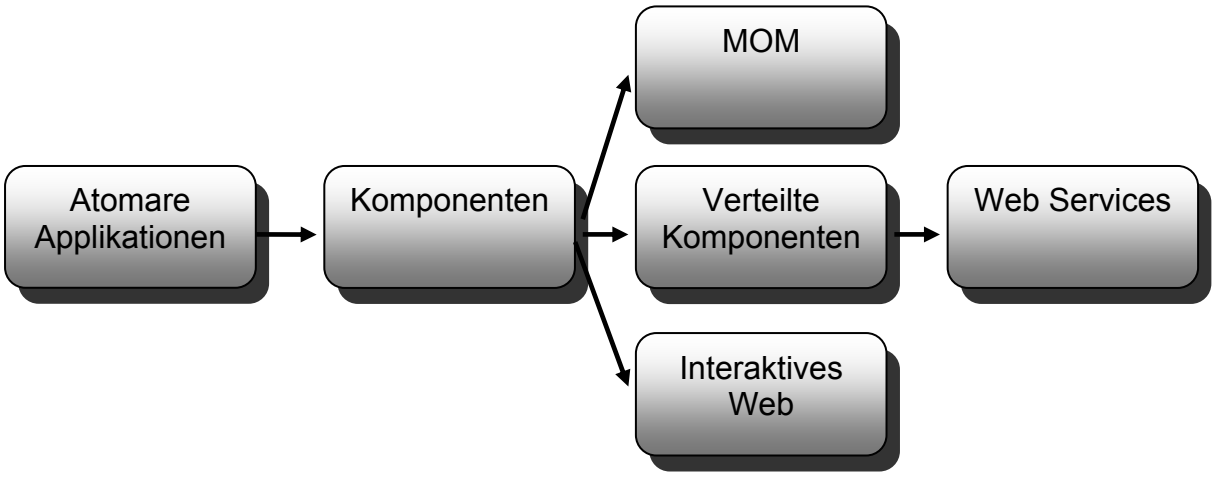

Abbildung 2-1 Entwicklungsgeschichte der Web Services

Hierbei ist jedoch zu beachten, dass Web Services als Ergänzung zu den bereits existierenden Technologien betrachtet werden müssen. Die Web Service Technologie dient dazu, Geschäftslogik über das Internet zur Verfügung zu stellen, wobei Technologien wie J2EE und CORBA der Implementierung dieser Geschäftslogik dienen.

Da Web Services von allen Global Players der IT, d.h. IBM, Microsoft, Sun, HP, Oracle usw. propagiert werden, existieren für den Begriff ´Web Service´ eine Reihe von Definitionen. Eine, nach Ansicht des Autors, geeignete findet sich in [CBDIWS]: “A web service provides programmable access to a capability that can be used independently of its implementation via a self-describing interface based on open internet standards.” Ein Web Service stellt eine Fähigkeit (capability) zur Verfügung – dies ist der geleistete Dienst (´service´). Der Zugriff auf diesen Dienst erfolgt via Internet-Protokollen – daher der Begriff ´web´. Bei diesen Protokollen handelt es sich um ´offene Standards´, d.h. Standards, auf welche sich die Mehrzahl der Unternehmen geeinigt haben und die vom W3C verwaltet werden. Weiterhin können Web Services dynamisch gefunden und aufgrund des ´programmierbaren Zugriffs´ in eigene Applikationen integriert werden.

2.1.1.1 Web Service Perspektiven

Ein Web Service kann aus zwei Perspektiven betrachtet werden: aus der Sicht des Konsumenten des Dienstes und aus der Sicht des Erbringers (Provider) des Dienstes. Für den Konsumenten ist nur der erbrachte Dienst entscheidend, d.h. es interessiert, welche Aufgabe der jeweilige Dienst besitzt, wie er benutzt werden kann

und auffindbar ist. Entscheidend für den Konsumenten ist also nur die Dienstbeschreibungsschnittstelle, so dass aus dieser Sicht ein Web Service ein Dienst ist, welcher über eine Schnittstelle im Internet angeboten wird. Für den Provider ist neben der Schnittstelle die Implementierung des Dienstes bestimmend. Der Provider entscheidet, welche Funktionalität der Dienst bietet, an welcher Stelle er sich zur Laufzeit befindet und wo sein Vorhandensein publiziert ist. Aus der Sicht des Providers handelt es sich also bei einem Web Service sowohl um die offen gelegte Schnittstelle als auch die Software, welche den Dienst implementiert. Die folgende Abbildung verdeutlicht dies.

