Technische Universität Darmstadt

WSQoSX – Eine dienstgütebasierte Web-Service-Architektur

Michael Spahn, Rainer Berbner, Oliver Heckmann,
Andreas Mauthe, Ralf Steinmetz

KOM Technical Report 07/2004
Version 1.1, Oktober 2004
Version 2.0, Mai 2005

Department of Electrical Engineering & Information Technology
Merckstraße 25 • D-64283 Darmstadt • Germany
Phone: +49 6151 166150
Fax: +49 6151 166152
Email: info@KOM.tu-darmstadt.de
URL: http://www.kom.tu-darmstadt.de/

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

3DES	Triple-DES
ADO	Active Data Objects
API	Application Programming Interface
ASP	Active Server Pages
B2B	Business-to-Business
BPE	Business Process Engine
BPEL4WS	Business Process Execution Language for Web Services
BPO	Business Process Outsourcing
CCITT	Commité Consultatif Internationale Télégrafique et Téléfonique
CSS	Cascading Style Sheets
DAO	Data Access Objects
DES	Data Encryption Standard
DiffServ	Differentiated Services
DOM	Document Object Model
DoS	Denial of Service
EAI	Enterprise Application Integration
EJB	Enterprise Java Bean
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDEA	International Data Encryption Algorithm
IIS	Internet Information Services
IntServ	Integrated Services
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
J2EE	Java 2 Platform Enterprise Edition
J2SE	Java 2 Platform Standard Edition
JAXR	Java API for XML Registries
JDBC	Java Database Connectivity
JSP	JavaServer Pages
JSTL	JavaServer Pages Standard Tag Library
JVM	Java Virtual Machine
MIME	Multipurpose Internet Mail Extensions
MPLS	Multiprotocol Label Switching
OASIS	Organization for the Advancement of Structured Information Standards
ODBC	Open DataBase Connectivity

PHP	PHP Hypertext Preprocessor
QoS	Quality of Service
RPC	Remote Procedure Call
RSVP	Resource Reservation Protocol
SLA	Service Level Agreement
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
STP	Straight Through Processing
TCP	Transmission Control Protocol
UDDI	Universal Discovery, Description and Integration
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VBA	Visual Basic for Applications
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Service Description Language
WSFL	Web Services Flow Language
WSLA	Web Service Level Agreement
WSML	Web Services Management Language
WSMN	Web Services Management Network
WSOI	Web Service Offerings Infrastructure
WSOL	Web Service Offerings Language
WSQoSX	Web Service Quality of Service Architectural Extension
WS-Security	Web Services Security
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language

1 Einführung

1.1 Motivation

Die Märkte, auf denen Unternehmen agieren, befinden sich in einem stetigen Prozess der zunehmenden Globalisierung und Deregulierung. Auf diesen sind sie einer großen Anzahl von Wettbewerbern und einem zunehmenden Innovations- und Kostendruck ausgesetzt. Im harten Wettbewerb um Marktanteile können sie nur bestehen, indem sie hochgradig innovativ sind und sich schnell an neue Wettbewerbssituationen anpassen. Geschäftsprozesse der Unternehmen müssen hierzu einerseits flexibel sein, um schnell an neue Anforderungen adaptiert werden zu können, und andererseits effizient und kosteneffektiv umgesetzt werden können [44] [8].

Hierbei können Unternehmen, die es verstehen sich selbst mit anderen Unternehmen zu vernetzen und sie in ihre eigenen Geschäftsprozesse zu integrieren, besondere Vorteile erringen. Hierzu zählen z.B. die Reduktion von Transaktionskosten durch eine hohe Rate der Automatisierung und „Straight Through Processing“ (STP) [85] [86], sowie die Nutzung von Synergie- und Skaleneffekten [9]. Ansatzpunkte sind hierbei die bessere Integration vor- und nachgelagerter Wertschöpfungsstufen der Supply-Chain, sowie die Auslagerung von Geschäftsprozessen, die nicht den Kernkompetenzen der Unternehmung zuzuordnen sind, im Rahmen des „Business Process Outsourcing“ (BPO) [67] [37].

Darüber hinaus kann die Fähigkeit der flexiblen Integration externer Unternehmen völlig neue Geschäftsfelder erschließen, wie z.B. die hochvolumige Abwicklung standardisierter Prozesse für Dritte in der Form einer „Prozessfabrik“. So versucht die Finanzindustrie im Rahmen der „Industrialisierung der Finanzindustrie“ [73] Transaktionsbanken zu schaffen, deren hauptsächliche Aufgabe in der Abwicklung eines hohen Volumens standardisierter Transaktionen des Zahlungsverkehrs und Wertpapiergeschäfts für kooperierende Unternehmen der Finanzindustrie besteht.

Grundvoraussetzung für die Integration externer Unternehmen in die eigenen Geschäftsprozesse und die Schaffung verteilter, unternehmensübergreifender Geschäftsprozesse [56] ist die Fähigkeit diese mithilfe einer flexiblen und kostengünstigen IT-Infrastruktur abbilden und ausführen zu können. Mit dem Internet steht eine leistungsfähige und kostengünstige Kommunikationsinfrastruktur zur Verfügung um die Netzwerke der Unternehmen zu verbinden. Die eigentliche Herausforderung liegt daher nicht in der Herstellung einer Verbindung zwischen den, an den Geschäftsprozessen beteiligten, Netzwerken, sondern vielmehr in der Überwindung der Heterogenität der beteiligten Systeme, Plattformen und Anwendungen.

Mit Methoden der „Enterprise Application Integration“ (EAI) [51] [3] [4] ist hierbei zunächst die Heterogenität innerhalb der unternehmenseigenen IT-Landschaft zu überwinden. Hierzu müssen heterogene, autonome Anwendungssysteme prozessorientiert in eine geeignete, vernetzte IT-Infrastruktur integriert werden. Über eine reine Schnittstellenadaption durch klassische Middleware hinaus, werden hierbei Funktionalitäten gekapselt und mittels Adapter über eine einheitliche Infrastruktur zugänglich gemacht. Als Systemarchitekturkonzept wird hier typischerweise eine „Service Oriented Architecture“ (SOA) verwendet, in welcher fachliche Dienste und Funktionalitäten in Form von Services bereitgestellt werden. Auf dieser Infrastruktur

wird die Prozesslogik der Geschäftsprozesse durch eine Orchestrierung der Services modelliert und mittels einer „Business Process Engine“ (BPE) ausgeführt.

Die Systeme externer Unternehmen können über Services, die sie nach außen anbieten, in die Geschäftsprozesse einer solchen Infrastruktur integriert werden. Je transparenter sich hierbei die Services dem Geschäftsprozess gegenüber darstellen, desto mehr verschwimmen die Grenzen zwischen den einzelnen Systemen der Unternehmen. Idealerweise lassen sich so Geschäftsprozesse aus Services orchestrieren ohne dass hierbei bekannt sein muss, welches System in welchem Unternehmen diesen Service bereitstellt.

Zur Umsetzung einer solchen SOA steht mit dem Internet eine leistungsstarke und kostengünstige Kommunikationsinfrastruktur zur Verfügung und mit der darauf basierenden Technologie der Web Services [5] ein plattformunabhängiges, auf offenen Standards basierendes, serviceorientiertes Komponentenmodell. Zur Orchestrierung von Geschäftsprozessen aus Web Services stehen Standards wie z.B. die „Business Process Execution Language for Web Services“ (BPEL4WS) [6] mit entsprechenden Implementierungen von BPEs zur Verfügung.

Ein wichtiger Aspekt bei der Orchestrierung von Geschäftsprozessen ist jedoch nicht nur die Integration einer Funktionalität durch einen Web Service an sich, sondern auch die Berücksichtigung der Dienstgüte (engl. „Quality of Service“, Abk. QoS) mit welcher die durch den Web Service integrierte Funktionalität bereitgestellt wird [53] [59] [79]. Unternehmen werden nicht bereit sein Web Services in kritischen Geschäftsprozessen zu verwenden, für die der Anbieter nicht bestimmte Eigenschaften garantiert. Nur so kann sichergestellt werden, dass sie bestimmten Anforderungen, z.B. hinsichtlich ihrer Verfügbarkeit, ihrer Antwortzeit, dem verarbeitbaren Volumen, ihrer Sicherheit oder des abgerechneten Nutzungsentgelts genügen. Dienstgütekriterien werden jedoch weder direkt durch die Basisstandards der Web-Service-Technologie, wie SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) oder UDDI (Universal Description, Discovery and Integration) berücksichtigt, noch durch Standards zur Orchestrierung von Web Services wie BPEL4WS. Ist es einem Unternehmen in gewissen Grenzen noch möglich Dienstgütekriterien von Web Services zu beeinflussen, welche durch es selbst und im eigenen Intranet zur Verfügung gestellt werden, so endet diese Möglichkeit der Einflussnahme spätestens mit dem Zugriff auf Web Services externer Web-Service-Provider über externe Netze, wie dem öffentlichen Internet.

Wird ein und dieselbe Funktionalität durch eine Vielzahl von verschiedenen Web Services, z.B. durch verschiedene Web-Service-Provider, bereitgestellt, so stellen Dienstgütekriterien ein wichtiges Differenzierungsmerkmal dar [66] [50] [89], anhand derer entschieden werden kann, welcher Web Service, gemäß individueller Präferenzen, am besten für einen speziellen Einsatzzweck geeignet ist. Beispielsweise könnten so für kritische Geschäftsprozesse nur hochverfügbare Web Services eingebunden werden, für weniger kritische Geschäftsprozesse hingegen schlichtweg die günstigsten. Im Rahmen flexibler und dynamisch anpassbarer Geschäftsprozesse [75] [12] sollte die Auswahl geeigneter Web Services zur Durchführung einer Funktionalität zur Laufzeit automatisch, ohne manuelle Eingriffe möglich sein. Hierzu ist es nötig Dienstgütekriterien für Web Services beschreiben, in vertragsähnlicher Form zusichern, dynamisch nach diesen auswählen und deren Einhaltung überwachen zu können [11].

Gegenstand dieses Technical Reports ist die Betrachtung von *WSQoSX (Web Service Quality of Service Architectural Extension)*. WSQoSX bezeichnet die Implementierung einer Dienstgüte unterstützenden Web-Service-Architektur für flexible Geschäftsprozesse, welche in der Lage ist Web Services dynamisch, unter Berücksichtigung von Dienstgütekriterien auszuwählen, in Geschäftsprozesse einzubinden und auszuführen.

1.2 Zielsetzung des Technical Reports

1.2.1 Hintergerund

Dieser Technical Report entstand im Rahmen von Forschungsarbeiten für das „E-Finance Lab Frankfurt am Main“[1]. Das E-Finance Lab ist eine Kooperation zwischen der Johann Wolfgang Goethe Universität Frankfurt am Main, der Technischen Universität Darmstadt und einem Netzwerk aus Industriepartnern. Das Ziel des E-Finance Lab ist die Entwicklung industrieller Methoden für den Umgestaltungsprozess der Finanzdienstleistungsindustrie, der allgemeinhin als „Industrialisierung der Finanzindustrie“ bezeichnet wird [73]. Der generelle Ansatz, der hierbei verfolgt wird, ist die Übertragung von Methoden die sich in anderen Industriezweigen, wie z.B. bei der Optimierung der Supply-Chain in der Automobilindustrie, bereits bewährt haben, auf Geschäftsprozesse der Finanzdienstleistungsindustrie. Eine wichtige Herausforderung besteht hierbei in der Entwicklung optimierter Geschäftsprozesse auf Basis einer geeigneten Produktionsinfrastruktur, welche die Anwendung besonders vorteilhafter Sourcing-Strategien ermöglicht.

Die Forschungstätigkeiten des E-Finance Lab verteilen sich auf vier Cluster, wobei jeder Cluster Forschungsarbeiten in einem speziellen Gebiet wahrnimmt. Dieser Technical Report entstand für den Cluster II, welcher sich mit der Entwicklung innovativer IT-Architekturen zur Unterstützung von Geschäftsprozessen in der Finanzindustrie beschäftigt. Der Cluster II ist am Fachgebiet „KOM Multimedia Kommunikation“[2] der Technischen Universität Darmstadt ansässig und wird von Prof. Dr.-Ing. R. Steinmetz geleitet. Ein Fokus der Forschungsaktivitäten dieses Clusters besteht in der Untersuchung der Dekomposition von Geschäftsprozessen in einzelne Prozessschritte, welche mittels komponentenbasierter Technologien in verteilten Umgebungen realisiert werden können.

1.2.2 Problemstellung

Um Geschäftsprozesse der Finanzindustrie optimieren zu können, müssen diese zunächst klar definiert werden. Hierzu wird der Gesamtprozess in einzelne Prozessbausteine zerlegt, die jeweils eine gewisse Funktionalität für den Gesamtprozess erbringen. Solche Prozessbausteine können mithilfe von Web Services realisiert und dezentral, z.B. durch externe Web Service Provider, zur Verfügung gestellt werden. Mittels Standards zur Orchestrierung von Geschäftsprozessen aus Web Services, wie z.B. BPEL4WS, können die verteilten Prozessbausteine wieder zu einem Gesamtgeschäftsprozess zusammengefügt werden.

[1] Siehe auch: http://www.efinancelab.de/
[2] Siehe auch: http://www.kom.e-technik.tu-darmstadt.de/

Eine solche Orchestrierung des Geschäftsprozesses ist jedoch statisch, da hierbei konkrete Web Services zur Realisierung einer Funktionalität angegeben werden. Um Geschäftsprozesse jedoch möglichst optimal gestalten zu können, wäre es wünschenswert zur Laufzeit den am besten geeigneten Web Service zur Erbringung einer bestimmten Funktionalität dynamisch einbinden zu können. Dies ermöglicht es, jeden Prozessschritt durch einen optimal geeigneten Web Service ausführen zu lassen und damit den Gesamtprozess als Ganzes in seiner Ausführung zu optimieren.

Hierbei ist es nötig bei der Ausführung eines Geschäftsprozesses vor dem Aufruf eines jeden Web Services zu prüfen, ob alternative Web Services zur Realisierung derselben Funktionalität bekannt sind und aus der Menge der Alternativen den besten Web Service auszuwählen. Da die Funktionalität der alternativen Web Services gleich ist, werden Dienstgütekriterien als Differenzierungsmerkmal verwendet. Mittels einer Präferenzstruktur auf Dienstgütebasis wird definiert, welches der am besten geeignete Web Service zur Erbringung einer Funktionalität ist, so dass zur Laufzeit aus der Menge von Alternativen dynamisch der Web Service eingebunden und ausgeführt werden kann, bei dem die Übereinstimmung zur definierten Präferenzstruktur am größten ist. So könnte für einen bestimmten Prozessschritt jeweils der günstigste bekannte Web Service gleicher Funktionalität verwendet werden, oder der schnellste, oder beliebig gewichtete Präferenzen bezüglich solcher Dienstgütekriterien.

Da Dienstgüte jedoch weder von den Basisstandards der Web Services, noch von Standards zur Orchestrierung von Geschäftsprozessen unterstützt werden, ergaben sich im Rahmen der Forschungstätigkeiten u.a. folgende zu lösende Problembereiche:

Welche Dienstgütekriterien für Web Services existieren und welche hiervon müssen berücksichtigt werden? Wie können diese Dienstgütekriterien beschrieben werden? In welcher Form kann eine Zusicherung von Dienstgütekriterien durch einen Web-Service-Provider erfolgen? Woher stammt das Wissen, welche Web Services momentan für die Nutzung durch einen Geschäftsprozess zur Verfügung stehen? Wie kann ermittelt werden welche dieser Web Services die gleiche Funktionalität zur Verfügung stellen? Wie können für einzelne Funktionalitäten des Geschäftsprozesses Präferenzstrukturen auf Dienstgütebasis definiert werden? Wie kann eine Bewertung der alternativen Web Services gemäß der definierten Präferenzstruktur erfolgen, so dass der am besten geeignete Web Service aus der Menge der Alternativen ermittelt werden kann?

Neben diesen grundlegend zu klärenden Fragen, ergaben sich weitere Problembereiche durch betriebliche Erfordernisse. Aufrufe von Web Services müssen durch eine Accounting-Komponente protokolliert werden, um Erfolg und Misserfolg solcher Aufrufe feststellen und die Abrechnung der Nutzungsentgelte durch die Web-Service-Provider kontrollieren zu können. Erbrachte Dienstgütekriterien bei der Ausführung der Web Services müssen gemessen werden um die Einhaltung der Dienstgüteversprechen der Web-Service-Provider kontrollieren zu können.

Darüber hinaus ergeben sich Anforderungen, welche die Nutzbarkeit, bzw. die Leichtigkeit des Einsatzes der zu erstellenden Infrastruktur betreffen. Die Infrastruktur sollte ihre Aufgaben so generisch wie möglich erfüllen. Änderungen an den zu verwendenden Web Services sollten nicht nötig sein, da diese von externen Web-Service-Providern stammen auf welche oftmals kein direkter Einfluss genommen werden kann. Auch Änderungen an bereits definierten Geschäftsprozessen

in BPEL4WS sollten nach Möglichkeit vermieden oder zumindest auf ein minimales Maß reduziert werden. Da also weder die Geschäftsprozesse maßgeblich, noch die beteiligten Web Services überhaupt verändert werden sollten, galt es eine infrastrukturelle Zwischenschicht zwischen Geschäftsprozessen und verwendeten Web Services zu konstruieren, welche in der Lage ist die bereits beschriebenen Anforderungen der dienstgütebasierten Auswahl von Web Services zu erfüllen.

Alle diese Anforderungen konnten im Rahmen der Forschungsarbeiten gelöst und in dem entwickelten System WSQoSX (Web Service Quality of Service Architectural Extension) umgesetzt werden.

1.3 Aufbau des Technical Reports

Nachdem im ersten Kapitel in den behandelten Themenkomplex eingeführt und die Zielsetzung des Technical Reports erläutert wurde, werden im zweiten Kapitel die nötigen Grundlagen für eine eingehendere Betrachtung des behandelten Themas erläutert. Im dritten Kapitel wird anschließend die konzeptionelle Funktionsweise des erarbeiteten Softwaresystems WSQoSX beschrieben, bevor im vierten Kapitel die Architektur und Implementierungsdetails des Systems erörtert werden. Im fünften Kapitel wird eine abschließende Zusammenfassung des Technical Reports gegeben.

2 Grundlagen

Bevor in den nachfolgenden Kapiteln auf die konkret entwickelte dienstgüteunterstützende Web-Service-Architektur für flexible Geschäftsprozesse eingegangen wird, werden in diesem Kapitel wichtige Grundlagen für das weitere Verständnis geschaffen. Zunächst werden Merkmale einer „Service Oriented Architecture“ (SOA) beschrieben und die Technologie der Web Services vorgestellt, mit deren Hilfe eine SOA aufgebaut werden kann. Zum tieferen Verständnis der Web-Service-Technologie werden die Basisstandards dieser Technologie (WSDL, SOAP und UDDI) erläutert. Anschließend wird der Standard BPEL4WS dargestellt, der eine Orchestrierung von Prozessen aus Web Services gestattet. Das Grundlagenkapitel schließt mit der Betrachtung von Dienstgüte, relevanten Dienstgütekriterien für Web Services und Möglichkeiten der Integration von Dienstgüte in Web Services ab.

2.1 Service Oriented Architecture (SOA)

Die „Service Oriented Architecture“ (SOA) ist ein Systemarchitektur-Konzept, welches den Aspekt der Serviceorientierung besonders hervorhebt. Anders als klassische Softwarearchitekturen, die eine komplette Systemstruktur beschreiben, beschränkt sich die SOA auf den Bereich der Bereitstellung fachlicher Dienste und Funktionalitäten in Form von Services, vornehmlich zum Zwecke der Applikationsintegration. Hierbei wird eine zusätzliche Schicht, der Service-Layer, über der eigentlichen Softwareinfrastruktur eingeführt, der bestimmte Funktionalitäten der vorhandenen Software in der Form von Services in einer Art und Weise zur Verfügung stellt, die es ermöglicht diese Funktionalität über Netzwerkkommunikation aufzurufen [83].

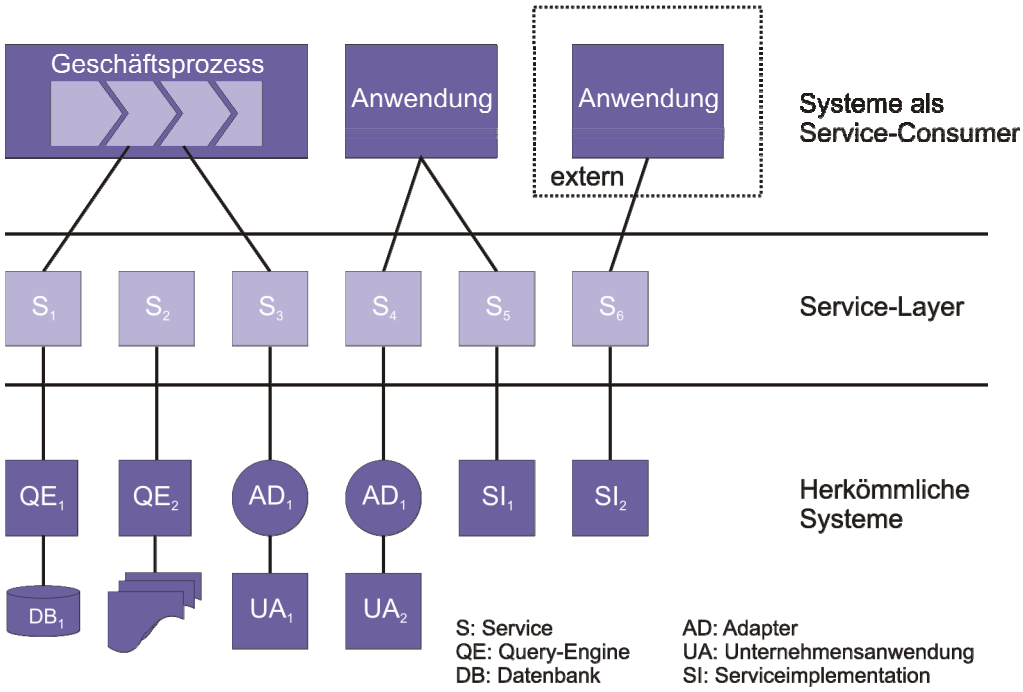

Abbildung 1 - Einführung eines Service-Layers

Jeder Service bietet hierbei seine Funktionalität nach außen durch eine Schnittstelle an. Die Schnittstelle eines Services beschreibt in einer Schnittstellenbeschreibungssprache die Operationen, welche der Service anbietet und wie diese aufgerufen werden können. Eine Adressierung des Service erfolgt durch einen Service-Endpunkt, welcher in seiner Ausprägung spezifisch zu der jeweils genutzten Netzwerkinfrastruktur ist (z.B. eine URL[3] in einem auf Internetstandards basierenden Netz). Für den Nutzer ist ein Service ein Endpunkt in einem Netz, an den er gemäß der Schnittstellenspezifikation Nachrichten senden und empfangen kann. Die Implementierung eines Services bleibt hierbei völlig verborgen. Ein Service könnte daher sowohl durch ein modernes objektorientiertes „Enterprise Java Bean“[4] (EJB) in einem J2EE-Container[5] auf einem Windows-Server bereitgestellt werden oder durch ein Legacy-System[6] in Form eines 30 Jahre alten COBOL-Programms[7] auf einem Mainframe-System[8].

Eine SOA basiert auf der Interaktion zwischen drei verschiedenen Rollen: Service-Provider, Service-Consumer und Service-Broker.

[3] Ein “Uniform Resource Locator” (URL) [16] [31] ist eine spezialisierte Form eines “Uniform Resource Indentifier” (URI) [15] und definiert eine Ressource in einem Netzwerk über einen Zugriffsmechanismus.
[4] Standardisierte Art einer Softwarekomponente, die in einer Architektur verwendet werden kann, welche die Spezifikationen der “Java 2 Platform, Enterprise Edition” (J2EE) erfüllt [48] [72].
[5] Kontext eines Anwendungsservers, der den Anforderungen der J2EE-Spezifikation genügt.
[6] Bezeichnung für bereits vorhandene, meist technologisch, architekturell oder anwendungstechnisch veraltete Systeme.
[7] COBOL (Common Business Oriented Language) bezeichnet eine an der Batch-Verarbeitung orientierte Programmiersprache, die seit den 60er Jahren eingesetzt wird. Bis zum heutigen Tag wird COBOL vor allem auf Mainframe-Systemen eingesetzt.
[8] In den 70er Jahren eingeführter Begriff für Großrechnersysteme, um sie von den damals aufkommenden kleineren Rechnersystemen (Minicomputer) abzugrenzen.

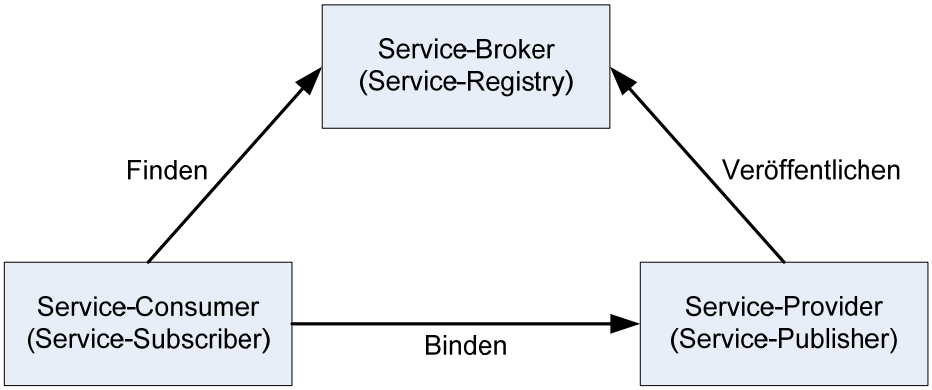

Abbildung 2 - Rollen in einer SOA (in Anlehnung an [83], S. 39)

Der Service-Provider implementiert einen Service und stellt ihn im Netzwerk zur Verfügung. Um die Verfügbarkeit eines Services anderen Teilnehmern im Netz zu signalisieren, kann der Service-Provider seinen Service bei einem oder mehreren Service-Brokern mit der Schnittstellenbeschreibung und weiteren Metadaten anmelden. Der Service-Consumer ist der Nutzer eines Services, der gemäß der Schnittstellenbeschreibung mit dem Service-Endpunkt im Netzwerk kommuniziert um die Funktionalität des Services in Anspruch zu nehmen. Ist ihm kein entsprechender Service bekannt, so kann er bei einem Service-Broker nach einem passenden Service suchen und diesen dynamisch einbinden. Der Service-Broker agiert als Vermittler zwischen Service-Consumer und Service-Provider. Er legt die vom Service Provider publizierten Services in einem Repository ab und bietet den Service-Consumern über eine Schnittstelle Suchmöglichkeiten für publizierte Services an. Der Service-Broker ist ein optionales Element in einer SOA. Sind den Service-Consumern alle für sie relevanten Services bereits bekannt und soll keine dynamische Suche und Einbindung von Services erfolgen, so kann auf einen Service-Broker als Vermittler verzichtet werden.

Insgesamt ergeben sich so eine Reihe von Merkmalen für eine SOA:

- Basis einer SOA bildet eine lose Kopplung zwischen Service-Provider und Service-Consumer durch Nachrichtenaustausch.
- Ein Service ist netzwerkadressierbar und bietet Lokationstransparenz.
- Ein Service kann dynamisch lokalisiert und eingebunden werden.
- Jeder Service wird als ein eigenständiges Modul gekapselt und bereitgestellt.
- Definierte Operationen einer Serviceschnittstelle sind meist grob granular.
- Durch Servicekomposition (Service-Assembly) können neue Services aus vorhandenen Services konstruiert werden.
- Durch den Einsatz standardisierter, interoperabler Netzwerkprotokolle, Transportprotokolle und Nachrichtenformaten kann ein hohes Maß von Interoperabilität innerhalb einer SOA erreicht werden.

Gerade die letztgenannte Eigenschaft, die Fähigkeit der Erreichung einer hohen Interoperabilität, prädestiniert eine SOA für den Einsatz zum Zwecke der EAI. Durch den Zugriff auf externe Netzwerke und die Verwendung externer Service-Broker ist der Einsatz einer SOA nicht nur auf eine Unternehmung beschränkt. Hierdurch können Services externer Service-Provider und damit externer Unternehmen in die eignen Geschäftsprozesse eingebunden werden.

Auch wenn eine SOA prinzipiell mittels beliebigen Technologien realisiert werden kann, so hat sich doch die Web-Service-Technologie zu einer der Kerntechnologien dieser Architekturform entwickelt und soll im Folgenden näher erläutert werden.

2.2 Web Services

Web Services sind wiederverwendbare, selbstbeschreibende, gekapselte Software-Komponenten, die eine Schnittstelle anbieten, über die ihre Funktionalität entfernt aufgerufen und die lose durch den Austausch von Nachrichten miteinander gekoppelt werden. Zur Erreichung universeller Interoperabilität werden für die Kommunikation die herkömmlichen Kanäle des Internets verwendet [7]. Die Grundlage der Web-Service-Technologie bilden drei, auf XML (Extensible Markup Language) [22] basierende Standards: WSDL (Web Service Description Language), SOAP (Simple Object Access Protocol) und UDDI (Universal Description, Discovery and Integration). Mit WSDL wird die Schnittstelle eines Web Services spezifiziert, mittels SOAP werden Nachrichten von und an Web Services übermittelt und mit UDDI, einem Standard für Verzeichnisdienste für Web Services, können Web Services aufgefunden werden.

Die Web-Service-Technologie erfüllt damit alle Anforderungen an den Aufbau einer SOA, wie sie im vorangegangenen Abschnitt beschrieben wurde. Da Web Services auf offenen XML-Standards basieren und zur Kommunikation die herkömmlichen Internetprotokolle verwenden, eignen sie sich in hervorragender Art und Weise zur Realisierung einer in hohem Maße interoperabelen SOA. Aus diesem Grund spielen Web Services im Bereich der EAI und auch der Orchestrierung von unternehmensübergreifenden Geschäftsprozessen eine wichtige Rolle.

Zum besseren Verständnis der Web-Service-Technologie werden die drei bereits erwähnten Basisstandards WSDL, SOAP und UDDI im Folgenden erläutert.

2.2.1 Web Service Description Language (WSDL)

Web Services bieten ein hohes Maß an Flexibilität und Wiederverwendbarkeit, da sie interoperabel und plattformunabhängig sind. Damit ein Web-Service-Consumer dynamisch auf einen Web Service zugreifen kann, muss die Schnittstelle des Web Services in einer standardisierten Form ebenso plattformunabhängig beschrieben und bereitgestellt werden, sowie dynamisch verarbeitet werden können. Eine solche Schnittstellenbeschreibung muss auf möglichst hoher Abstraktionsebene definieren, welche Methoden der Web Service anbietet, welche Parameter diese erwarten und über welche Transportmechanismen und Nachrichtenformate die Kommunikation stattfindet. Zu diesem Zwecke wurde die „Web Service Description Language“ (WSDL) [26] entwickelt. Die WSDL spezifiziert wie die Schnittstelle eines Web Services mithilfe eines XML-Dokuments, dem WSDL-Dokument des Web Services, beschrieben werden kann. Da XML-Dokumente plattformunabhängig verarbeitet werden können und der Aufbau eines WSDL-Dokuments standardisiert ist, können Web-Service-Consumer mithilfe eines WSDL-Dokuments dynamisch plattformspezifischen Code (z.B. Proxy-Klassen[9]) zum Zugriff auf einen Web Service erstellen und damit dynamisch Funktionalitäten durch Web Services einbinden.

[9] Ein Proxy (der englische Begriff für Stellvertreter) bezeichnet ein Computersystem oder –programm, welches als Zwischeninstanz zwischen Client und Server agiert und hierbei Anfragen des

Gemäß der WSDL-Spezifikation ist ein Web Service eine Menge abstrakter Netzwerkendpunkte, die Daten in Form von Nachrichten austauschen. Die Operationen des Web Services werden daher durch WSDL in Form von Ein- und Ausgabenachrichten definiert. Jede Ein- und Ausgabenachricht wird mit ihren Bestandteilen für Aufruf- und Rückgabeparameter und den jeweils verwendeten Datentypen definiert. Um den Web Service konkret aufrufen zu können wird darüber hinaus definiert wie diese Nachrichten kodiert und zwischen Web-Service-Consumer und Web-Service-Provider transportiert werden.

Die Definitionen werden in einem XML-Dokument, dem WSDL-Dokument, in XML-Syntax notiert. Das WSDL-Dokument wird hierbei hauptsächlich mittels sechs wichtiger Elemente aufgebaut: `definitions`, `types`, `message`, `portType`, `binding` und `service`. Der hierarchische Aufbau eines WSDL-Dokuments stellt sich grob wie folgt dar:

```
<?xml version="1.0"?>
<definitions>

      <types>
            <schema></schema>
      </types>

      <message>
            <part></part>
      </message>

      <portType>
            <operation>
                  <input></input>
                  <output></output>
            </operation>
      </portType>

      <binding>
            <operation>
                  <input></input>
                  <output></output>
            </operation>
      </binding>

      <service>
            <port></port>
      </service>

</definitions>
```

Abbildung 3 - Struktureller Aufbau eines WSDL-Dokuments (in Anlehnung an [83], S. 60)

Mittels der Elemente `types`, `message` und `portType` wird die abstrakte Schnittstelle des Web Services beschrieben, also seine Operationen und die verwendeten Datentypen. Die Beschreibung geschieht hierbei unabhängig vom verwendeten Transportprotokoll oder einer konkreten Implementierung. Mittels der

Clients an den Server weiterleitet, eingehende Antworten zurückübermittelt und dabei gewisse Zusatzfunktionalitäten übernimmt, wie z.B. Zwischenspeicherung, Filterung oder Vorverarbeitung.

Elemente `binding` und `service` wird beschrieben wie die Operationen konkret aufzurufen sind. Dabei beschreiben sie über welche URI und welche Protokolle der Web Service erreichbar ist und wie die Daten für einen Aufruf serialisiert und kodiert werden müssen. Die Bedeutung der einzelnen Elemente sei nun im Folgenden kurz erläutert.

Das Wurzelelement `definitions` dient als Container für die Servicebeschreibung und der Definition der beteiligten XML-Namensräume.

Das optionale `types`-Element definiert Datentypen, die in Nachrichten verwendet werden sollen. Da Datentypen in Nachrichten mittels „XML Schema“[10] definiert werden, müssen hier nur solche Datentypen definiert werden, die später in Nachrichten verwendet werden sollen, aber nicht bereits als Basistypen in XML Schema zur Verfügung stehen.

Mit `message`-Elementen werden Nachrichten beschrieben, die zwischen Web-Service-Consumer und Web-Service-Provider beim Aufruf einer Operation eines Web Services ausgetauscht werden. Hierbei erfolgt die Beschreibung der Methodenparameter, bzw. Rückgabewerte inklusiver ihrer Datentypen in `parts`-Elementen. Nachrichten, die einen Funktionsaufruf darstellen, definieren mit ihren `parts`-Kindelementen die benötigten Methodenparameter. Nachrichten, die eine Antwort auf einen Funktionsaufruf darstellen, definieren mit ihren `parts`-Kindelementen hingegen die von ihnen gelieferten Rückgabewerte.

Im `portType`-Element werden mittels `operations`-Elemente alle vom Web Service nach außen angebotenen Operationen beschrieben. Hierbei können für jede Operation maximal zwei Nachrichten, die zuvor mittels eines `message`-Elements definiert wurden, angegeben werden. Je nachdem ob diese Nachrichten in einem `input`- oder `output`-Element deklariert werden, können verschiedene Aufrufmechanismen für die Operation beschrieben werden. Auf die einzelnen Aufrufmechanismen soll hier jedoch nicht näher eingegangen werden.

Zu jeder Operation die im `portType`-Element deklariert wurde, wird mittels eines `bindings`-Element definiert in welchem Nachrichtenformat die Nachrichten ausgetauscht werden, mit welcher Kodierung die Daten in den Nachrichten kodiert werden und mit welchem Transportprotokoll die Nachrichten ausgetauscht werden. Diese Angaben werden als Bindung einer Operation bezeichnet. Zu einer Operation lassen sich hierbei mehrere Bindungen angeben, wie z.B. Bindungen an SOAP, HTTP[11] und MIME[12], die bereits explizit in der WSDL-Spezifikation definiert sind. WSDL ist jedoch erweiterbar, so dass hier auch andere Bindungen denkbar wären.

Für jede so definierte Bindung wird in einem `service`-Element definiert, an welche konkrete Netzwerkadresse ein Web-Service-Consumer seine gemäß der Bindung

[10] “XML Schema” ist ein auf XML basierender Standard des “World Wide Web Consortium” (W3C) zur Definition von XML-Dokumentstrukturen und XML-Datentypen [30] [78] [17] [46].

[11] Das “Hypertext Transfer Protocol” (HTTP) ist ein zustandsloses Request-Response-Protokoll zum Datenaustausch, welches auf dem TCP/IP-Protokollstapel aufsetzt und den Standard zum Austaussch von Web-Seiten im “World Wide Web” (WWW) darstellt [14] [32].

[12] "Multipurpose Internet Mail Extensions" (MIME) stellt einen Standard zur Kodierung von Inhalten in Nachrichten, wie z.B. E-Mails dar. Durch MIME kann der Typ der übermittelten Daten, sowie ein Kodierverfahren der Daten zum Zwecke der Übertragung spezifiziert werden [34] [35] [61] [36] [33].

erstellten Nachrichten zum Aufruf des Web Services senden kann. Hierbei wird die Netzwerkadresse durch ein `port`-Element definiert. Durch die Angabe mehrerer `port`-Elemente können mehrere Netzwerkadressen definiert werden, unter denen der Web Service erreicht werden kann.

2.2.2 Simple Object Access Protocol (SOAP)

In der WSDL-Beschreibung eines Web Services (siehe Kapitel 2.2.1) wird der Aufbau und Inhalt von Nachrichten, mit denen der Web Service angesprochen werden kann, nur abstrakt, unabhängig von einem konkreten Kommunikationsprotokoll beschrieben. Damit jedoch ein Web-Service-Consumer konkrete Nachrichten zum Aufruf des Web Services erstellen kann, muss der genaue Aufbau dieser Nachrichten bekannt sein. Das „Simple Object Access Protocol“ (SOAP) [60] [40] [41] [43] ist das Standard-Kommunikationsprotokoll der Web-Service-Technologie und spezifiziert den konkreten Aufbau von Nachrichten für die Kommunikation mit Web Services.

SOAP-Nachrichten sind XML-Dokumente, die in ihrem Aufbau und Inhalt der SOAP-Spezifikation genügen und zwischen Web-Service-Consumer und Web-Service-Provider zum Aufruf von Methoden eines Web Services ausgetauscht werden. Die Verwendung von XML-Nachrichten als Mittel zur Kommunikation bietet den großen Vorteil der Interoperabilität bei der Nachrichtenverarbeitung, da XML sich als plattformübergreifender Dokument-Standard etabliert hat und die Verarbeitung von XML-Dokumenten unabhängig von bestimmten Anwendungen, Programmiersprachen, Technologien und Betriebssystemen erfolgen kann.

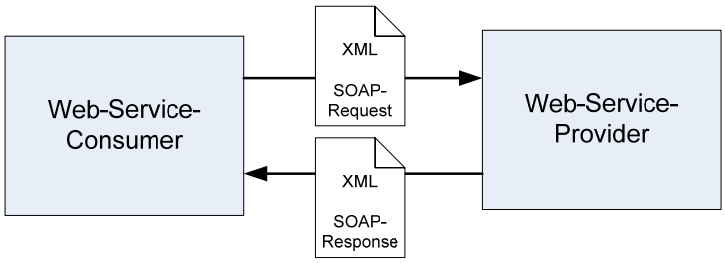

Abbildung 4 - Nachrichtenaustausch bei einem SOAP-RPC

Eine mögliche und häufige Anwendung von SOAP ist die Realisierung eines Methodenaufrufes in Form eines „Remote Procedure Calls“ (RPC). Ein RPC ermöglicht den Aufruf entfernter Methoden über ein Netzwerk gemäß des Client-Server-Paradigmas. Hierzu sendet der Client eine Nachricht mit Informationen über die aufzurufende Methode und die nötigen Parameterwerte an den Server, welcher die Methode ausführt und die Ergebniswerte wiederum in eine Nachricht verpackt und an den Client zurücksendet. Bei einem SOAP-RPC wird dieser Mechanismus dazu verwendet eine Methode eines Web Services aufzurufen. Der Web-Service-Consumer ist hierbei der Client und der Web-Service-Provider der Server (siehe Abbildung 4). Die aufzurufende Methode und die zugehörigen Parameterwerte werden in einem SOAP-Request und die Ergebniswerte in einem SOAP-Response als XML-Dokumente gemäß dem SOAP-Standard kodiert. Das Transportprotokoll, mit welchem die Nachrichten zwischen den Kommunikationspartnern transportiert werden, ist hierbei durch SOAP nicht fest vorgegeben. Meist wird HTTP als

Transportprotokoll verwendet, jedoch können hierzu auch Protokolle wie SMTP[13], FTP[14] oder auch TCP[15] direkt verwendet werden.

Eine SOAP-Nachricht besteht aus drei Hauptelementen: SOAP-Envelope, SOAP-Header und SOAP-Body.

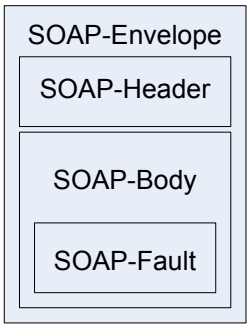

Abbildung 5 - Struktureller Aufbau einer SOAP-Nachricht (entnommen aus [83], S. 46)

Der SOAP-Envelope ist das Wurzelelement einer SOAP-Nachricht und enthält alle weiteren Elemente. Der SOAP-Header ist optionaler Bestandteil einer SOAP-Nachricht und dient der Übermittlung von Verwaltungsinformationen an den endgültigen Empfänger der Nachricht, oder an Zwischenstationen (Intermediaries) welche die Nachricht auf ihrem Weg zum Empfänger durchläuft. Der SOAP-Body enthält die eigentlichen Nutzdaten der Nachricht und kann im Falle eines SOAP-Response auch ein SOAP-Fault-Element enthalten, welches, vergleichbar eines Exception-Objekts[16] einer Programmiersprache, Fehler bei der Verarbeitung des vorangegangenen SOAP-Requests signalisiert und näher beschreibt.

Sollen mit einer SOAP-Nachricht auch Daten übermittelt werden, so müssen diese, bevor sie in die XML-Nachricht integriert werden können, vom Sender in ein XML-Format serialisiert werden. Damit der Empfänger der Nachricht erkennen kann, wie er die Daten aus dem XML-Format wieder deserialisieren und in ein für ihn verarbeitbares Format konvertiert kann, gibt der Sender den von ihm zur Kodierung der Daten verwendeten Standard mittels des so genannten Encodings an. Das Encoding wird üblicherweise im SOAP-Envelope deklariert und gilt danach für die ganze SOAP-Nachricht.