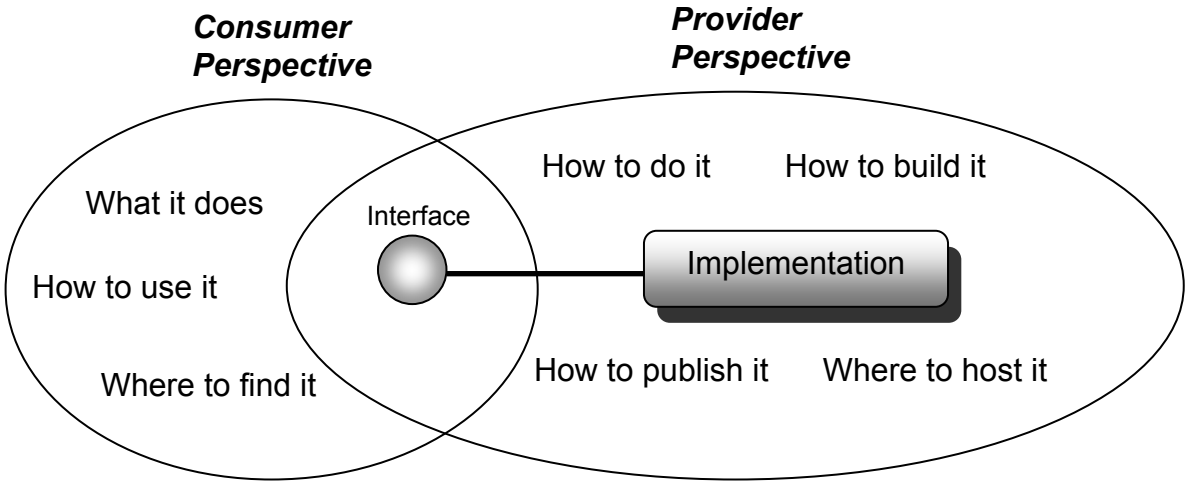

Abbildung 2-2 Web Service Perspektiven

2.1.1.2 Web-Applikations Topologien

Was unterscheidet die Web Service Technologie von anderen Topologien, wie zum Beispiel ´Application Service Providing´ (ASP) oder ´Software on Demand´ (SOD)? Application Service Provider stellen meist Komplettlösungen einer Software zur Verfügung, d.h. die gesamte Applikation im Sinne von Benutzeroberfläche, Workflow, Geschäfts- und Daten-Komponenten ist in einer Lösung gebunden. Diese Komplettlösung wird auf den Servern der ASPs gelagert, verwaltet und ausgeführt. Der Benutzer dieser Software greift meist mittels Browser auf sie zu, um damit zu arbeiten. Es existieren in den meisten Fällen wenige Möglichkeiten diese Software zu konfigurieren und sie in die eigenen Geschäftsprozesse einzubinden. Eine Alternative bietet die Software on Demand Lösung, bei welcher die Software auf Bedarf (on demand) auf den Rechner des Kunden geladen wird. Diese Software kann am Ende der Sitzung gelöscht werden oder auf dem Rechner verweilen, bis sie durch eine neue Version ersetzt wird oder ein bestehender Nutzungsvertrag ausläuft.

Diese Lösung bietet die Möglichkeit der unbeschränkten Konfiguration und Personalisierung. Web Services bieten eine weit flexiblere Lösung: Der Kern der Applikation, d.h. Geschäfts- und Datenkomponenten lagern weiterhin auf dem Server, es kann nun aber programmatisch über ihre Schnittstellen auf sie zugegriffen werden. So kann der Kunde seine eigenen Geschäftsprozesse und Benutzeroberflächen erstellen und in diese beliebige Web Services beliebiger Provider einbinden. Folgende Abbildung verdeutlicht dies.

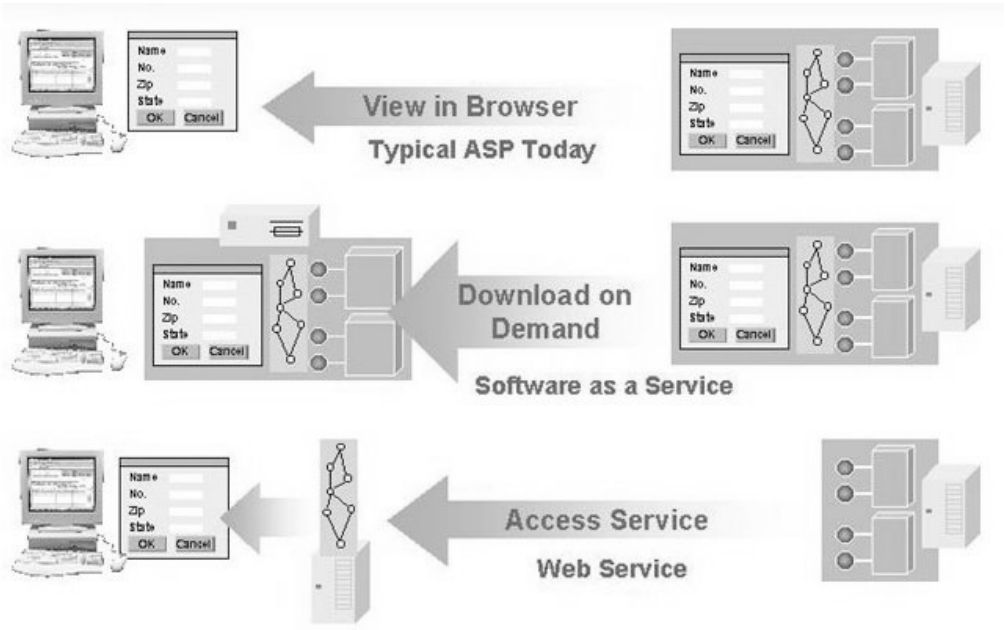

Abbildung 2-3 ASP, SOD und Web Services (Quelle: [CBDIWS], S.11)

2.1.1.3 Zukunfts-Szenario

Web Services können, wie bereits angesprochen, in allen Bereichen verwendet werden, wobei ihr größtes Potential sicher in den Bereichen B2B und B2C liegt. Ein E-Commerce-Szenario der Zukunft, bei welchem die Informationen der Wertschöpfungskette eines Unternehmens ausschließlich über Web Services ausgetauscht werden, wird in *Abbildung 2-4 E-Commerce Szenario der Zukunft* dargestellt. Die Vorteile stellen sich folgendermaßen dar: Zunächst ist es möglich, direkt mit der jeweiligen Informationsquelle zu kommunizieren, wodurch jedes Datum der Wertschöpfungskette immer akkurat und aktuell ist und damit die Notwendigkeit der regelmäßigen Replikation entfällt. Verlangt der Kunde Informationen zur Lieferbarkeit eines Produktes, wird diese Anfrage automatisch an den entsprechenden Lieferanten weitergeleitet und von diesem beantwortet. Auf gleichem Weg wird das Problem der ungenauen, falschen oder veralteten Daten behoben. So

ist es sowohl für den Kunden als auch für das Unternehmen möglich, bestimmte Web Services der Wertschöpfungskette bzw. des Kunden-Frontends unabhängig von Plattform und Endgerät in die eigene Applikation oder Kette einzubinden.