Der optionale SOAP-Header wird zur Übermittlung von zusätzlichen Informationen über die eigentliche Nachricht verwendet. Beispiele hierfür sind z.B. Daten die im Rahmen einer Authentifizierung, des Transaktionsmanagements oder der Kontextverwaltung benötigt werden. Mittels eines speziellen Attributs (`mustUnderstand`) kann der Sender in einem Element des SOAP-Headers angeben, ob es zwingend für die Verarbeitung der SOAP-Nachricht nötig ist, dass der Empfänger dieses Element auch verstehen kann. Ist dies zwingend nötig und der

[13] Das "Simple Mail Transfer Protocol" (SMTP) ist ein auf dem TCP/IP-Protokollstapel basierendes Protokoll zur Übermittlung von E-Mails [52].
[14] Das "File Transfer Protocol" (FTP) ist ein auf dem TCP/IP-Protokollstapel basierendes Protokoll zur Übermittlung von Dateien [65].
[15] Das "Transmission Control Protocol" (TCP) ist ein zuverlässiges, verbindungsorientiertes Transportprotokoll zur Übertragung beliebiger Daten [64].
[16] Eine Exception (engl. für Ausnahme) signalisiert das Auftreten bestimmter Programmzustände, meist Fehlerzustände, und stellt hierdurch die Basis für eine strukturierte Fehlerbehandlung zur Verfügung.

Empfänger kann dieses Element nicht verstehen, bzw. verarbeiten, so muss er in seiner Antwort mittels eines SOAP-Fault-Elements darauf hinweisen. Da es möglich ist, dass eine SOAP-Nachricht nicht direkt zum Empfänger gesendet wird, sondern auf seinem Weg dorthin andere zwischengeschaltete Systeme (Intermediaries) passiert, kann mittels des `actor`-Attributs zu jedem Header-Element spezifiziert werden, durch welches System es verarbeitet werden soll. Nachdem ein solches Zwischen-System die an es gerichteten Header-Elemente verarbeitet hat, entfernt es diese, bevor es die SOAP-Nachricht an die nachgelagerten Systeme weitergibt.

Im SOAP-Body sind die eigentlichen Nutzdaten für den Empfänger untergebracht. Im Falle des Aufrufs von Methoden eines Web Services sind dies abstrakte Beschreibungen der aufzurufenden Methode und ggf. serialisierte Parameterwerte, bzw. Ergebniswerte. Gelingt es dem Empfänger nicht die im Body gelieferten Informationen zu verarbeiten, oder tritt während der Verarbeitung der Informationen ein Fehler auf, so informiert der Empfänger den Sender in seiner Antwort mittels eines speziellen Fault-Elements über das Auftreten des Fehlers und liefert hierin noch eine Beschreibung des Fehlers.

2.2.3 Universal Description, Discovery and Integration (UDDI)

Die Abkürzung UDDI steht für „Universal Description, Discovery and Integration“ [10] und bezeichnet den von Microsoft, IBM und Ariba ins Leben gerufenen De-facto-Standard für Web-Service-Verzeichnisse. UDDI stellt neben WSDL und SOAP die dritte Basistechnologie der Web Services dar.

Damit in einer SOA die von Service-Providern angebotenen Services dynamisch von Service-Consumern aufgefunden und eingebunden werden können, ist es erforderlich für die Service-Consumer eine Möglichkeit zu schaffen, nach für sie geeigneten Services zu suchen. In einer SOA übernimmt daher ein Service-Broker den Betrieb einer Service-Registry, in welcher Service-Provider Informationen über ihre angebotenen Services hinterlegen können, und in der Service-Consumer nach geeigneten Services suchen können. Gerade im Bereich der B2B-Kollaboration[17] ist es wichtig externen Unternehmen Informationen über die aufrufbaren Services der eigenen SOA zukommen zu lassen, damit eine Integration der Services in unternehmensübergreifende Geschäftsprozesse möglich wird.

Der UDDI-Standard spezifiziert ein solches Verzeichnis für Web Services. Bestandteile der Spezifikation sind die Beschreibung der verwendeten Datenstrukturen und der API[18] zum Abfragen und Publizieren im Verzeichnis. Das UDDI-Verzeichnis ist hierbei selbst eine Web-Service-Anwendung und stellt seine Funktionalität als Web Service über das Kommunikationsprotokoll SOAP im Netzwerk zur Verfügung. Alle APIs und Datenstrukturen sind hierbei in XML, bzw. im XML-Schema-Format spezifiziert.

[17] B2B ist die Abkürzung des Begriffs Business-To-Business und bezeichnet Kommunikations- und Geschäftsbeziehungen zwischen Unternehmen, um sie von Beziehungen zu Privatpersonen, Mitarbeitern oder der öffentlichen Verwaltung abzugrenzen.

[18] Durch ein “Application Programming Interface” (API) definiert eine Software Schnittstellen zum Zugriff auf ihre Funktionalität durch andere Programme.

Ein UDDI-Verzeichnis bietet die Möglichkeit vielfältige Informationen aus verschiedenen Bereichen zu den Web Services abzulegen und abzufragen. Die drei wichtigsten Bereiche lassen sich hierbei wie folgt grob charakterisieren:

- White Pages: Name, Beschreibung und sämtliche Kontaktinformationen der Unternehmen die Web Services im Verzeichnis publiziert haben.
- Yellow Pages: Kategorisierungsinformationen der Unternehmen und der durch sie angebotenen Web Services. Die Kategorisierung ermöglicht es das UDDI-Verzeichnis wie ein Branchenbuch oder die Gelben Seiten zu benutzen.
- Green Pages: Technische Dokumentation zur Beschreibung der veröffentlichten Web Services. Hier können z.B. WSDL-Dokumente referenziert werden, welche die Schnittstelle des Web Services beschreiben.

Das hierbei zur Ablage der Informationen verwendete Datenmodell besteht im Wesentlichen aus fünf Datenstrukturen:

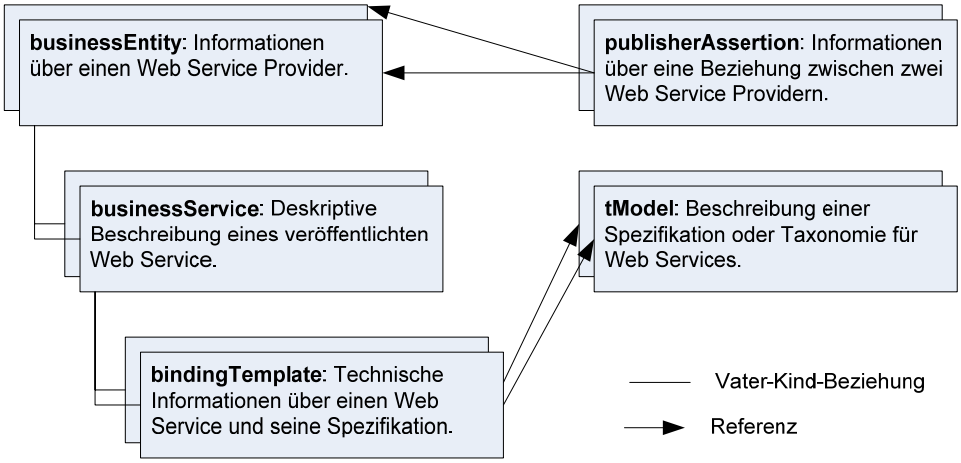

Abbildung 6 - Wesentliche Datenstrukturen in UDDI (in Anlehnung an [83], S. 71)

Eine `businessEntity` repräsentiert ein Unternehmen, bzw. eine reale Organisation, die Web Services im UDDI-Verzeichnis veröffentlicht hat. Stehen zwei Organisationen hierbei in einer gewissen Beziehung zueinander, weil sie z.B. gemeinsam an einem elektronischen Marktplatz teilnehmen, so kann diese Beziehung mittels einer `publisherAssertion` modelliert werden. Diese beschreibt die Art der Beziehung und referenziert die beteiligten Organisationen. Die von einer Organisation veröffentlichten Web Services werden durch `businessServices` repräsentiert und der jeweiligen `businessEntity` in einer Vater-Kind-Beziehung zugeordnet. Da ein Web Service auf verschiedene Arten und Weisen, z.B. über verschiedene Protokolle, aufgerufen werden kann, können einem `businessService` wiederum mehrere `bindingTemplates` zugeordnet werden, welche jeweils die technischen Details einer Zugangs-, bzw. Aufrufmöglichkeit des Web Services beschreiben. Hierbei wird unter einem Attribut `accessPoint` die Netzwerkadresse für den Zugriff auf den Web Service angegeben und unter der Datenstruktur `tModelInstanceDetails` eine technische Beschreibung des Web Services abgelegt. In der technischen Beschreibung wird u.a. auf Servicetypbeschreibungen (`tModels`) referenziert. Eine Servicetypbeschreibung kann beispielsweise eine Einordnung des Web Services in eine Taxonomie beinhalten oder eine Schnittstellenspezifikation durch ein referenziertes WSDL-Dokument. Auf die Kategorisierungs- und

Spezifizierungsmöglichkeiten durch `tModels` soll hier jedoch nicht näher eingegangen werden.

Damit Service Consumer das UDDI-Verzeichnis auf standardisierte Weise durchsuchen und Service Provider ihre Web Services auf standardisierte Weise in das UDDI-Verzeichnis eintragen können, beinhaltet der UDDI-Standard auch die Spezifikation einer entsprechenden API. Die API ist hierbei in Form von XML-Segmenten spezifiziert, die zum Aufruf einer Operation in eine SOAP-Nachricht verpackt an das UDDI-Verzeichnis gesendet werden müssen. Die gesamte API setzt sich hierbei aus zwei einzelnen APIs zusammen, der Inquiry-API und der Publisher-API. Das UDDI-Verzeichnis verwendet zum Empfangen von Nachrichten für beide APIs unterschiedliche Netzwerkendpunkte.

Die wichtigsten Operationen, die durch die Inquiry-API spezifiziert werden, sind Requests zum Auffinden von Informationen im UDDI-Verzeichnis. Hierbei dienen `find`-Requests zur Suche nach bestimmten UDDI-Datenstrukturen wie `businessEntity`, `businessService`, `bindingTemplate` und `tModel`. Nach dem Auffinden von Datenstrukturen können `get`-Requests zum Auslesen von Details aus diesen Datenstrukturen verwendet werden. Für das Suchen in einem UDDI-Verzeichnis wird keine Authentifizierung benötigt.

Die Publisher-API spezifiziert Requests zum Anlegen, Aktualisieren und Löschen von Datenstrukturen. Für das Anlegen und Aktualisieren von Datenstrukturen werden `save`-Requests und für das Löschen von Datenstrukturen `delete`-Requests verwendet. Um das UDDI-Verzeichnis vor unauthorisierten Veränderungen zu schützen ist bei der Benutzung der Publisher-API eine Authentifizierung nötig. Bevor Operationen der Publisher-API verwendet werden können, muss unter Angabe der Benutzeridentität ein Sicherheitstoken (`authToken`) beim UDDI-Verzeichnis angefordert werden. Das Sicherheitstoken muss beim Aufruf von Operationen der Publisher-API in die erzeugten SOAP-Nachrichten eingefügt werden und wird durch das UDDI-Verzeichnis zur Prüfung der Zugriffsrechte bei der Ausführung von Operationen verwendet.

Für den Zugriff auf UDDI-Verzeichnisse existieren verschiedene plattformspezifische APIs, welche die Kommunikation mit einem UDDI-Verzeichnis kapseln, sodass sich Entwickler nicht direkt mit XML-, bzw. SOAP-Dokumenten und deren Übermittlung und Empfang auseinandersetzen müssen. JAXR[19] (Java API for XML Registries) stellt z.B. eine solche API für die Java-Plattform dar.

2.3 Business Process Execution Language (BPEL)

Durch eine SOA auf Web-Service-Basis können zwar plattformunabhängig Funktionalitäten über ein Netzwerk zur Verfügung gestellt werden, jedoch bietet eine solche SOA keine direkte Möglichkeit Web Services zur Durchführung ganzer Prozesse zu verbinden. Um Geschäftsprozesse definieren und ausführen zu können muss das Zusammenspiel der Partner, ihrer Web Services und der Nachrichtenfluss zwischen diesen beschrieben werden können.

[19] Siehe auch: http://java.sun.com/xml/jaxr/

Zur standardisierten Definition von aus Web Services orchestrierten Geschäftsprozessen wurden im Jahre 2001 zwei konkurrierende Standards entwickelt: XLANG von Microsoft und die Web Services Flow Language (WSFL) von IBM. Um einen einheitlichen Standard zu schaffen arbeiteten Microsoft und IBM mit BEA, SAP und Siebel Systems zusammen und entwickelten aus den Ideen beider, vormals konkurrierenden, Standards einen neuen, gemeinsamen Standard, die „Business Process Execution Language for Web Services“ (BPEL4WS). Die Spezifikation von BPEL4WS erschien im Juli des Jahres 2002 in seiner ersten Version 1.0 und im Mai des Jahres 2003 in der Version 1.1 [6]. Die Version 1.1 der Spezifikation wurde zur Standardisierung und Weiterentwicklung an OASIS (Organization for the Advancement of Structured Information Standards) übergeben. Seitdem entwickelte sich BPEL4WS zu einem De-facto-Standard zur Definition von Geschäftsprozessen auf Web-Service-Basis und wird durch Produkte der wichtigsten Global Player unterstützt (z.B. Microsoft BizTalk Server, IBM WebSphere Process Choreographer, Oracle BPEL Process Manager, SAP NetWeaver, BEA WebLogic). OASIS beschloss im September des Jahres 2004 den Nachfolger von BPEL4WS 1.1 als WS-BPEL 2.0 zu bezeichnen, d.h. BPEL4WS zukünftig in „Web Services Business Process Execution Language“ (WS-BPEL) umzubenennen. Zum Zeitpunkt der Erstellung dieses Technical Reports lagen lediglich Entwürfe, jedoch noch keine endgültige Version von WS-BPEL 2.0 vor, wodurch die Version BPEL4WS 1.1 als die zur Zeit weiterhin gültige Spezifikation angesehen wird und im Rahmen dieses Technical Reports weiterhin BPEL4WS als Bezeichnung der Sprache verwendet wird.

Konkret handelt es sich bei BPEL4WS um die Spezifikation einer auf XML basierenden Sprache zur formalen Notation von Geschäftsprozessen in der Form von XML-Dokumenten. BPEL4WS gestattet die Definition von abstrakten, als auch von ausführbaren Geschäftsprozessen. Letztere werden im Groben durch die Festlegung ihres Workflows und der hieran beteiligten Aktivitäten, die durch Web Services gekapselt realisiert werden, definiert. Ein so definierter Geschäftsprozess kann mittels einer BPEL4WS-fähigen Business Process Engine (BPE) ausgeführt werden. Hierbei wird der Geschäftsprozess durch die BPE wiederum als Web Service zur Verfügung gestellt, wodurch Geschäftsprozesse im Rahmen von Aktivitäten anderer Geschäftsprozesse als Web Service eingebunden werden können. Die Schnittstellen des Geschäftsprozesses und der an ihm beteiligten Web Services werden in Form von WSDL-Dokumenten beschrieben.

Der Vorteil der Verwendung eines Standards für die Geschäftsprozessmodellierung liegt in der Interoperabilität der erstellten Geschäftsprozesse in Bezug auf die verschiedensten Plattformen und darauf realisierter BPEs. BPEL4WS kann daher nicht nur als Ausführungssprache, sondern auch als Austauschformat für die Definition von Geschäftsprozessen auf Web-Service-Basis betrachtet werden.

Durch BPEL4WS wird eine Reihe von Problembereichen, die sich in Bezug auf die Definition und Ausführung von Geschäftsprozessen ergeben, adressiert. Dies sind u.a.:

- Beschreibung des Kontrollflusses, z.B. mittels Konstrukten wie serielle / parallele Ausführung, Verzweigung, Schleife und Synchronisation (paralleler Aktivitäten).
- Beschreibung des Datenflusses, z.B. welche Daten aus welchen Nachrichten und/oder Zustandsinformationen wie in welche neue Nachrichten verpackt an welchen Web Service gesendet werden.

- Zuordnung von Nachrichten zu einem gemeinsamen Kontext, sowohl bei synchroner als auch langläufiger asynchroner Kommunikation.
- Verwaltung von Zustandsinformationen zu jedem Kontext.
- Fehlerbehandlungsmechanismen mit der Möglichkeit differenziert auf das Auftreten von Fehlern zu reagieren.
- Transaktionseigenschaften[20], wie z.B. Kompensationsmechanismen für den Fall des Abbruchs oder des Scheiterns einer bestimmten Menge von Operationen.

Wie konkret Geschäftsprozesse mittels BPEL4WS beschrieben werden können und in welcher konkreten Art und Weise BPEL4WS hierbei die oben genannten Problembereiche adressiert, soll im Rahmen dieses Grundlagenkapitels nicht weiter erläutert werden. Für weitergehende Informationen zum Thema BPEL4WS sei auf die Spezifikation [6], sowie auf entsprechende Literatur, wie z.B. [49] oder [84] verwiesen.

2.4 Service Level Agreement (SLA)

Im Bezug auf die Kooperation von Unternehmen in Geschäftsprozessen ist es für die beteiligten Partner relevant Vereinbarungen über die jeweils angebotene, bzw. geforderte Leistung zu treffen. Zur Definition von Leistungsvereinbarungen werden „Service Level Agreements“ (SLAs) [76] verwendet. SLAs sind formale Dokumente mit vertragsähnlichem Charakter, die eine Vereinbarung zwischen dem Anbieter und dem Nutzer einer Leistung darstellen und die Art und Weise der Leistungserbringung sowie relevanter Rahmenbedingungen festlegen, d.h. einen Service-Level definieren.

Bei SLAs im Allgemeinen handelt es sich meist um natürlichsprachige Dokumente für deren Aufbau keine genauen Vorgaben existieren. Je nach Art der Leistung, der Ausgestaltung der Leistungsinanspruchnahme und des Umfangs der Vereinbarungen können sich SLAs im Aufbau stark unterscheiden und mehr oder weniger komplex strukturiert sein. Typischerweise erfolgt die Strukturierung von SLAs ähnlich eines Vertrags mittels Klauseln, die jeweils einzelne Aspekte der Vereinbarung regeln. Übliche Klauseln betreffen hierbei die Qualität mit der eine Leistung erbracht werden muss, die Voraussetzungen und Rahmenbedingungen unter denen die Qualität garantiert wird, wie die Einhaltung der Qualitätsmerkmale überprüft wird, Konsequenzen die aus einem Über- oder Unterschreiten der vereinbarten Qualität erwachsen, Supportvereinbarungen, Haftungsbestimmungen, Nutzungsentgelte, Kündigungsbedingungen, u.ä..

SLAs stehen nicht zwangsläufig in Verbindung mit Leistungen die in elektronischen Geschäftsprozessen angeboten oder genutzt werden. SLAs können beispielsweise auch dazu verwendet werden um zu definieren in welcher Qualität Kunden in einem ausgelagerten Call-Center durch den Insourcer bedient werden müssen. Hierbei könnte z.B. die maximale Wartezeit eines Kunden bis zum Gespräch mit einem Call-Center-Agent als ein Qualitätsmerkmal der zu erbringenden Leistung definiert werden. In einem SLA zur Ausgestaltung der Lieferkonditionen eines Spediteurs

[20] Eine Transaktion ist eine Menge von Aktivitäten die zu einer kohärenten Arbeitseinheit zusammengefasst werden und für die gewisse Eigenschaften garantiert werden, z.B. dass diese entweder als Ganzes gelingen oder als Ganzes scheitern. Eine umfassende Einführung in das Thema Transaktionsmanagement kann z.B. [39] entnommen werden.

könnten als Qualitätsmerkmale Maximalzeiten für die Dauer bis zur Abholung der Ware und der anschließenden Auslieferung an den Kunden festgelegt werden.

Bei der Vereinbarung von Leistungsmerkmalen in SLAs ist zu beachten, dass diese nach Möglichkeit kontrollierbar sein sollten. Ist eine Kontrolle der Leistungsvereinbarung nicht möglich, so können Verstöße gegen die SLA nicht festgestellt werden und der Nutzen einer SLA beschränkt sich auf eine Absichtserklärung auf Vertrauensbasis.

Im Fokus elektronischer Geschäftsprozesse können SLAs dazu verwendet werden Leistungsvereinbarungen bezüglich der verwendeten Dienste einer SOA, wie sie durch Web Services zur Verfügung gestellt werden, zu spezifizieren. Die Verwendung von SLAs in der Form natürlichsprachiger Dokumente wäre hier zwar prinzipiell möglich, gestaltet sich aber durch den Mangel an automatischer Verarbeitbarkeit und dem Zwang zur menschlichen Interaktion als nachteilig. Sind an elektronischen Geschäftsprozessen eine Vielzahl von verschiedenen Web Services beteiligt, die dynamisch zu Geschäftsprozessen orchestriert werden sollen, so wird der Einsatz natürlichsprachiger SLAs aufgrund des hohen Bearbeitungs- und Koordinationsaufwandes nahezu unmöglich.

Damit SLAs effizient und effektiv für Web Services eingesetzt werden können, müssen diese in Form elektronischer Dokumente mit einem syntaktisch und semantisch standardisierten Format zur Verfügung stehen. Ebenso wie Web Services in einer standardisierten Art und Weise plattformunabhängig und interoperabel verwendet werden können, muss die Zusicherung von Leistungsmerkmalen für diese ebenso in einer standardisierten Art und Weise plattformunabhängig und interoperabel möglich sein, sofern die Leistungsvereinbarung eine sinnvolle, konsistente Ergänzung der bisherigen Web-Service-Technologie darstellen soll.

Eine standardisierte, elektronische Form von SLAs schafft die nötigen Voraussetzungen für

- das automatisierte Erstellen eigener und das automatisierte Auslesen fremder SLAs,
- die automatisierte Aushandlung von SLAs als Übereinkunft zwischen den Anforderungen des Nutzers und dem Angebot des Anbieters,
- die automatisierte Kontrolle der spezifizierten Leistungsmerkmale,
- die automatisierte Vergleichbarkeit von verschiedenen Web Services anhand ihrer Leistungsmerkmale, inklusive der Bildung eines Rankings als Maß der Übereinstimmung mit individuellen Nutzerpräferenzen.

Wesentlicher Bestandteil elektronischer SLAs für Web Services ist die Beschreibung der Leistungsmerkmale [47] [27]. Die Leistungsmerkmale eines Web Services werden hierbei hauptsächlich durch seine Dienstgüte definiert (siehe Kapitel 2.5). Der Begriff der Dienstgüte lässt sich jedoch nicht einheitlich definieren, wodurch sich die verschiedensten Dienstgütekriterien zur Definition einer Dienstgüte heranziehen lassen. Ein elektronisches SLA für Web Services sollte daher eine flexible Spezifikation von Dienstgüte mittels nahezu beliebigen Dienstgütekriterien ermöglichen [69]. Andererseits wird jedoch durch den Verzicht auf eine standardisierte Menge von Dienstgütekriterien zur Spezifikation von Dienstgüte die effektive Vergleichbarkeit von Dienstgüte verhindert. Die Dienstgüte, und damit die Leistungsmerkmale eines Web Services, werden erst dann wieder vergleichbar, wenn

sich die Partner auf eine einheitliche Menge von Dienstgütekriterien zur Spezifikation der Dienstgüte einigen.

Es existieren Ansätze die explizit elektronische SLAs für Web Services modellieren und eine Umgebung zur Kontrolle der Einhaltung dieser SLAs zur Verfügung stellen. IBM veröffentlichte hierzu ein spezialisiertes Framework[21], welches die XML-basierte Sprache „Web Service Level Agreement“ (WSLA) [57] zu Spezifikation von SLAs verwendet. HP entwickelte mit dem „Web Services Management Network“ (WSMN) [58] ein Overlay-Netzwerk[22] zum Management von Web Services, welches mit der „Web Services Management Language“ (WSML) ebenfalls eine XML-basierte Sprache zur Spezifikation von SLAs verwendet. Tosic [80] entwickelte mit der „Web Service Offerings Infrastructure“ (WSOI) eine Infrastruktur, in der zum Management von Web Services „Service Offerings“ verwendet werden, die in der XML-basierten Sprache „Web Service Offerings Language“ (WSOL) formuliert werden. Zur Verwaltung durch die WSOI können einem Web Service eine oder mehrere „Service Offerings“ zugeordnet werden, die hierbei eine Serviceklasse und deren Eigenschaften modellieren.

Allen Ansätzen ist gemein, dass sie zum Management von Web Services auch Aspekte der Dienstgüte einzelner Web Services modellieren und spezifizieren. Bevor jedoch weiter auf Beschreibungsmöglichkeiten der Dienstgüte von Web Services eingegangen werden kann, soll im Folgenden zunächst der Begriff der Dienstgüte sowie Dienstgütekriterien für Web Services erläutert werden.

2.5 Dienstgüte

Für den Begriff Dienstgüte (englisch „Quality of Service“, Abkürzung „QoS“) existiert keine einheitliche Definition. Je nachdem welche Art von Dienst unter welchen Aspekten betrachtet wird, setzt sich die Beurteilung der Güte dieses Dienstes aus anderen Teilaspekten zusammen. Auch wenn im Folgenden nur Dienste in Bezug auf verteilte Systeme und sie verbindende Netzwerke betrachtet werden, ergibt sich keine wesentliche Vereinfachung. Betrachtet man z.B. das Schichtenmodell eines Netzwerkes, wie das ISO-OSI-Refernzmodell[23], so werden in jeder Schicht eine Reihe von Diensten definiert, die wiederum durch die jeweils höhere Schicht verwendet werden. Hierbei werden durch die Dienste jeweils schichtspezifische Aufgaben adressiert und die Güte eines Dienstes daher in jeder Schicht anhand unterschiedlicher Kriterien beurteilt. So werden beispielsweise an einen Dienst der Bitübertragungsschicht (Übertragung einzelner Bits über ein Übertragungsmedium) andere Anforderungen gestellt als an einen Dienst der Transportschicht (Aufbau und Trennen von Verbindungen, Zerlegen von Daten in Transporteinheiten, Flusssteuerung der Datenströme) oder der Anwendungsschicht (z.B. Versenden von E-Mails oder Übertragen von Dateien).

Das CCITT[24] (Commité Consultatif Internationale Télégrafique et Téléfonique) bemühte sich im Jahre 1988 daher im Zusammenhang mit der ISDN-Technologie um

[21] Siehe auch: http://www.research.ibm.com/wsla/
[22] Ein Overlay-Netzwerk ist ein Netzwerk, welches auf einem anderen Netzwerk aufsetzt.
[23] Siehe [77], Seite 45 ff
[24] Das CCITT wurde 1992 unter dem Namen “ITU Telecommunication Standardization Sector” (ITU-T) in die “International Telecommunication Union” (ITU) integriert. Siehe auch: http://www.itu.int/ITU-T/

eine möglichst allgemeine Definition von Dienstgüte und entwickelte ein Dienstgütemodell, das die technischen Aspekte eines Dienstes ebenso berücksichtigt wie die Verfügbarkeit und die Bedienung der Endgeräte. In diesem Zusammenhang definierte das CCITT *Dienstgüte* als „The collective effect of service performances which determine the degree of satisfaction of a user of the service." [1]. Hierbei ist unter „user" nicht notwendigerweise ein menschlicher Benutzer zu verstehen, sondern auch nicht-menschliche Benutzer wie elektronische Geräte oder Software.

Im Folgenden soll sich das Grundverständnis von Dienstgüte an dieser allgemeinen Definition des CCITT orientieren. Da sich die genauen Kriterien an Dienstgüte jedoch aus dieser allgemeinen Definition nicht erschließen, müssen diese jeweils im Kontext, in welchem der Begriff Dienstgüte verwendet wird, definiert werden. Im Kontext von Netzwerken und Diensten die Netzwerke benutzen, bezeichnet der Begriff Dienstgüte zumeist nur technische Aspekte der Diensterbringung, wie z.B. die Übertragungsgeschwindigkeit in einem Netzwerk. Diese Art der Dienstgüte kann als *technische Dienstgüte*, oder *Dienstgüte im engeren Sinne* bezeichnet werden, wohingegen unter die *Dienstgüte im weiteren Sinne* auch nicht-technische Aspekte von Dienstgüte fallen können, wie z.B. die Reputation des Dienstanbieters oder das Entgelt, welches bei der Nutzung eines Dienstes an den Anbieter zu entrichten ist.

Dienstgüte in Netzwerken

In paketvermittelnden Netzwerken wie dem Internet werden Daten zur Übertragung in einzelne Transporteinheiten zerlegt und in Form einzelner, adressierter Pakete versendet. Hierbei passieren die Pakete auf dem Weg vom Sender zum Empfänger eine Vielzahl von Zwischensystemen, die jeweils durch Netzwerke mit den unterschiedlichsten Eigenschaften verbunden sind. So unterscheiden sich Teilstrecken z.B. hinsichtlich ihrer möglichen Transportgeschwindigkeit und Systeme bezüglich ihrer verfügbaren Puffergrößen und der Verarbeitungsgeschwindigkeit. Auf dem Weg durch diese Zwischenstationen können Pakete verloren gehen, sie können verzögert weitergeleitet werden, Paketinhalte können korrumpiert werden, die Empfangsreihenfolge der Pakete kann sich von der Sendereihenfolge unterscheiden und viele weitere Ereignisse, welche die Datenübertragung beeinflussen, können auftreten.

Die Güte eines Dienstes, der Daten in einem solchen Netz überträgt, kann anhand einer Vielzahl von Kriterien beurteilt werden, wie z.B. Latenz, Datendurchsatz, Fehlerrate, Unterstützung von Sicherheit und Nutzungskosten. Je nachdem zu welchem Zweck ein Dienst verwendet wird, sind die Dienstgütekriterien zur Beurteilung der Güte verschieden zu gewichten. Die Übertragung von Multimediadaten in Form von Video-Streaming erfordert eine möglichst konstante Datenrate, was jedoch wiederum bei der Übertragung von E-Mails nicht wichtig ist.

Dienstgütetypen

Eine sinnvolle Nutzung mancher Dienste ist nur möglich, falls sichergestellt werden kann, dass sich gewisse Dienstgüteparameter in bestimmten Grenzen bewegen. Beispielsweise stellt die Telefonie gewisse Anforderungen an eine niedrige Latenz, einen möglichst konstanten Datendurchsatz und eine niedrige Fehlerrate. Zu große Verzögerungen oder die häufige Unterbrechung der Übertragung würden kein befriedigendes Telefonieren ermöglichen. Manche Dienste bieten daher die

Möglichkeit gewisse Dienstgüteparameter mit dem Nutzer zu vereinbaren. Die Spezifikation der Dienstgüteparameter bestimmt dabei den Dienstgütetyp. Prinzipiell lassen sich hierbei drei Dienstgütetypen unterscheiden (vgl. [74], S. 242ff): Garantierte, vorhersagbare und Best-Effort-Dienste.

Garantierte Dienste stellen Garantien für Dienstgüte in der Art zur Verfügung, dass sie garantieren, dass Dienstgüteparameter während der Dienstnutzung einen festen Wert besitzen (z.B. möglicher Datendurchsatz 100 MBit/s) oder sich in einem durch Grenzwerte definierten Intervall bewegen (z.B. Latenzzeit von nicht weniger als 10ms und nicht mehr als 100ms).

Ein *vorhersagbarer Dienst* (historischer Dienst) stellt abgeschwächte Garantien auf Basis seines Verhaltens in der Vergangenheit aus. Die vorhergesagten Dienstgüteparameter sind Schätzungen aufgrund des vergangenen Verhaltens, die der Dienst mit einer gewissen Wahrscheinlichkeit auch in Zukunft erfüllen kann. Ein Beispiel hierfür ist ein Dienst, der in seiner Vergangenheit eine durchschnittliche Bandbreite von 100 MBit/s zur Verfügung stellte und daher die Bandbreite von 100 MBit/s als Dienstgüteparameter anbietet. Er kann jedoch nicht garantieren, dass die Bandbreite nicht zeitweise unter 100 MBit/s abfallen kann. Er garantiert lediglich, dass die Bandbreite im Durchschnitt in etwa 100 MBit/s betragen wird, falls seine Umgebungsparameter sich so wie in der Vergangenheit gestalten.

Best-Effort-Dienste sind solche, die entweder keine Garantien ermöglichen oder auf partiellen Garantien basieren. Sie versuchen die Dienstgüte lediglich „so gut es geht“ zu erbringen. Die meisten verfügbaren Netzwerkprotokolle stellen lediglich Best-Effort-Dienste zur Verfügung.

Durchsetzung von Dienstgüte

Damit garantierte Dienste die, den Nutzern zugesicherte, Dienstgüte sicherstellen können, müssen sie ihre verwendeten Ressourcen verwalten. Basiert beispielsweise ein Dienst auf der Nutzung einer Netzwerkverbindung über die 100 MBit/s übertragen werden können, so kann der Dienst nicht mehr als fünf Nutzern gleichzeitig eine Übertragung auf dieser Netzwerkverbindung mit jeweils 20 MBit/s garantieren. Der Dienst benötigt Informationen über die ihm zur Verfügung stehenden Ressourcen und die Garantien, welche ihm diese Ressourcen zusichern können. Anhand dieser Informationen ist der Dienst in der Lage die Ressourcen aktiv zu verwalten und entsprechende Garantien an seine Nutzer weiterzugeben. Im genannten Beispiel könnte der Dienst einen potentiellen sechsten Nutzer ablehnen oder aber seine Ressourcen durch den Aufbau einer neuen Netzwerkverbindung mit mindestens 20 MBit/s erweitern.

Nutzt ein garantierter Dienst wiederum Dienste, z.B. einer niedrigeren Schicht eines Schichtmodells, so müssen auch diese Dienste garantierte Dienste sein, für die rekursiv wiederum das gleiche gelten muss. Die ursprünglichen Dienstgüteanforderungen müssen durch den Dienst in Anforderungen an alle genutzten Dienste übersetzt und durch diese garantiert werden. Nur so ist der Dienst in der Lage die ursprünglich garantierte Dienstgüte erbringen zu können. In einem Schichtenmodell spricht man auch von der Übersetzung der Dienstgüte von Schicht zu Schicht ([74], S. 259ff). Existiert in der Kette der genutzten Dienste ein Dienst, der nicht in der Lage ist Dienstgüte zu garantieren, so kann ein Dienst, der diesem Dienst

in der Nutzungskette nachfolgt, keinerlei Dienstgütegarantien mehr aussprechen. Das Durchsetzen von Dienstgüte ist daher in der Praxis oftmals problematisch, da Dienste verschiedenster Schichten hieran beteiligt sind und sich diese Dienste über eine große Anzahl beteiligter Komponenten und Systeme verteilen können.

Im Bereich von Netzwerken existieren verschiedenste Ansätze Dienstgüte durchzusetzen. Die einfachste, aber auch kostspieligste, besteht darin eine hinreichende Anzahl von Ressourcen angemessener Kapazität bereit zu halten um die Dienstgüte selbst bei Spitzenbelastungen garantieren zu können (Overprovisioning). Problematisch ist hierbei jedoch die Vorhersage zukünftiger Spitzenbelastungen. Effizientere Verfahren basieren auf Reservierungsmechanismen und schalten der Dienstnutzung eine Reservierungsphase vor, in welcher der Dienst zunächst nötige Ressourcen reserviert und dem Nutzer nur dann die Nutzung gestattet, falls alle für die Erbringung der Dienstgüte nötigen Ressourcen verfügbar sind. Ein Beispiel für eine auf Reservierungen basierenden Architektur ist z.B. die IntServ-Architektur[25], welche RSVP[26] als Reservierungsprotokoll verwendet. Abgeschwächte Verfahren garantieren eine reservierte Dienstgüte nur mit einer gewissen Wahrscheinlichkeit, z.B. nur bei durchschnittlicher Belastung des Netzes, wodurch die Kapazitäten der verfügbaren Ressourcen besser ausgelastet werden können. Andere Verfahren beschränken sich darauf die verfügbaren Ressourcen ungleich an verschiedene Nutzer zu verteilen. Hierzu bilden sie Dienstgüteklassen und stellen je mehr Ressourcen zur Verarbeitung von Nutzerdaten zur Verfügung, desto höher die reservierte Dienstgüteklasse des Nutzers bewertet wird (z.B. gemäß Gold-Silber-Bronze-Priorisierung). Darüber hinaus existieren viele weitere Verfahren die hier aufgrund ihrer Fülle nicht erwähnt werden können. Ebenso soll auf eine genaue Darstellung der bereits genannten Verfahren im Rahmen dieses Grundlagenkapitels verzichtet werden. Für eine detailliertere Betrachtung des Themengebietes sei z.B. auf [45] verwiesen.

2.5.1 Dienstgütekriterien für Web Services

In etablierten Bereichen in denen garantierte Dienste schon seit langem eine wichtige Rolle spielen, wie z.B. Kommunikationsnetzwerke oder Multimedia- und Echtzeitanwendungen, ist Dienstgüte ein gut untersuchtes Thema. Wichtige Dienstgütekriterien zur Spezifikation von Dienstgüte wurden definiert, Beschreibungsmöglichkeiten für Dienstgüte geschaffen, Abbildungsmechanismen von Dienstgüte auf andere Schichten entwickelt und Verfahren zur Sicherstellung von Dienstgüte durch die Verwendung von spezialisierten Ressourcen-Managern entwickelt.

Dienstgüte in verteilten Systemen, wie sie z.B. in einer SOA auf Web-Service-Basis realisiert werden, ist im Gegensatz dazu noch ein relativ schlecht untersuchtes Gebiet in dem sich noch keine Standards etablieren konnten. Web Services können eine Fülle von Funktionalitäten bereitstellen und für die verschiedensten Zwecke in den verschiedensten Umgebungen eingesetzt werden. Zusätzlich ist es möglich Web Services unter einer Vielzahl von Aspekten zu betrachten, wie z.B. unter dem

[25] IntServ (Integrated Services) bezeichnet eine Architektur [20] zur Durchsetzung von Dienstgüte [71] [87] in einem, auf Internetstandards basierenden Netz. Siehe hierzu z.B. [74], S. 488ff oder [45], S. 17ff.

[26] RSVP (Resource Reservation Protocol) [21] [88] ist das in der IntServ-Architektur verwendete Protokoll zur Reservierung von Verbindungen, bzw. Datenpfaden, mit bestimmten Dienstgüteeigenschaften.

Gesichtpunkt ihrer Funktionalität, ihrer Eigenschaften als Softwarekomponente oder ihrer Eigenschaften als Komponente eines Netzwerkes. Es ist daher nicht möglich eine bestimmte Menge von Dienstgütekriterien für die Spezifikation der Dienstgüte eines Web Services für jeden Einsatzzweck zu definieren. Dementsprechend viele Varianten von Dienstgütekriterien für Web Services lassen sich in der Literatur finden (vgl. z.B. [53], [38], [66], [59]).

In [53] werden folgende relevanten Bereiche von Dienstgütekriterien für Web Services identifiziert:

- Performanz
- Zuverlässigkeit
- Skalierbarkeit
- Kapazität
- Robustheit
- Ausnahmebehandlung
- Akkuratesse
- Integrität
- Erreichbarkeit
- Verfügbarkeit
- Interoperabilität
- Sicherheit
- Netzwerkbezogene Kriterien

Performanz
Die Performanz eines Web Services repräsentiert, wie schnell eine Anfrage durch den Web Service verarbeitet wird. Tabelle 1 können einige Dienstgütekriterien entnommen werden, die der Performanz zuzuordnen sind.

Kriterium	Beschreibung
Durchsatz	Anzahl von Anfragen die in einem bestimmten Zeitintervall verarbeitet werden.
Antwortzeit	Zeitspanne die vom Absenden einer Anfrage an den Web Service bis zum Erhalt einer Antwort vergeht. Diese Zeitspanne beinhaltet die, vor allem durch das Netzwerk determinierte, Latenz und die Ausführungszeit des Web Services.
Latenz	Die Zeitspanne, die zwischen dem Absender einer Anfrage und dem Empfang einer Antwort zwangsläufig vergehen muss und unabhängig von der Größe der Anfrage und der Anzahl der auszuführenden Operationen durch den Web Service ist. Üblicherweise wird diese Zeitspanne von der Dauer determiniert, die eine Nachricht minimaler Länge benötigt um über das Netzwerk vom Nutzer zum Web Service und wieder zurück transportiert zu werden (minimale Round-Trip-Time).
Ausführungszeit	Zeit die ein Web Service zur Abarbeitung einer Anfrage benötigt. Sie wird gemessen vom Zeitpunkt des Erhalts der Anfrage durch den Web Service bis zum Absenden der Antwort an den Empfänger durch den Web Service.

Tabelle 1 - Dienstgütekriterien der Performanz

Zuverlässigkeit
Die Zuverlässigkeit eines Web Services repräsentiert die Fähigkeit des Web Services seine Funktionalität unter gegebenen Bedingungen für die Dauer eines bestimmten Zeitintervalls erfolgreich erbringen zu können. Zuverlässigkeit schließt ein, dass während dieses Zeitintervalls sichergestellt ist, dass ankommende Anfragen angenommen und verarbeitet werden, sowie korrekte Antworten erzeugt und in der richtigen Reihenfolge erfolgreich ausgeliefert werden. Als Indikator mangelnder Zuverlässigkeit kann das Auftreten von Fehlern angesehen werden. Als Fehler sind hierbei falsche Rückgabewerte, Ausfallzeiten, aber auch verloren gegangene Anfragen oder Antworten anzusehen.

Skalierbarkeit
Die Skalierbarkeit eines Web Services drückt die Fähigkeit aus, die Verarbeitungskapazität des Web Services so erhöhen zu können, dass eine gesteigerte Anzahl von Nutzeranfragen erfolgreich verarbeitet werden kann. Die Skalierbarkeit bezieht sich auf die Anzahl der durchführbaren Operationen und Transaktionen durch einen Web Service und steht in Zusammenhang mit dessen Performanz. Um eine gute Skalierbarkeit gewährleisten zu können muss der Web-Service-Provider in der Lage sein, die Ressourcen, der an der Ausführung des Web Services beteiligten Systeme, erweitern und die Last unter diesen Systemen verteilen zu können.

Kapazität
Die Kapazität bezeichnet die Anzahl von Anfragen die durch einen Web Service gleichzeitig und mit einer bestimmten Performanz verarbeitet werden können.

Robustheit
Mittels der Robustheit wird der Grad an Funktionstüchtigkeit ausgedrückt, die ein Web Service selbst beim Auftreten ungültiger, unvollständiger oder widersprüchlicher Anfragen erbringt. Ein Web Service sollte auch unter solchen Umständen dazu in der Lage sein deterministisch zu reagieren.

Ausnahmebehandlung
Die Unterstützung von Mechanismen zur Ausnahmebehandlung soll es einem Web Service ermöglichen auf unvorhergesehene Ausnahmen zu reagieren und seine Funktionstüchtigkeit zu erhalten. Da der Entwickler eines Web Services nicht alle möglichen Situationen, die beim Betrieb eines Web Services entstehen können, vorhersagen kann, ist die Verwendung von Mechanismen zur Ausnahmebehandlung ein hilfreiches Mittel die Robustheit des Web Services zu erhöhen.

Akkuratesse
Die Akkuratesse repräsentiert die durch einen Web Service verursachte Fehlerrate. Web Services sollten nach Möglichkeit eine hohe Akkuratesse, d.h. eine niedrige Rate an auftretenden Fehlern aufweisen.

Integrität
Die Integrität eines Web Services beschreibt die Fähigkeit unautorisierten Zugriff oder unautorisierte Veränderung von Daten und Programmen zu verhindern. Hierbei können zwei Arten der Integrität unterschieden werden: Datenintegrität und transaktionelle Integrität. Datenintegrität verhindert die unerlaubte Veränderung von übertragenen Daten, z.B. in Transitknoten. Transaktionelle Integrität schließt die Datenintegrität ein, sichert jedoch darüber hinaus die Konsistenz der Daten über eine

ganze Transaktion hinweg. Eine Transaktion[27] ist eine Menge von Aktivitäten die zu einer kohärenten Arbeitseinheit zusammengefasst werden und entweder als ganzes gelingen oder als ganzes scheitern. Im Erfolgsfalle überführt eine Transaktion ein System von einem konsistenten Zustand in einen anderen, wodurch die Integrität des Systems gewährleistet wird. Im Falle eines Fehlers oder gezielten Abbruchs wird die Transaktion zurückgesetzt (Rollback), wodurch das System in den zuletzt als konsistent angesehenen Zustand versetzt wird.

Erreichbarkeit
Unter der Erreichbarkeit eines Web Services wird seine Fähigkeit verstanden eingehende Anfragen von Clients anzunehmen und diese zu verarbeiten. Auch wenn das System, welches die Ausführungsumgebung eines Web Services beherbergt, funktionstüchtig und über das Netzwerk erreichbar ist, so ist es möglich, dass ein Web Service nicht erreichbar ist, z.B. wenn seine Implementierung in der Ausführungsumgebung durch das Eintreffen zu vieler Anfragen überlastet ist. Um eine hohe Erreichbarkeit zu sichern, sollten Web Services daher über eine gute Skalierbarkeit verfügen.

Verfügbarkeit
Die Verfügbarkeit eines Web Services bezeichnet die Wahrscheinlichkeit, mit der ein Web Service durch einen Client sofort in Anspruch genommen werden kann. Hierzu müssen das System und die Ausführungsumgebung, die den Web Service beherbergen, funktionstüchtig und über das Netzwerk erreichbar sein. Die Verfügbarkeit eines Web Services hängt eng mit dessen Zuverlässigkeit und Erreichbarkeit zusammen. Von besonderem Interesse im Zusammenhang mit der Verfügbarkeit ist die Time-to-Repair (TTR), welche die Zeit bezeichnet, die im Falle eines Ausfalls des Web Services benötigt wird um diesen wieder zur Verfügung zu stellen.

Interoperabilität
Web Services sollten plattformunabhängig und interoperabel sein. Die Realisierung eines Web Services sollte derart sein, dass Nutzer, welche einen Web Service verwenden wollen, nicht berücksichtigen müssen welche Programmiersprache zur Implementierung des Web Services verwendet wurde und in welcher Ausführungsumgebung in welcher Technologie auf welchem Betriebssystem dieser ausgeführt wird. Unter der Berücksichtigung der Interoperabilität ist Sorge dafür zu tragen, dass verwendete Nachrichtenformate und die Kodierung von Parameter- und Rückgabewerten nicht für eine spezifische Plattform oder Programmiersprache optimiert werden.

Sicherheit
Sicherheit, in Bezug auf Web Services, bezeichnet die Fähigkeit Sicherheitsmechanismen, gemäß den Sicherheitsbedürfnissen der an der Kommunikation beteiligten Partner, zur Verfügung stellen zu können. Bei der Übermittlung von Prozessdaten eines verteilten Geschäftsprozesses an die beteiligten Web Services über das öffentliche Internet ist die Bereitstellung grundlegender Sicherheitsmechanismen nahezu eine Voraussetzung ihrer Nutzung.

[27] Klassische Transaktionen sichern für alle gekapselten Operationen die sogenannten ACID-Eigenschaften zu, die jedoch bei verteilten Transaktionen in verteilten Umgebungen nur in abgeschwächter Form zugesichert werden können. Grundlagen zu Transaktionsmodellen und ihren Eigenschaften können z.B. [39] entnommen werden.

Sicherheitsmechanismen können hierbei eine Vielzahl von Aspekten adressieren, wie z.B. Authentifizierung, Autorisierung, Vertraulichkeit, Haftbarkeit, Rückverfolgbarkeit/Prüfbarkeit, Verschlüsselung und Unleugbarkeit.

Authentifizierung bezeichnet die Fähigkeit den Nutzer eines Web Services identifizieren zu können. Mittels der Autorisierung können Zugriffsberechtigungen auf Funktionen und Daten je nach identifiziertem Nutzer unterschiedlich festgelegt werden. Vertrauliche Daten sollten bei ihrem Austausch von und zu den autorisierten Nutzern entsprechend ihrer Vertraulichkeit übertragen und gehandhabt werden. Mittels der Verschlüsselung von Daten kann hierbei gewährleistet werden, dass nur der beabsichtigte Empfänger Daten entschlüsseln und verarbeiten kann. Ein Anbieter von Web Services sollte für die Funktionalität seines Web Services haftbar gemacht werden können. Voraussetzung hierfür ist die Rückverfolgbarkeit und Prüfbarkeit, die sicherstellt, dass Aufrufe von Web Services zurückverfolgt und die Ergebnisse des Aufrufs überprüft werden können. Mechanismen zur Unleugbarkeit garantieren, dass ein Nutzer nicht abstreiten kann einen Web Service aufgerufen, dessen Funktionalität beansprucht und entsprechende Reaktionen ausgelöst zu haben.

Netzwerkbezogene Kriterien
Um eine gewisse Dienstgüte für Web Services erreichen zu können, muss ein, auf der Web-Service-Anwendungsebene arbeitender Dienstgütemechanismus seine Dienstgüteanforderungen in Anforderungen an das verwendete Transportnetzwerk übersetzen und mit dessen Dienstgütemechanismen, wie z.B. IntServ/RSVP, DiffServ[28], MPLS[29], zusammenarbeiten können. Web Services kommunizieren ausschließlich mittels Nachrichten die über ein Netzwerk zwischen Web-Service-Consumer und Web-Service-Provider transportiert werden. Die Möglichkeit der Einflussnahme auf das zugrunde liegende Transportnetzwerk ist für die Dienstgüte von Web Services daher in besonderem Maße wichtig, da ihre Dienstgüte in vielerlei Hinsicht von den Eigenschaften des Transportnetzwerkes determiniert werden. So hängt z.B. die Geschwindigkeit und Fehleranfälligkeit der Nachrichtenübermittlung von Web Services hauptsächlich von den Eigenschaften des verwendeten Transportnetzwerks ab, die wiederum einen hohen Einfluss auf viele weitere Dienstgütekriterien von Web Services haben, wie z.B. die Performanz.

Preis
In Ergänzung zu [53] soll mit dem Preis noch ein sehr offensichtliches, jedoch betriebswirtschaftlich und nicht technisch motiviertes Dienstgütekriterium erwähnt werden. In einer Umgebung in der dieselbe Funktionalität durch mehrere Web Services zur Verfügung gestellt wird und die einzelnen Web-Service-Provider für die Erbringung der Funktionalität des Web Services unterschiedliche Nutzungsentgelte (Preise) verlangen, stellt der Preis und das dem Preis zugrundliegende Tarifmodell in der betrieblichen Praxis ein bedeutendes Auswahlkriterium für den konkret zu nutzenden Web Service dar. Der Preis, bzw. das Preis-Leistungs-Verhältnis kann der Dienstgüte im weiteren Sinne zugerechnet werden.

[28] DiffServ (Differentiated Services) [18] bezeichnet eine Architektur die Dienstgütemechanismen in einem Netzwerk zur Verfügung stellt, ohne hierbei (z.B. im Gegensatz zu IntServ) in jedem Transitknoten die Zustände aller über diesen Knoten geleiteten Datenpfade einzeln verwalten zu müssen.
[29] MPLS (Multiprotocol Label Switching) [68] bezeichnet einen dienstgüteunterstützenden Datentransportmechanismus, der unterhalb der IP-Schicht agiert und sowohl Daten für paket- als auch verbindungsorientierte Dienste transportieren kann.

2.5.2 Problembereiche bei der Integration von Dienstgüte in Web Services

Um Dienstgüte in Web Services unterstützen zu können müssen u.a. folgende Problembereiche beachtet werden:

- Die Definition und Spezifikation von Dienstgütekriterien.
- Die Definition und Spezifikation der Dienstgüte unter Verwendung definierter Dienstgütekriterien.
- Die Einführung von Mechanismen zum Abgleich von Dienstgüteanforderungen von Web-Service-Consumern (Nutzern) und Dienstgüteangeboten von Web-Service-Providern (Anbietern) zur Auswahl geeigneter Web Services.
- Die Entwicklung von Methoden zur Durchsetzung und Kontrolle vereinbarter Dienstgüte.

Definition und Spezifikation von Dienstgütekriterien

Die erste Voraussetzung, die zur Integration von Dienstgüte in Web Services zu schaffen ist, ist die Definition eines einheitlichen Vokabulars für diesen Themenbereich. Nur so kann im Rahmen einer Beschreibung von Dienstgüte und der Vereinbarung von Dienstgüte zwischen Anbieter und Nutzer eine Verständigungsmöglichkeit geschaffen werden. Je nach Einsatzzweck und der Art der zu formulierenden Dienstgüte müssen hierzu konkrete Dienstgütekriterien definiert werden, z.B. was genau durch das Dienstgütekriterium „Durchsatz“ ausgedrückt werden soll. Gegenstand solcher Definitionen muss insbesondere auch die Art und Weise der Spezifikation des jeweiligen Dienstgütekriteriums sein. Beispiele hierfür sind u.a. der Ausdruck mittels einer geeigneten Maßzahl (z.B. Preis „0,01 EUR/Aufruf“) oder einer Relation (z.B. Antwortzeit „< 5000 ms“), die Bekanntgabe eines Leistungsmerkmals durch sein Vorhandensein (z.B. SSL-Verschlüsselung „ja“) oder die Auflistung von Leistungsmerkmalen durch ihre Art (z.B. Authentifizierung „Plain Text, Kerberos, X.509“).

Definition und Spezifikation der Dienstgüte

Zur Definition einer Dienstgüte reicht die Spezifikation einzelner Dienstgütekriterien nicht aus. Einzelne Dienstgütekriterien müssen zur Definition einer Dienstgüte kombiniert werden können (z.B. „Durchsatz > 1000 Aufrufe/Tag und Antwortzeit < 5000 ms“). Hierzu ist ein Format zu definieren, welches je nach Einsatzzweck die spezifischen Anforderungen an die Ausdrucksmächtigkeit einer Dienstgütedefinition erfüllt und zur Spezifikation, bzw. Notation der Dienstgüte verwendet werden kann. Hierbei sollte versucht werden einen guten Kompromiss zwischen der Ausdrucksmächtigkeit und Flexibilität auf der einen und der Komplexität auf der anderen Seite zu finden. Ein Format zur Spezifikation einer Dienstgüte sollte zwar universell und flexibel für die verschiedensten Zwecke eingesetzt werden können, jedoch nicht zu komplex in Aufbau und Handhabung sein.

Neben der Spezifikation der eigentlichen Dienstgüte ist es für die meisten praxisrelevanten Einsatzsituationen notwendig zusätzliche Informationen mit der Dienstgüte zu verknüpfen. Hierbei kann es sich beispielsweise um Verwaltungsdaten (z.B. Angaben zur Identifikation des Anbieters) oder vertragliche Rahmendaten zur Erbringung der Dienstgüte (z.B. Gültigkeitsdauer des Dienstgüteangebotes, Nutzungsentgelte) handeln. Dokumente, die nicht nur die Dienstgüte, sondern auch

solche zusätzlichen Angaben enthalten, sollen im Folgenden als „Service Level Agreement“ (SLA) bezeichnet werden (vgl. Kapitel 2.4).

Die Spezifikation von Dienstgüte kann in einem eigenständigen SLA-Dokument erfolgen, oder als Erweiterung in bestehende Standards der Web-Service-Technologie integriert werden. Eine Erweiterung bestehender Standards ermöglicht es, die Informationen eines SLA über die herkömmlichen Mechanismen der Web-Service-Technologie zu verwalten, zu transportieren und zu verarbeiten. In Abhängigkeit des Standards, der um Informationen einer SLA erweitert wird, werden hierbei spezifische Einsatzzwecke adressiert. Durch die Erweiterung von WSDL-Dokumenten können die Informationen eines SLAs jenen Komponenten zur Verfügung gestellt werden, welche die Schnittstellenspezifikation eines Web Services verwalten, transportieren oder verarbeiten. Dadurch erhalten diese Komponenten zusätzlich zur Schnittstellenspezifikation des Web Services auch Informationen über die Dienstgüte und die Rahmenbedingungen der Nutzung. Eine Erweiterung der UDDI-Datenstrukturen um SLA-Informationen, ermöglicht es bereits bei der Suche nach Web Services nach gewissen Dienstgütekriterien zu selektieren. Erfolgt eine Anreicherung von SOAP-Nachrichten mit Informationen über eine gewünschte Dienstgüte, so können sich während des Aufrufs eines Web Services auch alle Zwischensysteme, die an der Verarbeitung der SOAP-Nachrichten beteiligt sind, an die geforderte Dienstgüte anpassen.

Abgleich von Dienstgüteanforderungen und Dienstgüteangeboten zur Auswahl geeigneter Web Services

Methoden zum Abgleich von Dienstgüte können zwischen den Systemen der Nutzer und Anbieter direkt realisiert werden oder unter der Verwendung spezialisierter Zwischeninstantzen (Service-Broker), wie z.B. einem erweiterten UDDI-Verzeichnis mit der Möglichkeit Web Services nach Dienstgütekriterien zu selektieren. Aufzurufende Web Services können hierbei einmalig a priori ermittelt werden, oder wiederkehrend dynamisch zur Zeit des Aufrufs.

Sollen Web Services statisch eingebunden werden, so können Nutzer von Web Services manuell oder semi-manuell nach Anbietern von Web Services mit entsprechend geeigneter Dienstgüte suchen, konkrete SLAs mit den Anbietern aushandeln und diese Web Services manuell einbinden.

Bei der dynamischen Einbindung von Web Services gemäß ihrer Dienstgüte ist ein solches Vorgehen jedoch aufgrund der nötigen manuellen Interaktion nicht mehr möglich. Hier muss eine Möglichkeit geschaffen werden einen Abgleich zwischen der durch den Nutzer geforderten und der durch die Anbieter zugesicherte Dienstgüte automatisch zur Laufzeit herbeizuführen. Hierzu müssen die Zusicherungen bezüglich der Dienstgüte und der Rahmenbedingungen, unter denen diese Zusicherung erfolgt, in der Form von standardisierten SLA-Dokumenten oder als Erweiterungen bestehender Standardformate bereitgestellt und automatisiert zugreifbar gemacht werden. Darüber hinaus werden Methoden benötigt, die den eigentlichen Abgleich zwischen der konkret vorliegenden Anforderung und den verfügbaren Angeboten vornehmen und so die konkret zu nutzenden Web Services bestimmen. Ist dem Nutzer keine Menge konkreter Web Services bekannt, die eine benötigte Funktionalität realisieren, so benötigt er zunächst eine Methode um Web Services zur Realisierung dieser Funktionalität aufzufinden. Erst in einem nächsten Schritt keine eine Auswahl

aus der Menge der funktionell geeigneten Web Services unter dem Aspekt der Dienstgüte erfolgen. Kommen hierbei mehrere Web Services in Frage, die allesamt den Anforderungen eines Nutzers genügen, so sind Methoden zu bestimmen, die aus der Menge der möglichen Web Services den am besten geeigneten Web Service selektieren. Dies kann mittels eines Rankings erfolgen, jedoch ist hierzu das Wissen über die individuellen Präferenzen des Nutzers nötig, um zu bestimmen, was für diesen Nutzer „der beste“ Web Service ist. Eine zusätzliche Komplexität wird in Szenarien erzeugt in denen die zu erbringende Dienstgüte dynamisch, in einer der Nutzung vorausgehenden Verhandlungsphase, mit dem Anbieter ausgehandelt werden kann.

Bei der dynamischen Einbindung von Web Services sind die Systeme der Nutzer von Web Services entsprechend flexibel zu gestalten. Diese zusätzliche Flexibilität erzeugt jedoch eine Vielzahl weiterer Problembereiche. Beispielsweise ist bei einer vollständig dynamischen Einbindung von Web Services nicht sicher, welcher Web Service zur Realisierung einer bestimmten Funktionalität zur Laufzeit konkret verwendet wird. So könnte anhand der Dienstgüteeigenschaften jeweils der momentan schnellste Web Service zur Realisierung einer Funktionalität verwendet werden, wobei sich die Performanz der zur Verfügung stehenden Web Services stetig ändern kann. Dadurch kann insbesondere nicht sichergestellt werden, dass bei der mehrmaligen Nutzung einer Funktionalität, diese in allen Fällen durch denselben Web Service realisiert wird. Wird beispielsweise die Funktionalität eines Reisebuchungssystems durch einen Web Service eingebunden, so ist es möglich dass zur Reservierung einer Reise im Rahmen einer Transaktion der Web Services eines anderen Anbieters verwendet wird, als später bei der Übermittlung eines Stornos im Rahmen eines Rollbacks der Transaktion. Der Rollback der Transaktion scheitert in diesem Fall, da keine Koordination zwischen der dynamischen Einbindung von Web Services und der verwendeten Transaktions- und Koordinationsprotokolle erfolgt ist.

Methoden zur Durchsetzung und Kontrolle vereinbarter Dienstgüte

Die alleinige Vereinbarung einer Dienstgüte zwischen dem Anbieter und dem Nutzer eines Web Services und die Ermittlung des, aufgrund seiner Dienstgüte am besten für einen Nutzer geeigneten, Web Services genügt noch nicht einer befriedigenden Unterstützung von Dienstgüte im Umfeld von Web Services. Die genutzte Infrastruktur muss über Methoden und Mechanismen verfügen die vereinbarte Dienstgüte bei der Nutzung von Web Services auch effektiv durchzusetzen. Die tatsächlich erbrachte Dienstgüte muss hierbei gemessen werden um einerseits die Einhaltung der vereinbarten Dienstgüte überwachen und andererseits die beteiligten Systeme zur Durchsetzung der Dienstgüte proaktiv steuern zu können.

Während der Nutzung eines Web Services, mit einer mittels SLA vereinbarten Dienstgüte und Rahmenbedingungen, muss der Nutzer des Web Services in der Lage sein, die Einhaltung der Dienstgüte und die Abrechnung der Nutzungsentgelte durch den Anbieter zu kontrollieren. Hierzu benötigt der Nutzer Komponenten die Informationen über die Nutzung von Web Services aufzeichnen und hieraus die Parameter der tatsächlich vom Anbieter erbrachten Dienstgüte ableiten, so dass diese mit der vereinbarten Dienstgüte verglichen werden können. Auch die Kontrolle der Abrechnung gemäß des vereinbarten Tarifmodells und der Nutzungsentgelte sollte möglich sein.

Der Anbieter eines Web Services muss sicherstellen, dass die mit dem Nutzer vereinbarte Dienstgüte erbracht werden kann. Er benötigt Komponenten um die definierten Dienstgüteanforderungen auf Anforderungen an durch ihn kontrollierbare Ressourcen umsetzen und diese entsprechend ansteuern zu können. So hat er z.B. die Gesamtheit der von ihm an alle Nutzer zugesicherten Dienstgüte zu berücksichtigen um entsprechende Systeme mit der nötigen Kapazität bereitstellen und die Last zwischen diesen Systemen verteilen zu können. Zur Kontrolle und proaktiven Steuerung der erbrachten Dienstgüte, sowie zur Abrechnung der Inanspruchnahme seiner Web Services durch die Nutzer, ist der Anbieter ebenfalls auf Komponenten zum Sammeln und Auswerten von Informationen bezüglich der Nutzung seiner Web Services angewiesen.

Ein besonderer Problembereich bei der Durchsetzung der vereinbarten Dienstgüte stellt hierbei für den Anbieter eines Web Services ein Zugriff auf seine Web Services über das öffentliche Internet dar. Web Services kommunizieren in der Regel mittels SOAP-Nachrichten die mittels HTTP über die TCP/IP-Infrastruktur des Internets übermittelt werden. Da weder durch den Nutzer, noch durch den Anbieter alle hierbei an der Kommunikation beteiligten Systeme kontrolliert werden können, und die Übermittlung der Nachrichten lediglich über einen Best-Efford-Dienst erfolgt, weist die Unterstützung der vereinbarten Dienstgüte zwangsläufig an dieser Stelle eine Lücke auf. Ein Anbieter von herkömmlichen Web Services kann daher über die üblichen Kanäle des Internets keine garantierten Dienste anbieten und wird bei der Zusicherung von Dienstgüte in SLAs auf Erfahrungswerte der Vergangenheit zurückgreifen, also lediglich vorhersagbare Dienste anbieten. Dies äußert sich in der Regel in den Rahmenbedingungen unter denen eine gewisse Dienstgüte in einer SLA angeboten wird. Hier werden üblicherweise nicht durch ihn beeinflussbare Faktoren, wie z.B. DoS-Attacken[30], Netzausfälle, unerwartete Verzögerungen, u.ä. als Umstände aufgeführt, unter denen ein Anbieter nicht verpflichtet ist, die vereinbarte Dienstgüte zu erbringen. Kalkuliert der Anbieter jedoch solche Ereignisse zu einer gewissen Wahrscheinlichkeit in seine, in der SLA vorhergesagten Dienstgüte ein, so ist er zu einer hohen Wahrscheinlichkeit in der Lage diese Dienstgüte im Schnitt auch erbringen zu können. Auch Telefonanbieter und Internetprovider garantieren die Verfügbarkeit eines Telefonanschlusses oder Internetzugangs nicht unter Einflüssen die außerhalb ihrer Kontrolle liegen, wie z.B. beim Einfluss höherer Gewalt durch Naturkatastrophen.

Ansätze

Entsprechend der Vielzahl der zu adressierenden Problembereiche bei der Integration von Dienstgüte in eine Web-Service-Umgebung existiert auch eine Vielzahl von Lösungsansätzen, die sich aus den verschiedensten Methoden zur Lösung der einzelnen Problembereiche zusammensetzen und hierbei verschiedene Problembereiche fokussieren. Konkrete Ansätze zur Berücksichtigung von Dienstgüte in Web-Service-Umgebungen sind z.B.

- Web Service Level Agreement (WSLA) von IBM
- Web Service Offering Language (WSOL) der Carleton University in Kanada
- SLAng des University College London in England
- UDDI eXtension (UX) der Nanyang Technological University in Singapur

[30] DoS ist die Abkürzung für "Denial of Service" und bezeichnet Systemangriffe, in denen Systeme oder Dienste durch eine hohe Anzahl von Anfragen in kurzer Zeit überlastet werden und dadurch nicht mehr zur Verfügung stehen [29].

- UDDIe der Cardiff University in England
- Web-Service-QoS (WS-QoS) der Freien Universität Berlin

Ein Vergleich dieser Ansätze kann [79] entnommen werden.

3 Konzept und Funktionsweise von WSQoSX

Gegenstand von WSQoSX ist die Realisierung einer Dienstgüte unterstützenden Web-Service-Architektur für flexible Geschäftsprozesse, welche in der Lage ist Web Services dynamisch, unter Berücksichtigung ihrer Dienstgüte und weiterer Rahmenbedingungen, auszuwählen, in Geschäftsprozesse einzubinden und auszuführen. Hierzu werden Web Services, die in einem Geschäftsprozess dynamisch eingunden werden sollen, gemäß ihrer Funktionalität in Kategorien gruppiert. Zu jeder Kategorie können eine individuelle Präferenzstruktur für Web-Service-Eigenschaften (z.B. Eigenschaften der technischen Dienstgüte oder Nutzungsentgelt), sowie entsprechende Mindestanforderungen definiert werden. Anbieter von Web Services können über ein Portal Web Services in Kategorien anmelden und hierbei in einer SLA die Dienstgüte eines Web Services und seine Nutzungsbedingungen definieren. Nach einer Bewertung der angemeldeten Web Services ist das System in der Lage ein Maß der Übereinstimmung eines Web Services mit der definierten Präfernzstruktur zu ermitteln. Während der Laufzeit des Geschäftsprozesses wird zur Durchführung eines Prozessschrittes dynamisch der Web Service aufgerufen, der am besten mit der definierten Präferenzstruktur übereinstimmt. Hierbei werden Web Services, die nicht den definierten Mindestanforderungen genügen, vom System nicht verwendet. Alle Web-Service-Aufrufe werden protokolliert und die Einhaltung der vom Anbieter garantierten Dienstgüte kontrolliert. Eine proaktive Steuerung von Ressourcen zur Durchsetzung von Dienstgüte erfolgt im Rahmen des WSQoSX-Systems jedoch nicht.

Ein Beispiel: Der Geschäftsprozess einer Bank zur Abwicklung einer Kreditgewährung enthält einen Prozessschritt zur Überprüfung der Kreditwürdigkeit des Kunden. Dieser Prozessschritt wird durch einen Web Service gekapselt, der intern als „Schufa-Service“ bezeichnet wird. Nachdem dem Schufa-Service Daten zur Identifizierung des Kunden übermittelt wurden, liefert er die Kreditwürdigkeit des Kunden in Form eines Ratings zurück. Die Bank ist daran interessiert diesen Prozessschritt durch einen möglichst optimalen Web Service ausführen zu lassen. Der Web Service sollte möglichst preiswert sein, jedoch gewissen Mindestanforderungen an seiner Dienstgüte und der Seriosität des Anbieters genügen. Die Bank veröffentlicht daher in ihrem Web-Service-Portal eine Ausschreibung in Form einer Kategorie „Schufa-Service“. Hier wird die Funktionalität des Schufa-Service beschrieben und seine WSDL-Beschreibung zur Spezifikation der Schnittstelle veröffentlicht. Darüber hinaus definiert die Bank Mindestanforderungen an Web Services dieser Art und ihre Präferenz bezüglich solcher Web Services durch die Gewichtung von Eigenschaften. Anbieter, die einen Web Service mit der gewünschten Funktionalität und Schnittstelle bereitstellen, können diesen Web Service am Portal der Bank anmelden. Hierbei geben sie in einem SLA-Dokument die zugesicherte Dienstgüte, sowie weitere Rahmenbedingungen der Nutzung (z.B. Nutzungsentgelte) an. Zur Laufzeit werden automatisch alle Anfragen des Geschäftsprozesses an den Schufa-Service dynamisch an den besten zur Zeit bekannten Web Service der Kategorie „Schufa-Service“ weitergeleitet. Durch die Protokollierung der Web-Service-Aufrufe kann die Abrechnung der Nutzungsentgelte mit dem jeweiligen Anbieter kontrolliert werden. Sollte ein Web Service bei Aufrufen

gegen die, durch den Anbieter garantierte Dienstgüte verstoßen, wird der System-Administrator hiervon in Kenntnis gesetzt und kann weitere Maßnahmen ergreifen.

Um ein Szenario wie im geschilderten Beispiel realisieren zu können, sind Informationen über verfügbare Web Services zu verwalten und diese in mehreren Phasen in die Infrastruktur der Geschäftsprozessausführung zu integrieren. Im Folgenden werden die Integrationsphasen eines Web Services, der in die Infrastruktur von WSQoSX eingebunden wird, dargelegt, sowie die verwendeten Auswahl- und Aufrufmechanismen erläutert.

3.1 Integrationsphasen und Verwaltung von Web Services

Ein Web Service wird in mehreren Phasen in die Infrastruktur von WSQoSX integriert. Eine Übersicht der Integrationsphasen kann hierbei dem Zustandsdiagramm in Abbildung 7 entnommen werden.

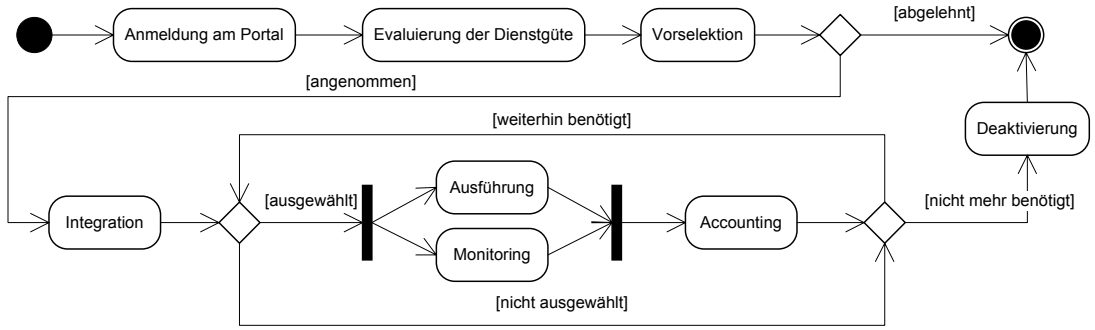

Abbildung 7 - Integrationsphasen eines Web Services (in Anlehnung an [11])

1. Phase: Anmeldung am Portal des Nutzers
Damit ein Web Service durch das WSQoSX-System dynamisch eingebunden werden kann, muss der Web Service zunächst erfasst werden. Der Anbieter eines Web Services meldet hierzu seinen Web Service über das Portal des Nutzers unter Bekanntgabe der verbindlich zugesicherten Dienstgüte am System an.

2. Phase: Evaluierung der Dienstgüte
Durch den Administrator des Systems wird die Anmeldung geprüft und Dienstgütekriterien, die nicht quantitativ spezifiziert werden können (z.B. Sicherheit oder Seriösität des Anbieters), werden mittels Punkten bewertet. Anschließend ermittelt das System eine Bewertung des Web Services gemäß einer zuvor definierten Gewichtung einzelner Dienstgüteparameter.

3. Phase: Vorselektion
Nach der Bewertung des Web Services wird in dieser Phase geprüft ob der Web Service zuvor definierten Mindestanforderungen genügt, die in Form von Regeln formuliert sind. Kann ein Web Service nicht alle definierten Regeln erfüllen, so genügt er nicht den Mindestanforderungen und gelangt nicht in die nächste Integrationsphase.

4. Phase: Integration des Web Services in das interne Verzeichnis
Erfüllt ein Web Service die definierten Mindestanforderungen, so wird der Web Service in die Menge der, zur Verwendung durch das System zu Verfügung

stehenden, Web Services integriert. Das System verwendet nur Web Services aus dieser Menge um anstehende Web-Service-Aufrufe abzuarbeiten.

5. Phase: Selektion eines Web Services zur Laufzeit
Wird vom System zur Laufzeit ein zu tätigender Web-Service-Aufruf empfangen, so wählt das System dynamisch den besten bekannten Web Service aus, der in der Lage ist den entgegengenommenen Aufruf abzuarbeiten. Die Auswahl wird hierbei anhand der Bewertungen der Web Services vorgenommen.

6. Phase: Ausführung
Der durch das System zu tätigende Web-Service-Aufruf wird an den, in der vorherigen Phase selektierten, Web Service weitergeleitet. Die Antwort des Web Services wird empfangen und an den ursprünglichen Aufrufer übermittelt. Während des Aufrufs wird die tatsächlich erbrachte Dienstgüte eines Web Services anhand einiger direkt messbarer Dienstgütekriterien erfasst (Monitoring des Aufrufs).

7. Phase: Accounting
Informationen über den Aufruf des Web Services werden durch das System protokolliert. Anhand dieser Protokollinformationen ist es möglich eine Übersicht über die aufgerufenen Web Services, ihre Aufrufer, den Erfolg der Aufrufe und die erbrachte Dienstgüte durch den jeweiligen Web Service zu ermitteln. Die Zuordnung von Kosten zu Aufrufern, die Kontrolle von abgerechneten Nutzungsentgelten, sowie die Einhaltung der SLA durch die Web Services ist durch diese Informationen möglich.

8. Phase: Deaktivierung des Web Services
Sollte die Funktionalität eines im System registrierter Web Services nicht mehr benötigt werden, so kann ein Web Service durch einen Administrator aus dem System gelöscht oder vorübergehend deaktiviert werden. Auch der Anbieter eines registrierten Web Service kann diesen über das Portal entfernen, z.B. falls dieser in Zukunft durch ihn nicht mehr angeboten wird.

Wie genau Informationen über Web Services über die einzelnen Integrationsphasen hinweg verarbeitet werden, soll im Folgenden näher erläutert werden.

3.1.1 Portal als Anbieterschnittstelle

Damit durch das WSQoSX-System Web Services, die aus Geschäftsprozessen heraus aufgerufen werden, dynamisch ausgewählt und eingebunden werden können, muss das System über Informationen verfügen, welche Web Services mit welchen Eigenschaften zur Einbindung zur Verfügung stehen. Um dem System entsprechende Informationen über Web Services zukommen lassen zu können, wird eine Schnittstelle benötigt über die vor allem die Anbieter ihre Web Services am System des Nutzers anmelden können. Eine solche Anbieterschnittstelle wird mit dem *„Web Service Portal“ (WSPortal)* bereitgestellt. Das Portal kann über HTTP erreicht und mittels eines Webbrowsers über eine Weboberfläche plattformunabhängig verwendet werden.

Bevor ein Anbieter das Portal verwenden kann, muss sich der Anbieter einmalig unter Bekanntgabe seiner Kontaktinformationen registrieren. Dieser Vorgang kann mit dem Veröffentlichen einer `businessEntity` in einem UDDI-Verzeichnis verglichen

werden (siehe Kapitel 2.2.3). Bei der Registrierung des Anbieters werden folgende Daten erfasst (siehe auch Abbildung 8):

- Name der Unternehmung
- Vor- und Nachname einer Kontaktperson
- Anschrift
- Telefon- und Fax-Nummer
- E-Mail-Adresse

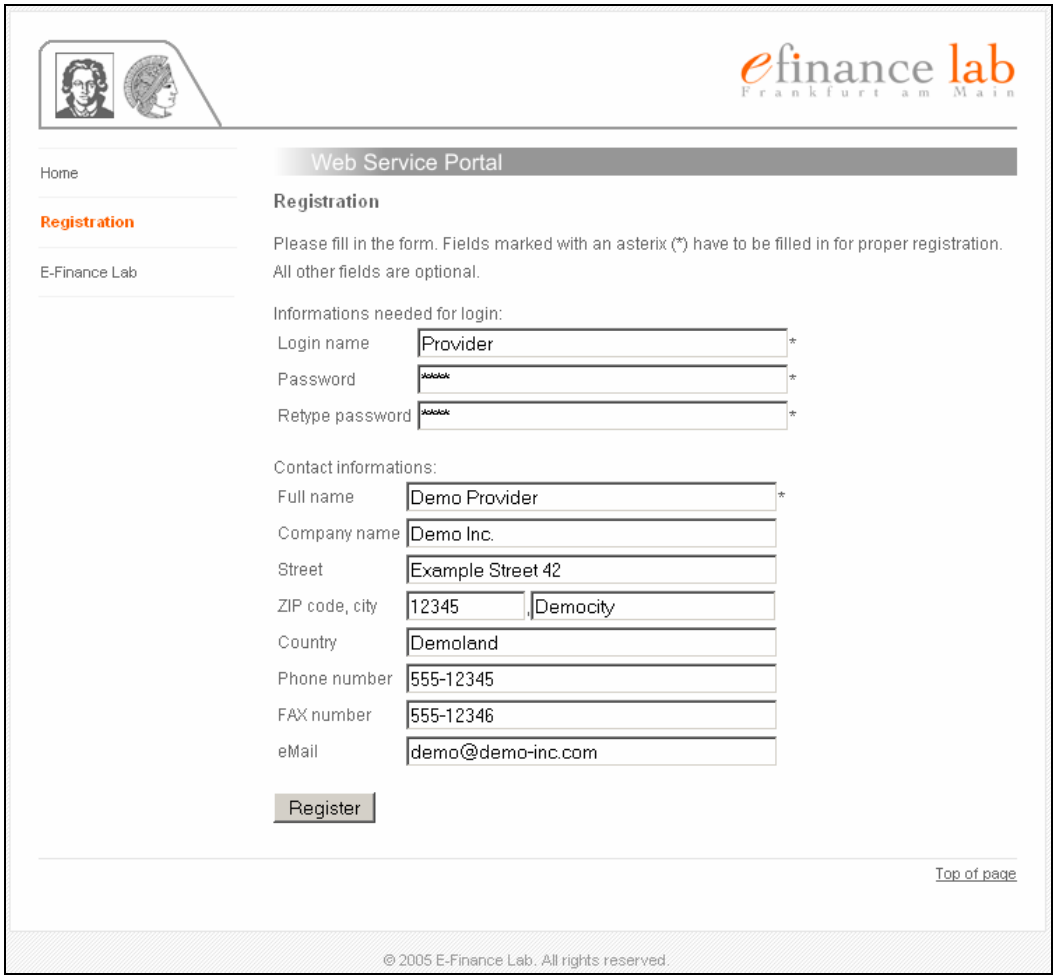

Abbildung 8 - Registrierung eines Anbieters im Portal

Diese Daten werden durch den Nutzer benötigt um in einem späteren Schritt entscheiden zu können, ob er mit diesem Anbieter zusammenarbeiten möchte oder um ihn im Falle von Problemen technischer oder organisatorischer Natur erreichen zu können.

Bei seiner Registrierung wählt der Anbieter zusätzlich einen Login-Namen und ein Passwort um sich bei einem späteren Besuch am Portal identifizieren zu können. Hat sich der Anbieter am Portal registriert oder durch einen gültigen Login identifiziert, so kann er die Anbieterfunktionalitäten nutzen. Hierzu zählen:

- Aufrufen einer Übersicht von Kategorien, in denen Web Services angemeldet werden können.
- Anmelden von Web Services in einer Kategorie.
- Aufrufen einer Übersicht von Web Services, die durch den Anbieter bereits am System angemeldet wurden.

- Löschen von Web Services, die durch den Anbieter bereits am System angemeldet wurden.
- Bearbeiten der Kontaktdaten des Anbieters.
- Löschen der Anbieterregistrierung (bedingt die Löschung aller durch den Anbieter angebotenen Web Services).
- Logout des Anbieters vom Portal.

Ein Anbieter kann am Portal nicht beliebige Web Services anmelden, sondern nur solche, die vom Nutzer benötigt werden. Welche Web Services generell durch das System verwaltet werden, wird durch Kategorien festgelegt. Welche dieser Kategorien einem Anbieter für die Anmeldung von Web Services zur Verfügung stehen, wird vom Administrator des Systems festgelegt. Das Konzept der Kategorien wird im Folgenden näher erläutert.

3.1.2 Kategorisierung der Web Services

Hintergrund

Durch das WSQoSX-System können Aufrufe an Web Services, die im Rahmen eines, aus Web Services orchestrierten Geschäftsprozesses aufgerufen werden, dynamisch an konkrete Web Services weitergeleitet werden. Hierbei wird derjenige konkrete Web Service zur Weiterleitung ausgewählt, der aufgrund seiner Bewertung am besten zur Abarbeitung des Aufrufs geeignet ist. Da Web Services einen bestimmten Prozessschritt, d.h. eine ganz bestimmte Funktionalität, im Geschäftsprozess ausführen, muss sichergestellt werden, dass Aufrufe nur an solche Web Service weitergeleitet werden, welche dieselbe Funktionalität bereitstellen, d.h. in der Lage sind den konkreten Prozessschritt ausführen zu können. Dem WSQoSX-System muss daher bekannt sein, welche Web Services dieselbe Funktionalität bereitstellen.

Gruppierung nach Funktionalität und Schnittstelle

Um dem System mitzuteilen welche Funktionalität durch einen Web Service erbracht wird, werden Web Services bei ihrer Anmeldung einer bestimmten *Kategorie* zugeordnet. Kategorien gruppieren Web Services gleicher Funktionalität und Schnittstelle. Jede Kategorie verfügt über eine informelle Beschreibung ihrer Funktionalität, sowie eine Schnittstellenspezifikation in Form eines WSDL-Dokuments. Meldet ein Anbieter einen Web Service über das Portal in einer Kategorie an, so sichert er zu, dass die dort beschriebene Funktionalität durch die dort spezifizierte Schnittstelle erbracht wird. Einer Anmeldung eines Web Services in einer Kategorie entspricht daher der Definition seiner funktionalen Eigenschaften und seiner Schnittstelle.

Die Anmeldung eines Web Services in einer Kategorie lässt sich mit dem Eintragen eines Web Services in ein UDDI-Verzeichnis wie folgt vergleichen: Die WSDL-Schnittstellenspezifikation der Kategorie wird zunächst als ein tModel veröffentlicht. Bei jeder Veröffentlichung eines Web Services aus dieser Kategorie wird angegeben, dass dieser Web Service das Kategorie-tModel implementiert. Durch eine Anfrage an das UDDI-Verzeichnis, welche Web Services dieses tModel implementieren, können alle Web Services einer Kategorie gefunden werden.

Verwaltung der Kategorien

Für jeden Web Service eines Geschäftsprozesses, der durch das System zur Laufzeit dynamisch durch den optimalen Web Service ersetzt werden soll, muss der Nutzer eine Kategorie definieren, welche die Funktionalität und die Schnittstelle des Web Services spezifiziert. Hierbei werden zu jeder Kategorie folgende Daten erfasst (vgl. Abbildung 9):

- Kategoriename
- Kurzbeschreibung
- URL zu einem Dokument mit ausführlicher Beschreibung der Funktionalität
- URL zu dem WSDL-Dokument der Schnittstellenspezifikation
- URL zu einer Vorlage für ein SLA-Dokument
- Status der Erlaubnis von Neuanmeldungen

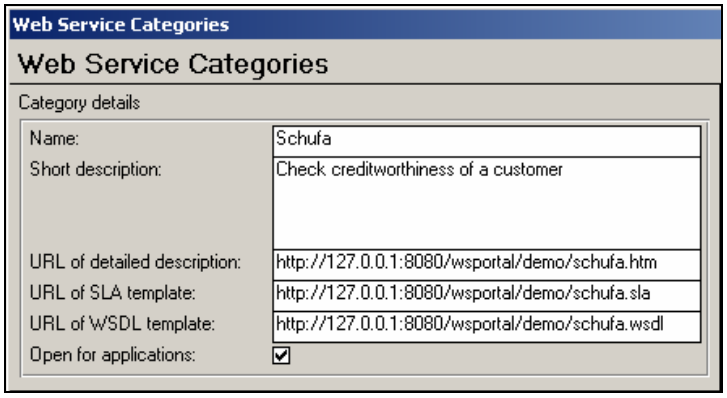

Abbildung 9 - Verwaltung von Kategorien

Kategoriename und Kurzbeschreibung

Der Kategoriename und die Kurzbeschreibung sollten die Kategorie grob charakterisieren. In dem eingangs in Kapitel 3 genannten Beispiel des „Schufa-Service“ könnte als Kategoriename „Schufa“ und als Kurzbeschreibung „Überprüfung der Kreditwürdigkeit eines Kunden“ angegeben werden.