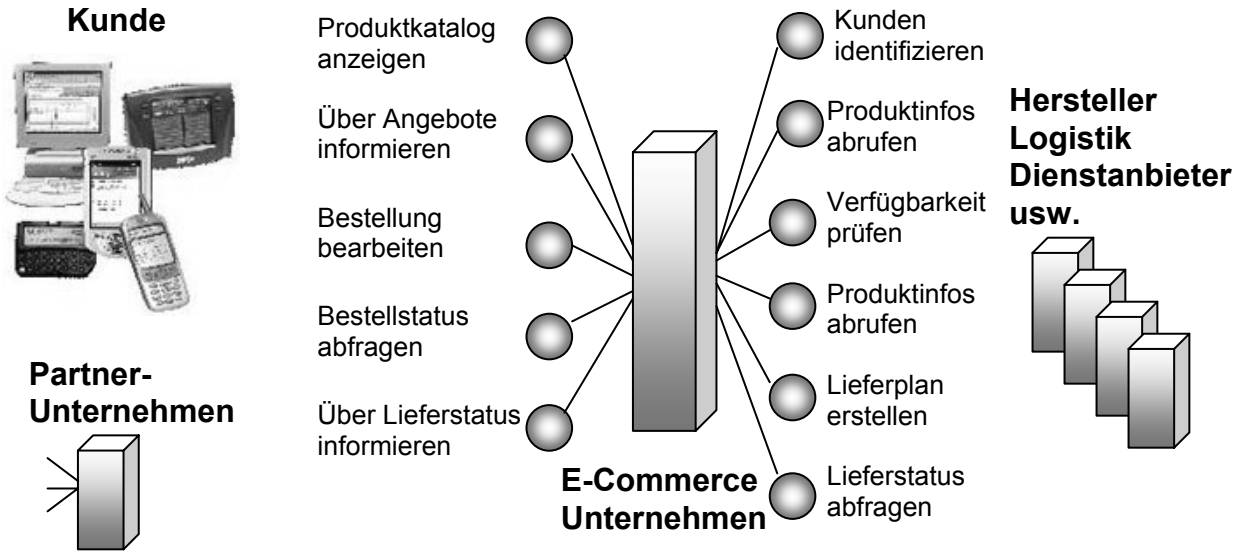

Abbildung 2-4 E-Commerce Szenario der Zukunft

2.1.1.4 Web Services Modell

Die Web Services Architektur basiert auf Interaktionen zwischen drei Instanzen, welche jeweils eine bestimmte Rolle einnehmen. Diese Instanzen sind:

- Service Provider

  Der Service Provider stellt den Web Service zur Verfügung, welcher einen spezifischen Dienst leistet. Aus der Geschäftsperspektive ist dies der Besitzer des Dienstes, aus der Architekturperspektive derjenige, welcher den Dienst hostet und den Zugriff auf diesen ermöglicht.

- Service Requestor (Consumer)

  Der Service Requestor ist der Konsument des Dienstes. Er findet und benutzt den Dienst, um eine bestimmte Aufgabe zu erfüllen. Aus der Geschäftsperspektive ist dies das Unternehmen, welches eine bestimmte Funktionalität benötigt, aus der Architekturperspektive ist dies die Applikation, welche nach einem Dienst sucht und eine Interaktion mit diesem Dienst anstrebt. Die Rolle des Service Requestors kann hierbei von einem Browser, welcher auf Benutzereingaben reagiert, oder einem Programm ohne Benutzerschnittstelle gespielt werden. Natürlich kann der Service Requestor selbst ein Web Service sein, wodurch Workflows realisiert werden können.

- Service Registry (Broker)

  Die Service Registry bietet ein durchsuchbares Verzeichnis für Web Services Schnittstellen. Service Requestors können in diesem Verzeichnis Dienste auffinden, um diese zu nutzen.

Um das volle Potential dieser Architektur auszunutzen, müssen drei Interaktionen stattfinden, welche sowohl einzeln als auch iterativ auftreten können:

- publish

  Damit auf einen Web Service zugegriffen werden kann und der Service Requestor diesen finden kann, muss dessen Schnittstelle veröffentlicht werden. An welcher Stelle die Schnittstelle veröffentlicht wird, hängt von verschiedenen Faktoren ab. Ist eine dynamische Bindung gewünscht, geschieht dies in der Service Registry. Andernfalls kann die Schnittstellenbeschreibung auch direkt zwischen Service Provider und Service Requestor ausgetauscht werden (zum Beispiel über E-Mail, FTP, Veröffentlichung auf Website usw.).