Dokument mit ausführlicher Beschreibung der Funktionalität

Um die funktionalen Eigenschaften einer Kategorie näher zu spezifizieren, wird die URL eines Dokuments (z.B. HTML-Dokument, PDF-Dokument) angegeben, welches eine ausführlichen Beschreibung der Funktionalität enthält, die durch alle Web Services dieser Kategorie erbracht werden muss. Im bereits genannten Beispiel des „Schufa-Service“ würde in diesem Dokument erläutert werden, welche Eingabeparameter zur Identifikation eines Kunden übermittelt werden, wie genau die Kreditwürdigkeit des Kunden ermittelt werden soll und welche Rückgabewerte zu liefern sind.

WSDL-Dokument der Schnittstellenspezifikation

Damit sichergestellt werden kann, dass eine Anfrage an jeden Web Service innerhalb einer Kategorie versendet werden kann, müssen alle Web Services einer Kategorie dieselbe Schnittstelle implementieren. Hierzu wird die URL zu einem WSDL-Dokument angegeben, welches die Schnittstellenspezifikation der zu implementierende Schnittstelle der Kategorie enthält. Ein Anbieter kann dieses WSDL-Dokument verwenden um einen Web Service mit entsprechender Schnittstelle zu erstellen oder bestehende Web Services an diese Schnittstelle anzupassen.

Vorlage für ein SLA-Dokument
Da die funktionalen Eigenschaften aller Web Services in einer Kategorie identisch sind, wird die Auswahl eines konkreten Web Service zur Laufzeit anhand der Bewertung nicht-funktionaler Eigenschaften vorgenommen. Zur Übermittlung der nicht-funktionalen Eigenschaften eines Web Services kann hierbei ein spezielles SLA-Dokument verwendet werden. Um dem Anbieter die Erstellung eines solchen SLA-Dokuments zu erleichtern, wird in jeder Kategorie die URL zu einer Vorlage eines SLA-Dokuments angegeben. In dieses Vorlagendokument kann der Anbieter die konkreten Eigenschaften seines Web Services eintragen und später bei der Anmeldung am Portal übermitteln.

Erlaubnis von Neuanmeldungen
Jede Kategorie verfügt über einen Statuswert, der entweder Neuanmeldungen von Web Services in dieser Kategorie erlaubt oder diese unterbindet. Ist die Kategorie für Neuanmeldungen freigegeben, so handelt es sich um eine offene Kategorie. Nur offene Kategorien erscheinen im Portal in der Übersichtsliste der Kategorien und können von einem Anbieter für die Anmeldung von Web Services ausgewählt werden.

Präferenzen und Anforderungen an nicht-funktionale Eigenschaften

In jeder Kategorie können neben den Anforderungen an die funktionalen Eigenschaften eines Web Services auch Mindestanforderungen und Präferenzen bezüglich nicht-funktionaler Eigenschaften definiert werden. Durch Mindestanforderungen können Kriterien definiert werden, die ein Web Service mindestens erbringen muss um in dieser Kategorie als produktiver Web Service verwendet zu werden. Mittels der Präferenzen kann definiert werden welche nicht-funktionalen Eigenschaften bei der Auswahl eines Web Services zur Laufzeit mit welcher Gewichtung zu berücksichtigen sind.

Bevor jedoch auf Mindestanforderungen und Präferenzen bezüglich nicht-funktionaler Eigenschaften eingegangen wird (siege Kapitel 3.1.4), werden im Folgenden zunächst die nicht-funktionalen Eigenschaften eines Web Services näher erläutert.

3.1.3 Nicht-funktionale Eigenschaften von Web Services

Das System verwaltet zu jedem angemeldeten Web Service eine Reihe nicht-funktionaler Eigenschaften, die sich wie folgt untergliedern lassen.

Dienstgütekriterien des Web Services:
- Verfügbarkeit
- Durchsatz
- Antwortzeit
- Verschlüsselung
- Authentifizierung
- Autorisierung

Rahmenbedingungen der Diensterbringung:
- Preis
- Gültigkeitsdauer

Zusatzinformationen zur Beurteilung des Anbieters:

- Referenzen
- Reputation

Im Einzelnen sind diese nicht-funktionalen Eigenschaften wie folgt definiert.

Verfügbarkeit (engl. availability)
Die Verfügbarkeit eines Web Services bezeichnet die Mindestwahrscheinlichkeit, mit der ein Web Service durch einen Nutzer erfolgreich in Anspruch genommen werden kann. Die Verfügbarkeit wird als numerischer, prozentualer Wert angegeben, z.B. 99%. Das System geht davon aus, dass mindestens der durch die Verfügbarkeit angegebene Prozentsatz an Aufrufen des Web Services erfolgreich vorgenommen werden kann. Von der Nichtverfügbarkeit eines Web Services wird ausgegangen, falls der spezifizierte Netzwerkendpunkt des Web Services nicht erreichbar ist oder nach der Übermittlung einer Anfrage an den Web Service eine erforderliche Antwort vom Web Service nicht innerhalb der doppelten zugesicherten Antwortzeit erhalten wurde (Timeout des Aufrufs).

Durchsatz (engl. throughput)
Der Durchsatz bezeichnet die Mindestanzahl von Aufrufen, die durch einen Web Service an einem Tag verarbeitet werden können. Der Durchsatz wird als numerischer Wert in der Einheit Aufrufe pro Tag spezifiziert, z.B. 1000 Aufrufe/Tag. Der Durchsatz spezifiziert hierbei nicht, wie sich die Aufrufe über den Tag verteilen. Das System geht davon aus, dass es mindestens die durch den Durchsatz spezifizierte Anzahl von Aufrufen innerhalb eines Tages an den Web Service senden kann ohne diesen hierdurch zu überlasten.

Antwortzeit (engl. response time)
Die Antwortzeit ist die Zeit, die zwischen dem Beginn der Übermittlung der Anfrage an den Web Service und dem vollständigen Empfang der Antwort des Web Services höchstens vergeht. Die Antwortzeit wird als numerischer Wert in der Einheit Millisekunden spezifiziert, z.B. 1500 ms. Das System geht davon aus, dass höchstens die Antwortzeit benötigt wird um eine Anfrage an einen Web Service zu senden und eine Antwort zu erhalten. Sollte mehr als die doppelte Antwortzeit vergehen, so nimmt das System an, dass der Web Service nicht verfügbar ist (Timeout des Aufrufs).

Verschlüsselung (engl. encryption), Authentifizierung (engl. authentication) und Autorisierung (engl. authorisation)
Angaben zu Sicherheitsaspekten des Web Services können über die Eigenschaften Verschlüsselung, Authentifizierung und Autorisierung gemacht werden. Die Angaben erfolgen hierbei in Textform und unterliegen keinerlei Bindung an eine feste Syntax oder Semantik. Eine Nutzungsmöglichkeit besteht z.B. in der Angabe verpflichtend zu verwendender oder optional unterstützter Verfahren der Verschlüsselung, Authentifizierung und Autorisierung. Durch die Angabe „WS-Security, 3DES“ in der Eigenschaft Verschlüsselung könnte beispielsweise die Verwendung von WS-Security[31] in den ausgetauschten SOAP-Nachrichten und die Verwendung des Triple-

[31] WS-Security [62] ist ein vom OASIS herausgegeber Standard zur Erweiterung des SOAP-Protokolls um Mechanismen zur Sicherstellung der Integrität und Vertraulichkeit in SOAP-Nachriten.

DES-Algorithmus (3DES)[32] zur Verschlüsselung der Parameter und Rückgabewerte signalisiert werden.

Preis (engl. price)
Der Preis bezeichnet das Nutzungsentgelt, welches der Nutzer an den Anbieter für jeden erfolgreich durchführten Web-Service-Aufruf zu entrichten hat. Der Preis wird als numerischer Wert in der Einheit Preis in EUR pro Aufruf spezifiziert, z.B. 0,02 EUR/Aufruf. Bei der Spezifikation des Nutzungsentgelts können keine hiervon abweichenden Tarifmodelle, wie z.B. Staffel- oder Flatrate-Tarifierung, verwendet werden.

Gültigkeit (engl. validity)
Die Gültigkeit wird in Form eines Datums angegeben und bezeichnet den Zeitpunkt, bis zu welchem die Zusicherungen des Anbieters bezüglich des Web Services gültig sind. Mit Ablauf des Datums läuft das SLA zwischen Nutzer und Anbieter aus und das System kann nicht mehr von der Richtigkeit der gemachten Angaben zu einem Web Service ausgehen. Für den Nutzer bedeutet dies insbesondere, dass er nicht mehr die Einhaltung der, in einem abgelaufenen SLA spezifizierten Eigenschaften vom Anbieter verlangen kann.

Referenzen (engl. references)
Mittels eines formfreien Textes hat der Anbieter die Möglichkeit Referenzen anzugeben und so den Nutzer von seiner Professionalität und Erfahrung zu überzeugen. Eine denkbare Nutzungsmöglichkeit der Referenzen besteht beispielsweise in der Angabe einer durch Kommata separierten Liste von bedeutenden Geschäftspartnern des Anbieters.

Reputation (engl. reputation)
Die Reputation eines Anbieters ist die einzige nicht-funktionale Eigenschaft eines Web Services, die nicht vom Anbieter selbst angegeben werden kann. Diese Eigenschaft ist vielmehr ein Platzhalter für formfreie Anmerkungen des Nutzers, die er während der Bewertung eines Web Services machen kann. Der Nutzer vermerkt hier wie er den Ruf des Anbieters einschätzt und ob er bereits in der Vergangenheit positive wie negative Erfahrungen mit Web Services dieses Anbieters sammeln konnte. Nach der Bewertung durch den Nutzer geht diese Eigenschaft zusammen mit den anderen nicht-funktionalen Eigenschaften in die Gesamtbewertung des Web Services mit ein.

Übermittlung nicht-funktionaler Eigenschaften

Alle nicht-funktionalen Eigenschaften, mit Ausnahme der Reputation, müssen durch einen Anbieter bei der Anmeldung eines Web Services spezifiziert werden. Der Anbieter kann im Anmeldeformular des Portals alle Eigenschaften einzeln eintragen, oder stattdessen die URL eines SLA-Dokuments angeben, welches alle Eigenschaften als ganzes enthält. Das Format des SLA-Dokuments wird hierbei durch eine Untermenge der Sprache WSLA, die von IBM zur Spezifikation von SLA-Dokumenten entwickelt wurde, definiert und wird in Kapitel 4.4.6 näher erläutert.

[32] 3DES (Triple DES) ist eine Variante des Verschlüsselungsverfahrens DES (Data Encryption Standard) zur Steigerung der effektiven Schlüssellänge. Sowohl DES als auch 3DES werden in [2] spezifiziert.

3.1.4 Präferenzen und Mindestanforderungen in einer Kategorie

Notwendigkeit der Definition von Präferenzen

Alle Web Services einer Kategorie stellen dieselbe Funktionalität über dieselbe Schnittstelle zur Verfügung. Wird zur Laufzeit die Funktionalität einer Kategorie benötigt, so muss unter allen in dieser Kategorie verfügbaren Web Services derjenige ausgewählt und aufgerufen werden, der diese Funktionalität möglichst optimal erbringen kann. Was jedoch als optimal anzusehen ist, kann nicht über alle Kategorien hinweg verallgemeinert werden. Ein Web Service mit einer kürzeren Antwortzeit ist gegenüber einem Web Service mit einer längeren Antwortzeit meist im Vorteil, jedoch könnte in bestimmten Anwendungsfällen die Antwortzeit gegenüber den Sicherheitsaspekten eines Web Services als irrelevant eingestuft werden. Da sich also die Anforderungen an nicht-funktionale Eigenschaften eines Web Services von Kategorie zu Kategorie unterscheiden können, muss es möglich sein zu jeder Kategorie spezifische Präferenzen zur Auswahl des optimalen Web Services anzugeben.

Definition von Präferenzen

Die *Definition von Präferenzen* für die Auswahl eines Web Services erfolgt über eine *Gewichtung der nicht-funktionalen Eigenschaften*. Die Gewichtung kann durch den Nutzer über die Kategorieverwaltung des Administrations-Frontends spezifisch für jede Kategorie vorgenommen werden. Hierbei kann für jede nicht-funktionale Eigenschaft ein numerischer Wert als Gewicht festgelegt werden (siehe Abbildung 10). Je höher das Gewicht einer Eigenschaft, desto stärker wird die Eigenschaft bei der Bewertung, und damit auch bei der Auswahl, eines Web Services berücksichtigt.

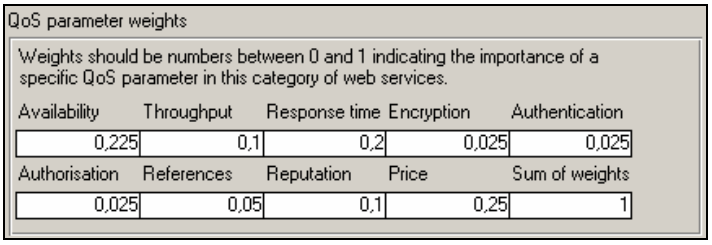

Abbildung 10 - Definition der Gewichtung nicht-funktionaler Eigenschaften in einer Kategorie

Obwohl durch das System selbst keine näheren Anforderungen an die numerischen Werte der Gewichte gestellt werden, wird empfohlen für Gewichte nur Werte zwischen 0 und 1 zu verwenden und sie so auf die einzelnen Eigenschaften zu verteilen, dass sie in der Summe 1 ergeben. Bezeichnet man die neun definierbaren Gewichte einer Kategorie als w_j (mit j=1,…,9), so sollte nach Möglichkeit gelten:

$$\left(\forall_{j=1,\ldots,9}.0 \le w_j \le 1\right) \wedge \left(\sum_{j=1}^{9} w_j = 1\right)$$

Formel 1 - Empfehlung zur Gewichtung nicht-funktionaler Eigenschaften in einer Kategorie

Das Administrations-Frontend unterstützt den Nutzer bei einer derartigen Aufteilung der Gewichte, indem es die bei jeder Änderung der Gewichte die Summe dieser neu berechnet und anzeigt.

Wie genau die Gewichte in die Bewertung, und damit auch in die Auswahl, eines Web Services eingehen, wird in Kapitel 3.1.6 ausführlich erläutert.

Mindestanforderungen

Neben der Definition von Präferenzen bei der Auswahl eines Web Services ist es wünschenswert über eine Vorselektion Web Services mit gewissen Eigenschaften von der produktiven Nutzung durch das System ausschließen zu können. So kann z.B. verhindert werden, dass in einem kritischen Geschäftsprozess Web Services verwendet werden, die gewissen Anforderungen an die Verfügbarkeit nicht genügen. Zu diesem Zweck erlaubt das System spezifisch für jede Kategorie *Mindestanforderungen* an Web Services *in der Form von Regeln* zu definieren.

Entsprechende Regeln können durch den Nutzer über die Kategorieverwaltung des Administrations-Frontends definiert werden (siehe Abbildung 11). Zu jeder Kategorie lassen sich beliebig viele Regeln definieren. Jede Regel besteht hierbei aus drei Teilen: Dem Namen einer nicht-funktionalen Eigenschaft, einem Vergleichsoperator und einem numerischen Vergleichswert.

QoS Rules

	Parameter	Operator	Value
▶	Throughput	>=	1000
	Availability	>=	98
	Response time	<=	10000
*			

Abbildung 11 - Definition von Mindestanforderungen durch Regeln

Um zu entscheiden ob eine Regel durch einen Web Service erfüllt wird, vergleicht das System die definierte nicht-funktionale Eigenschaft des Web Services mithilfe des definierten Vergleichsoperators mit dem definierten Vergleichswert. Der Wahrheitswert dieser Vergleichsoperation gibt an, ob die Regel erfüllt wurde oder nicht. Ein Web Service erfüllt genau dann die definierte Mindestanforderung einer Kategorie, wenn er alle definierten Regeln dieser Kategorie erfüllt.

Ein Beispiel für die Definition von Regeln kann Abbildung 11 entnommen werden. Im Einzelnen werden dort folgende Regeln definiert:

- Ein Web Service muss einen Durchsatz von mindestens 1.000 Aufrufen pro Tag bewältigen können.
- Ein Web Service muss eine Verfügbarkeit von mehr als 98% aufweisen.
- Ein Web Service muss über eine Antwortzeit von höchstens 10.000 ms verfügen.

Im Beispiel wurden nur Regeln für nicht-funktionale Eigenschaften definiert, deren Eigenschaftswerte numerische Werte sind und daher direkt mittels eines Vergleichsoperators mit dem numerischen Vergleichswert verglichen werden können. Es lassen sich jedoch auch Regeln für nicht-funktionale Eigenschaften definieren, die nicht durch numerische Werte beschrieben werden können, wie z.B. für die Eigenschaft Verschlüsselung. In einem solchen Fall erfolgt der Vergleich nicht mit dem eigentlichen (nicht-numerischen) Eigenschaftswert, sondern mit der numerischen Punktbewertung, die ein Nutzer diesem Eigenschaftswert während der Bewertung des Web Services zugeordnet hat. Um sicherstellen zu können, dass solche

Punktbewertungen für entsprechende Vergleichsoperationen verfügbar sind, werden die Regeln erst nach der Bewertung eines Web Services durch den Nutzer überprüft. Die Bewertung von Web Services wird im Kapitel 3.1.6 näher erläutert.

Flexibilität von Präferenzen und Mindestanforderungen

Präferenzen und Mindestanforderungen müssen durch den Nutzer bereits zum Zeitpunkt der Erstellung einer Kategorie festgelegt werden, können jedoch jederzeit an die aktuellen Bedürfnisse angepasst werden. So kann z.B. in einer Phase hohen Transaktionsaufkommens in einer Kategorie die Präferenz auf Web Services verschoben werden, die einen hohen Durchsatz sowie eine hohe Verfügbarkeit bieten. Fällt das Transaktionsaufkommen wieder geringer aus, so könnte die Präferenz wieder auf preisgünstigere Web Service verlegt werden.

3.1.5 Anmeldung eines Web Services

Die Anmeldung von Web Services kann durch einen Anbieter über das Portal erfolgen. Im Portal kann sich ein Anbieter die Liste von Kategorien anzeigen lassen, die momentan für die Anmeldung von Web Services freigegeben sind (offene Kategorien, siehe Abbildung 12).

Abbildung 12 - Liste offener Kategorien im Portal

Aus der Sicht des Anbieters handelt es sich bei Kategorien um eine Ausschreibung für Web Services mit einer gewünschten Funktionalität und Schnittstelle. Details über die, in einer Kategorie verlangten Funktionalität erhält der Anbieter über ein verlinktes Dokument mit einer ausführlichen Beschreibung. Die erforderliche Schnittstelle ist mittels eines verlinkten WSDL-Dokuments spezifiziert.

Verfügt ein Anbieter über einen Web Service, der den Anforderungen der Kategorie genügt, so kann er diesen Web Service in dieser Kategorie anmelden. Die Anmeldung stellt für den Anbieter eine Bewerbung dar, da er nach einer Anmeldung noch nicht davon ausgehen kann, dass der angemeldete Web Service vom Nutzer auch tatsächlich verwendet wird. Erst nachdem die Anmeldung durch den Nutzer überprüft, Eigenschaften des Web Services bewertet und die Erfüllung von Mindestanforderungen an den Web Service geprüft wurde, wird ein Web Service zur produktiven Verwendung im System freigegeben (oder abgelehnt). Der Anbieter kann in der Liste seiner angemeldeten Web Services ersehen, ob ein Web Service bereits überprüft wurde, oder ob eine Überprüfung noch aussteht. Das Ergebnis einer solchen Überprüfung wird ihm jedoch nicht explizit mitgeteilt.

Um sich mit einem Web Service in einer Kategorie zu bewerben, muss der Anbieter nach der Auswahl der entsprechenden Kategorie eine Reihe von Informationen in ein Formular eintragen (siehe Abbildung 13).

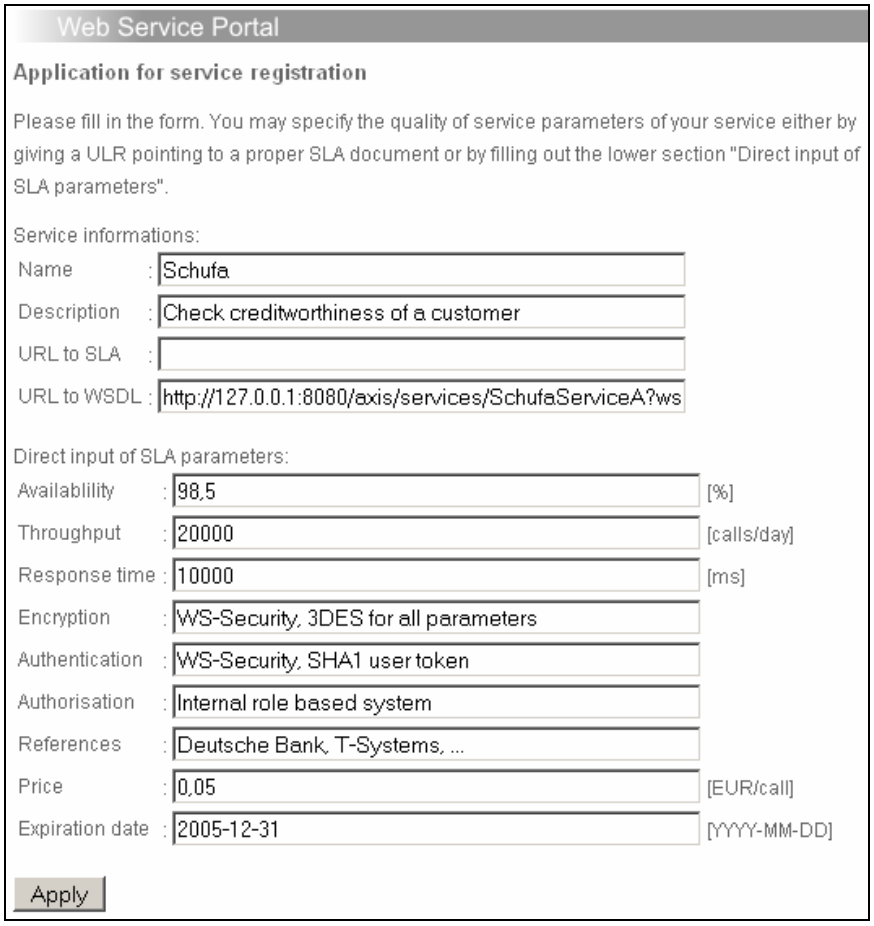

Abbildung 13 - Anmeldung eines Web Services am Portal

Die anzugebenden Informationen lassen sich wie folgt untergliedern:

- Verwaltungsinformationen
- Schnittstellendefinition des Web Services
- Nicht-funktionale Eigenschaften des Web Services

Verwaltungsinformationen

Der Anbieter hat die Möglichkeit dem anzumeldenden Web Service einen Namen und eine Kurzbeschreibung zuzuordnen. Diese Angaben erleichtern sowohl dem Anbieter als auch dem Nutzer die Identifikation und das spätere Auffinden des Web Services. Es ist einem Anbieter nicht möglich zwei verschiedene Web Services in der gleichen Kategorie unter demselben Namen anzumelden. Sollten Nutzer und Anbieter in direkten Kontakt bezüglich eines Web Services treten, so genügt daher die Angabe der Kategorie und des Namens eines Web Services um ihn in diesem Kontext eindeutig zu identifizieren. Der Name eines Web Services dient lediglich zur Identifikation durch menschliche Benutzer. Das System verwendet intern andere Merkmale um einen Web Service eindeutig zu identifizieren.

Schnittstellenspezifikation

Die Angabe der Schnittstellenspezifikation des Web Services erfolgt über die Angabe der URL des entsprechenden WSDL-Dokuments. Nach dem Absender der

Anmeldeinformationen wird das WSDL-Dokument von dieser URL eingelesen, geparsed und der Netzwerkendpunkt des Web Services aus den `port`-Elementen des WSDL-Dokuments extrahiert. Da das System ausschließlich Aufrufe von Web Services über das HTTP-Protokoll vornehmen kann, erwartet es einen Netzwerkendpunkt in Form einer entsprechenden URL. Das WSDL-Dokument wird durch das System lokal gesichert, da es einen Teil der Vereinbarung darstellt, die der Anbieter dem Nutzer zum Zeitpunkt der Anmeldung des Web Services verbindlich anbietet.

Nicht-funktionale Eigenschaften des Web Services
Nicht-funktionale Eigenschaften des Web Services können bei der Anmeldung durch zwei alternative Methoden durch den Anbieter übermittelt werden. Entweder der Anbieter gibt die nicht-funktionalen Eigenschaften einzeln im Anmeldeformular an (siehe Abbildung 13) oder er gibt die URL zu einem SLA-Dokument an, welches alle Angaben zu den nicht-funktionalen Eigenschaften enthält. Wurde die URL eins SLA-Dokuments angegeben, so wird das Dokument von der angegebenen URL gelesen und die nicht-funktionalen Eigenschaften daraus extrahiert. Die einzelnen nicht-funktionalen Eigenschaften sind innerhalb des SLA-Dokuments[33] in Form von `ServiceLevelObjective`-Elementen gekapselt, die jeweils einen eigenen Gültigkeitszeitraum definieren können. Das Ende des Gültigkeitszeitraums für das gesamte SLA-Dokument wird hierbei als frühstes Verfallsdatum aller `ServiceLevelObjective`-Elemente bestimmt. Das SLA-Dokument wird, wie auch das WSDL-Dokument, durch das System lokal gesichert, da es einen Teil der Vereinbarung zwischen Anbieter und Nutzer darstellt.

3.1.6 Bewertung der Web Services

Alle Web Services innerhalb einer Kategorie bieten dieselben funktionalen Eigenschaften, da sie dieselbe Funktionalität über dieselbe Schnittstelle anbieten. Damit das System zur Laufzeit feststellen kann, welcher Web Service innerhalb einer Kategorie am besten zur Erbringung der Funktionalität geeignet ist, wird jeder Web Service anhand seiner nicht-funktionalen Eigenschaften bewertet. Die Bewertung erfolgt hierbei durch die Berechnung eines numerischen Wertes, der Score, des Web Services. Je höher die Score eines Web Services, desto besser entspricht der Web Service den Anforderungen des Nutzers an einen Web Service in seiner Kategorie.

Quantifizierbare und „weiche" Eigenschaften
Die Score eines Web Services kann nicht automatisch nach der Anmeldung eines Web Services berechnet werden, da das System nicht in der Lage ist selbständig aus allen nicht-funktionalen Eigenschaften eines Web Services eine Bewertung abzuleiten. Nach der Fähigkeit des Systems aus den Eigenschaften automatisch eine Bewertung ableiten zu können, können alle Eigenschaften disjunkt in die Menge der quantifizierbaren und der „weichen" Eigenschaften aufgeteilt werden.

Quantifizierbare Eigenschaften enthalten intrinsisch, bedingt durch ihre Spezifikation in quantitativer Form, eine Vergleichsmöglichkeit durch das System. Beispielsweise kann das System unter den beiden Antwortzeiten 1.000 ms und 5.000 ms selbständig das bessere Antwortzeitverhalten erkennen und die kürzere Antwortzeit besser bewerten als eine längere Antwortzeit.

[33] Der genaue Aufbau eines SLA-Dokuments wird in Kapitel 4.4.6 beschrieben.

Weiche Eigenschaften[34] können im Gegensatz dazu nicht automatisch miteinander verglichen werden. Dem System ist weder bekannt wie weiche Eigenschaften durch einen Anbieter syntaktisch beschrieben werden, noch was diese Angaben semantisch bedeuten oder wie diese vom Nutzer im konkreten Einsatzbereich des Web Services beurteilt werden. So ist das System beispielsweise nicht in der Lage festzustellen, ob eine Verschlüsselung mittels „DES“ vorteilhafter ist als eine Verschlüsselung mittels „IDEA“[35].

Eine Aufteilung, aller zur Bewertung eines Web Services herangezogen Eigenschaften, in quantifizierbare und weiche Eigenschaften kann Tabelle 2 entnommen werden.

Quantifizierbare Eigenschaften	Weiche Eigenschaften
Verfügbarkeit	Verschlüsselung
Durchsatz	Authentifizierung
Antwortzeit	Autorisierung
Preis	Referenzen
	Reputation

Tabelle 2 - Quantifizierbare und weiche Eigenschaften zur Bewertung von Web Services

Manuelle Punktvergabe für weiche Eigenschaften

Um eine Vergleichbarkeit von weichen Eigenschaften im Rahmen der Bewertung eines Web Services zu erreichen, müssen diese durch den Nutzer individuell bewertet werden. Erst nachdem alle weichen Eigenschaften bewertet wurden, kann eine Bewertung des Web Services als Ganzes erfolgen.

Die Bewertung der weichen Eigenschaften erfolgt durch den Nutzer über das Administrations-Frontend in der Form von ganzzahligen *Punkten* im Wertebereich zwischen 0 und 10 (siehe grün hinterlegte Felder in der Spalte „Points“ in Abbildung 14). Hierbei entsprechen 0 Punkte der Bewertung „ungenügend“ und 10 Punkte der Bewertung „hervorragend“. Punkte zwischen 0 und 10 entsprechen den linearen Abstufungen zwischen den Extrembewertungen.

[34] Aus Gründen der Lesbarkeit wird im Folgenden darauf verzichtet das Adjektiv “weich” im Zusammenhang mit dem Begriff der weichen Eigenschaften in Anführungszeichen zu setzen.

[35] Der “International Data Encryption Algorithm” (IDEA) bezeichnet einen Verschlüsselungsalgorithmus der dem DES-Algorithmus in Bezug auf seine Sicherheit, nicht nur durch eine doppelte Länge des Schlüssels, überlegen ist (siehe hierzu z.B. auch [70] Kapitel 12 und Kapitel 13.9).

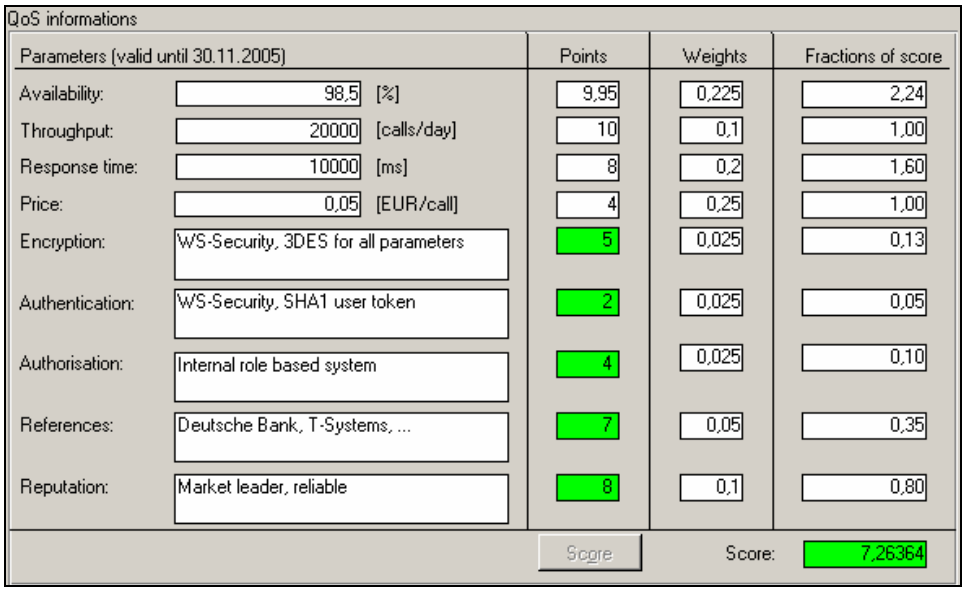

Abbildung 14 - Punktevergabe für weiche Eigenschaften im Administrations-Frontend

Normierung - Berechnung der Punkte quantifizierbarer Eigenschaften

Die Vergabe von Punkten für quantifizierbare Eigenschaften erfolgt automatisch durch das System und wird in Form einer *Normierung* der Eigenschaftswerte unter allen Web Services einer Kategorie vorgenommen. Hierbei erhält der beste Eigenschaftswert innerhalb einer Kategorie 10 Punkte. Alle anderen Eigenschaftswerte erhalten eine Punktzahl (zwischen 0 und 10), welche die relative Größe des Eigenschaftswerts zum besten Eigenschaftswert der Kategorie repräsentiert.

Je nach betrachteter Eigenschaft werden entweder kleinere oder größere Werte als bessere Eigenschaftswerte interpretiert. Im Falle der Verfügbarkeit und des Durchsatzes werden höhere Werte besser eingestuft als niedrigere Werte. Eine Normierung dieser Eigenschaften erfolgt daher am jeweiligen Maximalwert der Eigenschaften in einer Kategorie. Die Antwortzeit und der Preis werden hingegen als besser eingestuft, je niedriger die spezifizierten Werte sind. Eine Normierung erfolgt in diesem Fall an dem jeweiligen Minimalwert in einer Kategorie.

Existieren in einer Kategorie n Web Services und werden die Eigenschaftswerte der quantifizierbaren Eigenschaften eines Web Services i mit $v_{i,j}$ bezeichnet, wobei i=1, …, n und j=1, …, 4, so ergibt sich die Berechnung der Punkte $p_{i,j}$ eines Eigenschaftswertes gemäß folgender Tabelle 3.

Eigenschaft	Eigenschaftswert	Punkte
Verfügbarkeit	$v_{i,1}$	$p_{i,1} = \frac{v_{i,1}}{\max(v_{1,1}, v_{2,1}, ..., v_{n,1})} \bullet 10$
Durchsatz	$v_{i,2}$	$p_{i,2} = \frac{v_{i,2}}{\max(v_{1,2}, v_{2,2}, ..., v_{n,2})} \bullet 10$
Antwortzeit	$v_{i,3}$	$p_{i,3} = \frac{\min(v_{1,3}, v_{2,3}, ..., v_{n,3})}{v_{i,3}} \bullet 10$
Preis	$v_{i,4}$	$p_{i,4} = \frac{\min(v_{1,4}, v_{2,4}, ..., v_{n,4})}{v_{i,4}} \bullet 10$

Tabelle 3 - Berechnung der Punkte quantifizierbarer Eigenschaften durch Normierung

Ein einfaches Beispiel zur Berechnung der Punkte in einer Kategorie mit zwei Web Services kann der folgenden Tabelle 4 entnommen werden. In der Tabelle sind die Eigenschaftswerte und die Punkte der Maxima und Minima kursiv dargestellt. Die übrigen Punkte ergeben sich als relative Größe zum Maximum, bzw. Minimum, ausgedrückt in zehntel Prozent.

	Web Service 1		Web Service 2	
Eigenschaft	**Eigenschaftswert**	**Punkte**	**Eigenschaftswert**	**Punkte**
Verfügbarkeit [%]	98,5	9,95	***99***	***10***
Durchsatz [Aufrufe/Tag]	***20.000***	***10***	18.000	9
Antwortzeit [ms]	10.000	8	***8.000***	***10***
Preis [EUR/Aufruf]	0,05	4	***0,02***	***10***

Tabelle 4 - Berechnungsbeispiel der Punkte quantifizierbarer Eigenschaften

Wird ein Web Service einer Kategorie hinzugefügt, oder aus dieser gelöscht, so können sich neue Maxima oder Minima für Eigenschaftswerte in dieser Kategorie ergeben. Die Punktvergabe durch die Normierung der Eigenschaftswerte ist daher bei jedem Hinzufügen oder Löschen eines Web Services nötig und wird vom System in der betroffenen Kategorie automatisch durchgeführt. Da sich hierbei die Punkte der quantifizierbaren Eigenschaften ändern können, muss auch die Score für jeden Web Service erneut berechnet werden.

Berechnung der Score

Nachdem die Punkte zur Bewertung der quantifizierbaren Eigenschaften vom System durch Normierung berechnet wurden und der Nutzer die Punkte zur Bewertung der weichen Kriterien spezifiziert hat, kann durch das System die Gesamtbewertung des Web Services, die *Score*, berechnet werden. Die Score eines Web Services ist hierbei die gewichtete Summe aus den Punkten aller seiner Eigenschaften. Zur Gewichtung der Punkte werden die in der Kategorie definierten Gewichte für die Eigenschaften verwendet (vgl. Kapitel 3.1.4).

Die Punkte der Eigenschaftswerte eines Web Services i und die Gewichte der Eigenschaften sollen im Folgenden wie in Tabelle 5 bezeichnet werden.

Eigenschaft	Punkte	Gewicht
Verfügbarkeit	$p_{i,1}$	w_1
Durchsatz	$p_{i,2}$	w_2
Antwortzeit	$p_{i,3}$	w_3
Preis	$p_{i,4}$	w_4
Verschlüsselung	$p_{i,5}$	w_5
Authentifizierung	$p_{i,6}$	w_6
Autorisierung	$p_{i,7}$	w_7
Referenzen	$p_{i,8}$	w_8
Reputation	$p_{i,9}$	w_9

Tabelle 5 - Bezeichnung der Variablen für Punkte und Gewichte der Eigenschaften

Mit diesen Definitionen ergibt sich die Berechnungsvorschrift der Score s_i eines Web Services i als:

$$s_i = \sum_{j=1}^{9} p_{i,j} \bullet w_j$$

Formel 2 - Berechnung der Score eines Web Services

Wurden die Gewichte der Eigenschaften in einer Kategorie wie empfohlen so festgelegt, dass sie in der Summe 1 ergeben, so impliziert dies im Zusammenhang mit dem Wertebereich der Punkte (0 bis 10) eine Score die ebenfalls einen Wertebereich von 0 bis 10 aufweist. Diese Implikation lässt sich wie folgt ausdrücken:

$$\left(\forall_{i=1,\ldots,n}.\forall_{j=1,\ldots,9}.0 \le p_{i,j} \le 10\right) \wedge \left(\sum_{j=1}^{9} w_j = 1\right) \Rightarrow \forall_{i=1,\ldots,n}.0 \le s_i \le 10$$

Formel 3 - Eine Gewichtssumme von 1 impliziert eine Score zwischen 0 und 10

Überprüfung von Mindestanforderungen

Bevor einem Web Service seine Score zugeteilt wird, erfolgt zunächst noch eine Überprüfung der definierten Mindestanforderungen, die durch alle Web Services seiner Kategorie erfüllt werden müssen. Nacheinander werden alle Regeln, durch welche die Mindestanforderungen definiert sind, überprüft (vgl. Kapitel 3.1.4). Kann der Web Service alle Regeln erfüllen, so wird ihm seine berechnete Score zugeteilt und ist damit in das interne Verzeichnis produktiv einsetzbarer Web Services des Systems aufgenommen. Kann auch nur eine Regeln nicht erfüllt werden, so wird der Web Service abgelehnt und ihm die Score -1 zugeteilt. Dieser Wert konfligiert nicht mit Scores von Web Services die alle Regeln erfüllen, da er außerhalb des Wertebereichs einer normalen Score liegt. Anhand der Score lassen sich daher drei Zustände eines Web Services im internen Verzeichnis eindeutig unterscheiden:

Score	Status im internen Verzeichnis
Noch nicht vergeben	Die Bewertung des Web Services durch den Nutzer steht noch aus.
≥ 0	Der Web Service wurde durch den Nutzer bewertet und erfüllt die Mindestanforderungen an Web Services seiner Kategorie.
-1	Der Web Service wurde durch den Nutzer bewertet, konnte jedoch nicht die Mindestanforderungen an Web Services seiner Kategorie erfüllen.

Tabelle 6 - Zusammenhang zwischen Score und Status im internen Verzeichnis

3.1.7 Löschung und Deaktivierung eines Web Services

Löschung

Die Löschung eines Web Services kann aus den verschiedensten Gründen erfolgen. Aus der Sicht des Nutzers könnten Gründe hierfür darin bestehen, dass der Web Service nicht mehr benötigt wird, die Zusammenarbeit mit dem Anbieter eingestellt werden soll, oder die Gültigkeitsdauer des Angebots abgelaufen ist. Aus der Sicht eines Anbieters könnte ein Grund für die Löschung darin bestehen, dass der Web Service in Zukunft nicht mehr, bzw. diesem konkreten Nutzer nicht mehr, angeboten werden soll. Hierbei sollte der Anbieter jedoch bedenken, dass er bei der Anmeldung seines Web Services am Portal, durch die Übermittlung einer Gültigkeitsdauer, dem Nutzer die Nutzung des Web Services für einen bestimmten Zeitraum verbindlich garantiert hat.

Ein Anbieter kann seine angebotenen Web Services über das Portal löschen. Der Nutzer verwendet zum Löschen von Web Services das Administrations-Frontend. Im Administrations-Frontend besteht für den Nutzer zusätzlich die Möglichkeit ganze Kategorien inklusive aller enthaltenen Web Services oder Anbieter inklusiver aller angebotener Web Services zu löschen.

Durch eine Löschung eines Web Services werden alle Informationen bezüglich des Web Services aus dem System entfernt. Der Web Service steht daraufhin nicht mehr zur Verwendung durch das System zur Verfügung. Da eine Löschung nicht rückgängig gemacht werden kann, muss der gelöschte Web Service erneut in der betreffenden Kategorie angemeldet werden, sollte er in Zukunft noch einmal benötigt werden.

Durch die Löschung eines Web Services aus einer Kategorie können sich die minimalen und maximalen Eigenschaftswerte der quantifizierbaren Eigenschaften in dieser Kategorie verändern. Hierdurch müssen die Punkte für die quantifizierbaren Eigenschaften durch eine erneute Normierung neu berechnet werden. Da durch die Änderung der Punkte die bisherige Score eines Web Services ungültig wird, ist auch die Score aller Web Services in dieser Kategorie neu zu berechnen.

Deaktivierung

In der Praxis existiert eine Vielzahl von Fällen, in denen ein Web Service nicht endgültig gelöscht, sondern nur vorübergehend deaktiviert werden soll. Ein Beispiel

für einen solchen Fall ist z.B. das akute Auftreten von Problemen bei der Nutzung eines Web Services. Durch eine Deaktivierung des Web Services kann die Funktionalität an einen anderen Web Service derselben Kategorie übertragen werden und es besteht die Möglichkeit der genauen Problemanalyse, bevor über eine endgültige Löschung des Web Services entschieden werden kann.