- find

  Mit Hilfe dieser Operation erlangt der Service Requestor die Schnittstellenbeschreibung des gewünschten Dienstes. Dies kann direkt oder über das Suchen innerhalb der Service Registry geschehen. Weiterhin kann diese Operation in zwei verschiedenen Lebenszyklusphasen aufgerufen werden: zur Designzeit, um die Schnittstellenbeschreibung zur Programmentwicklung zu verwenden und zur Laufzeit, um eine dynamische Nutzung der Architektur zu ermöglichen.

- bind

  Durch diese Operation baut der Service Requestor eine Verbindung zum Service Provider auf und initiiert eine Interaktion. Das Initiieren der Interaktion geschieht entweder durch einen direkten Aufruf einer Methode des Web Service (RPC) oder dadurch, dass sich der Service Requestor für asynchrone Ereignisse beim Service Provider registriert (Messaging).

Wie bereits in *Kapitel 2.1.1.1 – Web Service Perspektiven* angesprochen, hat ein Web Service zwei grundlegende Bestandteile:

- Service

  Die Implementierung des Web Service liefert den Service selbst. Dieser Service ist ein Softwaremodul, welches auf dem Server des Service Providers abgelegt ist

und per Netzwerk zugreifbar ist. Es interagiert mit dem Service Requestor oder kann selbst als Service Requestor fungieren, indem es andere Web Services benutzt.

- Service Description

  Die Beschreibung eines Web Service beinhaltet die Details der Schnittstelle und der Implementierung des Dienstes. Dies umfasst Datentypen, Operationen, Bindungsinformationen und Informationen über den Server, auf welchem sich der Dienst befindet. Es können weiterhin Metainformationen über den Dienst und dessen Provider enthalten sein, um den Web Service zu kategorisieren und das Auffinden durch den Service Requestor zu vereinfachen.

Das typische Szenario sieht folgendermaßen aus: Ein Service Provider stellt unterschiedliche Dienste, wie zum Beispiel einen Aktienkurs-Informationsdienst, auf seinem Server zur Verfügung. Desweiteren veröffentlicht er die Web Service Beschreibung in einer globalen Service Registry, wo diese von jedem aufgefunden werden kann. Ein Service Requestor sucht in dieser Registry nach für ihn geeigneten Diensten und findet zum Beispiel diesen Aktienkurs-Informationsdienst. Er erhält als Antwort auf seine Anfrage an die Registry die Schnittstellenbeschreibung des Dienstes und ist nun in der Lage auf diesen zuzugreifen, um ihn zum Beispiel in eine eigene Applikation einzubinden. Das Web Services Modell stellt sich also folgendermaßen dar:

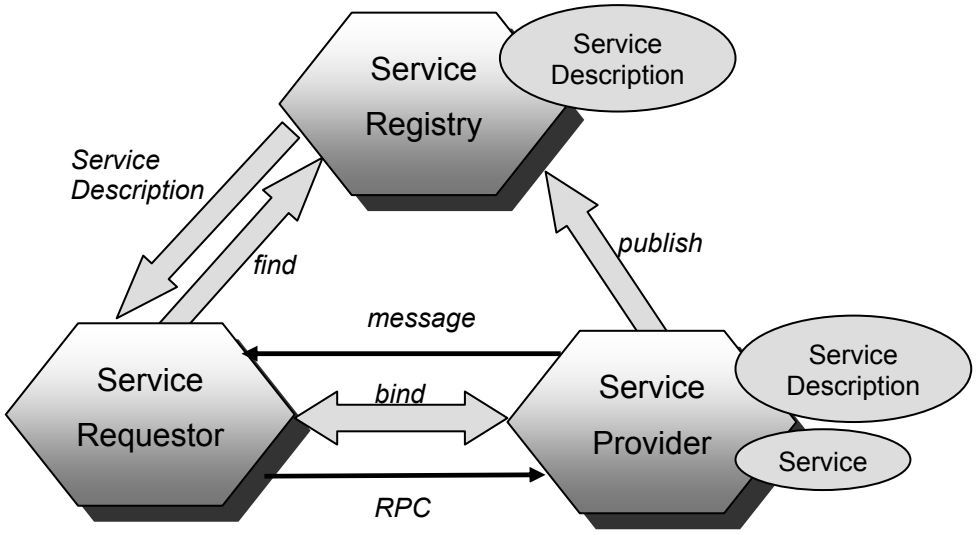

Abbildung 2-5 Web Services Modell

2.1.1.5 Warum Web Services?

Es existiert eine Reihe von Vorteilen, welche sich durch die Nutzung von Web Services ergeben. Einige dieser Vorteile können auch durch andere Ansätze und bereits existierende Technologien erreicht werden. Nachfolgend verschiedene für die Web Service Technologie spezifische Vorteile:

- Loose technologie coupling – zwischen Requestor- und Provider-Plattform besteht keine Technologieabhängigkeit mehr.
- Late binding – ermöglicht den Beziehungsaufbau zwischen Systemen zur Laufzeit.
- Dynamic Inspection – die Fähigkeiten und Funktionalitäten von Web Services können auch zur Laufzeit aufgefunden werden.
- Programmable – Web Services ermöglichen es, dass Systeme Funktionen über das Internet aufrufen, statt nur Daten auszutauschen und zu präsentieren.
- Use of standard protocols – es wird eine Reihe standardisierter und allgemein anerkannter Protokolle verwendet, statt wie bisher viele verschiedene proprietäre Technologien.
- Industry consensus – alle Key Player akzeptieren und propagieren die Web Service Technologie.
- Protecting investments – Web Services ersetzen keine bestehenden Systeme, sondern fügen eine Schicht weiterer Funktionalität hinzu. Dadurch sind bereits getätigte Investionen geschützt.

2.1.1.6 Web Service Standards

Web Services sind in jeder Netzwerk-Umgebung verwendbar, egal ob es sich dabei um Internet, Extranet oder Intranet handelt. Ihr alleiniger Fokus liegt auf dem Zusammenspiel der Anwendungskomponenten. Weiterhin spielen weder Betriebssystem noch Programmiersprache oder Komponentenmodell eine Rolle. Es liegt nahe, dass eine solche Offenheit der Zusammenarbeit einige Standards voraussetzt. Diese Standards, welche in den nachfolgenden Kapiteln näher vorgestellt werden sollen, sind: XML, SOAP, WSDL und UDDI. Es ist jedoch zu beachten, dass diese vier Standards nur die erste Stufe und damit die Basis der Web Services Evolution darstellen. Um die vollständige Vision zu verwirklichen, bedarf es

einer Reihe weiterer Standards, welche sich zum Teil bereits in der Entwicklung befinden. Der Ansatz, welcher insbesondere von IBM propagiert wird, besteht in einer modularen Annäherung an die Gesamtvision, d.h. die Evolution verläuft in Phasen. Dies impliziert, dass jedes Protokoll separat eingeführt wird und Zeit hat sich selbständig zu entwickeln.

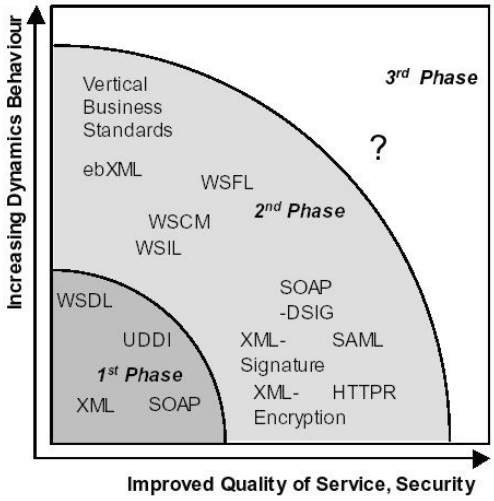

Abbildung 2-6 Web Services Evolution (Quelle: [WSRHRN],S.5)

Eine Erklärung der dargestellten Technologie liefert die folgende Tabelle:

XML	Extensible Markup Language; Metasprache zur Definition von Markup-Sprachen
SOAP	Simple Object Access Protocol; XML-Protokoll zum Transport von XML-Daten
WSDL	Web Services Description Language; XML-Protokoll zur Beschreibung von Web Services
UDDI	Universal Description, Discovery, Integration; Basis für Web Services Registry
WSIL	Web Services Inspection Language; Durchsuchen einer Website nach verfügbaren Web Services
WSCM	Web Services Component Model; Komposition und Präsentation von Web Services
ebXML	Electronic Business Extensible Markup Language; XML-basierte Infrastruktur für den elektronischen Handel
XML-Signature	Beschreiben und Prüfen von digitalen Signaturen auf XML-Basis