Die vorübergehende Deaktivierung eines Web Services ist dem Anbieter nicht möglich, da dies eine vorübergehende Aussetzung seiner Garantie bezüglich der Dienstnutzung bedeuten würde. Ein Nutzer kann jedoch über das Administrations-Frontend Web Services vorübergehend deaktivieren. Hierzu löscht der Nutzer die Score eines Web Services. Dadurch erhält der Web Service denselben Status wie ein Web Service, dessen Bewertung durch den Nutzer noch aussteht und wird daher nicht als produktiver Web Service durch das System verwendet. Um den Web Service zu reaktivieren, lässt der Benutzer die Score des Web Services erneut berechnen, wodurch er vom System wieder als produktiver Web Service verwendet wird.

Im Gegensatz zu einer Löschung müssen bei einer Deaktivierung eines Web Services nicht die Scores aller Web Services in der betroffenen Kategorie neu berechnet werden. Der deaktivierte Web Service bleibt bei seiner Deaktivierung Bestandteil einer Kategorie und wird lediglich innerhalb dieser Kategorie nicht mehr produktiv verwendet. Da sich hierdurch keine Änderung der Minima und Maxima bei den quantifizierbaren Eigenschaften einer Kategorie ergibt, bleibt die Gültigkeit aller Punkte und Scores in der Kategorie erhalten.

3.2 Auswahl und Aufruf eines Web Services

Nachdem im vorangegangenen Kapitel 3.1 erläutert wurde, welche Informationen zu den einzelnen Web Services erfasst werden und wie Web Services durch das System verwaltet werden, wird in diesem Kapitel erläutert wie das System prinzipiell Web-Service-Aufrufe entgegennimmt, ein Ziel für die Weiterleitung des Web-Service-Aufrufs bestimmt, den Aufruf eines externen Web Service vornimmt, Dienstgütekriterien während des Aufrufs misst und Informationen über den Aufruf protokolliert.

3.2.1 Funktionsweise und Systemkomponenten

Um die Systemkomponenten von WSQoSX und ihr Zusammenspiel mit der Umgebung möglichst anschaulich erläutern zu können, wird im Folgenden zunächst eine klassische Web-Service-Umgebung skizziert. Anschließend wird die skizzierte Umgebung um die Systemkomponenten von WSQoSX erweitert und die hierdurch zusätzlich erbrachte Funktionalität erläutert.

Klassische Web-Service-Umgebung

In einem Unternehmen stellt sich eine klassische Umgebung zur Nutzung von Web Services grob wie in Abbildung 15 beschrieben dar.

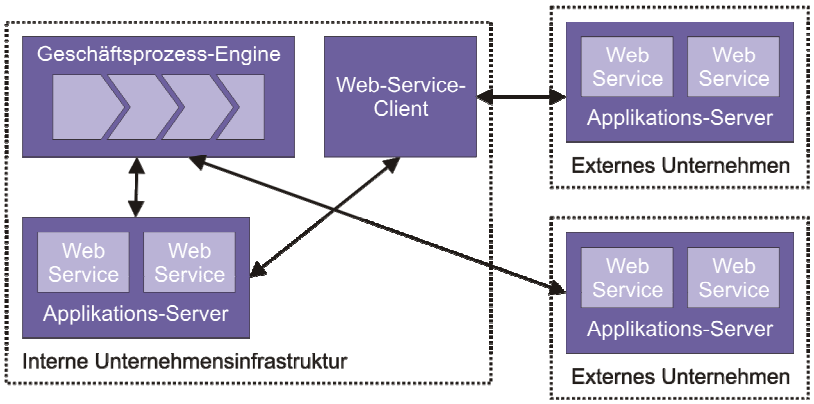

Abbildung 15 - Klassische Web-Service-Umgebung

In einem Unternehmen existieren verschiedenste Web-Service-Clients, die zur Erbringung ihrer Funktionalität Web Services aufrufen. Zu diesen Web-Service-Clients ist auch die Geschäftsprozess-Engine zu zählen, welche die Ausführungsumgebung der Geschäftsprozesse des Unternehmens darstellt. Durch die Web-Service-Clients werden sowohl interne Web Services des Unternehmens, als auch externe Web Services von Anbietern außerhalb des Unternehmens aufgerufen. Da zum Aufruf von Web Services durch die Web-Service-Clients nur statische Netzwerkendpunkte verwendet werden, ist eine dynamische Adressierung von Web Services nicht möglich. Um dynamische Aufrufe von Web Services in einer solchen Umgebung realisieren zu können, müssten alle verwendeten Web-Service-Clients aufwändig überarbeitet und erweitert werden.

WSQoSX-Umgebung

Um mit einem minimalen Aufwand dynamische Web-Service-Aufrufe in einer solchen Umgebung realisieren zu können, wird diese um die Systemkomponenten von WSQoSX ergänzt. Das WSQoSIX-System besteht aus drei Kernkomponenten, die über eine gemeinsame Datenhaltung in einer Datenbank verknüpft sind: Ein generischer *Proxy für Web Services (WSProxy)*, ein *Administrations-Frontend (WSProxyAdmin)* und ein *Web-Portal (WSPortal)*.

Der Aufbau der erweiterten Umgebung wird durch Abbildung 16 veranschaulicht.

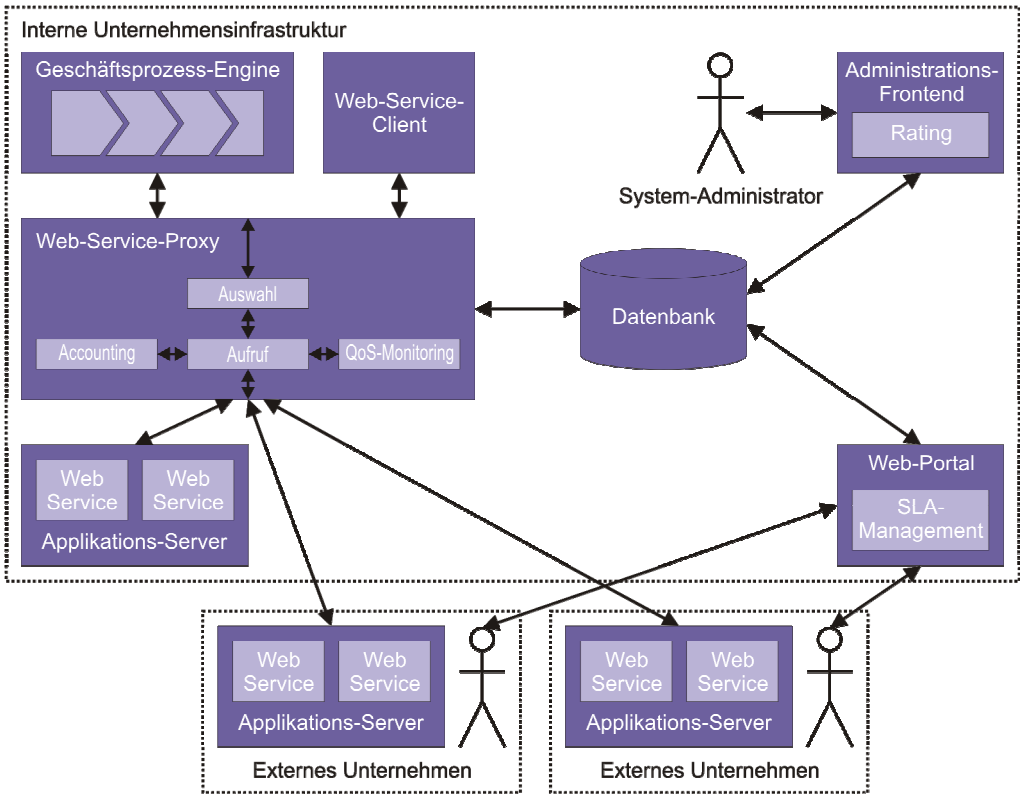

Abbildung 16 - Durch WSQoSX erweiterte Web-Service-Umgebung

Um den Web-Service-Clients den dynamischen Aufruf von Web Services zu ermöglichen, wird die Konfiguration der Netzwerkendpunkte in diesen so verändert, dass sie ihre Web-Service-Aufrufe an den WSProxy senden. Da alle Web-Service-Clients mit statischen Netzwerkendpunkten umgehen können, kann der statische Netzwerkendpunkt des WSProxy im Unternehmensnetzwerk problemlos konfiguriert werden. Für jeden Web Service, der durch einen Web-Service-Client aufgerufen werden soll, wird hierbei der Netzwerkendpunkt des WSProxy unterschiedlich parametrisiert. Ein solcher parametrisierter Netzwerkendpunkt soll im Folgenden als *Request-URL* bezeichnet werden.

Alle auf diese Weise umkonfigurierten Web-Service-Clients senden ihre Web-Service-Aufrufe an den WSProxy, anstatt sie direkt an den vormals eingebundenen Web Service zu senden. Der WSProxy empfängt die Web-Service-Aufrufe und erkennt anhand der übermittelten Request-URL welchen Web Service der Web-Service-Client aufrufen möchte. Über das Administrations-Frontend kann der System-Administrator festlegen wie der WSProxy die Aufrufe der einzelnen Web Services vorzunehmen hat. Ein Aufruf kann hierbei an einen festen Web Service weitergeleitet werden, oder das Weiterleitungsziel kann dynamisch zur Zeit des Aufrufs ermittelt werden. Soll das Ziel der Weiterleitung dynamisch bestimmt werden, so kann der System-Administrator eine Kategorie (vgl. Kapitel 3.1.2) definieren, aus der ein geeigneter Web Service als Ziel bestimmt werden soll. Bei der Auswahl eines Web Services aus einer Kategorie ermittelt die Auswahl-Komponente des WSProxy den am besten in der Kategorie verfügbaren Web Service. Was hierbei als optimal angesehen wird, kann durch den System-Administrator im Administrations-Frontend spezifisch für jede Kategorie mittels einer Präferenzstruktur festgelegt werden.

Steht das Ziel der Weiterleitung des Web-Service-Aufrufes fest, so ruft die Aufruf-Komponente des WSProxy den Web Service auf, empfängt die Antwort vom aufgerufenen Web Service und gibt diese an den Web-Service-Client zurück. Während des Aufrufes werden durch die QoS-Monitoring-Komponente des WSProxy bestimmte Dienstgüteparameter des Aufrufs gemessen und protokolliert. Verstoßen hierbei gemessene Dienstgüteparameter gegen die Dienstgüte, die ein Anbieter in einem SLA garantiert hat, so wird der System-Administrator benachrichtigt. Die Accounting-Komponente des WSProxy protokolliert während der gesamten Verarbeitung eines Aufrufs eine Vielzahl von Informationen der einzelnen Abarbeitungsschritte. Aufgrund dieser Informationen ist u.a. möglich den verarbeiteten Web-Service-Aufrufen die einzelnen aufrufenden Web-Service-Clients zuzuordnen und mithilfe dieser Zuordnung eine hausinterne Kostenrechnung durchzuführen. Eine Auswertung der Datenbasis ermöglicht zudem die Kontrolle der Abrechnung von Nutzungsentgelten mit den Anbietern von Web Services.

Die Interaktion der Anbieter von Web Services mit dem System erfolgt über das Web-Portal. Hier können sich die Anbieter über offene Ausschreibungen für Web Services in Form von Kategorien informieren und sich mit passenden Web Services am Portal bewerben. Bei der Bewerbung garantieren die Anbieter mittels einem verbindlichen SLA eine gewisse Dienstgüte und die Rahmenbedingungen der Nutzung des Web Services. Bewerbungen werden durch den System-Administrator über das Administrations-Frontend bearbeitet. Akzeptiert der System-Administrator die Bewerbung, so bewertet er die weichen Eigenschaften (vgl. Kapitel 3.1.6) eines Web Services. Das System prüft daraufhin die Einhaltung gewisser Mindestanforderungen durch den Web Service. Erfüllt der Web Service alle Mindestanforderungen, so wird ihm eine Gesamtbewertung zugeteilt und in die Menge der produktiv durch das System verwendeter Web Services aufgenommen.

Betrachtung

Der WSProxy agiert in dieser Umgebung als Zwischeninstanz zwischen Web-Service-Client und Web Service. Durch diesen zusätzlichen Indirektionsschritt beim Aufrufen von Web Services wird ein dynamisches Routing von Web-Service-Aufrufen möglich. Der WSProxy könnte daher auch als Web-Service-Router bezeichnet werden. Ein herkömmlicher Router bestimmt das Ziel von Datenpaketen gemäß ihres Ziels und den bekannten Informationen über die Wege, die zu diesem Ziel führen. Der WSProxy bestimmt das Ziel von Web-Service-Aufrufen gemäß des benötigten Web Services und den bekannten Informationen über Web Services die dieselbe Funktionalität bereitstellen.

Der WSProxy lässt sich leicht in eine bestehende Web-Service-Infrastruktur einbinden, da sowohl von den Web-Service-Clients, als auch von den Web Services keine speziellen Protokolle zum Aufruf verwendet werden müssen. Es wird keine spezielle API zur Programmierung von Clients und Web Services benötigt und alle beteiligten Protokolle (TCP/IP, HTTP, SOAP) werden in unveränderter Form verwendet. Das Handling der Aufrufe bleibt für die Clients unverändert, mit der einzigen Ausnahme, dass sie den Aufruf an ein anderes Ziel senden. Für aufgerufene Web Services ist der Umweg der Aufrufe über den WSProxy vollkommen transparent, d.h. sie erhalten Aufrufe auf dieselbe Art und Weise wie zuvor und antworten auf diese in gewohnter Art und Weise, wodurch sich für sie keinerlei Änderungen ergeben.

Das Routing der Web-Service-Aufrufe kann durch den System-Administrator über das Administrations-Frontend bequem und flexibel festgelegt werden. Die Menge der Web Services, die durch das System als potentielle Weiterleitungsziele verwendet werden, lässt sich über das Administrations-Frontend ebenso bequem verwalten. Über Ausschreibungen von Web Services in einem Portal können auch externe Unternehmen ihre Web Services zur Verwendung durch das System vorschlagen. Durch die automatische Messung von Dienstgüteparametern und dem Vergleich mit SLA-Zusicherungen wird der System-Administrator über mögliche Performance-Probleme frühzeitig informiert. Eine Kostenkontrolle wird durch die Auswertung von Logging-Informationen möglich. Sowohl eine interne Kostenrechnung als auch die Überprüfung der Abrechnung von Nutzungsentgelten externer Anbieter ist möglich.

Die durch WSQoSX erweiterte Web-Service-Umgebung bietet daher eine Vielzahl zusätzlicher Funktionalitäten bei einem Minimum an nötigen Voraussetzungen der Integration.

3.2.2 Routing der Web-Service-Aufrufe

Der WSProxy fungiert bei einem Web-Service-Aufruf als Zwischeninstanz zwischen Web-Service-Client und Web Service und übernimmt das *Routing* des Web-Service-Aufrufs. Der hierbei auftretende Ablauf des Aufrufvorgangs wird durch das Sequenzdiagramm in Abbildung 17 veranschaulicht.

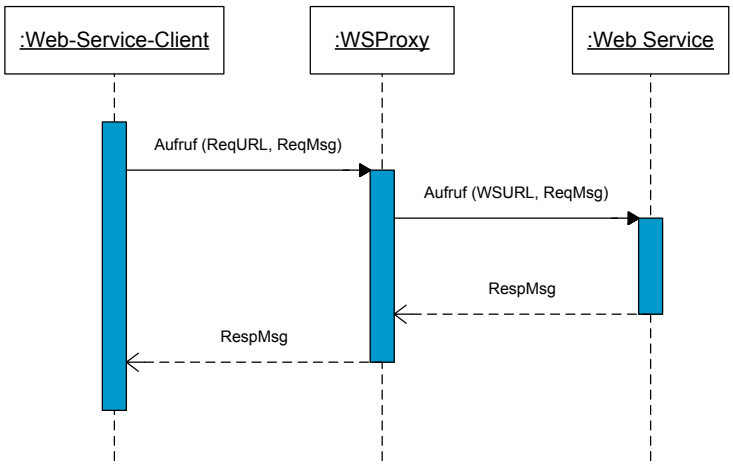

Abbildung 17 - Indirekter Aufruf eines Web Services über den WSProxy

Zum Aufruf eines Web Services generiert der Client eine (SOAP-)Nachricht `ReqMsg` und sendet diese über eine statische Request-URL `ReqURL` an den WSProxy. Die Request-URL signalisiert dem WSProxy über einen parametrisierten Teil den aufzurufenden Web Service. Im WSProxy ist für jeden Web Service eine Routing-Methode hinterlegt, mit der eine URL zum Aufruf des Web Services bestimmt werden kann. Durch Anwendung dieser Methode bestimmt der WSProxy die konkrete URL eines Web Services (`WSURL`) an den der Aufruf weiterzuleiten ist. Die Nachricht `ReqMsg` wird anschließend vom WSProxy an die `WSURL` gesendet und dort von einem Web Service empfangen. Der Web Service verarbeitet die Nachricht und sendet eine Antwortnachricht (`RespMsg`) zum WSProxy zurück. Dieser leitet die Antwortnachricht wiederum an den Client weiter.

Übergabe des Web-Service-Identifiers (WSID) in der Request-URL

Wie bereits erwähnt, wird das Routing des WSProxy durch einen Teil der URL bestimmt, die zum Aufruf des WSProxy verwendet wurde. Da der WSProxy nur über das Protokoll HTTP angesprochen werden kann, handelt es sich bei diesen URLs ausschließlich um URIs [15] nach dem HTTP-Schema [32]:

HTTP-URL = `http://<host>[:<port>][<abs_path>[?<query>]]`

Die Basis-URL zur Adressierung des WSProxy umfasst dabei den Teil „`http://<host>:<port>`“. In der Basis-URL wird das Protokoll festegelegt mit welchem der WSProxy angesprochen wird (HTTP), der Rechner auf dem der WSProxy als Server ausgeführt wird (Host), sowie der Port unter dem der WSProxy TCP-Verbindungen annimmt. Diese Angaben sind für die Adressierung und den Aufbau einer HTTP-Verbindung zum WSProxy zwingend erforderlich. Alle weiteren Teile, die eine HTTP-URL enthalten kann, wie Pfad- und Parameterangaben, werden beim Aufruf des WSProxy im HTTP-Header übergeben und stehen dem WSProxy zur Weiterverarbeitung zur Verfügung. Derjenige Teil der URL, welcher über die Basis-URL hinausgeht, soll im Folgenden als *Web-Service-Identifier (WSID)* bezeichnet werden. Das Schema für URLs zum Aufruf von Web Services über den WSProxy lautet daher:

WSProxy-URL = `http://<host>:<port><WSID>`

Ein Beispiel: Der WSProxy-Server wurde auf einem Rechner, der im Netzwerk unter der IP-Adresse 192.168.0.10 erreichbar ist, in seinen Standard-Einstellungen installiert und ist daher unter Port 8081 erreichbar. Die Basis-URL des WSProxy lautet in diesem Fall „`http://192.168.0.10:8081`“. Ruft ein Client einen Web Service mittels der URL „`http://192.168.0.10:8081/creditprocess/bank`“ über den WSProxy auf, so interpretiert WSProxy den Teil „`/creditprocess/bank`“ als WSID.

Unique Service Identifier (USID) und Suchverfahren

Mittels des WSIDs teilt der aufrufende Client dem WSProxy mit, welchen Web Service er aufrufen möchte, jedoch nicht, über welche URL dieser Web Service erreicht werden kann. Wie der WSProxy zu einem WSID eine entsprechende URL ermitteln soll, wird über ein Target definiert, welches dem WSID über einen USID zugeordnet wird. Ein *Unique Service Identifier (USID)* ist hierbei nichts anderes als ein eindeutiger Bezeichner für einen Web Service. Jedem USID wird ein *Target* zugeordnet, das aus einem Suchverfahren und einem Suchparameter besteht. Um zur Laufzeit die URL eines geeigneten Web Service zur Weiterleitung des Web-Service-Aufrufs zu bestimmen, wird die im Target spezifizierte *Suchmethode* mit dem *Suchparameter* ausgeführt. Die Syntax zur Angabe eines Targets ist:

Target = `<Suchverfahren>:<Suchparameter>`

Als Suchverfahren können „static“ und „portal“ angegeben werden, wobei die Suchparameter spezifisch für jedes Suchverfahren sind.

Suchverfahren „static“

Das Suchverfahren „static“ wird verwendet um statische Weiterleitungen von Web-Service-Aufrufen an bestimmte Web Services zu realisieren. Bei diesem „Suchverfahren“ handelt es sich nicht um ein Suchverfahren im eigentlichen Sinne, da keine wirkliche Suche stattfindet, sondern lediglich der im Target angegebene Suchparameter als Ergebnis zurückgegeben wird. Wird in einem Target als Suchparameter die URL eines Web Services angegeben, so ist das Ergebnis der „Suche“ stets die angegebene URL.

Suchverfahren „portal“

Im Gegensatz zum Suchverfahren „static“ stellt das Suchverfahren „portal“ eine Suche im eigentlichen Sinne dar. Als Suchparameter wird die Kategorie eines Web Services angegeben. Zur Laufzeit ermittelt das Suchverfahren dynamisch den bestmöglichen Web Service aus der angegebenen Kategorie. Konkret wählt das Suchverfahren den Web Service aus, der zur Zeit der Suche in dieser Kategorie die höchste Score besitzt. Die URL des ermittelten Web Services wird als Ergebnis der Suche zurückgeliefert.

Konfiguration

Die Konfiguration des Routings wird durch den System-Administrator über das Administrations-Frontend vorgenommen. Ein Beispiel für eine Routing-Konfiguration kann der Abbildung 18 entnommen werden.

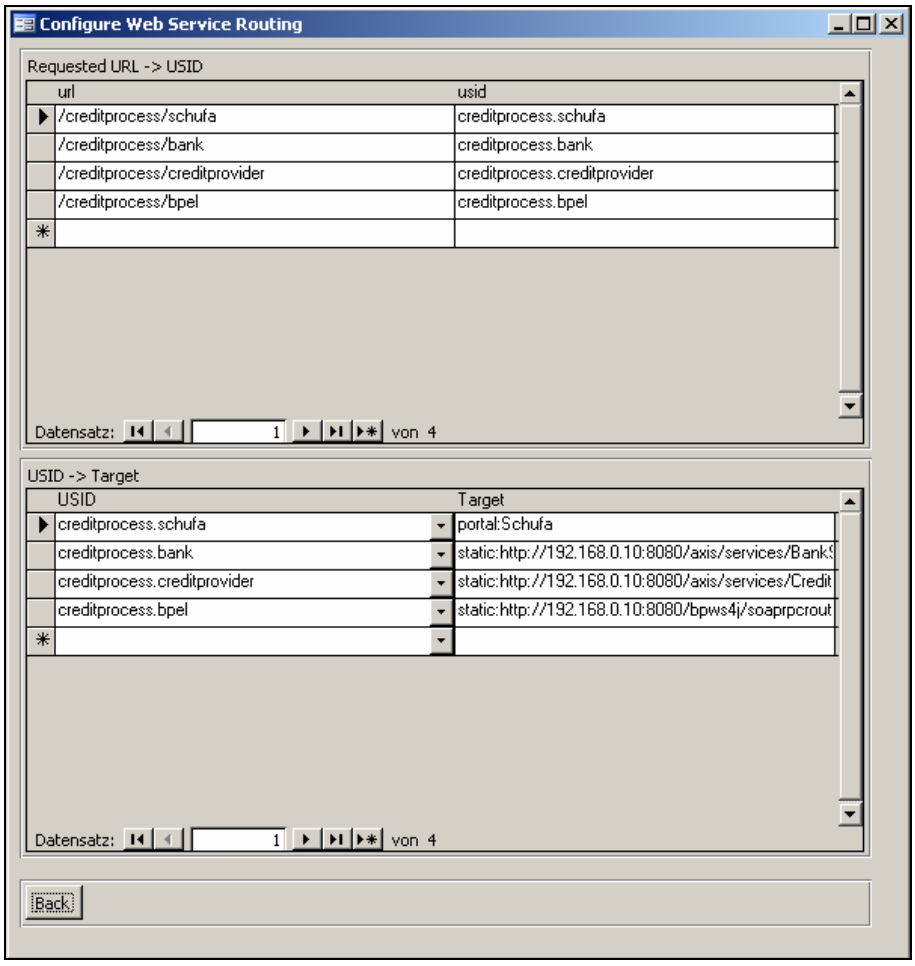

Abbildung 18 - Konfiguration des Routings über WSID, USID und Target

Im Beispiel wird dem WSID „`/creditprocess/schufa`“ der USID „`creditprocess.schufa`“ zugeordnet. Dem USID „`creditprocess.schufa`“ wird wiederum das Target „`portal:Schufa`“ zugeordnet. Durch diese beiden Zuordnungen ergibt sich eine Zuordnung des WSIDs „`/creditprocess/schufa`“ zum Target „`portal:Schufa`“. Ist die Basis-URL des WSProxy-Servers beispielsweise „`http://192.168.0.10:8081`“ und ein Web-Service-Client sendet die SOAP-Nachricht eines Web-Service-Aufrufs an die URL „`http://192.168.0.10:8081/creditprocess/schufa`“, so ermittelt der WSProxy den Web Service in der Kategorie „Schufa“ der zu diesem Zeitpunkt die höchste Score besitzt und sendet den Web-Service-Aufruf an die URL des ermittelten Web Services. Nach Erhalt der Antwort des Web Services durch den WSProxy wird diese an den ursprünglich aufrufenden Web-Service-Client weitergegeben.

Dem WSID „`/creditprocess/bank`“ wird im Beispiel über den USID „`creditprocess.bank`“ das Target „`static:http://192.168.0.10:8080/axis/services/BankService`“ zugeordnet. Bei einem Web-Service-Aufruf an die URL „`http://192.168.0.10:8081/creditprocess/bank`“ erfolgt daher stets eine Weiterleitung des Aufrufs an die URL „`http://192.168.0.10:8080/axis/services/ BankService`“.

In diesen Beispielen macht die Verwendung eines USIDs scheinbar keinen Sinn, da ein Target auch direkt einem WSID zugeordnet werden könnte. Soll ein und derselbe

Web Service jedoch unter mehreren WSIDs zur Verfügung gestellt werden, so kann mehreren WSIDs derselbe USID zugeordnet werden. Wird dann das Target der USID verändert, so wirkt sich dies auf alle WSIDs aus, die dem USID zugeordnet sind. Dem System-Administrator wird hierdurch die Arbeit erspart alle WSIDs durchgehen und alle Targets einzeln korrigieren zu müssen. USIDs dienen einzig und alleine dem Zweck solche Konfigurationsszenarien komfortabel zu unterstützen.

3.2.3 QoS-Monitoring

Während des Aufrufs eines Web Services durch den WSProxy wird die tatsächlich bei diesem Aufruf durch den Web Service erbrachte technische Dienstgüte gemessen, bzw. anhand von Logging-Informationen berechnet. Die Ergebnisse werden einzeln für jeden Aufruf in einer *Call-History* gespeichert. Zusätzlich wird zu jedem Web Service eine *History-Statistik* geführt, in der alle Informationen über das bisherige Dienstgüteverhalten in Kennzahlen zusammengefasst werden.

Die QoS-Monitoring-Komponente sammelt hierbei zu jedem Web Service Informationen über folgende Dienstgütekriterien:

- Antwortzeit
- Verfügbarkeit
- Durchsatz

Antwortzeit

Bevor der WSProxy die Aufrufnachricht an einen Web Service sendet, wird die Startzeit des Aufrufs protokolliert. Nachdem die Antwort des Web Services vollständig empfangen wurde, wird die Endzeit des Aufrufs protokolliert. Aus der Differenz zwischen Start- und Endzeit wird die Dauer des Aufrufs, und damit die Antwortzeit des Web Services, berechnet. Die ermittelte Antwortzeit wird in der Call-History gespeichert und die durchschnittliche Antwortzeit in der History-Statistik aktualisiert.

Wurde ein Web Service i m-mal erfolgreich aufgerufen und bezeichnet t_{ij} die j-te gemessene Antwortzeit (j=1, …, m) des Web Services, so wird die durchschnittliche Antwortzeit $\bar{t}_i$ des Web Services i folgendermaßen berechnet:

$$\bar{t}_i = \frac{\sum_{j=1}^{m} t_{ij}}{m}$$

Formel 4 - Durchschnittliche Antwortzeit des Web Services i

Verfügbarkeit

Bei jedem Aufruf eines Web Services, für den eine Antwortzeit ermittelt werden kann, wird von der Verfügbarkeit des Web Services ausgegangen, da in diesem Fall die Aufrufnachricht erfolgreich an den Web Service gesendet und eine Antwort empfangen werden konnte. Ein Web-Service wird als nicht verfügbar angesehen, falls die Aufrufnachricht nicht an den Web Service übermittelt werden konnte, weil z.B. keine HTTP-Verbindung zum Web Service aufgebaut werden konnte. Kommt es während der Übermittlung der Aufrufnachricht oder während des Empfangs der Antwort zu einem Abbruch der HTTP-Verbindung, so gilt der Web Service ebenfalls

als nicht verfügbar. Bricht die HTTP-Verbindung nicht ab, es trifft jedoch innerhalb der doppelten, in der SLA garantierten Antwortzeit keine Antwort vom Web Service ein, so gilt der Web Service ebenso als nicht verfügbar.

Die Verfügbarkeit eines Web Services ist unabhängig davon, ob die Übermittlung der Antwort des Web Services an den Client fehlschlägt. Nachdem der Web Service erfolgreich aufgerufen wurde ist es für dessen Verfügbarkeit nicht relevant ob die weitere Verarbeitung seiner Antwort durch den WSProxy oder den Client fehlschlägt.

Nachdem feststeht, ob ein Web Service bei einem Aufruf als verfügbar oder nicht verfügbar angesehen werden kann, wird die Verfügbarkeit in der Call-History gespeichert und die durchschnittliche Verfügbarkeit des Web Services in der History-Statistik aktualisiert.

Wurde ein Web Service i m-mal aufgerufen und wird hierbei die Verfügbarkeit als a_{ij} (i=1, …, n und j=1, …, m) festgehalten, indem $a_{ij} = 1$ falls Web Service i bei Aufruf j verfügbar war und $a_{ij} = 0$ falls Web Service i bei Aufruf j nicht verfügbar war, so wird die durchschnittliche Verfügbarkeit $\overline{a}_i$ eines Web Services i wie folgt berechnet:

$$\overline{a}_i = \frac{\sum_{j=1}^{m} a_{ij}}{m} \bullet 100$$

Formel 5 - Durchschnittliche Verfügbarkeit des Web Services i

Die durchschnittliche Verfügbarkeit wird hierbei als prozentualer Anteil der Aufrufe des Web Services i, bei denen der Web Service als verfügbar angesehen wurde, zu der Anzahl aller Aufrufe des Web Services i berechnet.

Durchsatz

Jeder Aufruf eines Web Services wird in der Call-History protokolliert. Anhand der Angabe zur Verfügbarkeit können erfolgreiche Web-Service-Aufrufe identifiziert werden. Nach jedem erfolgreichen Aufruf wird die Call-History nach weiteren erfolgreichen Aufrufen des soeben aufgerufenen Web Services selektiert und nach verschiedenen zeitlichen Intervallen gruppiert. Hieraus wird der maximale, bisher durch diesen Web Service erzeugte Durchsatz pro Monat, Tag und Stunde ermittelt und in der History-Statistik des Web Services aktualisiert.

Im Gegensatz zu der durchschnittlichen Antwortzeit und der durchschnittlichen Verfügbarkeit, können die Angaben zum maximalen Durchsatz in den verschiedenen Zeitintervallen nicht zur Kontrolle der Garantien der SLA verwendet werden. Lediglich in dem Fall, dass der durch das System verursachte Durchsatz über dem Durchsatz liegt, den der Anbieter in der SLA garantiert hat, könnte festgestellt werden, dass der Web Service in der Lage ist den Durchsatz leistungsmäßig zu erbringen. Sinkt die Verfügbarkeit und die Antwortzeit bereits vor dem Erreichen des garantierten Durchsatzes ab, so kann dies vielfältige Ursachen haben. Ein direkter Rückschluss darauf, dass der Web Service des Anbieters nicht in der Lage ist den garantierten Durchsatz zu erbringen, kann nicht geschlossen werden. Eine Messung in der Form der Erfassung anderer, messbarer Dienstgütekriterien bei einer künstlich herbeigeführten Spitzenbelastung verbietet sich alleine schon aufgrund der Tatsache,

dass für jeden Web-Service-Aufruf ein Nutzungsentgelt zu entrichten ist. Spitzenbelastungstests wären daher extrem kostenintensiv.

Die Angaben bezüglich der Maximaldurchsätze in der History-Statistik können jedoch in hervorragender Weise dazu verwendet werden um die Spitzenbelastungen, die ein Web Service verursacht, abschätzen zu können und entsprechende Zusicherungen von den Anbietern von Web Services zu verlangen.

Kontrolle der Dienstgüteparameter

Das QoS-Monitoring beschränkt sich nicht nur auf die Messung von Dienstgüteparametern, sondern schließt auch das Warnen des Nutzers bei kritischen gemessenen Werten ein. Zu jeder *Warnung* wird der genaue Zeitpunkt ihres Auftretens, ein *Warnungstyp*, sowie eine *Warnmeldung* festgehalten.

Es lassen sich drei Warnungstypen unterscheiden:

- Nicht-Verfügbarkeit (not available)
- SLA-Verletzung (SLA violation)
- SLA abgelaufen (SLA expired)

Eine Warnung des Typs „Nicht-Verfügbarkeit“ wird generiert, falls das System bei der Weiterleitung des Web-Service-Aufrufs die Nicht-Verfügbarkeit des Web Services festgestellt hat. Eine Warnung des Typs „SLA-Verletzung“ wird generiert, falls bei einem Web-Service-Aufruf Dienstgüteparameter gemessen wurden, welche die durch den Anbieter zugesicherten Dienstgüteparameter verletzten. Hierbei wird unterschieden ob die Dienstgüteparameter einmalig verletzt wurden, oder ob das durchschnittliche Verhalten des Web Services die Dienstgüteparameter verletzt. Eine Warnung des Typs „SLA abgelaufen“ wird bei jedem Aufruf eines Web Services ausgelöst, dessen Zusicherungen durch den Anbieter abgelaufen sind.

Ein einziger Web-Service-Aufruf kann hierbei eine Reihe von Warnmeldungen auslösen. Wurde z.B. ein Web Service aufgerufen der nicht verfügbar war, dadurch seine durchschnittliche Verfügbarkeit unter den per SLA garantierten Wert abgefallen ist und dessen SLA nicht mehr gültig ist, würde Warnungen aller drei Typen erzeugen.

Die generierten Warnungen sind spezifisch für das Suchverfahren, welches zum Routing des Web-Service-Aufrufs verwendet wurde.

Warnungen bei „static“-Routing

Wurde das Suchverfahren „static“ im Target für das Routing verwendet, so ist dem System lediglich die URL bekannt, an welche der Web-Service-Aufruf weitergeleitet werden soll. Weitere Informationen über den aufgerufenen Web Service liegen dem System nicht vor. Dadurch ist es dem System nicht möglich bei „static“-Routing gemessene Dienstgüteparameter mit Vorgaben, z.B. aus einem SLA, zu vergleichen. Warnungen des Typs „SLA-Verletzung“ und „SLA abgelaufen“ können daher bei „static“-Routing nicht generiert werden. In diesem Falle können lediglich Warnungen des Typs „Nicht-Verfügbarkeit“ generiert werden.

Ein Beispiel: Einem WSID wird über einen USID das Target „`static: http://192.168.0.10:8080/bpws4j/soaprpcrouter`“ zugeordnet. Die ersten

drei Aufrufe des Clients der WSID gelingen, jedoch kann beim vierten Aufruf unter der statisch definierten URL kein Web Service erreicht werden. Das System generiert daher unter Angabe des Zeitpunkts des Aufrufs und des Typs „Nicht-Verfügbarkeit“ folgende Warnmeldung: „`Static routing: Service has not been available at endpoint URL 'http://192.168.0.10:8080/bpws4j/soaprpcrouter'. Status: 'remote call io error'. Average availability is 75.0% (measured over 4 calls).`“

Warnungen bei „portal“-Routing
Wurde das Suchverfahren „portal“ im Target verwendet, so wird der Web Service mit der höchsten Score aus der angegebenen Kategorie zur Weiterleitung des Web-Service-Aufrufes herangezogen. Da durch die Suche über das Portal auch Informationen über die durch den Anbieter garantierte Dienstgüte vorliegen, können hierbei zusätzlich zu Warnungen des Typs „Nicht-Verfügbarkeit“ auch Warnungen des Typs „SLA-Verletzung“ und „SLA abgelaufen“ generiert werden.

Warnungen des Typs „SLA-Verletzung“ werden in den folgenden Fällen generiert:

- Die bei dem Aufruf gemessene Antwortzeit überschreitet die durch den Anbieter garantierte Antwortzeit.
- Die sich nach einem Aufruf ergebende durchschnittliche Antwortzeit überschreitet die durch den Anbieter garantierte Antwortzeit.
- Die sich nach einem Aufruf ergebende durchschnittliche Verfügbarkeit unterschreitet die durch den Anbieter garantierte Verfügbarkeit.

Wird bei dem Vergleich mit den Garantien des Anbieters festgestellt, dass die Garantien des Anbieters abgelaufen sind, so wird eine Warnung des Typs „SLA abgelaufen“ erzeugt.

3.2.4 Logging und Accounting

Während der Abarbeitung eines Aufrufes durch den WSProxy werden neben der Messung der technischen Dienstgüte und der eventuellen Ausgabe von Warnungen eine Vielzahl weiterer Informationen in einer *Logging-Tabelle* erfasst. Die hier gesammelten Informationen können bei dem Auftreten von Fehlern zur Analyse der Ursache verwendet werden und können als Datenbasis für weitere Auswertungen, z.B. im Zusammenhang mit dem Accounting verwendet werden.

Die in der Logging-Tabelle zu jedem durch den WSProxy abzuarbeitenden Aufruf gespeicherten Informationen können wie folgt untergliedert werden.

Informationen zum Aufruf durch den Client
Zur Initiierung eines Aufrufs über den WSProxy sendet der Client eine HTTP-Nachricht an den WSProxy. Hierbei wird die eingehende HTTP-Nachricht, die IP-Adresse und der Port des Clients sowie der Aufrufzeitpunkt protokolliert.

Informationen zur Bestimmung des Routing-Ziels
Nachdem die HTTP-Nachricht des Clients empfangen wurde, wird der WSID aus dem HTTP-Header extrahiert. Zu dem übermittelten WSID wird über den verknüpften USID das zu verwendende Target ermittelt. Der extrahierte WSID, der verknüpfte USID und das verknüpfte Target werden protokolliert.

Nachdem das im Target spezifizierte Suchverfahren verwendet wurde um den Web Service zur Weiterleitung zu suchen, wird ein, für das jeweilige Suchverfahren eindeutiger, Identifikator für den gefundenen Web Service, sowie die URL zum Aufruf des Web Services protokolliert.

Informationen zum Aufruf des externen Web Services
Die ursprünglich vom Client empfangene HTTP-Nachricht wird an die ermittelte URL des Web Services gesendet. Hierbei wird die gesendete HTTP-Nachricht, sowie der Sendezeitpunkt protokolliert.

Nachdem eine Antwort vom aufgerufenen Web Service empfangen wurde, wird der Empfangszeitpunkt, sowie die empfangene HTTP-Nachricht protokolliert. Aus der Differenz zwischen Empfangs- und Sendezeitpunkt wird zusätzlich die Antwortzeit des Web Services protokolliert.

Informationen zur Rücksendung der Antwort an den Client
Die vom Web Service empfangene HTTP-Nachricht wird an den Client gesendet. Hierbei wird die gesendete HTTP-Nachricht, sowie der Absendezeitpunkt an den Client protokolliert.

Generelle Informationen zur Abarbeitung des Aufrufs
Bei jeder Änderung des Protokolleintrages wird der Zeitpunkt dieser letzten Änderung festgehalten. Der momentane Status der Abarbeitung eines Web-Service-Aufrufs wird durch spezielle Status-Codes protokolliert (siehe auch Kapitel 4.3.9). Hierdurch ist jederzeit ersichtlich in welcher Phase der Abarbeitung durch den WSProxy sich ein Web-Service-Aufruf befindet. Sollte die Verarbeitung durch das Auftreten von unvorhergesehenen Fehlern unkontrolliert abbrechen, so kann anhand des Statuscodes ersehen werden in welcher Phase die Verarbeitung diese unerwartet unterbrochen wurde. Wird eine HTTP-Fehlermeldung an den Client gesendet, z.B. weil der durch den Client gesendete WSID unbekannt ist, oder der Aufruf des Web Services durch den WSProxy fehlgeschlagen ist, so wird der gesendete HTTP-Fehlercode protokolliert. Anhand des HTTP-Fehlercodes können auch bei einer erfolgreichen Verarbeitung eines Web-Service-Aufrufes über alle Phasen hinweg aufgetretene Fehler im Zusammenhang mit dem aufrufenden Client oder dem aufgerufenen Web Service erkannt werden.

Accounting

Durch das Festhalten von Detailinformationen zu jedem einzelnen durch den WSProxy verarbeiteten Web-Service-Aufruf in der Logging-Tabelle und der Call-History, entsteht eine umfangreiche Datenbasis, die zu den verschiedensten Zwecken ausgewertet werden kann. Ein mögliches Auswertungsszenario besteht in der internen oder externen Kostenabrechnung. Auswertungen dieser Art werden durch das WSQoSX-System selbst nicht vorgenommen, jedoch stellt das System eine entsprechende Datenbasis zur Verfügung um Auswertungen dieser Art vornehmen zu können. Möglichkeiten der internen und externen Kostenabrechnungen werden daher im Folgenden nur kurz skizziert.

Interne Kostenabrechnung
Intern, im Netzwerk des Nutzers, kann ein Aufrufer eines Web Services durch seine IP-Adresse identifiziert werden. Die durch einen Aufruf des Web Services

entstehenden Kosten können über die hinterlegten SLA-Informationen gewonnen werden. Kosten die durch die Web-Service-Aufrufe an sich entstehen, können so über identifizierte Clients in die Kostenstellenrechnung des Unternehmens integriert werden. Ist für die Ermittlung der Kosten von Web-Service-Aufrufen auch das hierdurch erzeugte Transfervolumen relevant, so können über die in der Logging-Tabelle protokollierten, gesendeten und empfangenen HTTP-Nachrichten die Transfervolumina im internen und externen Netz ermittelt und verrechnet werden.

Externe Kostenabrechnung
Anbieter, die Web Services über das Portal am System angemeldet haben, haben einen Preis pro Aufruf ihres Web Services hinterlegt. In regelmäßigen Abständen werden daher die Anbieter den Nutzer für die Inanspruchnahme der angebotenen Web Services belasten. Um die abgerechneten Beträge kontrollieren zu können, können die erfolgreichen Aufrufe des abgerechneten Web Services im Abrechnungszeitraum ermittelt, mit dem Preis pro Aufruf multipliziert und so das zu entrichtende Nutzungsentgelt berechnet werden. Hieran können leicht Diskrepanzen mit den Abrechungen der Anbieter erkannt werden. Sollte der Web Service im Abrechnungszeitraum gegen die garantierte Dienstgüte verstoßen haben, so sind hier entsprechende Abzüge einrechenbar, bzw. Zahlungen gänzlich zu verweigern.

4 Architektur und Implementierung von WSQoSX

Nachdem in Kapitel 3 das Konzept von WSQoSX erläutert wurde, wird im vorliegenden Kapitel näher auf dessen Architektur und Implementierung eingegangen. Zu Beginn wird ein Überblick über die Architektur und der an ihr beteiligten Komponenten gegeben. Bevor weiter auf die einzelnen Komponenten eingegangen wird, werden einige Vorbemerkungen zur Distribution des Systems, sowie dessen Konfiguration und seinen Loggingfähigkeiten gemacht. Anschließend werden zunächst solche Komponenten erläutert die das System verwenden, bevor detailliert auf die Kernkomponenten eingegangen wird, welche die eigentliche dienstgüteorientierte Infrastruktur zum Aufruf von Web Services aufbauen.

4.1 Überblick

Die Architektur und Implementierung von WSQoSX lässt sich in zwei Bereiche unterteilen: Einerseits die *Kernkomponenten* des Systems, welche die eigentliche dienstgüteunterstützende Infrastruktur für die Web-Service-Umgebung bereitstellen und andererseits *weitere Komponenten*, welche als Web-Service-Consumer und Web-Service-Provider auf die Infrastruktur des Systems zugreifen. Eine architekturelle Übersicht der Komponenten kann Abbildung 19 entnommen werden (vergleiche auch Kapitel 3.2.1, Abbildung 16).

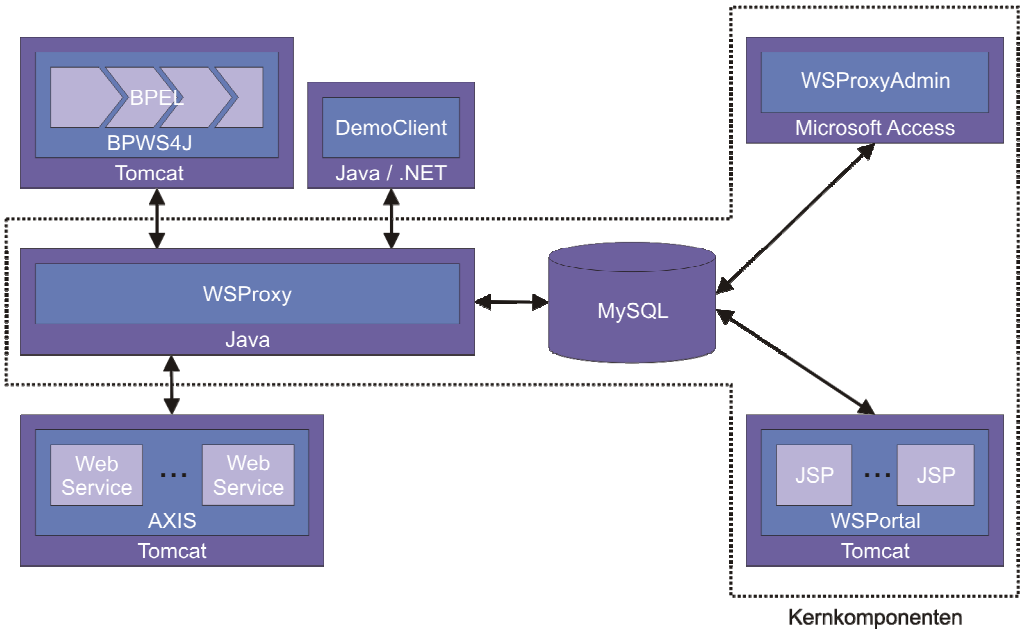

Abbildung 19 - WSQoSX-Systemarchitektur

Zu den Kernkomponenten zählen der Web-Service-Proxy-Server WSProxy, das Portal WSPortal, das Administrations-Frontend WSProxyAdmin, sowie ein Datenbankserver zur gemeinsamen Datenablage aller Kernkomponenten. Zur Demonstration der Nutzung der, durch die Kernkomponenten realisierte Infrastruktur wurde ein Beispielgeschäftsprozess modelliert, der im Zuge seiner Ausführung in einer BPE auf Web Services über die Infrastruktur zugreift. Die durch den Geschäftsprozess benötigten Web Services wurden ebenso implementiert wie zwei Clients zur Nutzung des Geschäftsprozesses.

Mit Ausnahme des WSProxyAdmin und einem zusätzlichen Demo-Client zum Aufruf des Geschäftsprozesses in .NET[36] basieren alle Implementierungen auf Java 1.4.1[37]. Als Applikations-Server, bzw. Servlet-Container, wird Tomcat 5.0.28[38] [25] verwendet. Tomcat wird von dem Portal WSPortal, dem Web-Service-Server AXIS[39] [83] und der BPE BPWS4J[40] als Ausführungsumgebung genutzt. Der in BPEL formulierte Geschäftsprozess wird durch BPWS4J ausgeführt, Web Services über AXIS bereitgestellt und WSPortal bietet durch JavaServer Pages (JSP)[41] eine Web-Schnittstelle für Anbieter von Web Services. Zur Datenablage wird von allen Komponenten der Datenbankserver MySQL 4.1.9[42] verwendet. Das Administrations-Frontend WSProxyAdmin, zur Administration der Kernkomponenten, wurde in Microsoft Access 2003[43] entwickelt.

Bevor die einzelnen Komponenten und ihre Implementierung im Weiteren näher erläutert werden, sind zunächst einige Vorbemerkungen über die allgemeine Konfiguration, die Konfigurierbarkeit und des Loggings nötig.

[36] Siehe auch: http://www.microsoft.com/net/ und z.B. [24].
[37] Siehe auch: http://java.sun.com/ und z.B. [81].
[38] Siehe auch: http://jakarta.apache.org/tomcat/ und z.B. [25].
[39] Siehe auch: http://ws.apache.org/axis/ und z.B. [83].
[40] Siehe auch: http://www.alphaworks.ibm.com/tech/bpws4j/
[41] Siehe auch: http://java.sun.com/products/jsp/ und z.B. [13].
[42] Siehe auch: http://www.mysql.com/ und z.B. [28].
[43] Siehe auch: http://office.microsoft.com/de-de/FX010857911031.aspx und z.B. [82] und [23].

4.1.1 Distribution und Konfiguration

Um das WSQoSX-System leicht installieren und schnell in einen lauffähigen Zustand versetzen zu können, wurde eine *Distribution* von WSQoSX erstellt. Die Distribution enthält, mit Ausnahme von Microsoft Access, jede durch das System verwendete Basissoftware, alle Komponenten des Systems, sowie eine ausführliche Installationsanleitung[44] in englischer Sprache mit einer Vielzahl von Hinweisen für Entwickler und zur Konfiguration.

Wird WSQoSX gemäß der Installationsanleitung installiert, so sind alle Komponenten vorkonfiguriert und betriebsbereit. Um ein möglichst einfaches Testen des Systems zu ermöglichen wurden hierbei alle Komponenten auf einem Rechner installiert und entsprechend konfiguriert. Die Komponenten des Systems lassen sich jedoch über mehrere Rechner in einem Netzwerk verteilen. Um hiernach erneut ein korrektes Zusammenspiel der Komponenten gewährleisten zu können, ist es notwendig die einzelnen Komponenten gemäß den neuen Gegebenheiten zu konfigurieren. Viele Konfigurationsoptionen lassen sich über Properties-Files[45] oder XML-basierte Konfigurationsdateien festlegen. Konfigurationsmöglichkeiten reichen hierbei von der URL, unter der Komponenten im Netzwerk erreicht werden können, über Ports, auf denen Server angesprochen werden können, bis hin zu einzelnen Datenbanknamen und Tabellennamen zur Datenablage.

Aufgrund der Flexibilität der Komponenten und des hierdurch entstehenden Umfangs an Konfigurationsmöglichkeiten, können diese im Folgenden nicht einzeln betrachtet werden. Es wird daher, soweit nicht anders erwähnt, von der Standard-Konfiguration des Systems gemäß der Installationsanleitung ausgegangen. Dies stellt keine Einschränkung beim Einsatz des Systems dar, sondern dient lediglich der Vereinfachung der Betrachtung.

4.1.2 Logging

Alle javabasierten Komponenten verwenden zum Logging, bzw. der Ausgabe von Debug-Informationen Log4J[46]. Hierbei werden über ein differenziertes Ausgabesystem nur einige Logging-Informationen direkt auf der Konsole der Java Virtual Machine (JVM) ausgegeben, jedoch alle Logging-Informationen über ein UDP-Socket versendet. Über das graphische Logging-Frontend Chainsaw[47] von Log4J können die Logging-Informationen aller Komponenten empfangen und übersichtlich angezeigt und ausgewertet werden. Das Log4J-Logging der Komponenten kann über ein Properties-File im jeweiligen Verzeichnis der Komponente frei konfiguriert werden.

Das Betrachten der Logging-Informationen liefert sehr detaillierte Informationen über die ausgeführten Funktionalitäten der einzelnen Komponenten. Zu einem Verständnis des Systems, welches über den Rahmen dieses Technical Reports hinausgeht, oder zur Fehleranalyse, wird das Studium der Logging-Informationen während des Systembetriebs empfohlen. Zur weiteren Vertiefung sollte anschließend die Analyse der Komponentenquelltexte erfolgen.

---

[44] Siehe Datei "`ReadMe.txt`" im Wurzelverzeichnis der Distribution.
[45] Durch Java verwendetes Format für Textdateien zur Ablage von Schlüssel-Wert-Paaren.
[46] Siehe auch: http://logging.apache.org/log4j/ und z.B. [42].
[47] Siehe auch: http://logging.apache.org/log4j/docs/chainsaw.html

4.2 Geschäftsprozess, Client und AXIS

Zur Demonstration der Nutzung sind in der Standard-Installation des Systems zwei Clients zum Aufrufen eines BPEL-Beispielgeschäftsprozesses, der BPEL-Beispielgeschäftsprozess, eine entsprechende Ausführungsumgebung BPWS4J, die vom Beispielgeschäftsprozess benötigten Web Services und der Web-Service-Server AXIS enthalten. Diese sollen im Folgenden näher erläutert werden.

4.2.1 BPEL-Geschäftsprozess

Um die Möglichkeit der flexiblen Nutzung von Web Services durch Geschäftsprozesse zu demonstrieren, wurde zunächst ein einfacher Beispielgeschäftsprozess für die Bearbeitung eines Kreditantrages aus Sicht eines Finanzinstituts erstellt. Im Rahmen des Geschäftsprozesses wird zunächst die Kreditwürdigkeit des Antragstellers durch eine Schufa-Auskunft überprüft. Erweist sich der Antragsteller als kreditwürdig, so wird der Kreditantrag angenommen, im Finanzinstitut registriert und der Kreditbetrag auf das Konto des Antragstellers überwiesen. Anschließend erfolgt die Benachrichtigung über das Ergebnis der Bearbeitung des Kreditantrages.

Die Aktivitäten des Geschäftsprozesses wurden in Web Services gekapselt und der Geschäftsprozess als BPEL-Geschäftsprozess aus diesen Web Services orchestriert.

Bei der Definition eines BPEL-Geschäftsprozesses [49] werden verwendete Web Services nur mittels `partnerLink`-Elementen unter Angabe eines `portType`-Elements ihrer abstrakte Schnittstelle (siehe Kapitel 2.2.1) eingebunden, d.h. insbesondere ohne Angabe eines Bindings und eines konkret adressierten Web Services. Bei der Ausführung eines BPEL-Prozesses könnte daher jeder beliebige Web Service, welcher die benötigte abstrakte Schnittstelle implementiert, verwendet werden.

```
<invoke     name="callSchufa"
            partnerLink="Schufa"
            portType="ns1:SchufaWS"
            operation="validCustomer"
            inputVariable="schufaRequest"
            outputVariable="schufaResponse">
      …
</invoke>
```

Abbildung 20 - Aufruf des Schufa-Web-Service durch den BPEL-Prozess

Es ist die Aufgabe der Ausführungsumgebung des BPEL-Prozesses für jedes `partnerLink`-Element einen konkreten Web Service aufzurufen. Bei der verwendeten Ausführungsumgebung BPWS4J geschieht das Verknüpfen eines `partnerLink`-Elements mit einem konkreten Web Service durch die Angabe des WSDL-Dokuments des konkreten Web Services. Eine solche Verknüpfung muss zum Zeitpunkt der Veröffentlichung eines BPEL-Prozesses in der Ausführungsumgebung vorgenommen werden. Erst danach kann der BPEL-Prozess als Web Service über die Ausführungsumgebung aufgerufen und damit ausgeführt werden.
Damit bei der Ausführung des Geschäftsprozesses die Web Services trotz dieser statischen Verknüpfung flexibel eingebunden werden können, wird vor der Verknüpfung eines `partnerLink`-Elements mit einem WSDL-Dokument, das

`service`-Element des Web Services im WSDL-Dokument angepasst. Im `service`-Element des Web Services ist der konkret aufzurufende Web Service mittels einer URL adressiert. Statt der URL des eigentlichen Web Services wird hier die Basis-URL des WSProxy mit einem entsprechenden WSID zur Identifikation des Web Services eingetragen (siehe Kapitel 3.2.2). Wird nun zur Laufzeit des Geschäftsprozesses ein Web Service durch die Ausführungsumgebung aufgerufen, so werden die Web-Service-Anfragen an den WSProxy gesendet und können durch diesen dynamisch weitergeleitet werden.

Durch den Geschäftsprozess werden folgende Web Services aufgerufen:

- `CreditProviderService`. Dieser Web Service repräsentiert den Kreditgeber und verwaltet die einzelnen Kreditanträge, ihre Vertragsmodalitäten und den Status der jeweiligen Kreditrückzahlung.
- `SchufaService`. Durch diesen Web Service wird die Institution repräsentiert, welche die Kreditwürdigkeit des Antragstellers überprüft. Nach der Übermittlung der Kundendaten des Antragstellers wird die Kreditwürdigkeit in Form eines Wahrheitswertes (wahr oder falsch) zurückgeliefert.
- `BankService`. Der BankService repräsentiert eine Bank, über die Zahlungen im Rahmen des Kreditprozesses vorgenommen werden. Nach der Gewährung eines Kredites wird über diesen Web Service der Kreditbetrag vom Konto des Kreditgebers auf das Konto des Kreditnehmers überwiesen.

Für jeden dieser Web Services ist über das Administrations-Frontend konfiguriert worden, unter welchem WSID er angesprochen werden kann und wie im Falle eines Aufrufes das Routing an den konkreten Web Service durchzuführen ist. Hierbei wurde für alle Web Services, mit Ausnahme des Schufa-Web-Services, ein statisches Routing definiert. Der Schufa-Web-Service liegt in zwei Varianten vor (`SchufaServiceA` und `SchufaServiceB`), die mit unterschiedlichen nicht-funktionalen Eigenschaften über das Portal in der Kategorie „Schufa“ angemeldet wurden. Der Schufa-Web-Service wird im WSProxy über den WSID „`/creditprocess/schufa`“) angesprochen und anschließend über das Target „`portal:Schufa`“ geroutet. Dies bedeutet, dass bei einem Aufruf des Schufa-Web-Services über den WSProxy der Web-Service-Aufruf dynamisch zur Laufzeit an den konkreten Schufa-Web-Service der Kategorie „Schufa“ mit der höchsten Score weitergeleitet wird.

Im konkreten, vorkonfigurierten Fall der Standard-Installation des Systems, wird der Schufa-Web-Service durch die Ausführungsumgebung des Geschäftsprozesses über die URL „`http://127.0.0.1:8081/creditprocess/schufa`“ aufgerufen. Der Web-Service-Aufruf wird durch den WSProxy empfangen und anhand des WSID „`/creditprocess/schufa`“ ermittelt, dass das Target „`portal:Schufa`“ zum Routing des Aufrufs verwendet werden soll. WSProxy verwendet das „portal“-Suchverfahren und sucht in der Kategorie „Schufa“ den Web Service mit der höchsten Score. Die höchste Score dieser Kategorie besitzt der „`SchufaServiceB`“ mit 8,775 Score-Punkten. Der Web-Service-Aufruf wird daher an die Endpunkt-URL „`http://127.0.0.1:8080/axis/ services/SchufaServiceB`“ des `SchufaServiceB` weitergeleitet. Das vom `SchufaServiceB` empfangene Ergebnis wird durch den WSProxy wieder an die Ausführungsumgebung des Geschäftsprozesses zurückgegeben. Die Umleitung des Web-Service-Aufrufes durch

den WSProxy ist für die Ausführungsumgebung des Geschäftsprozesses hierbei vollkommen transparent, d.h. für sie erscheint es, als hätte sie den Schufa-Web-Service direkt aufgerufen.

4.2.2 BPEL-Ausführungsumgebung BPWS4J

Als Ausführungsumgebung für BPEL-Geschäftsprozesse wurde die Business Process Execution Language for Web Services Java Run Time (BPWS4J) von IBM alphaWorks in der Version 2.0 verwendet. Bei BPWS4J handelt es sich um die Referenzimplementierung einer Ausführungsumgebung für die, durch IBM, Microsoft und BEA entwickelte Sprache BPEL4WS (kurz BPEL) in der Version 1.1 (siehe auch Kapitel 2.3).

BPWS4J ist vollständig in Java implementiert und wird in einem Servlet-Container installiert. In der Standard-Installation von WSQoSX ist BPWS4J unter dem Kontext „`/bpws4j`“ im Servlet-Container Tomcat 5.0.28 vorinstalliert. Die Administrations-Web-Oberfläche kann über die URL „`http://127.0.0.1:8080/bpws4j/`“ erreicht werden.

Um BPEL-Geschäftsprozesse ausführen zu können, müssen diese über die Administrations-Oberfläche von BPWS4J veröffentlicht werden. Neben dem eigentlichen BPEL-Dokument, welches den BPEL-Geschäftsprozess definiert, wird zur Veröffentlichung ein zugehöriges WSDL-Dokument benötigt, welches die abstrakte Schnittstelle des Geschäftsprozesses beschreibt, über die Clients nach der Veröffentlichung auf den BPEL-Geschäftsprozess als Web Service zugreifen können. Während des Veröffentlichungsvorgangs wird das BPEL-Dokument geparsed und nach `partnerLink`-Elementen durchsucht, die einen Partner, d.h. einen Web Service mit welchem der Geschäftsprozess interagiert, definieren. Für jeden Partner muss über die Web-Oberfläche ein konkreter Web Service durch Angabe seines WSDL-Dokuments verknüpft werden. Um einen solchen Web Service dynamisch über den WSProxy aufrufen zu können, ist die Endpunkt-URL im WSDL-Dokument entsprechend anzupassen, bevor das WSDL-Dokument zur Spezifikation eines konkreten Partners während des Veröffentlichungsvorganges verwendet wird.

In der Standard-Installation von WSQoSX ist BPWS4J bereits vollständig installiert, konfiguriert und der BPEL-Beispielgeschäftsprozess veröffentlicht.

4.2.3 Geschäftsprozess-Client

Der BPEL-Beispielgeschäftsprozess wird über BPWS4J als Web Service veröffentlicht. Über die Administrations-Web-Oberfläche von BPWS4J kann zu jedem veröffentlichten Geschäftsprozess das WSDL-Dokument abgerufen werden, welches seine Schnittstelle als Web Services beschreibt. Mithilfe dieses WSDL-Dokuments lassen sich Clients zum Aufruf des Web Services in verschiedensten Programmiersprachen auf den verschiedensten Plattformen erstellen.

Zur Demonstration der Interoperabilität enthält die Standard-Installation von WSQoSX nicht nur einen in Java implementierten Client zum Aufruf des Beispielgeschäftsprozesses, sondern zusätzlich noch einen in C# auf dem .NET-Framework implementierten Client.

.NET-Client

Der .NET-Client wurde mit „Microsoft Visual Studio .NET 2003“[48] in der Programmiersprache C# [54] erstellt. „Microsoft Visual Studio .NET 2003“ bietet die komfortable Möglichkeit eine Klasse zum Aufruf eines Web Services automatisch aus einem angegebenen WSDL-Dokument zu erstellen. Alle auf diese Art erstellten Klassen erben von der Klasse `WebClientProtocol` aus dem Assembly `System.Web.Services.Protocols`. Je nachdem welches Transportprotokoll und Nachrichtenformat zur Kommunikation mit dem Web Service im WSDL-Dokument defininiert wurde, erbt die erstellte Klasse von weiteren Sub-Klassen. Da der BPEL-Geschäftsprozess-Web-Service das Transportprotokoll HTTP und das Nachrichtenformat SOAP verwendet, erbt die konkret erstellte Klasse `loanapprovalServiceBP` von der Klasse `SoapHttpClientProtocol`. Alle Operationen die der Web Service nach außen anbietet, werden als Methoden dieser Klasse definiert. Zur Durchführung des Kreditprüfungsprozesses dient die Methode `submit`:

```
public String submit (  String firstname, String surname ,
                        Int32 salery, Int32 creditHeight,
                        String telNumber, Int32 emplStat,
                        String maritalStatus, Int32 instHeight,
                        Int32 autoBankTransfer )
```

Um die für den Aufruf nötigen Parameter bequem erfassen zu können, wurde ein entsprechendes Formular erstellt. Der Nutzer gibt hier die Parameterdaten ein und kann mittels eines Klicks auf einen „Approve“-Button die Daten an den Web Service senden.

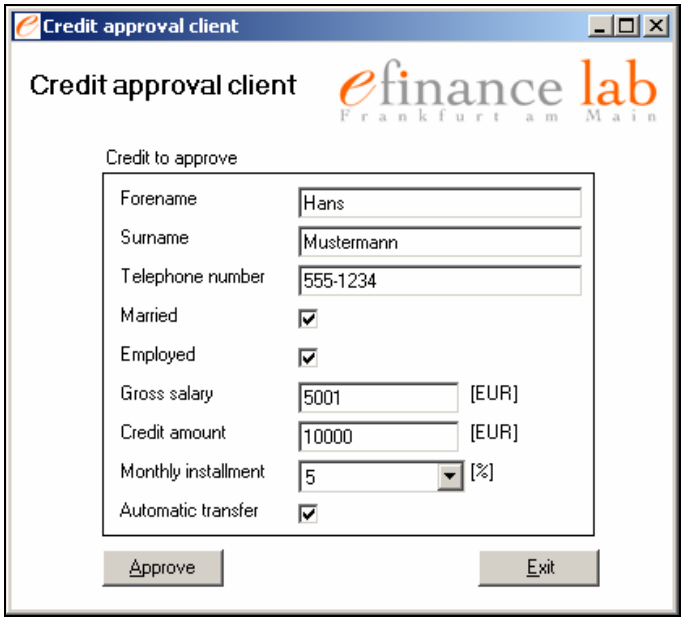

Abbildung 21 - Screenshot des .NET-Clients zum Aufruf des Geschäftsprozesses

Um den Client flexibel bezüglich der Endpoint-URL des Web Services zu halten, wird diese vor dem Absenden aus einem XML-Dokument in ein `Settings`-Objekt gelesen. Durch das anpassen der URL in diesem XML-Dokument kann daher sehr

[48] Siehe auch: http://msdn.microsoft.com/vstudio/

einfach festgelegt werden ob der Web Service direkt oder indirekt über den WSProxy aufgerufen werden soll. Durch die Indirektion über den WSProxy wäre es möglich bei dem Vorhandensein mehrerer alternativer Geschäftsprozesse dynamisch den am besten geeigneten Geschäftsprozesse zur Bearbeitung des Kreditantrags zu verwenden.

Um die Daten tatsächlich an den Web Service zu senden, wird ein eine Instanz der Klasse `loanapprovalServiceBP` erzeugt, die im `Settings`-Objekt eingelesene Endpunkt-URL des Web Services über die Eigenschaft `Url` des Objekts der Klasse `loanapprovalServiceBP` gesetzt und anschließend die Funktion `submit` auf diesem Objekt aufgerufen:

```
loanapprovalServiceBP approvalService = new loanapprovalServiceBP();
approvalService.Url=this.settings.BPELProcessURL;
String result = approvalService.submit(...);
```

Das Ergebnis der Abarbeitung des Kreditantrages, oder etwaige Fehlermeldungen, werden dem Benutzer anschließend durch einen Dialog präsentiert.

Java Client

Die Funktionalität des Java-Client ist identisch mit der des .NET-Clients, weshalb hier nicht nochmals näher auf die Funktionalität eingegangen werden soll. Bei der Erstellung des Java-Clients wurde zur automatischen Generierung von Klassen zum Aufruf des Geschäftsprozess-Web-Services das Tool „WSDL2Java“ (`org.apache.axis.wsdl.WSDL2Java`) des Web-Service-Servers AXIS verwendet. Die generierten Klassen wurden zur Verwendung einer frei definierbaren Endpunkt-URL angepasst. Bevor die Daten an den Web Service gesendet werden, wird die konkret für den Aufruf zu verwendende Endpunkt-URL aus einer Properties-Datei ausgelesen. Der Aufruf geschieht dann unter Angabe dieser URL als zusätzlicher erster Parameter:

```
String url = BPELServerSettings.getURL();
boolean result = SoapBPELCommMgr.requestForCredit(url, ...);
```

4.2.4 Web-Service-Server AXIS

Der BPEL-Beispiel-Geschäftsprozess führt Aktivitäten mittels Web Services aus. Damit diese Web Services aufgerufen werden können, müssen sie im System durch einen Web-Services-Server angeboten werden. Zu diesem Zweck wird in der Standard-Installation von WSQoSX Apache AXIS in der Version 1.1 verwendet.

AXIS ist die Abkürzung von „Apache eXtensible Interaction System“ und bezeichnet die vollständig in Java implementierte Open-Source-SOAP-Engine des „Apache Web Services Project“ [83]. Zum einen ist AXIS ein Server für Web Services, der in Form eines eigenständigen Servers oder einer Web-Applikation in einem Servlet-Container betrieben werden kann. Darüber hinaus ist AXIS ein Framework zur Entwicklung von Anwendungen die SOAP zur Kommunikation verwenden. Die Transportprotokolle, die zur Übermittlung der SOAP-Nachrichten zum Einsatz kommen können, sind hierbei nicht fest vorgegeben. AXIS unterstützt die wichtigsten Transportprotokolle (wie z.B. HTTP) direkt und lässt sich um weitere Transportprotokolle ergänzen. Beim Umgang mit Web Services und deren SOAP-Nachrichten unterstützt AXIS 1.1

folgende Standards: SOAP 1.1, WSDL 1.1, „Java API for XML-based RPC“[49] (JAX-RPC) 1.0 und „SOAP with Attachments API for Java“[50] (SAAJ) 1.2.

Um eine Java-Klasse über AXIS als einen Web Service zur Nutzung nach außen anzubieten, d.h. zu veröffentlichen, genügt es die Java-Klasse zu kompilieren, in eine bestimmte Verzeichnisstruktur zu kopieren und AXIS über das Administrations-Tool `AdminClient` eine XML-Datei mit der „Web Service Deployment Description“ (WSDD) zukommen zu lassen. Nach dem Veröffentlichen empfängt AXIS automatisch SOAP-Nachrichten zum Aufruf des Web Services, übersetzt die SOAP-Nachricht in einen Aufruf der Java-Klasse, ruft die Java-Klasse auf, übersetzt die Ergebnisse wieder in eine SOAP-Nachricht und sendet sie an den Aufrufer zurück. Über die WSDD-Datei wird AXIS beim Veröffentlichen zu jeder Java-Klasse mitgeteilt, unter welchem Web-Service-Namen sie welche Methoden nach außen anbietet. AXIS ist in der Lage zu jeder als Web Service veröffentlichten Java-Klasse ein entsprechendes WSDL-Dokument zur Spezifikation der jeweiligen Schnittstelle automatisch zu generieren.

In der Standard-Installation von WSQoSX wurde AXIS 1.1 als Web-Anwendung im Servlet-Container Tomcat 5.0.28 im Kontext „`/axis`“ installiert. Hierbei wurden bereits allen Web Services, die durch den BPEL-Beispiel-Geschäftsprozess benötigt werden, vorinstalliert und vorkonfiguriert. Die Web-Oberfläche von AXIS kann über die URL „`http://127.0.0.1:8080/axis/`“ erreicht werden. Dort kann eine Liste aller veröffentlichten Web Services und deren Spezifikation in Form von WSDL-Dokumenten eingesehen werden. Die WSDL-Dokumente eines veröffentlichten Web Service sind hierbei stets über eine URL der Form „`http://127.0.0.1:8080/axis/services/<Web-Service-Name>?wsdl`“ erreichbar.

Tabelle 7 können die konkret über AXIS als Web Service veröffentlichten Java-Klassen entnommen werden.

Name der Java-Klasse	Name des Web Services
`BankWS`	`BankService`
`CreditProviderWS`	`CreditProviderService`
`SchufaWSA`	`SchufaServiceA`
`SchufaWSB`	`SchufaServiceB`

Tabelle 7 - Über AXIS als Web Service veröffentlichte Java-Klassen

Zum Zugriff auf die Web Services ist hierbei eine Endpunkt-URL der Form „`http://127.0.0.1:8080/axis/services/<Web-Service-Name>`“ zu verwenden.

4.3 WSProxy

Der Web-Service-Proxy-Server WSProxy bildet den Kern der dienstgüteorientierten Web-Service-Infrastruktur von WSQoSX. Seine Aufgabe ist es Web-Service-Aufrufe zu Empfangen, den gewünschten Web Service aus den übermittelten Informationen des Aufrufes zu ermitteln, das zugeordnete Routing-Verfahren zu bestimmen, ein Ziel

[49] Siehe auch: http://java.sun.com/xml/jaxrpc/
[50] Siehe auch: http://java.sun.com/xml/saaj/

für das Routing durch ein Suchverfahren abzuleiten, den empfangenen Aufruf an das Ziel weiterzuleiten, eine Antwort zu Empfangen und an den Aufrufer zurückzuliefern. Hierbei wird durch den WSProxy jeder Aufruf mitsamt den ermittelten Dienstgüteparametern protokolliert und Statistiken bezüglich der Dienstgüteparameter verwendeter Web Services aktualisiert. Weichen die Dienstgüteparameter eines Aufrufes oder im durchschnittlichen Verhalten von den zugesicherten Dienstgüteparametern in einer SLA ab, so werden Warnungen generiert. Ebenso werden Warnungen im Falle der Nichterreichbarkeit eines Web Services oder bei der Verwendung eines Web Services mit abgelaufener SLA geniert.

Der WSProxy ist vollständig in Java implementiert. Zur Entwicklung wurde die „Java 2 Platform, Standard Edition“ (J2SE) in der Version 1.4.1 verwendet. Tabelle 8 kann eine Übersicht der Pakete und der jeweiligen Klassen des WSProxy entnommen werden.

Paket	Klassen
de.tud.kom.dynws.proxy	WSProxy WSProxyClientThread WSProxyDaemonThread WSProxyHTTPRequest
de.tud.kom.dynws.proxy.history	WSHistory WSWarnings WSWarningsFactory WSWarningsPortal WSWarningsStatic
de.tud.kom.dynws.proxy.router	WSRouter WSRouterRemoteTimeoutException WSRouterUnroutableException
de.tud.kom.dynws.proxy.search	WSSearch.java WSSearchFactory WSSearchPortal WSSearchStatic
de.tud.kom.dynws.proxy.util	ChunkedInputStream CopyWrapInputStream CopyWrapOutputStream DBHelper

Tabelle 8 - Pakete und Klassen des WSProxy

Im Folgenden wird ein typischer Ablauf der Verarbeitung eines Web-Service-Aufrufs durch den WSProxy geschildert. Hierbei werden die wichtigsten Klassen und die durch sie erbrachte Funktionalität näher erläutert.

4.3.1 Start des WSProxy

Zum Start des WSProxy ist die Klasse `de.tud.kom.dynws.proxy.WSProxy` auszuführen. Hierbei muss der Port, auf welchem der WSProxy-Server Web-Service-Anfragen entgegennehmen soll, als Kommandozeilenparameter angegeben werden. Weitere grundlegende Konfigurationsdaten entnimmt WSProxy aus Properties-

Dateien im selben Paket. Zu den Konfigurationsdaten zählen das Log4J-Logging-Verhalten in der Datei „`log4j.properties`“, die Datenbankkonfiguration (JDBC-Datenbanktreiber[51], JDBC-URL zum Datenbankserver, Datenbankname und Tabellennamen) in der Datei „`db.properties`“, sowie die Standard-Timeout-Dauer[52] bei Web-Service-Aufrufen in der Datei „`WSRouter.properties`“. Alle weiteren Daten entnimmt WSProxy aus den Tabellen der konfigurierten Datenbank.

Während des Starts wird ein neuer Thread der Klasse `WSProxyDaemonThread` als Daemon-Thread[53] gestartet. Die Aufgabe des Daemon-Threads ist es, auf dem per Kommandozeilenparameter spezifizierten Port, in einer Endlosschleife auf eingehende TCP-Verbindungen zu warten. Nach dem Start des Daemon-Threads wartet WSProxy im Anwendungs-Thread auf das Betätigen der Eingabetaste, worauf der Anwendungs-Thread und damit WSProxy beendet wird.

4.3.2 Verbindungsannahme

Solange der Anwendungs-Thread noch nicht beendet wurde, ist der Daemon-Thread aktiv und wartet auf eingehende TCP-Verbindungen. Wird eine eingehende Verbindung festgestellt, so wird ein neuer Thread der Klasse `WSProxyClientThread` gestartet und das Socket mit der angenommenen Verbindung an diesen Thread übergben. Die Aufgabe des `WSProxyClientThread` ist es die weitere Verarbeitung der angenommenen Verbindung nebenläufig zu übernehmen. Der Daemon-Thread steht direkt nach dem Start des `WSProxyClientThread` wieder für die Annahme von neuen Verbindungen zur Verfügung.

4.3.3 Empfang des HTTP-Requests

Durch den `WSProxyClientThread` wird die Nachricht, die ein Client über die angenommene Verbindung sendet, empfangen. Hierbei wird eine Nachricht in Form eines HTTP-Requests gemäß dem HTTP-Protokoll 1.0 [14] oder 1.1 [32] erwartet. Zunächst wird der HTTP-Header zeilenweise aus dem eingehenden Strom von ASCII-Zeichen gelesen. Der konkret erwartete Aufbau der Nachricht ist wie folgt:

- 1. Zeile: `<http command> <request uri> <http version>`
- Folgende Zeilen: `<header property>: <header value>`
- Leerzeile: Separator zwischen HTTP-Header und HTTP-Body
- Alle folgenden Zeichen: HTTP-Body

Wird als HTTP-Kommando weder „`GET`“ noch „`POST`“ übermittelt, so wird die Verarbeitung des HTTP-Requests abgebrochen und der HTTP-Fehlercode 501 („Not implemented“) an den Client gesendet, welcher dem Client signalisiert, dass das gewünschte Kommando durch den WSProxy nicht verarbeitet werden kann.

---

[51] JDBC (Java Database Connectivity) ist ein API der Java-Plattform zur Realisierung einer einheitlichen Schnittstelle zum Zugriff auf Datenbankserver verschiedener Hersteller. Siehe hierzu auch: http://java.sun.com/products/jdbc/

[52] Falls der WSProxy bei dem Aufruf eines externen Web Services keine Informationen über die vom Anbieter garantierte Antwortzeit besitzt, so bricht er den Aufruf nach Ablauf der Standard-Timeout-Dauer ab.

[53] Beim Beenden des Anwendungs-Threads wartet die JVM nicht auf das Ende eines Daemon-Threads. Deamon-Threads eignen sich als Hintergrund-Threads zur Realisierung von Serverfunktionalitäten in einer Endlosschleife.

Wurde als HTTP-Kommando „`GET`“ oder „`POST`“ übermittelt, so werden alle Header-Zeilen bis zum Auftreten einer Leerzeile eingelesen. Alle Zeichen nach der Leerzeile werden als Body des HTTP-Requests interpretiert und Byte für Byte binär ausgelesen. Hat der Client zuvor im Header über den „Content-Length“-Header[54] die Größe des Body in Bytes angegeben, so wird der Body in einem Stück gelesen, bis die angegebene Anzahl an Bytes empfangen wurde. Wurde durch den Client zuvor im HTTP-Header über den „Transfer-Encoding“-Header das Übermittlungsverfahren „chunked“[55] definiert, so wird der Body in mehreren, speziell markierten Teilen (Chunks) eingelesen. In diesem Fall steht die Größe des Body bis zum Erhalt des letzten Chunks nicht fest.

Der empfangene HTTP-Request wird in einem Objekt der Klasse `WSProxyHTTPRequest` abgelegt.

4.3.4 Routing des HTTP-Requests

Zur Weiterverarbeitung des empfangenen HTTP-Requests wird eine Instanz der Klasse `WSRouter` erzeugt und hierbei der empfangene HTTP-Request in Form eines `WSProxyHTTPRequest`-Objekts im Konstruktor übergeben. Durch den Aufruf der Methode `route()` des `WSRouter`-Objekts wird das Routing des HTTP-Requests ausgelöst.

Die ersten Zeile des HTTP-Requests enthält (u.a.) die durch den Request beim Server adressierte URI. Im Falle des WSProxy handelt es sich bei dieser URI um den WSID des adressierten Web Service. Zu dem übermittelten WSID wird nun in der Datenbank über den zugeordneten USID ein Traget ermittelt, welches angibt, wie nach einem Ziel für den HTTP-Request gesucht werden soll. Kann ein solches Target nicht ermittelt werden, ist das im Target angegebene Suchverfahren unbekannt oder kann über das angegebene Suchverfahren keine Endpunkt-URL für die Weiterleitung ermittelt werden, so wird eine `WSRouterUnroutableException` ausgelöst und an den Client der HTTP-Fehlercode[56] 404 („Not found“) zurückgegeben.

Konnte ein Target ermittelt werden, so wird eine Instanz der Klasse `WSSearchFactory` erzeugt und über die Methode `getSearchForTarget(target)` eine Objekt zu diesem Target instanziiert, welches das Interface `WSSearch` implementiert. Für jedes unterstützte, in einem Target spezifizierbare Suchverfahren existiert eine Klasse, welches das Interface `WSSearch` implementiert und das spezifische Suchverfahren nach einer Endpunkt-URL über die Funktion `getEndpointURL` bereitstellt (z.B. `WSSearchPortal` oder `WSSearchStatic`). Neue Suchverfahren können daher leicht über weitere Klassen, die das Interface `WSSearch` implementieren, ergänzt werden.

Das Suchverfahren „static“ wird durch die Klasse `WSSearchStatic` implementiert und liefert den im Target angegebenen Suchparameter als Suchergebnis, d.h. als Endpunkt-URL zur Weiterleitung des HTTP-Requests. Das Suchverfahren „portal“

[54] Die Verwendung des „Content-Length“-Headers entspricht dem Standard der HTTP-Version 1.0.
[55] Das Übermittlungsverfahren „chunked“ steht erst seit der HTTP-Version 1.1 zur Verfügung.
[56] Enthielt der Body des HTTP-Requests eine SOAP-Nachricht, so wird im Body der HTTP-Fehlermeldung eine entsprechende SOAP-Fault-Nachricht geliefert, ansonsten eine HTML-Fehlerseite.

wird durch die Klasse `WSSearchPortal` implementiert und durchsucht die, durch den Suchparameter spezifizierte Kategorie nach dem Web Service mit der höchsten Score. Die Endpunkt-URL des gefundenen Web Services wird als Ergebnis der Suche zurückgegeben.

Neben der Methode `getEndpointURL` zur Ermittlung der Endpunkt-URL stellt jede Klasse, die das Interface `WSSearch` implementiert, auch die Funktion `getResponseTime` zur Verfügung. Nach der Ermittlung der Endpunkt-URL liefert diese Funktion die zu erwartende Antwortzeit des Web Services unter der ermittelten Endpunkt-URL. Das Suchverfahren „static“ liefert auf eine solche Anfrage stets die Antwortzeit -1, da ihm keinerlei Informationen über die Antwortzeit eines Web Services vorliegen. Das Suchverfahren „portal“ liefert hier die durch den Anbieter bei der Anmeldung spezifizierte garantierte Antwortzeit seines Web Services.

4.3.5 Aufruf des Web Services

Nachdem die Endpunkt-URL zur Weiterleitung des HTTP-Requests ermittelt wurde, wird der vom Client empfangene HTTP-Request über die Funktion `call` des `WSRouter`-Objekts an diese Endpunkt-URL gesendet. Vor dem Versenden des HTTP-Requests werden die Header des Requests überprüft und ggf. angepasst. Existiert im Header des Requests ein „Host“-Header, so verweist dieser noch auf die Basis-URL des WSProxy. Vor dem Versenden wird daher der „Host“-Header so angepasst, dass dieser den, in der Endpunkt-URL spezifizierten Rechner adressiert. Enthielt der weiterzuleitende HTTP-Request einen Body und verwendete der Client das Übermittlungsverfahren „chunked“ zu dessen Übermittlung, so wird der entsprechende „Transfer-Encoding“-Header entfernt und durch einen „Content-Length“-Header ersetzt, da der `WSRouter` bei der Weiterleitung des HTTP-Requests den Body stets in einem Stück und nicht in einzelnen Chunks versendet.

Zum Versenden des HTTP-Requests an die Endpunkt-URL wird eine leicht angepasste Version des „Apache Jakarta Commons HTTP Client“[57] verwendet. Hierbei wurde die Klasse `org.apache.commons.httpclient.HttpMethodBase` so überarbeitet, dass ein „User-Agent“-Header nicht zwingend in den HTTP-Headern vorhanden sein muss. Die Klasse `org.apache.commons.httpclient.HttpConnection` wurde um das Wrapping aller Eingabe- und Ausgabe-Streams in ein Byte-Array erweitert. Hierdurch ist die Protokollierung jedes Bytes möglich, das durch den HTTP-Client auf TCP-Ebene gesendet und empfangen wird. Dieses Feature wird intern für das Logging aller gesendeten und empfangenen HTTP-Nachrichten auf Byteebene verwendet.

Zum Versenden des HTTP-Requests wird ein HTTP-Client als Objekt der Klasse `org.apache.commons.httpclient.HttpClient` erzeugt. Mit Methoden des Objekts wird anschließend das, durch den HTTP-Request vorgegebene HTTP-Kommando „`GET`“ oder „`POST`“ gesetzt, die ermittelte Endpunkt-URL als Ziel des Aufrufs festgelegt, die angepassten Header des HTTP-Requests gesetzt, sowie die Timeout-Zeit für den bevorstehenden Aufruf gesetzt. Sollte die maximale Antwortzeit des, über die Endpunkt-URL aufzurufenden, Web Service bekannt sein, so wird die Timeout-Zeit auf die doppelte Antwortzeit gesetzt. Sollte die Antwortzeit unbekannt

[57] Siehe auch: http://jakarta.apache.org/commons/httpclient/

sein, so wird der Standard-Timeout-Wert[58] gesetzt, der in der Datei „`WSRouter.properties`“ gesetzt wurde.