XML-Encryption	Verschlüsselung von digitalen Inhalten auf XML-Basis
SOAP-DSIG	Simple Object Access Protocol – Digital Signatures; Signieren von SOAP Nachrichten
SAML	Security Assertion Markup Language; Austausch von Authentifikations- und Autorisierungsinformationen
HTTPR	Hypertext Transfer Protocol Reliable; Zuverlässiger Datentransport über HTTP
WSFL	Web Services Flow Language; Workflow auf Web Services Basis

Tabelle 2-1 Web Services Standards

Wir befinden uns zurzeit in der ersten Phase (2001 – dynamic business integration, simple transactions). Die Basisprotokolle (XML, SOAP, WSDL und UDDI) sind standardisiert oder zumindest de facto Standards. In dieser Phase sind bereits folgende Szenarien möglich:

- Verbesserte Enterprise Application Integration (EAI) – Web Services werden intern zur vereinfachten Anbindung an Legacy-Systeme verwendet.
- Business to Business (B2B) Integration zwischen Partnern – Web Services ermöglichen Echtzeit-Kommunikation zwischen ausgewählten Organisationen, unabhängig von der für die Implementierung verwendeten Technologien.
- Erstellung von intelligenten Endpunkten – Web Services liefern strukturierte Informationen zur externen Weiterverarbeitung in fremden Applikationen ohne die bisherige Abhängigkeit von Benutzerinteraktionen.
- Einfache Informationsdienste und Transaktionen sind für eine weitreichendes Publikum verfügbar.

Obwohl die Web Service Technologie noch in ihren Kinderschuhen steckt, wird sie von allen Seiten akzeptiert und durch eine Vielzahl von Entwicklungsumgebungen, Adaptern und Konnektoren unterstützt. “It is difficult to find a previous technology that the whole industry has been so quick to endorse and adopt without major dissent.”[1]

[1] Vgl. [WSRHRN], S.5.

Phase zwei (2002-2004 – improved infrastructure, more complex transactions) wird weitere Protokolle einführen, welche eine robustere Infrastruktur und komplexere Transaktionsszenarios ermöglichen. Eine weitere wichtige Aufgabe dieser Protokolle ist die Schaffung einer Sicherheitsarchitektur für Web Services.

In Phase drei (2004+ – adaptive systems) haben Web Services eine weite Verbreitung gefunden und Organisationen werden nun nicht nur ihre Systeme, sondern auch ihre Geschäftsmodelle anpassen, um die Vorteile des dynamischen Verhaltens der Web Service Technologie zu nutzen. Dazu gehören das dynamische Auffinden von Geschäftspartnern, die dynamische Nutzung von entfernten Diensten und die dynamische Anpassung an Änderungen aller Art. Es werden sich selbst beschreibende und sich selbst heilende Systeme möglich, welche sich permanent optimieren, um neue Möglichkeiten zu nutzen.

2.1.1.7 Web Services Stack

Um die Operationen publish, find und bind interoperabel auszuführen, wird ein Web Services Stack benötigt, welcher auf jeder Ebene entsprechende Standards und Protokolle zur Verfügung stellt. *Abbildung 2-7 Web Services Stack* zeigt diesen Web Services Stack. Die jeweils über einer anderen liegende Schicht nutzt die Fähigkeiten der darunter liegenden Schichten und baut auf diesen auf. Die vertikalen Balken repräsentieren Anforderungen, welche in jeder Schicht erfüllt sein müssen. Der Text auf der linken Seite nennt die Standards bzw. aufkommende Standards, die die jeweilige Schicht adressieren.

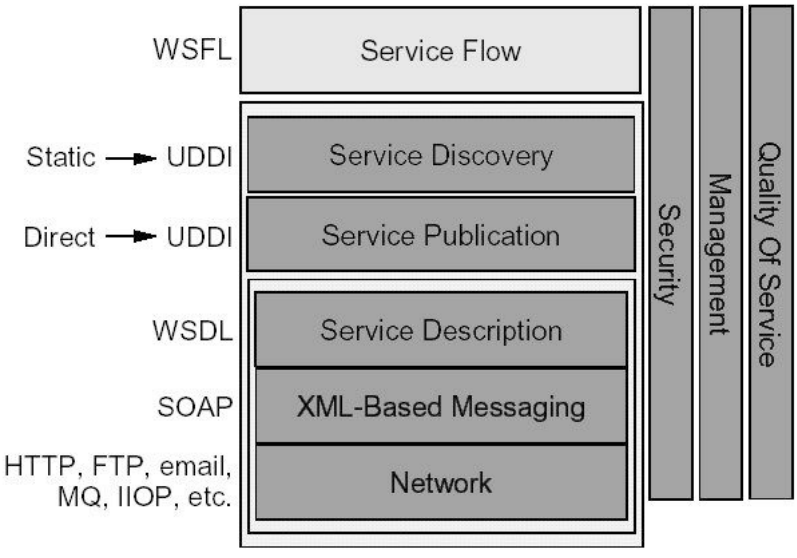

Abbildung 2-7 Web Services Stack (Quelle: [WSCA], S.10)

Die Grundlage des Web Services Stacks bildet das Netzwerk. Web Services müssen über das Netzwerk zugreifbar sein, um den Informationsaustausch zwischen Service Requestor und Service Provider zu ermöglichen. Web Services im Internet-Bereich nutzen weit verbreitete Netzwerkprotokolle wie HTTP, SMTP und FTP, während Web Services im Intranet Bereich weitere Protokolle wie IIOP[1] oder MQ (Message Queuing) benutzen können. Die zweite Schicht, das XML-basierte Messaging, repräsentiert die Benutzung von XML (siehe *Kapitel 2.1.2 – XML (Extensible Markup Language)*) als Basis für ein Messaging-Protokoll. In diesem Bereich hat sich SOAP (siehe *Kapitel 2.1.3 – SOAP (Simple Object Access Protocol)*) aus den verschiedensten Gründen als alleiniger Standard durchgesetzt. Die Service Description Schicht besteht eigentlich selbst wiederum aus einem Stack von Web Service Beschreibungs-Dokumenten, wobei WSDL (siehe *Kapitel 2.1.4 WSDL (Web Service Description Language)*) der minimale de facto Standard ist, um Web Services zu beschreiben. Weitere Beschreibungen auf höherer Ebene, wie zum Beispiel den Unternehmenskontext, Servicegüte oder Dienst-Dienst-Beziehungen bieten andere Protokolle. Der Unternehmenskontext wird zum Beispiel durch UDDI (siehe *Kapitel 2.1.5 – UDDI (Universal Description, Discovery, Integration)*) beschrieben, welches in Verbindung mit WSDL eingesetzt wird. Service Komposition und Workflow werden in einem WSFL Dokument beschrieben. Jede Aktion, welche ein WSDL Dokument einem Service Requestor zugänglich macht, wird als ´Service Publication´ bezeichnet. Dabei besteht die einfachste und zugleich am wenigsten dynamische Möglichkeit darin, das WSDL Dokument direkt an den Service Requestor zu senden. Alternativ kann der Service Provider das Dokument in einer öffentlich zugänglichen UDDI-Registry veröffentlichen, wo es für den Service Requestor auffindbar ist. Das Auffinden des Dokumentes geschieht ebenfalls mit Hilfe des UDDI Protokolls und wird als ´Service Discovery´ bezeichnet.

Da die Implementierung eines Web Services ein Softwaremodul ist, liegt es nahe, Web Services durch Komposition anderer Web Services zu erzeugen. Hierbei existieren eine Reihe von Anwendungsgebieten. So könnten unternehmensinterne Web Services kollaborieren, um Workflows abzubilden, wobei nach außen hin nur ein Web Service sichtbar ist, welcher die Workflow-Kette anstößt. Eine weitere wichtige Anwendung ist die Kollaboration von Web Services verschiedener Unternehmen zum

[1] IIOP (Internet Inter ORB Protocol) ist das CORBA-Protokoll für die Kommunikation im Internet.