Wurden alle Aufrufparameter im HTTP-Client-Objekt gesetzt, so wird die aktuelle Systemzeit festgehalten und der HTTP-Request durch einen Aufruf der Funktion `executeMethod` durchgeführt. Der HTTP-Request wird an die Endpunkt-URL des Web Services gesendet und die Antwort des Web Services empfangen. Nachdem die Antwort des Web Services vollständig empfangen wurde, wird erneut die Systemzeit ermittelt und aus der Differenz zur Startzeit des Aufrufs die Antwortzeit des aufgerufenen Web Services ermittelt und protokolliert.

4.3.6 Rückübermittlung der Antwort an den Client

Die vom aufgerufenen Web Service erhaltene Antwort wird vom `WSRouter` an den `WSProxyClientThread` in Form eines Byte-Array zurückgegeben und von diesem über die noch bestehende TCP-Verbindung zum Client an diesen zurückgesendet.

Sollten während des Aufrufs des Web Services Fehler aufgetreten sein, so wird der `WSProxyClientThread` in Form von Exceptions hiervon in Kenntnis gesetzt. Dem Client wird in diesem Fall je nach aufgetretenem Fehler eine HTTP-Fehlernachricht gesendet. Ist während des Aufrufs ein Timeout aufgetreten, so wird eine `WSRouterRemoteTimeoutException` ausgelöst und der HTTP-Fehlercode 504 („Gateway timeout“) an den Client gesendet. Ist während des Aufrufs ein Fehler bei einer Ein-/Ausgabeoperation aufgetreten (Ausnahme `IOException` oder Erben hiervon), so wird der HTTP-Fehlercode 502 („Bad Gateway“) an den Client gesendet. Sollte ein anderer, nicht erwarteter Fehler während des Aufrufes aufgetreten sein, so wird der HTTP-Fehlercode 500 („Internal server error“) an den Client übermittelt.

Wurde die Antwort vollständig an den Client übermittelt, so wird die TCP-Verbindung zum Client beendet.

4.3.7 Nachbearbeitung eines Aufrufes / QoS-Monitoring

Nachdem die TCP-Verbindung zum Client beendet wurde, beginnt die Nachbearbeitung des soeben abgearbeiteten Aufrufs. Hierzu zählen das Eintragen der gemessenen Dienstgüteparameter in die Call-History, das Aktualisieren der Statistik des aufgerufenen Web Services in der History-Statistik und das Generieren von Warnungen, sollten Dienstgüteparameter in einem kritischen Bereich festgestellt worden sein. Call-History, History-Statistik und Warnungen wurden bereits in Kapitel 3.2.3 ausführlich beschrieben, weswegen an dieser Stelle auf weitere Einzelheiten verzichtet werden soll.

Zur Nachbearbeitung des Aufrufs wird ein Objekt der Klasse `WSHistory` instantziiert und der soeben verarbeitete HTTP-Request als `WSProxyHTTPRequest`-Objekt im Konstruktor übergeben. Durch den Aufruf der Methode `update` auf dem Objekt wird die Nachbearbeitung vollständig ausgeführt.

Die Generierung von Warnungen geschieht hierbei spezifisch für das Suchverfahren, welches verwendet wurde um die Endpunkt-URL zur Weiterleitung des HTTP-

[58] Die Standard-Timeout-Dauer in der Standard-Installation des WSProxy beträgt 30.000 ms.

Requests zu ermitteln. Das Interface `WSWarnings` definiert eine gemeinsame Schnittstelle für alle Klassen, die Warnmeldungen generieren können. Innerhalb der Klasse `WSHistory` wird die Klasse `WSWarningsFactory` verwendet um eine zum verwendeten Suchverfahren passende Implementierung von `WSWarnings` zu instanziieren (z.B. `WSWarningsPortal` und `WSWarningsStatic`). Durch den Aufruf der Methode `createWarnings` auf dem so erzeugten Objekt werden die Aufrufdaten entsprechend analysiert und, falls möglich und nötig, Warnungen generiert und in der Datenbank abgelegt.

4.3.8 Beendigung der Verarbeitung des HTTP-Requests

Mit dem Abschluss der Nachbearbeitung des Aufrufs ist dieser vollständig durch den `WSProxyClientThread` abgearbeitet und der Thread wird beendet.

4.3.9 Logging

Bereits während der Verarbeitung des HTTP-Requests werden Informationen über den HTTP-Request und den Status seiner Verarbeitung in einer Logging-Tabelle persistent[59] protokolliert (siehe Kapitel 3.2.4). Da Informationen dieser Art auch für eine Fehleranalyse eingesetzt werden können sollen, erfolgt die Protokollierung der Informationen jeweils direkt durch die Klasse, welche eine bestimmte Information ermittelt hat und direkt zum Zeitpunkt der Feststellung der Information. Sollte der WSProxy unerwartet die Verarbeitung eines HTTP-Requests abbrechen, so spiegeln die Logging-Informationen den letzten Wissensstand über den verarbeiteten HTTP-Request wieder. Darüber hinaus werden einige dieser Logging-Informationen in der Nachbearbeitung eines Aufrufs weiterverarbeitet.

Die einzelnen Klassen greifen zur Protokollierung der Informationen auf die Klasse `DBHelper` zurück, die neben Methoden zum vereinfachten Zugriff auf die vom System genutzten Datenbanktabellen auch zahlreiche Methoden speziell für das Logging in die Logging-Tabelle bereitstellt. Eine ausführliche Darstellung aller Aspekte des Loggings kann jedoch aufgrund des Umfangs im Rahmen dieses Technical Reports nicht gegeben werden.

Besonders erwähnt werden soll jedoch der Status eines HTTP-Requests. Sobald der Status eines HTTP-Requests über die Methode `setStatus` des `WSProxyHTTPRequest`-Objekts, welches ihn repräsentiert, geändert wird, wird der neue Status sofort persistent in der Logging-Tabelle aktualisiert. Der Status spiegelt die Phase der Verarbeitung wieder in der sich ein HTTP-Request befindet. Die zur Darstellung des Status verwendeten Status-Codes können Tabelle 9 entnommen werden.

[59] Logging-Informationen, die über das Log4J-Logging-System ausgegeben werden, sind hingegen nicht persistent und daher nur zur Laufzeit des Systems verfügbar.

Status-Code	Beschreibung	Ausgelöster HTTP-Error
`received`	Ein Request wurde vom `WSProxyClientThread` empfangen und in einem `WSProxyHTTPRequest`-Objekt abgelegt.	
`routing`	Das `WSProxyHTTPRequest`-Objekt wurde an den `WSRouter` übergeben.	
`no route found`	Das Routing des Requests durch den `WSRouter` hat eine Exception ausgelöst. Möglich Ursachen sind: • Der zu routende Request war `NULL`. • Zu dem angefragten WSID konnte keine USID gefunden werden. • Zu der USID konnte kein Target gefunden werden. • Das Target spezifiziert kein, oder kein bekanntes Suchverfahren für die Suche nach der Endpunkt-URL des Web Services.	`404 (file not found)`
`searching endpoint url`	Das Target spezifiziert ein gültiges Suchverfahren und der Suchalgorithmus sucht momentan nach der Endpunkt-URL des Web Services.	
`calling remote`	Die Endpunkt-URL des Web Services wurde ermittelt. Der Request wird nun an den Web Service weitergeleitet und die Antwort empfangen.	
`remote call timeout`	Während des Aufrufs des Web Services trat ein Timeout auf und die Kommunikation mit dem Web Service wurde abgebrochen.	`504 (gateway timeout)`
`remote call io error`	Beim Aufruf des Web Services wurde eine `IOException` ausgelöst. Eine mögliche Ursache hierfür ist z.B. dass der Verbindungsaufbau zum Web Service gescheitert ist.	`502(bad gateway)`
`remote call error`	Während des Aufrufs des Web-Services kam es zu einer Exception die weder auf einen Timeout, noch auf einen IO-Error zurückzuführen ist. Die Ursache des Fehlers ist unbekannt.	`500(internal server error)`
`empty response`	Nach dem Aufruf des Web Services wurde eine leere Antwort vom Web-Service empfangen.	`500 (internal server error)`
`sending response`	Der Aufruf des Web Services war erfolgreich und die empfangene (nicht leere) Antwort wird nun an den Client gesendet.	
`sending response failed`	Beim Senden der Antwort an den Client ist eine Exception aufgetreten. Es ist unklar ob die Antwort ganz, teilweise oder gar nicht übermittelt werden konnte.	
`finished`	Die Antwort wurde fehlerfrei an den Client übermittelt. Die Verarbeitung des Requests durch den `WSProxyClientThread` wurde komplett und fehlerfrei abgeschlossen.	

Tabelle 9 - Übersicht der beim Logging verwendeten Status-Codes

4.3.10 Generizität des WSProxy

Der WSProxy ist in der Lage alle HTTP-Requests mit dem HTTP-Kommando „`GET`“ oder „`POST`“ dynamisch weiterzuleiten. Die im HTTP-Request übermittelten Header und der Inhalt des Body sind für den WSProxy bei der Durchführung der Weiterleitung nicht relevant. Lediglich im Fehlerfalle wird überprüft, ob der Inhalt des Body als SOAP-Nachricht interpretiert werden kann, um zu entscheiden, ob im Body der HTTP-Fehlermeldung ein HTML-Dokument oder eine SOAP-Fault-Nachricht eingebettet werden soll.

Der WSProxy wurde zwar zum Zwecke der Weiterleitung von SOAP-Nachrichten im Rahmen der Kommunikation von Web Services über HTTP entwickelt, jedoch

beschränken sich die Fähigkeiten des WSProxy nicht allein auf die dynamische Weiterleitung von SOAP-Nachrichten.

Die Entscheidung zu welcher Endpunkt-URL ein empfangener HTTP-Request weitergeleitet wird, wird über die im HTTP-Request am WSProxy angefragte URI festgelegt, die der WSProxy als ein Identifikator für einen Web Service interpretiert (WSID). Zu jedem WSID lassen sich über ein Target mit dem Suchverfahren „portal“ und einer Kategorie als Suchparameter eine Menge von möglichen Endpunkt-URLs definieren, die anhand einer Score zur Weiterleitung eines HTTP-Requests ausgewählt werden.

Diese Form der dynamischen Weiterleitung von HTTP-Requests ließe sich in einem entsprechenden Umfeld nicht nur zur Weiterleitung von SOAP-Nachrichten verwenden, sondern von nahezu beliebigen, über HTTP transportierten Inhalten, deren Ziel abhängig von gewissen Dienstgüteparametern und einer individuellen Gewichtung derer ausgewählt werden soll.

4.4 WSPortal

Das Portal WSPortal stellt eine Web-Oberfläche zur Verfügung, die als Schnittstelle zwischen den Anbietern von Web Services und dem WSQoSX-System dient. Die einzelnen Seiten der Web-Oberfläche wurden als „JavaServer Page“ (JSP) [13] realisiert und generieren den Inhalt einer aufgerufenen Web-Seite dynamisch zum Zeitpunkt ihres Aufrufes. Die JSP-Seiten, sowie die hierin verwendeten Java-Klassen werden in Form einer Web-Applikation durch den Applikations-Server Tomcat 5.0.28 nach außen zur Verfügung gestellt. In der Standard-Installation von WSQoSX ist das WSPortal unter dem Kontext „`/wsportal`“ in Tomcat vorinstalliert. Das Portal, bzw. seine Web-Oberfläche kann über die URL „`http://127.0.0.1:8080/wsportal/`“ erreicht werden.

4.4.1 Technologische Grundlagen des Portals

Bei dem Aufruf einzelner JSP-Seiten einer Web-Applikation durch einen Browser dient Tomcat gleichermaßen als Web-Server zur Kommunikation zwischen dem Browser und der Web-Applikation, als auch als Ausführungsumgebung der Web-Applikation selbst. Damit eine aufgerufene JSP-Seite durch Tomcat verarbeitet werden kann, erzeugt Tomcat zunächst, sofern nicht bereits geschehen, beim Aufruf der JSP-Seite durch den internen JSP-Compiler[60] Jasper aus der angeforderten JSP-Seite ein Servlet. Das Servlet wird kompiliert und anschließend über den internen Servlet-Conatiner[61] Catalina ausgeführt. Bei der Ausführung des Servlets erhält das Servlet von der Ausführungsumgebung alle relevanten Informationen über den abzuarbeitenden Aufruf. Das Servlet generiert anschließend eine passende Ausgabe und übergibt diese wieder an die Ausführungsumgebung. Schließlich sendet Tomcat die Ausgabe des Servlets zurück an den aufrufenden Browser.

[60] Tomcat unterstützt JSP-Seiten gemäß der JSP-Spezifikation 2.0. Siehe auch : http://java.sun.com/products/jsp/

[61] Tomcat unterstützt Servlets gemäß der Servlet-Spezifikation 2.4. Siehe auch: http://java.sun.com/products/servlet/

In JSP-Seiten ist es möglich statischen Inhalt, z.B. (X)HTML-Code, sowie dynamische Elemente zu mischen. Dynamische Elemente werden direkt an der Stelle in die JSP-Seite eingefügt, an welcher der Output, der durch sie dynamisch erzeugt wird, im Ausgabedokument erscheinen soll. Dynamische Elemente können z.B. JSP-Tags oder Sciptlets mit Java-Code sein. Dies ist vergleichbar mit der Platzierung von PHP-[62] oder ASP-Code[63] innerhalb von Dokumenten, der bei einem Abruf des Dokuments vom Server dynamisch ausgeführt und im Dokument mit seiner Ausgabe ersetzt wird.

Dynamische Elemente in JSP-Seiten sollten nach Möglichkeit deklarativen Charakter haben, d.h. beschreiben was an einer Stelle dynamisch eingefügt werden soll, jedoch nicht beschreiben wie dieser Vorgang auszuführen ist. Auf diese Weise wird versucht die Implementierung der dynamischen Funktionen aus den JSP-Seiten heraus in Java-Klassen auszulagern, wodurch die Wiederverwendbarkeit verwendeter Code-Fragmente stark erhöht und eine bessere Trennung zwischen Präsentations- und Applikationslogik erreicht wird. Zu diesem Zweck werden JSP-Tags als eine Art parametrisierbare Makros für Java-Code verwendet. Beispielweise kann ein JSP-Tag zum Einfügen der aktuellen Uhrzeit geschaffen werden und in einer Vielzahl von JSP-Seiten verwendet werden. Die Implementierung des Tags wird jedoch außerhalb der JSP-Seite in einer einzigen Java-Klasse definiert.

JSP-Tags wurden für eine Vielzahl von Aufgaben und Einsatzzwecke entwickelt und in „JSP Tag Libraries“ zusammengefasst. Damit eine Nutzung von JSP-Tags in einer Vielzahl von Ausführungsumgebungen möglich wird, wurde mit der „Java Server Pages Standard Tag Library“[64] (JSTL) eine Spezifikation für eine standardisierte Sammlung von JSP-Tags erstellt, welche die typischen Funktionalitäten von Web-Applikationen effektiv unterstützen sollen. Im Rahmen der JSP-Seiten des WSPortal wurde hierbei die Referenzimplementierung der JSLT-Spezifikation 1.1 des „Apache Jakarta Taglib“-Projekts[65] verwendet.

Weitere Grundlagen zu JSP, JSLT, Servlets und Tomcat können aufgrund des Umfanges an dieser Stelle nicht vermittelt werden. Eine Einführung in das Themengebiet kann z.B. [13] entnommen werden.

4.4.2 Session-Handling und Gültigkeit von Variablen

Besucht ein Nutzer mehrere Seiten einer Web-Site, so ist es dem Web-Server aufgrund der Verwendung des zustandslosen Kommunikationsprotokolls HTTP nicht direkt möglich zu erkennen, ob diese Seiten von demselben Nutzer aufgerufen werden. Bei einer Web-Site wie dem WSPortal ist dies jedoch zur Realisierung seiner Funktionalität zwingend erforderlich, da hier bestimmte Aktionen nur für registrierte Benutzer angeboten werden sollen und jeder Benutzer nur seine Daten im Portal abrufen und verändern können soll. Das Portal ist darauf angewiesen einen Nutzer

[62] "PHP Hypertext Preprocessor" (PHP) ist eine Open-Source-Programmiersprache die hauptsächlich für Zwecke der Entwicklung serverseitiger Anwendung zur Erstellung von dynamischen Web-Inhalten verwendet wird. Siehe auch: http://www.php.net/

[63] "Active Server Pages" (ASP) ist eine von Microsoft entwickelte Technologie zur serverseitigen dynamischen Generierung von Web-Inhalten und stellt eine Erweiterung des Web-Servers der "Internet Information Services" (IIS) dar. Siehe auch: http://www.asp.net/

[64] Siehe: http://java.sun.com/products/jsp/jstl/

[65] Siehe: http://jakarta.apache.org/taglibs/

identifizieren und über den Aufruf verschiedener Seiten hinweg, d.h. im Verlauf einer Session, erkennen zu können.

Um einen Nutzer über eine Session hinweg erkennen zu können, ist es nötig den Nutzer dazu zu veranlassen einen Identifikator für diese Session (Session-ID) bei jedem Aufruf an die Ausführungsumgebung zu übermitteln. Eine Session-ID kann entweder als Cookie im Browser des Nutzers abgelegt werden oder wird in den Links einer Web-Seite als Parameter integriert. Im ersten Fall wird der Cookie bei jedem Aufruf einer Web-Seite an den Web-Server übermittelt und im zweiten Fall wird die Session-ID als Bestandteil der aufgerufenen URI an den Web-Server übermittelt. Durch Kenntnis der Session-ID kann der Web-Server, oder die nachgelagerte Ausführungsumgebung einer Web-Applikation, einen Nutzer über eine Session hinweg erkennen.

In JSP-Seiten muss ein Session-Handling nicht explizit implementiert werden. Die Ausführungsumgebung nimmt das Session-Handling auf Wunsch vollkommen transparent und selbständig vor. Innerhalb der JSP-Seiten können Variablen bestimmten Geltungsbereichen (Scopes) zugeordnet werden. Hierbei kann die Gültigkeit einer Variablen z.B. auf einen einzigen Aufruf, auf eine ganze Session, oder auf fortwährende globale Gültigkeit in der ganzen Web-Applikation festgesetzt werden.

Wird die Gültigkeit einer Variablen auf eine Session beschränkt, so steht der Inhalt der Variablen in allen JSP-Seiten zur Verfügung, die durch denselben Nutzer aufgerufen werden. Dieser Mechanismus wird verwendet um einen angemeldeten Benutzer über alle JSP-Seiten des WSPortal hinweg zu identifizieren. Nach dem Login des Benutzers werden seine Benutzerdaten in dem Java-Bean `currentUser` festgehalten, welches während der gesamten Session des Benutzers gültig bleibt. Jede JSP-Seite kann daher bei ihrer Ausführung erkennen, ob sie durch einen gültigen, angemeldeten Benutzer ausgeführt wird, und um welchen Benutzer es sich hierbei handelt. Ruft der Benutzer für eine bestimmte Zeitdauer keine JSP-Seiten des WSPortal mehr auf, oder führt er ein Logout durch, so verliert die Session-Variable ihre Gültigkeit und der Nutzer wird als ein nicht angemeldeter Benutzer erkannt.

Konfigurationsdaten, die in allen JSP-Seiten des WSPortal und bei der Benutzung durch alle Nutzer benötigt werden, werden in dem Java-Bean `appSettings` festgehalten, welches eine Gültigkeit für die ganze Web-Applikation besitzt.

```
<jsp:useBean id="appSettings" scope="application"
class="de.tud.kom.dynws.portal.beans.SettingsBean"/>

<jsp:useBean id="currentUser" scope="session"
class="de.tud.kom.dynws.portal.beans.CurrentUserBean"/>

<% if (!currentUser.isLoggedIn()) { %>
    <jsp:forward page="index.jsp" />
<% } %>
```

Abbildung 22 - Deklaration des Gültigkeitsbereichs von Java-Beans und Überprüfung des aktuellen Benutzers (Quelltextausschnitt JSP-Seite)

Abbildung 22 zeigt einen Ausschnitt aus dem Quelltext einer JSP-Seite. Im dargestellten Ausschnitt ist ersichtlich wie die Gültigkeitsbereiche der Java-Beans

`currentUser` und `appSettings` festgelegt werden und eine Überprüfung auf einen aktuell angemeldeten Benutzer erfolgt. Wird festgestellt, dass der aufrufende Nutzer momentan am Portal angemeldet ist, so wird mit der Verarbeitung der JSP-Seite fortgefahren. Wird jedoch festgestellt, dass der Nutzer momentan nicht angemeldet ist, so wird er zur Einstiegsseite des Portals (`index.jsp`) weitergeleitet. Dort hat er die Möglichkeit sich zu registrieren oder, falls er bereits registriert ist, sich anzumelden.

4.4.3 Verarbeitung von Formulardaten

In einigen JSP-Seiten des Portals werden Formulare eingesetzt, die vom Nutzer ausgefüllt werden müssen, bevor der Nutzer eine bestimmte Aktion ausführen kann. Die Verarbeitung von Formulardaten, sowie die Signalisierung der auszuführenden Aktion wird in allen JSP-Seiten des Portals auf die gleiche Weise durchgeführt und soll im Folgenden kurz erläutert werden.

Damit ein Nutzer Aktionen ausführen kann, für deren Ausführung zusätzliche Informationen nötig sind, wird dem Nutzer ein Formular zur Eingabe der nötigen Informationen, sowie ein Button zur Durchführung der Aktion präsentiert. Nachdem ein Nutzer Daten in das Formular eingegeben hat, wählt er über einen Button die Aktion aus, die auszuführen ist. Durch die Betätigung des Buttons wird der Browser des Nutzers dazu veranlasst die Formulardaten, sowie die vom Nutzer gewählte Aktion durch ein HTTP-Post-Kommando an dieselbe JSP-Seite zu senden, die dem Nutzer vormals das Formular präsentiert hat. Um die Formulardaten beim Empfang durch die JSP-Seite abzulegen, wird ein Java-Bean erstellet, dessen Attribute genau dieselben Namen tragen, wie die Bezeichner der übermittelten Formulardaten. Wird ein solches Java-Bean in der JSP-Seite entsprechend deklariert, so übernimmt die Ausführungsumgebung automatisch die Ablage der empfangenen Formulardaten in die Attribute des Java-Beans.

Da bei der Übermittlung der Formulardaten ebenfalls die vom Benutzer ausgewählte Aktion übermittelt wurde, ist bekannt zu welchem Zweck die Formulardaten übermittelt wurden. Je nach auszuführender Aktion kann jetzt überprüft werden, ob die übermittelten Formulardaten vollständig und richtig spezifiziert sind, d.h. ob für die auszuführende Aktion alle nötigen Parameter im korrekten Format vorliegen.

Fehlen nötige Formulardaten oder sind diese falsch spezifiziert, so werden dem Nutzer entsprechende Meldungen angezeigt und er zur Korrektur der Daten aufgefordert. Um dem Nutzer die Korrektur der Daten so angenehm wie möglich zu gestalten, wird ihm das ursprüngliche Formular angezeigt, in welchem die bereits durch ihn übermittelten Daten vorgeblendet sind. Nach der Korrektur der Daten kann der Nutzer durch die Auswahl eines Aktions-Buttons die erneute Übermittlung und Verarbeitung der Formulardaten veranlassen.

Hat der Nutzer alle zur Ausführung der gewünschten Aktion nötigen Daten im korrekten Format spezifiziert, so wird die Aktion durch die JSP-Seite ausgeführt und der Benutzer durch eine entsprechende Meldung über die Ausführung und deren Ergebnis informiert.

4.4.4 Übersicht verwendeter JSP-Seiten

Die Benutzerschnittstelle des WSPortal wird durch fünf JSP-Seiten realisiert, die dem Nutzer jeweils verschiedene Funktionalitäten bieten und untereinander verlinkt sind. Jede JSP-Seite generiert als Ausgabe XHTML-Dokumente[66] in denen Inhalt und Layout strikt getrennt werden. Das Layout wird einheitlich für alle Seiten über eine verknüpfte CSS-Datei[67] festegelegt.

Die einzelnen JSP-Seiten werden im Folgenden kurz beschrieben.

index.jsp
Die Indexseite ist die Einstiegsseite in das Portal. Wird die Seite durch einen Nutzer aufgerufen, der sich noch nicht angemeldet hat, so wird dem Nutzer ein Login-Formular zur Anmeldung, sowie ein Link zu einer Seite, auf der er sich registrieren kann, angezeigt. Das Menü eines nicht angemeldeten Benutzers beschränkt sich auf die Auswahl zwischen der Login-Seite und der Seite auf der er sich registrieren kann.

Wird die Seite durch einen angemeldeten Benutzer aufgerufen (was automatisch nach einem Login oder einer Registrierung geschieht), so wird der Benutzer begrüßt, ihm das Datum seines letzten Logins angezeigt und ihm kurze Hinweise zur Benutzung des Portals gegeben. Das Menü eines angemeldeten Benutzers beinhaltet die Auswahl einer Seite zur Anzeige der bereits angemeldeten Web Services und der für Bewerbungen geöffneten Kategorien, eine Seite zur Veränderung der Daten des Nutzerprofils, sowie einen Logout-Link, welcher den Benutzer wieder abmeldet und zur Einstiegsseite zurückbringt.

registeruser.jsp
Diese Seite dient der Registrierung eines neuen Nutzers am Portal. Zur Registrierung muss der Nutzer einen Login-Namen, ein Passwort, sowie seinen echten Namen angeben. Alle weiteren Angaben bezüglich der Kontaktinformationen sind optional. Bei der Registrierung erfolgt eine Prüfung, ob alle Pflichtfelder ausgefüllt wurden und der Login-Name noch nicht verwendet wurde. Im Fehlerfalle wird der Nutzer durch die Anzeige entsprechender Meldungen zur Korrektur aufgefordert. Nach erfolgreicher Registrierung wird der Nutzer als angemeldeter Benutzer zur Indexseite weitergeleitet.

edituser.jsp
Diese Seite bietet jedem angemeldeten Benutzer die Möglichkeit seine, bei der Registrierung angegebenen Informationen zu überarbeiten. Eine Überprüfung der Angaben erfolgt analog zur Registrierung. Zusätzlich hat der Nutzer die Möglichkeit seine Registrierung zu löschen. In diesem Fall werden alle durch den Nutzer bereits am System angemeldeten Web Services gelöscht und die Score der in den jeweiligen Kategorien verbleibenden Web Services neu berechnet. Nach Abschluss des Vorgangs wird der Benutzer als nicht angemeldeter Benutzer auf die Indexseite weitergeleitet.

[66] Die “Extensible Hypertext Markup Language” (XHTML) [63] ist eine Reformulierung und Erweiterung der Auszeichnungssprache “Hypertext Markup Language” (HTML) in XML-Syntax.
[67] "Cascading Style Sheets" (CSS) [55] [19] bezeichnet einen Mechanismus und eine Sprache zur Beschreibung der Präsentation strukturierter Dokumente, die in Auszeichnungssprachen wie HTML oder XHTML erstellt wurden.

servicelist.jsp
Diese Seite zeigt einem angemeldeten Benutzer eine Liste aller durch ihn bereits am Portal angemeldeten Web Services, sowie eine Liste der Kategorien in denen momentan Bewerbungen möglich sind.

Zu jedem angemeldeten Web Service werden die Informationen angezeigt, die der Nutzer bei der Anmeldung des Web Services angegeben hat. Zusätzlich wird dem Nutzer angezeigt, ob der Web Service bereits durch den Systemadministrator bewertet wurde. Jeder der angezeigten Web Services kann durch Auswahl eines Links gelöscht werden. Nach dem Löschen wird die Score aller, in der jeweiligen Kategorie verbleibenden Web Services neu berechnet und dem Nutzer eine aktualisierte Version der Seite präsentiert.

Zu jeder Kategorie, in welcher momentan Bewerbungen neuer Web Services erlaubt sind, wird ein Link angezeigt, über den der Nutzer zu einer Seite weitergeleitet wird, die zur Anmeldung eines Web Services in dieser Kategorie verwendet werden kann.

applyservice.jsp
Diese Seite ist über einen Link zu einer, für Bewerbungen offenen Kategorie in der Seite `servicelist.jsp` erreichbar und dient der Erfassung von Bewerbungsdaten eines neuen Web Services (vgl. Abbildung 13). Nachdem eine Bewerbung über den Apply-Button abgesendet wurde, wird überprüft ob alle nötigen Daten angegeben wurden. Fehlen Daten, oder wurden diese falsch angegeben (z.B. ungültige Angabe einer Zahl oder eines Datums), so wird der Nutzer über entsprechende Meldungen zur Korrektur aufgefordert. Hat der Nutzer die Rahmendaten der Diensterbringung durch ein SLA-Dokument spezifiziert, so wird das angegebene SLA-Dokument eingelesen, die Informationen aus dem Dokument extrahiert und derselben Prüfung unterzogen wie Daten, die direkt in das Bewerbungsformular eingegeben wurden. Sind alle nötigen Bewerbungsunterlagen vollständig und formal richtig, wird der Web Service als zu bewertender Web Service im System gespeichert und der Nutzer zu der Seite `servicelist.jsp` weitergeleitet.

4.4.5 Übersicht verwendeter Java-Klassen

Viele Funktionalitäten des WSPortal sind nicht direkt in den JSP-Seiten implementiert, sondern in Java-Klassen, die durch die JSP-Seiten verwendet werden. JSP-Seiten dienen der Realisation der Präsentationsschicht des WSPortal und werden daher primär dazu verwendet Web-Seiten zur Interaktion mit dem Benutzer zu erzeugen, auf Benutzerinteraktionen zu reagieren und entsprechende Verarbeitungsprozesse anzustoßen. JSP-Seiten werden hierbei nicht dazu eingesetzt alle Verarbeitungsprozesse vollständig zu implementieren. Der Großteil der Implementierung der Verarbeitungsprozesse ist in JSP-Tags oder Java-Klassen ausgelagert, die innerhalb der JSP-Seiten lediglich verwendet werden.

Tabelle 10 kann eine nach Paketen gruppierte Übersicht der, im Rahmen des WSPortal verwendeten, Java-Klassen entnommen werden.

Paket	Beschreibung
`de.tud.kom.dynws.portal.beans`	Java-Beans zur Verwaltung und Verarbeitung von Informationen in und durch JSP-Seiten. • `CurrentUserBean` • `ServiceBean` • `SettingsBean` • `SLABean` • `UserBean`
`de.tud.kom.dynws.portal.scoring`	Java-Klassen zur Durchführung des Scorings aller Web Services einer Kategorie. • `WSParameters` • `WSPoints` • `WSService` • `WSServiceCategory` • `WSWeights`
`de.tud.kom.dynws.portal.sla`	Java-Klassen zum Einlesen, Parsen und Überprüfen von SLA-Dokumenten. • `End` • `Predicate` • `ServiceLevelObjective` • `SLA` • `SLAParameter` • `SLAParser` • `Value`
`de.tud.kom.dynws.portal.util`	Java-Klassen zum Einlesen grundlegender Konfigurationsparameter des WSPortal und dem Zugriff auf die Datenbanktabellen. • `Settings` • `DBHelper` Properties-Dateien zur Konfigurierung des Log4J-Loggings und des Datenbankzugriffes. • `log4j.properties` • `db.properties`

Tabelle 10 - Durch das WSPortal verwendete Java-Klassen, gruppiert nach Paketen.

Eine genauere Darstellung aller Pakete und der ihnen angehörenden Klassen kann aufgrund des Umfanges an dieser Stelle nicht erfolgen. Im Folgenden soll jedoch die Verarbeitung von SLA-Dokumenten durch die Java-Klassen des Pakets `de.tud.kom.dynws.portal.sla` näher beschrieben werden.

4.4.6 Verarbeitung der SLA-Dokumente

Wählt ein Nutzer in der JSP-Seite `servicelist.jsp` einen Link zur Anmeldung eines neuen Web Services in einer Kategorie, so wird der Nutzer zur JSP-Seite

`applyservice.jsp` weitergeleitet, die ihm ein Anmeldeformular zur Anmeldung eines Web Services in der gewählten Kategorie präsentiert. In diesem Anmeldeformular muss ein Nutzer alle durch das System verwalteten nicht-funktionalen Eigenschaften des Web Services, mit Ausnahme der Reputation, angeben. Der Nutzer kann hierbei diese Eigenschaften einzeln in entsprechende Formularfelder des Anmeldeformulars eintragen, oder stattdessen die URL eines SLA-Dokuments angeben, welches alle diese Eigenschaften enthält.

Aufbau von SLA-Dokumenten

Das Format des SLA-Dokuments wird hierbei durch eine Untermenge der XML-basierten Sprache „Web Service Level Agreement“ (WSLA) [57], die von IBM zur Spezifikation von SLA-Dokumenten entwickelt wurde, definiert. Jede anzugebende Eigenschaft wird im SLA-Dokument durch ein `ServiceLevelObjective`-Element und seine Kindelemente gekapselt. Der Aufbau eines SLA-Dokuments kann der Abbildung 23 entnommen werden.

```
<SLA>
      <ServiceLevelObjective>
            ...
            <Validity>
                  ...
                  <End>2005-12-31</End>
            </Validity>
            <Expression>
                  <Predicate type="Greater">
                        <SLAParameter>Availability</SLAParameter>
                        <Value>98.5</Value>
                  </Predicate>
            </Expression>
            ...
      </ServiceLevelObjective>

      <ServiceLevelObjective>
            ...
      </ServiceLevelObjective>

      ...
</SLA>
```

Abbildung 23 - Aufbau eines SLA-Dokuments

Jedes `ServiceLevelObjective`-Element (SLO-Element) umschließt die Spezifikation einer anzugebenden Eigenschaft. Das SLO-Element besitzt hierbei zwei direkte Kindelemente: Das `Validity`-Element zur Spezifikation der Gültigkeitsdauer der Garantie einer Eigenschaft, sowie das `Expression`-Element zur Formulierung eines Ausdrucks, der die garantierte Eigenschaft spezifiziert.

Die Gültigkeitsdauer der Garantie wird innerhalb des `Validity`-Elements durch ein `End`-Element definiert, welches ein Datum im Format „<Jahr>-<Monat>-<Tag>“ umschließt.

Die eigentliche garantierte Eigenschaft wird durch drei Einzelangaben definiert: Eine bezeichnete Eigenschaft, ein Wert und ein Vergleichsoperator, der die Eigenschaft mit

dem Wert verknüpft. Die Garantie, dass die Verfügbarkeit eines Web Services größer als 98,5% beträgt, würde durch die Eigenschaftsbezeichnung „Availability“, den Wert „98.5“, sowie den Vergleichsoperator „Greater“ definiert werden. Der Vergleichsoperator wird durch das Attribut `type` des `Predicate`-Kindelements des `Expression`-Elements definiert. Erlaubte Angaben für einen Vergleichoperator sind hierbei „Less“ (kleiner), „Equal“ (gleich) und „Greater“ (größer). Innerhalb des `Predicate`-Elements wird der Eigenschaftsbezeichner durch ein `SLAParameter`-Element umschlossen und der Wert durch ein `Value`-Element.

Tabelle 11 kann entnommen werden, welche nicht-funktionalen Eigenschaften eines Web Services unter Angabe welcher Eigenschaftsbezeichner in SLO-Elemente gekapselt in einem vollständigen SLA-Dokument erwartet werden.

Nicht-funktionale Eigenschaft	Eigenschaftsbezeichner
Verfügbarkeit	`availability`
Durchsatz	`throughput`
Antwortzeit	`responsetime`
Verschlüsselung	`encryption`
Authorisierung	`authorisation`
Authentifizierung	`authentication`
Referenzen	`references`
Preis	`price`

Tabelle 11 - Eigenschaftsbezeichner für nicht-funktionale Eigenschaften eines Web Services in einem vollständigen SLA-Dokument

Für die Eigenschaftsbezeichner `availability`, `throughput`, `responsetime` und `price` wird die Angabe eines numerischen Wertes im `Value`-Element erwartet.

Die nicht-funktionale Eigenschaft der Gültigkeit, welche den Ablauf der Gültigkeit der durch die SLA ausgesprochenen Garantien beschreibt, wird als frühstes Verfallsdatum aller `ServiceLevelObjective`-Elemente eines SLA-Dokuments bestimmt.

Ein SLA-Dokument wird nur als vollständig und korrekt angesehen, falls es alle genannten nicht-funktionalen Eigenschaften eines Web Services in der soeben beschriebenen Form spezifiziert.

Parsing eines SLA-Dokuments

Wurde zur Spezifikation der nicht-funktionalen Eigenschaften auf der JSP-Seite `applyservice.jsp` eine URL zu einem SLA-Dokument angegeben, so wird während der Prüfung der Anmeldedaten eines Web Services eine Instanz der Klasse `SLAParser` erzeugt und das SLA-Dokument über die Funktion `readSlaDocument` eingelesen. Konnte das SLA-Dokument von der angegebenen URL eingelesen werden, so wird über die Funktion `parse` ein Parsing des eingelesenen Dokuments durchgeführt. Hierzu wird zunächst versucht eine DOM-Repräsentation[68] des

[68] Das “Document Object Model” (DOM) ist ein Standard des "World Wide Web Consortium" (W3C) für ein Modell zum programmiersprachenunabhängigen Zugriff auf eine baumorientierte Repräsentation strukturierter Dokumente (z.B. XML-Dokumente). Siehe auch: http://www.w3.org/DOM/

Dokuments zu erstellen. Kann das eingelesene Dokument als DOM-Baum dargestellt werden und besitzt der DOM-Baum ein SLA-Element als Wurzelelement, so wird mit dem eigentlichen Parsing begonnen.

Für jedes zu parsende Element des SLA-Dokuments existiert eine eigene Klasse, die in der Lage ist ihr spezifisches Element über die Funktion `parse` zu parsen. Enthält ein zu parsendes Element zu parsende Kindelemente, so wird durch die Klasse, die das Element parsed, für jedes Kindelement eine entsprechende Klasse für das Parsing instanziiert und das Parsing der Kindelemente wiederum mit der Funktion `parse` auf der erzeugten Instanz durchgeführt. Diese Klassen parsen Kindelemente wiederum analog. Werden im DOM-Baum auf diese Weise die Blätter erreicht, so wird der Inhalt des Elements, sowie dessen Gültigkeit ermittelt und das Parsing des Elements beendet. Nachdem der aufrufenden Klasse alle Ergebnisse der Kindelemente bekannt sind, bestimmt sie ihre Inhalte und deren Gültigkeit aus den Inhalten und der Gültigkeit ihrer Kindelemente. Das Parsing ist abgeschlossen, wenn auf diesem Wege die Verarbeitung wieder das SLA-Wurzelelement erreicht.

Konkret wird der Klasse `SLA` das SLA-Wurzelelement übergeben. Sie selektiert alle `ServiceLevelObjective`-Kindelemente und instanziiert für jedes dieser Kindelemente ein Objekt der Klasse `ServiceLevelObjective`. Dieses selektiert aus seinem `ServiceLevelObjective`-Element das `End`-Kindelement des `Validity`-Kindelements und übergibt es an eine neue Instanz der `End`-Klasse. Das `Predicate`-Kindelement des `Expression`-Kindelements übergibt es an eine neue Instanz der `Predicate`-Klasse. Die `End`-Klasse extrahiert das Datum, welches das `End`-Element umschließt und prüft dessen Gültigkeit. Die `Predicate`-Klasse instanziiert für ihre Kindelemente `SLAParameter` und `Value` Objekte der Klassen `SLAParameter` und `Value`. Die Klasse `SLAParameter` ermittelt den Eigenschaftsbezeichner, der durch das `SLAParameter`-Element umschlossen wird. Die Klasse `Value` ermittelt den Wert, den das `Value`-Element umschließt. Die Klasse `Predicate` speichert die Inhalte der Kindelemente `SLAParameter` und `Value`, sowie den Wert des Attributs `type` des `Predicate`-Elements. Sind alle Kindelemente und der Attributwert vorhanden und mit gültigen Werten belegt, so wird auch das `Predicate`-Element für gültig erklärt. Die Klasse `ServiceLevelObjective` sammelt auf diese Weise ebenfalls die Inhalte der relevanten Subelemente `End` und `Predicate` und bestimmt seine Gültigkeit aus der Gültigkeit aller Subelemente. Die `SLA`-Klasse wiederum sammelt alle Objekte der Klasse `ServiceLevelObjective` in einem `Vector`-Objekt. Sind alle `ServiceLevelObjective`-Objekte im `Vector` gültig, so wird das ganze SLA-Dokument für gültig erklärt und das Verfallsdatum des gesamten SLA-Dokuments als frühestes Verfallsdatum unter allen `ServiceLevelObjective`-Objekten ermittelt.

Nach dem Parsing wird in der JSP-Seite, über welche die Anmeldung des Web Services ausgelöst wurde, über ein Objekt der Klasse `ServiceBean` geprüft, ob alle benötigten Eigenschaften durch SLO-Objekte spezifiert wurden und ob diese gültig sind. Wurde ein nicht vollständiges oder nicht korrektes SLA-Dokument übermittelt, bekommt der Nutzer ausführliche Informationen darüber angezeigt, welche nicht-funktionalen Eigenschaften in seinem SLA-Dokument fehlen oder falsch spezifiziert wurden. Wurde ein vollständiges und korrektes SLA-Dokument übermittelt, so wird mit der Verarbeitung der Anmeldung des Web Services so fortgefahren, als wären die nicht-funktionalen Eigenschaften direkt in das Anmeldeformular eingegeben worden.

4.5 WSProxyAdmin

WSProxyAdmin ist das Administrations-Frontend von WSQoSX. Seine Aufgabe ist es alle relevanten Informationen in der gemeinsamen Datenbasis des Web-Service-Proxy WSProxy und des Web-Portals WSPortal zu verwalten und zu pflegen. Funktionalitäten die durch den WSProxyAdmin angeboten werden umfassen u.a.:

- Konfiguration des Routings von Web Services durch den WSProxy.
- Bewertung von neu am WSPortal angemeldeten Web Services.
- Verwaltung der Web-Service-Kategorien inklusive der Gewichtung der nicht-funktionalen Eigenschaften und der Regeln zur Definition der Mindestanforderungen an Web Services.
- Verwaltung der durch das System verwendeten Web Services.
- Verwaltung der am WSPortal registrierten Anbieter.
- Einsicht der beim Aufruf von Web Services generierten Warnungen, sowie der Call-History, der History-Statistik und der Logging-Tabelle.

Das Administrations-Frontend WSProxyAdmin wurde auf Basis von Microsoft Access entwickelt, und bietet dem System-Administrator eine graphische Benutzeroberfläche über die er die genannten Funktionalitäten in einer übersichtlichen und benutzerfreundlichen Art und Weise nutzen kann. Bei der Entwicklung des WSProxyAdmin wurde „Microsoft Access 2003“ eingesetzt, dieser jedoch aus Gründen besserer Abwärtskompatibilität im Format von „Microsoft Access 2000“ gespeichert. WSProxyAdmin kann daher mit allen Versionen von Microsoft Access ab der Version 2000 betrieben werden.

Bevor im Folgenden näher auf das Administrations-Frontend eingegangen wird, wird zunächst das Datenbanksystem Microsoft Access kurz erläutert, auf dessen Basis das Administrations-Frontend implementiert wurde.

4.5.1 Microsoft Access als verwendete Implementationsbasis

Microsoft Access (im Folgenden kurz Access) ist ein relationales Datenbanksystem, welches neben der Verwaltung der Daten in Form von Tabellen und der Abfragemöglichkeit durch SQL-Queries auch die Entwicklung von Datenbankanwendungen ermöglicht. Zu diesem Zweck lassen sich zu einer Datenbank (u.a.) Formulare, Berichte und Quellcode-Module hinzufügen. Durch Formulare kann eine graphische Benutzerschnittstelle aufgebaut werden und mittels Berichten lassen sich Daten in eine, für den Ausdruck geeigneten Art und Weise aufbereiten. Sowohl Formulare, als auch Berichte ermöglichen das Einbetten von Quellcode der Programmiersprache „Microsoft Visual Basic for Applications“ (VBA). Neben der Integrationsmöglichkeit von VBA-Code in Formularen und Berichten kann VBA-Code auch in eigenständigen Quellcode-Modulen organisiert und an anderer Stelle der Datenbankanwendung genutzt werden.

Access speichert alle Daten einer Datenbankanwendung (Tabellen, Abfragen, Formulare, Berichte, Code-Module) in einer einzigen Datei. Für den Zugriff auf die Daten der Tabellen und der Auswertung der Abfragen verwendet Access eine eigne als „Jet“ bezeichnete Datenbankengine. Der Zugriff auf externe Datenbanken und Datenbankserver ist über entsprechende ODBC-Treiber[69] und die ODBC-API

[69] "Open Database Connectivity" (ODBC) bezeichnet eine standardisierte Schnittstelle zum einheitlichen Zugriff auf Datenbankserver verschiedener Hersteller. Die erste Spezifikation von ODBC

möglich. Für die Entwicklung von Datenbankanwendungen bietet Access ein umfangreiches Framework an Klassen und Objekten. Für den Zugriff auf Daten stellt das Framework hierbei u.a. Zugriffsmethoden über die Schnittstellen „Data Access Objects“ (DAO) und „Active Data Objects“ (ADO) bereit.

Auf eine umfassendere Darstellung von Access und relevanter Technologien soll im Rahmen dieses Technical Reports verzichtet werden. Eine detailliertere Einführung kann z.B. [82] oder [23] entnommen werden.

4.5.2 Gemeinsame Datenhaltung der Kernkomponenten

Die Kernkomponenten von WSQoSX verwenden zur Ablage ihrer gemeinsamen Datenbasis gemeinschaftlich den Datenbankserver MySQL (siehe auch Abbildung 19). In jeder Kernkomponente lässt sich der Zugriff auf den Datenbankserver individuell konfigurieren, so dass es auf einfache Art und Weise möglich wird die einzelnen Kernkomponenten und den Datenbankserver auf mehrere unterschiedliche Systeme zu verteilen. In jeder Kernkomponente können folgende Angaben für den Zugriff auf die gemeinsame Datenbasis individuell festgelegt werden:

- IP-Adresse und Port des Datenbankservers.
- Zu verwendender Treiber[70] für den Zugriff auf den Datenbankserver.
- Benutzername und Passwort für die Anmeldung am Datenbankserver.
- Name der zu verwendenden Datenbank auf dem Datenbankserver.
- Name der zu verwendenden Tabellen in der Datenbank.

Die Kernkomponenten erwarten einen gewissen Aufbau der in der Konfiguration spezifizierten Tabellen, sind jedoch vollkommen flexibel bezüglich der Benennung der Tabellen, der Datenbank, der Lage des Datenbankservers, sowie der zum Zugriff nötigen Benutzeridentifikation.

In der Standard-Installation von WSQoSX wird der Datenbankserver MySQL in der Version 4.1.9 verwendet. Er wird lokal betrieben und verwendet zur Kommunikation den MySQL-Standard-Port 3306. Der Name der Datenbank zur gemeinsamen Datenhaltung ist „wsproxy“. Bei einem Verbindungsaufbau wird der Benutzer „prototyp“ und das Passwort „webservice“ verwendet. Dieser Benutzer besitzt nur Rechte zum Zugriff auf die Datenbank „wsproxy“, wobei diese Rechte nur solche umfassen die zwingend für die Datenmanipulationen durch die Kernkomponenten nötig sind.

Eine Übersicht der Tabellen, die zur Datenablage in der Datenbank „wsproxy“ verwendet werden, kann zusammen mit einer kurzen Beschreibung ihres Inhalts der Tabelle 12 entnommen werden.

wurde 1992 durch die "SQL Access Group" erarbeitet und setzte sich seitdem vor allem im Umfeld des Betriebssystems Windows durch.

[70] Im Falle des WSProxy und WSPortal ist ein JDBC-Treiber anzugeben. Im Falle des WSProxyAdmin ein ODBC-Treiber.

Tabellennname	Kurzbeschreibung des Inhalts
ws_mapping	Zuordnung eines WSID zu einem USID.
ws_routing	Zuordnung eines USID zu einem Target.
ws_portal_categories	Web-Service-Kategorien inkl. der jeweiligen Gewichtung der nicht-funktionale Eigenschaften.
ws_portal_categories_rules	Regeln zur Definition von Mindestanforderungen in den einzelnen Kategorien.
ws_portal_services	Am System angemeldete Web Services inkl. ihrer jeweiligen Score.
ws_portal_services_parameters	Garantien bzgl. der nicht-funktionalen Eigenschaften eines Web Services, die bei der Anmeldung durch den Anbieter zugesichert wurden.
ws_portal_services_points	Punkte für die nicht-funktionalen Eigenschaften eines Web Services, die vom System-Administrator vergeben oder durch Normierung aus den garantieren Eigenschaften berechnet wurden.
ws_logging	Logging-Tabelle mit detaillierten Informationen zu jedem durch den WSProxy verarbeiteten HTTP-Request.
ws_history_calls	Daten der Call-History.
ws_history_statistics	Daten der History-Statistik.
ws_history_warnings	Warnungen, die bei einem Web-Service-Aufruf übder den WSProxy durch das QoS-Monitoring erzeugt wurden.
ws_portal_user	Registrierte Anbieter des WSPortal.

Tabelle 12 - Kurzbeschreibung der zur Datenablage verwendeten Tabellen

Auf eine ausführliche Darstellung der Tabellenstrukturen soll im Rahmen dieses Technical Reports verzichtet werden.

4.5.3 Datenzugriff

Um aus Access heraus auf den Datenbankserver MySQL zugreifen zu können wurde der ODBC-Treiber MyODBC[71] in der Version 3.51.10 verwendet. Alle Tabellen der Datenbank „wsproxy“ wurden über den ODBC-Treiber in der Access-Datenbank des WSProxyAdmin verknüpft. Stellt WSProxy beim Start fest, dass auf die Datenbank oder einzelne Tabellen nicht zugegriffen werden kann, so wird dem Nutzer ein Formular angezeigt, über welches er zu jeder einzelnen Tabelle alle nötigen ODBC-Verbindungsparameter zu ihrer Verknüpfung bearbeiten kann. Auf diese Weise ist nach einer Veränderung der Systemkonfiguration eine bequeme Einrichtung der neuen Verbindungsparameter der Datenbank möglich.

4.5.4 Verwendete Formulare

Die Benutzeroberfläche des WSProxyAdmin wurde durch Formulare realisiert. Durch Access werden hierbei eine Vielzahl von Komponenten angeboten, die sich in

[71] Siehe: http://dev.mysql.com/downloads/connector/odbc/

Formulare integrieren lassen und grundlegende Funktionalitäten für die Entwicklung einer Datenbankanwendung zur Verfügung stellen. Hierzu zählen z.B. Komponenten zum Anzeigen, Hinzufügen, Ändern und Löschen von Daten, Komponenten zum Navigieren in Daten und Komponenten zur Realisierung von allgemeinen Benutzerinteraktionen. Jede Komponente ist in der Lage auf bestimmte Ereignisse (Events) zu reagieren, wobei das Verhalten der Komponenten beim Auftreten von bestimmten Events über VBA-Code individuell angepasst werden kann. Auf diese Art und Weise lässt sich mit VBA-Code eine individuelle Anwendungslogik über Formulare realisieren. VBA-Code lässt sich nicht nur in Formulare integrieren, sondern kann auch eigenständig in Form von Modulen erfasst werden. Hierbei kann ein Modul entweder eine Sammlung von Funktionen und Methoden oder eine Klassendefinition enthalten. Über Module definierte Funktionen und Methoden lassen sich in Formularen verwenden und definierte Klassen instanziieren.

Die wichtigsten, zur Realisierung der Benutzeroberfläche des WSProxyAdmin verwendeten Formulare werden im Folgenden genannt und ihre Funktionalität kurz beschrieben.

FormMainMenu
Dieses Formular realisiert das Hauptmenü des WSProxyAdmin, welches dem Nutzer direkt nach dem Start angezeigt wird. Von hier aus können alle anderen Formulare über entsprechende Buttons aufgerufen oder der WSProxyAdmin beendet werden.

FormMapping
Dieses Formular dient der Konfiguration des Routings von HTTP-Requests durch den WSProxy. Das Routing wird über die Zuordnung von WSID zu USID und USID zu Target festgelegt. Das Formular zeigt die bestehenden Zuordnungen, welche bearbeitet oder gelöscht werden können. Neue Zuordnungen können über das Formular definierent werden, wobei sich ein Target nur einem USID zuordnen lässt, falls der USID bereits einem WSID zugeordnet wurde.

FormCategories
Dieses Formular dient der Verwaltung von Web-Service-Kategorien. Zu jeder Kategorie werden Details wie Name, Kurzbeschreibung, sowie die URL-Adressen der ausführlichen Beschreibung, des WSDL-Dokuments der zu implementierenden Schnittstelle und der Vorlage des SLA-Dokuments angezeigt. Weiterhin lassen sich in jeder Kategorie die Gewichtung der nicht-funktionalen Eigenschaften von Web Services, sowie die Regeln zur Definition von Mindestanforderungen an Web Servcies festlegen. Kategorien können angelegt, verändert oder gelöscht werden. Im Falle der Löschung eine Kategorie werden ebenfalls alle Web Services gelöscht, die in dieser Kategorie angemeldet wurden.

FormServices
Dieses Formular dient der Verwaltung aller angemeldeten Web Services. Zu einem Web Service werden alle bei seiner Anmeldung angegebenen Daten angezeigt, die Daten seines Anbieters, die Regeln für die Mindestanforderungen in seiner Kategorie, die Gewichtung der nicht-funktionalen Eigenschaften seiner Kategoerie, die Bewertung seiner nicht-funktionalen Eigenschaften in Form von Punkten und die hieraus berechnete Score. Wurde zu einem Web Service noch keine Score berechnet, d.h. der Web Service ist unbewertet, so werden die Punkte zur Bewertung seiner quantifizierbaren Eigenschaften automatisch durch eine Normierung in der Kategorie

berechnet und angezeigt. Punkte zur Bewertung der weichen Eigenschaften (vgl. Kapitel 3.1.6) müssen durch den Administrator selbst in entsprechende Felder des Formulars eingetragen werden (siehe hierzu auch Abbildung 14). Anschließend kann die Score eines Web Services über den Button „Score“ berechnet werden. Die Bewertung eines Web Services kann über den Button „Unscore“ wieder aufgehoben werden. Wird ein Web Service gelöscht, so werden die Bewertungspunkte aller in derselben Kategorie verbleibenden Web Services neu normiert und ihre Score neu berechnet.

Das Formular kann über das Hauptmenü in einem speziellen Modus aufgerufen werden, in welchem es nur Web Services zeigt, die noch nicht bewertet wurden. Hierdurch kann ein Administrator schnell erkennen, ob neue, bisher noch unbewertete Web Servcies vorliegen und diese direkt über das Formular bewerten.

FormUsers
Dieses Formular dient der Verwaltung aller am WSPortal registrierten Anbieter. Zu jedem Anbieter wird der Zeitpunkt seiner letzten Anmeldung am Portal angezeigt, sowie alle Daten, die er bei seiner Registrierung am Portal angegeben hat. Wird ein Anbieter gelöscht, so werden auch alle durch ihn angebotenen Web Services gelöscht. Durch das Löschen eines Web Services werden hierbei die Bewertungspunkte aller in der jeweiligen Kategorie verbleibenden Web Services neu normiert und ihre Score neu berechnet.

FormOverview
Das Formular zeigt einer Übersicht aller Kategorien, angemeldeten Web Services und registrierten Anbieter. Durch die Auswahl einer Kategorie oder eines Anbieters kann die Anzeige der Web Services auf die Web Services der gewählten Kategorie oder des gewählten Anbieters beschränkt werden. Wird ein Doppelklick auf eine Kategorie, einen Web Service oder einen Anbieter ausgeführt, so wird das entsprechende Formular zur Verwaltung der Elemente dieses Typs geöffnet und das gewählte Element angesprungen.

FormHistory
Das Formular zeigt die Historty-Statistik, die Call-History und alle Warnungen, die nach der Durchführung von Web-Service-Aufrufen im Rahmen des QoS-Monitorings generiert wurden. Durch die Auswahl eines Web Services in der History-Statistik kann die Ansicht in der Call-History und der Liste der Warnungen auf Einträge beschränkt werden, die den gewählten Web Service betreffen. Über einen Doppelklick auf ein Element der Call-History kann der entsprechende Eintrag der Logging-Tabelle aufgerufen und angezeigt werden.

FormLoggingTable
Dieses Formular zeigt alle Einträge der Logging-Tabelle (vgl. Kapitel 3.2.4), in einer nach Eintragsdatum sortierten Liste.

FormConfigureODBCTables
Sollte bei einem Start des WSProxyAdmin festgestellt werden, dass die Verknüpfung von mindestens einer Tabelle aus der Datenbank des MySQL-Servers fehlgeschlagen ist, so wird dem Nutzer dieses Formular angezeigt. Hier kann er die Konfiguration der ODBC-Verbindungsparameter für die vom MySQL-Server zu verknüpften Tabellen bearbeiten und die Verknüpfung erneut veranlassen.

4.5.5 Scoring der Web Services

Da aufgrund des Umfangs nicht auf die Details der einzelnen Formulare eingegangen werden kann, sollen hier exemplarisch einige Aspekte der Implementierung des Scorings durch das Formular `FormService` erläutert werden.

Wie bereits in Kapitel 3.1.6 geschildert, müssen vor der Berechnung der Score eines Web Services seine nicht-funktionalen Eigenschaften in Form von Punkten bewertet werden. Die Punkte der quantifizierbaren Eigenschaften können durch eine Normierung anhand der Minimal- und Maximalwerte aller quantifizierbaren Eigenschaften einer Kategorie berechnet werden, wohingegen die weichen Eigenschaften manuell durch die Vergabe von Punkten im Wertebereich zwischen null und zehn bewertet werden müssen.

Das Formular `FormService` zeigt zu jedem Web Service seine Bewertungsmatrix an (siehe Abbildung 14). In der Spalte „Points“ werden hierbei die Punkte angezeigt, mit denen die einzelnen nicht-funktionalen Eigenschaften des Web Services bewertet wurden. Die Spalte „Weights“ zeigt die, über die Kategorie definierten, Gewichte der einzelnen nicht-funktionalen Eigenschaften. Die Spalte „Fractions of Score“ zeigt den Anteil an der Score an, der durch die Multiplikation des Gewichts einer nicht-funktionalen Eigenschaft mit seinen Bewertungspunkten entsteht.

Wird im Formular zu einem anderen Web Service navigiert, so müssen die Informationen in der Bewertungsmatrix aktualisiert werden. Bei einem Wechsel des angezeigten Web Services werden zunächst alle Formularfelder, die direkt mit der Datenquelle des Formulars verknüpft sind, aktualisiert. Diese Aktualisierung geschieht automatisch und muss nicht durch VBA-Code implementiert werden. Auf diese Weise wird die Spalte „Weights“, sowie die Score des Web Services aktualisiert. Nachdem die verknüpften Anzeigefelder aktualisiert wurden wird im Formular das Event `OnCurrent` ausgelöst und die hiermit verknüpfte Methode `Form_Current()` des Formulars aufgerufen. Hierin müssen nun die Spalte „Points“ und „Fractions of score“ aktualisiert werden. Wird anhand der Existenz einer Score festgestellt, dass es sich um einen bereits bewerteten Web Service handelt, so werden die Punkte aus der Tabelle `ws_portal_services_points` ausgelesen und den entsprechenden Anzeigefeldern als Inhalt zugewiesen. Der Inhalt der Anzeigefelder der Spalte „Fractions of score“ wird anschließend durch die Multiplikation der Punkte mit dem jeweiligen Gewicht berechnet.

Handelt es sich bei dem anzuzeigenden Web Service jedoch um einen noch nicht bewerteten Web Service, so liegen noch keine Punktbewertungen für diesen Web Service vor. Um dies zu signalisieren werden die Felder zur Anzeige der Punktbewertungen für die weichen Eigeneschaften rot hinterlegt und für Benutzereingaben freigeschaltet. Die Punktbewertungen für die quantifizierbaren Eigenschaften werden anschließend durch eine Normierung berechnet und angezeigt.

Zur Normierung der quantifizierbaren Eigenschaften wird eine neue Instanz der Klasse `WSWebServiceCategory` instanziiert und die Kategorie, welcher der aktuell anzuzeigende Web Service angehört, an das erzeugte Objekt übergeben. Über die Methode `load` werden nun alle Web Services dieser Kategorie aus den Tabellen der Datenbank ausgelesen und in Form von `WSWebService`-Objekten in einem Array abgelegt. In jedem `WSWebService`-Objekt sind die Werte der nicht-funktionalen

Eigenschaften über ein `WSParameters`-Objekt verknüpft und die Punktbewertungen (soweit vorhanden) über ein `WSPoints`-Objekt. Durch den Aufruf der Methode `normalizePoints` auf dem `WSWebServiceCategory`-Objekt werden die Punkte für die quantifizierbaren Eigenschaften für alle Web Services der Kategorie durch eine Normierung berechnet. Die Normierung wird wie in Kapitel 3.1.6 beschrieben anhand der eingelesenen Informationen der Web Services durchgeführt. Nach der Normierung wird das `WSWebService`-Objekt, welches den momentan im Formular anzuzeigenden Web Service repräsentiert, aus dem Array der `WSWebService`-Objekte des `WSWebServiceCategory`-Objekts entnommen. Hieraus werden die normierten Punkte der quantifizierbaren Eigenschaften aus dem verknüpften `WSPoints`-Objekt entnommen und in den entsprechenden Anzeigefeldern der Spalte „Points“ im Formular angezeigt.

Um die Score des angezeigten Web Services berechnen zu können, muss der Administrator zunächst noch die Punktbewertung der weichen Eigenschaften in die entsprechenden Felder des Formulars eintragen. Die Felder besitzen eine Eingabemaske, welche nur die Eingabe von Zahlenwerten zwischen null und zehn zuläßt. Wurde ein Feld mit einer gültigen Punktbewertung gefüllt, so ändert es die Hintergrundfarbe von rot auf grün und veranlasst die Aktualisierung des, direkt mit der Punktbewertung zusammenhängenden Feldes der Spalte „Fractions of score“.

Wurden auf diese Weise alle Punktbewertungen für die nicht-funktionalen Eigenschaften des Web Services ermittelt, so kann, durch einen Klick auf den Button „Score“, die Score des Web Services ermittelt werden. Der Klick löst das `OnClick`-Event des Buttons aus, der mit der Methode `cmdScore_Click()` des Formulars verknüpft ist, die daraufhin ausgeführt wird. Zunächst werden die Punktbewertungen des Web Services in der Tabelle `ws_portal_services_points` gespeichert. Anschließend wird geprüft, ob der Web Service alle Regeln der Mindestanforderung an Web Services seiner Kategorie erfüllt. Diese Überprüfung wird durch die Funktion `checkRulesForService` vorgenommen, die in dem Code-Modul `ModulGlobal` außerhalb des Formulars definiert ist und auf deren Funktionsweise an dieser Stelle nicht näher eingegangen werden soll. Erfüllt der Web Service alle Mindestanforderungen, so wird seine Score als Summe der Summanden bestimmt, die in der Spalte „Fractions of score“ bereits berechnet wurden. Wird auch nur eine Regel der Mindestanforderungen durch den Web Service verletzt, so wird als Score -1 definiert.

Nachdem die Score des Web Services ermittelt wurde, wird diese dem Administrator über eine Dialogbox angezeigt. Dieser kann hier entscheiden ob er die Score des Web Services in der Datenbank speichern möchte, oder sie verwerfen möchte. Wird die Score verworfen, so werden auch alle Punktbewertungen für Eigenschaften wieder aus der Tabelle `ws_portal_services_points` entfernt. Wird die Score hingegen akzeptiert, so wird sie in der Tabelle `ws_portal_services` zum entsprechenden Web Service gespeichert und im Formular angezeigt. Hierbei wird der Hintergrund des Feldes zur Anzeige der Score grün gefärbt, falls die Score nicht -1 beträgt, d.h. falls der Web Service die Mindestanfoderungen seiner Kategorie erfüllt, ansonsten wird der Hintergrund rot gefärbt. Das Scoring eines Web Services ist damit abgeschlossen.

Durch diese Art des Scorings ist es dem WSProxy zur Laufzeit auf sehr einfache Art und Weise möglich den besten Web Service einer Kategorie im Rahmen des „portal“-

Suchverfahrens zu selektieren. Hierzu verwendet er eine SQL-Query um aus der Tabelle `ws_portal_services` eine, nach der Score sortieten Liste aller Web Services zu erhalten, deren Score größer oder gleich null ist. Das erste Element der Ergebnisliste ist stets der Web Service mit der höchsten Score und damit der am besten geeignete Web Service seiner Kategorie.

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

Im Rahmen unserer Forschungsarbeiten wurde eine Dienstgüte unterstützende Web-Service-Architektur für flexible Geschäftsprozesse (WSQoSX) entwickelt, welche in der Lage ist Web Services dynamisch, unter Berücksichtigung ihrer Dienstgüte und weiterer Rahmenbedingungen, auszuwählen, in Geschäftsprozesse einzubinden und auszuführen.

Die Verwendung von Web Services zur Abbildung von Prozessbausteinen wurde gewählt, da durch die Möglichkeit diese plattformunabhängig und interoperabel einsetzen zu können eine leichte Integration verschiedenster Systeme, auch über Unternehmensgrenzen hinweg, ermöglicht wird. Die Anbindung externer Systeme ist über die herkömmlichen Kanäle des Internets problemlos möglich.

Die Flexibilisierung von Geschäftsprozessen wurde durch die dynamische Einbindung von Prozessbausteinen, in der Form von Web Services, in einen aus Web Services orchestrierten Geschäftsprozess realisiert. Hierbei wurde die Flexibilisierung unter dem Fokus der Optimierung des Geschäftsprozesses durch den Einsatz möglichst optimaler Web Services für jeden Prozessschritt betrachtet und realisiert.

Um den Einsatz möglichst optimaler Web Services für jeden Prozessschritt zu ermöglichen wurde ein Konzept zur Beschreibung eines Web Services erarbeitet, der über die syntaktische Beschreibung seiner Schnittstelle in Form von WSDL-Dokumenten hinausgeht. Web Services werden durch das System nach ihrer Funktionalität und Schnittstelle in Kategorien gruppiert, sowie ihre nicht-funktionalen Eigenschaften und Rahmenbedingungen ihrer Nutzung in der Form von SLAs erfasst. Externe Anbieter können Web Services über ein Web-Portal zur Nutzung durch das System anmelden, wobei zur elektronischen Übermittlung der SLA ein Format, bzw. Beschreibungssprache eingeführt wurde, welches auf einer Teilmenge der XML-basierten Sprache WSLA basiert.

Damit zur Laufzeit des Geschäftsprozesses ein Web Service dynamisch eingebunden werden kann, wurde mittels eines Web-Service-Proxy eine Zwischenschicht zwischen der Ausführungsumgebung des Geschäftsprozesses und der durch sie verwendeten Web Services eingeführt. Der generische Web-Service-Proxy ist in der Lage beliebige Web-Service-Aufrufe dynamisch zu routen, wobei das Routing über Suchverfahren für das Weiterleitungsziel konfiguriert werden kann. Es wurde ein Routingverfahren vorgestellt, welches aus der Menge potentiell geeigneter Web Services, durch ein vorgestelltes Modell für Präferenzstrukturen und der Bewertung von Web Services, das Maß der Übereinstimmung von Web Services mit der definierten Präferenzstruktur erkennen kann und dadurch in der Lage ist, den Web-Service-

Aufruf an den, gemäß der Präferenzstruktur am besten geeigneten Web Service weiterzuleiten.

Zur Kontrolle der durch den Anbieter eines Web Services in der SLA garantierten technischen Dienstgüte wird die tatsächlich bei Web-Service-Aufrufen erbrachte technische Dienstgüte gemessen, mit den Garantien der SLA verglichen und im Falle von Diskrepanzen entsprechende Warnungen erzeugt. Durch ein Logging aller Web-Service-Aufrufe und das Führen von Statistiken über die Performanz der einzelnen Web Services wird hierbei eine umfangreiche Datenbasis geschaffen, die sich für die verschiedensten Zwecke des Accountings und der Performazanalyse auswerten lässt.

Die Architektur und alle Komponenten des entwickelten WSQoSX wurden erläutert, wobei in besonderem Maße erwähnenswert ist, dass es zur Einführung von WSQoSX in eine bestehende Web-Service-Umgebung lediglich der Veränderung einer URL in den Web-Service-Clients bedarf. Durch dieses Maximum an Kompatibilität zur bisherigen Web-Service-Umgebung fällt der Integrationsaufwand minimal aus.

5.2 Ausblick

Im Falle der Ausführung eines Geschäftsprozesses, geschieht die Auswahl eines geeigneten Web Services lokal auf Prozessschrittebene, d.h. im Moment der Ausführung eines Prozessschrittes wird der momentan geeignetste unter allen verfügbaren Web Services zu dessen Ausführung gewählt. Da bei der Auswahl der zu verwendenden Web Services ausschließlich der aktuelle Prozessschritt, unabhängig von bereits ausgeführten und zukünftig auszuführenden Prozessschritten betrachtet und optimiert wird, findet, bezogen auf den gesamten Geschäftsprozess, lediglich eine lokale Optimierung statt.

Diese lokale Optimierung auf der Ebene einzelner Prozessschritte weist jedoch zwei entscheide Nachteile auf. Zum einen führt die Optimierung einzelner Prozessschritte nicht zwingendermaßen zu einer optimalen Ausführung des Prozesses in seiner Gänze. Zum anderen ermöglicht es eine lokale Optimierung nicht, globale Dienstgüteeigenschaften für den Geschäftsprozess in seiner Gänze sicherzustellen, wie z.B. die Einhaltung einer bestimmten maximalen Ausführungsdauer für den gesamten Geschäftsprozess. Globale Eigenschaften können nur sichergestellt werden, wenn bereits zu Beginn der Ausführung des Geschäftsprozesses Web Services für jeden Prozessschritt so festgelegt werden, dass die sich aus der Gesamtauswahl ergebenden Dienstgüteeigenschaften des Geschäftsprozesses die an sie gestellten Anforderungen erfüllen.

Im Rahmen unserer zukünftigen Forschungsaktivitäten werden wir daher vermehrt Algorithmen zur globalen, dienstgütebasierten Optimierung von Geschäftsprozessen auf Web-Service-Basis untersuchen.

Literaturverzeichnis

[1] *CCITT Blue Book, VIII.2, Data Communication Networks: Services and Facilities, Interfaces*, Band 2, Genf, ITU, 1989.

[2] *Data Encryption Standard (DES)*, Federal Information Processing Standard 46-3, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf, U.S. Department of Commerce, National Institute of Standards and Technology, 25.10.1999, Abruf am 16.05.2005.

[3] Aier S., Schönherr M.: *Enterprise Application Integration – Flexibilisierung komplexer Unternehmensarchitekturen*, Band 1, Berlin, GITO Verlag, 2003.

[4] Aier S., Schönherr M.: *Enterprise Application Integration – Serviceorientierung und nachhaltige Architekturen*, Band 2, Berlin, GITO Verlag, 2004.

[5] Alonso G., Casati F., Kuno H., Machiraju V.: *Web Services. Concepts, Architectures and Applications.*, Berlin, Heidelberg, Springer Verlag, 2004.

[6] Andrews T., Curbera F., Dholakia H., Goland Y., Klein J., Leymann F., Liu K., Roller D., Smith D., Thatte S., Trickovic I., Weerawarana S.: *Business Process Execution Language for Web Services Version 1.1*, ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf, BEA Systems, IBM, Microsoft, SAP AG, Siebel Systems, 05.05.2003, Abruf am 12.05.2005.

[7] Austin D., Barbir A., Ferris C., Garg S.: *Web Services Architecture Requirements*, W3C Working Group Note, http://www.w3.org/TR/wsa-reqs/, W3C, 11.02.2004, Abruf am 07.04.2005.

[8] Becker J., Kugler M., Rosemann M.: *Process Management: A Guide for the Design of Business Processes*, Berlin, New York, Springer, 2003.

[9] Beimborn D., Franke J., Weitzel T.: *Drivers and Inhibitors for Outsourcing Financial Processes – A Comparative Survey of Economies of Scale, Scope, and Skill*, in: Proceedings of the 11th Americas Conference on Information Systems (AMCIS 2005), Omaha, NE, USA, 2005.

[10] Bellwood T., Capell S., Clement L., Colgrave J., Dovey M. J., Feygin D., Hately A., Kochman R., Macias P., Novotny M., Paolucci M., Riegen C. v., Rogers T., Sycara K., Wenzel P., Wu Z.: *UDDI Version 3.0.2*, UDDI Spec Technical Committee Draft, http://uddi.org/pubs/uddi_v3.htm, OASIS, 19.10.2004, Abruf am 13.05.2005.

[11] Berbner R., Heckmann O., Mauthe A., Steinmetz R.: *Eine Dienstgüte unterstützende Web Service-Architektur für flexible Geschäftsprozesse*, in: Wirtschaftsinformatik, 47 (2005) 4, Wiesbaden, Vieweg Verlag.

[12] Berbner R., Mauthe A., Steinmetz R.: *Unterstützung dynamischer E-Finance-Geschäftsprozesse*, in: Konferenzband der Konferenz Elektronische Geschäftsprozesse 2004 (EGP 2004), Klagenfurt, Österreich, 2004, S. 44-54.

[13] Bergsten H.: *JavaServer Pages*, 2. Aufl., Sebastopol, O'Reilly & Associates, Inc., 2002.

[14] Berners-Lee T., Fielding R., Frystyk H.: *Hypertext Transfer Protocol -- HTTP/1.0*, Request for Comments 1945, http://www.ietf.org/rfc/rfc1945.txt, The Internet Society, Mai 1996, Abruf am 13.05.2005.

[15] Berners-Lee T., Fielding R., Masinter L.: *Uniform Resource Identifier (URI): Generic Syntax*, Request for Comments 3986, http://www.ietf.org/rfc/rfc3986.txt, The Internet Society, Jan. 2005, Abruf am 28.04.2005.

[16] Berners-Lee T., Masinter L., McCahill M.: *Uniform Resource Locators (URL)*, Request for Comments 1738, http://www.ietf.org/rfc/rfc1738.txt, The Internet Society, Dez. 1994, Abruf am 12.05.2005.
[17] Biron P. V., Malhotra A.: *XML Schema Part 2: Datatypes Second Edition*, W3C Recommendation, http://www.w3.org/TR/xmlschema-2/, W3C, 28.10.2004, Abruf am 13.05.2005.
[18] Blake S., Black D., Carlson M., Davies E., Wang Z., Weiss W.: *An Architecture for Differentiated Services*, Request for Comments 2475, http://www.ietf.org/rfc/rfc2475.txt, The Internet Society, Dez. 1998, Abruf am 16.05.2005.
[19] Bos B., Çelik T., Hickson I., Lie H. W.: *Cascading Style Sheets, level 2 revision 1 - CSS 2.1 Specification*, W3C Candidate Recommendation, http://www.w3.org/TR/CSS21/, W3C, 25.02.2004, Abruf am 17.05.2005.
[20] Braden R., Clark D., Shenker S.: *Integrated Services in the Internet Architecture: an Overview*, Request for Comments 1633, http://www.ietf.org/rfc/rfc1633.txt, The Internet Society, Juni 1994, Abruf am 14.05.2005.
[21] Braden R., Zhang L., Berson S., Herzog S., Jamin S.: *Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification*, Request for Comment 2205, http://www.ietf.org/rfc/rfc2205.txt, The Internet Society, Sep. 1997, Abruf am 14.05.2005.
[22] Bray T., Paoli J., Sperberg-McQueen C. M., Maler E., Yergeau F., Cowan J.: *Extensible Markup Language (XML) 1.1*, W3C Recommendation, http://www.w3.org/TR/xml11/, W3C, 15.04.2004, Abruf am 13.05.2005.
[23] Cardoza P., Hennig T., Seach G., Stein A.: *Access 2003 VBA Programmer's Reference*, Hoboken, Wrox Press, 2004.
[24] Chappell D.: *Understanding.NET: A Tutorial and Analysis*, Boston, London, Addison-Wesley Professional, 2002.
[25] Chopra V., Bakore A., Eaves J., Galbraith B., Li S., Wiggers C.: *Professional Apache Tomcat 5*, Hoboken, Wrox Press, 2004.
[26] Christensen E., Curbera F., Meredith G., Weerawarana S.: *Web Services Description Language (WSDL) 1.1*, W3C Note, http://www.w3.org/TR/wsdl, W3C, 15.03.2001, Abruf am 13.05.2005.
[27] Dan A., Ludwig H., Pacifici G.: *Web service differentiation with service level agreements*, http://www-106.ibm.com/developerworks/webservices/library/ws-slafram/, IBM, Mai 2003, Abruf am 14.05.2005.
[28] DuBois P.: *MySQL*, 2. Aufl., Indianapolis, Sams Publishing, 2003.
[29] Eckert C.: *IT-Sicherheit*, 3. Aufl., Wien, R. Oldenbourg Verlag, 2004.
[30] Fallside D. C., Walmsley P.: *XML Schema Part 0: Primer Second Edition*, W3C Recommendation, http://www.w3.org/TR/xmlschema-0/, W3C, 28.10.2004, Abruf am 13.05.2005.
[31] Fielding R.: *Relative Uniform Resource Locators*, Request for Comments 1808, http://www.ietf.org/rfc/rfc1808.txt, The Internet Society, Juni 1995, Abruf am 12.05.2005.
[32] Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., Berners-Lee T.: *Hypertext Transfer Protocol -- HTTP/1.1*, Request for Comments 2616, http://www.ietf.org/rfc/rfc2616.txt, The Internet Society, Juni 1999, Abruf am 28.05.2005.
[33] Freed N., Borenstein N.: *Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples*, Request for Comments 2049,

http://www.ietf.org/rfc/rfc2049.txt, The Internet Society, Nov. 1996, Abruf am 13.05.2005.

[34] Freed N., Borenstein N.: *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, Request for Comments 2045, http://www.ietf.org/rfc/rfc2045.txt, The Internet Society, Nov. 1996, Abruf am 13.05.2005.

[35] Freed N., Borenstein N.: *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, Request for Comments 2046, http://www.ietf.org/rfc/rfc2046.txt, The Internet Society, Nov. 1996, Abruf am 13.05.2005.

[36] Freed N., Klensin J., Postel J.: *Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*, Request for Comments 2048, http://www.ietf.org/rfc/rfc2048.txt, The Internet Society, Nov. 1996, Abruf am 13.05.2005.

[37] Fröschl F.: *Vom IuK-Outsourcing zum Business Process Outsourcing*, in: Wirtschaftsinformatik, 41 (1999) 5, Wiesbaden, Vieweg Verlag, S. 458-460.

[38] Gouscos D., Kalikakis M., Georgiadis P.: *An Approach to Modeling Web Service QoS and Provision Price*, in: Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops (WISEW'03), Rom, Italien, 2003, S. 121-130.

[39] Gray J., Reuter A.: *Transaction Processing: Concepts and Techniques*, San Fransisco, Morgan Kaufmann Publishers, 1993.

[40] Gudgin M., Hadley M., Mendelsohn N., Moreau J.-J., Nielsen H. F.: *SOAP Version 1.2 Part 1: Messaging Framework*, W3C Recommendation, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/, W3C, 24.06.2003, Abruf am 13.05.2005.

[41] Gudgin M., Hadley M., Mendelsohn N., Moreau J.-J., Nielsen H. F.: *SOAP Version 1.2 Part 2: Adjuncts*, W3C Recommendation, http://www.w3.org/TR/2003/REC-soap12-part2-20030624/, W3C, 24.06.2003, Abruf am 13.05.2005.

[42] Gupta S.: *Pro Apache Log4j*, 2. Aufl., Berkeley, Apress, 2005.

[43] Haas H., Hurley O., Karmarkar A., Mischkinsky J., Jones M., Thompson L., Martin R.: *SOAP Version 1.2 Specification Assertions and Test Collection*, W3C Recommendation, http://www.w3.org/TR/2003/REC-soap12-testcollection-20030624/, W3C, 24.06.2003, Abruf am 13.05.2005.

[44] Hammer M., Champy J.: *Rengineering the Corporation: A Manifesto for Business Revolution*, New York, HarperBusiness, 2003.

[45] Heckmann O.: *A System-oriented Approach to Efficiency and Quality of Service for Internet Service Providers*, Dissertationsschrift, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, 2004.

[46] Holstege M., Vedamuthu A. S.: *XML Schema: Component Designators*, W3C Working Draft, http://www.w3.org/TR/xmlschema-ref/, W3C, 29.03.2005, Abruf am 13.05.2005.

[47] Jin L.-j., Machiraju V., Sahai A.: *Analysis on Service Level Agreement of Web Services*, http://www.hpl.hp.com/techreports/2002/HPL-2002-180.pdf, HP Laboratories Palo Alto, 21.06.2002, Abruf am 14.05.2003.

[48] Johnson R.: *Expert One-on-One J2EE Design and Development*, Indianapolis, Wrox Press Ltd., 2002.

[49] Juric M. B., Mathew B., Sarang P.: *Business Process Execution Language for Web Services: BPEL and BPEL4WS*, Birmingham, Packt Publishing Ltd., 2004.

[50] Kalepu S., Krishnaswamy S., Loke S. W.: *Verity: A QoS Metric for Selecting Web Services and Providers*, in: Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops (WISEW'03), Rom, Italien, 2003, S. 131-139.
[51] Keller W.: *Enterprise Application Integration. Erfahrungen aus der Praxis.*, Heidelberg, dpunkt.verlag GmbH, 2002.
[52] Klensin J.: *Simple Mail Transfer Protocol*, Request for Comments 2821, http://www.ietf.org/rfc/rfc2821.txt, The Internet Society, April 2001, Abruf am 13.05.2005.
[53] Lee K., Jeon J., Lee W., Jeong S.-H., Park S.-W.: *QoS for Web Services: Requirements and Possible Approaches*, W3C Working Group Note, http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/, W3C, 25.11.2003, Abruf am 15.04.2005.
[54] Liberty J.: *Programming C#*, 4. Aufl., Sebastopol, O'Reilly, 2005.
[55] Lie H. W., Bos B.: *Cascading Style Sheets, level 1*, W3C Recommendation, http://www.w3.org/TR/CSS1, W3C, 11.01.1999, Abruf am 17.05.2005.
[56] Lindert F., Wiedeler M.: *Organisationsübergreifendes Geschäftsprozessmanagement*, in: IT Information Technology, 46 (2004) 4, München, Oldenbourg Wissenschaftsverlag, S. 175-183.
[57] Ludwig H., Keller A., Dan A., King R. P., Franck R.: *Web Service Level Agreement (WSLA) Language Specification*, http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf, IBM Corporation, 28.01.2003, Abruf am 14.05.2005.
[58] Machiraju V., Sahai A., Moorsel A. v.: *Web Services Management Network: An Overlay Network for Federated Service Management*, http://www.hpl.hp.com/techreports/2002/HPL-2002-234.pdf, HP Laboratories Palo Alto, 21.08.2002, Abruf am 14.05.2005.
[59] Mani A., Nagarajan A.: *Understanding quality of service for Web services*, http://www-106.ibm.com/developerworks/webservices/library/ws-quality.html, IBM developerWorks, 01.01.2002, Abruf am 15.04.2005.
[60] Mitra N.: *SOAP Version 1.2 Part 0: Primer*, W3C Recommendation, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/, W3C, 24.06.2003, Abruf am 13.05.2005.
[61] Moore K.: *Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text*, Request for Comments 2047, http://www.ietf.org/rfc/rfc2047.txt, The Internet Society, Nov. 1996, Abruf am 13.05.2005.
[62] Nadalin A., Kaler C., Hallam-Baker P., Monzillo R.: *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)*, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf, OASIS, März 2004, Abruf am 27.04.2005.
[63] Pemberton S., u.a.: *XHTML 1.0 The Extensible HyperText Markup Language*, W3C Recommendation, http://www.w3.org/TR/xhtml1/, W3C, 01.08.2002, Abruf am 17.05.2005.
[64] Postel J.: *Transmission Control Protocol*, Request for Comments 793, http://www.ietf.org/rfc/rfc793.txt, The Internet Society, Sep.1981, Abruf am 13.05.2005.
[65] Postel J., Reynolds J.: *File Transfer Protocol (FTP)*, Request for Comments 959, http://www.ietf.org/rfc/rfc959.txt, The Internet Society, Okt. 1985, Abruf am 13.05.2005.

[66] Ran S.: *A model for web services discovery with QoS*, in: ACM SIGecom Exchanges, 4 (2003) 1, New York, ACM Press, S. 1-10.
[67] Riedl R.: *Begriffliche Grundlagen des Business Process Outsourcing*, in: Information Management & Consulting, 18 (2003) 3, Saarbrücken, mc information multimedia communication AG, S. 6-10.
[68] Rosen E., Viswanathan A., Callon R.: *Multiprotocol Label Switching Architecture*, Request for Comments 3031, http://www.ietf.org/rfc/rfc3031.txt, The Internet Society, Jan. 2001, Abruf am 16.05.2005.
[69] Sahai A., Durante A., Machiraju V.: *Towards Automated SLA Management for Web Services*, http://www.hpl.hp.com/techreports/2001/HPL-2001-310R1.pdf, HP Laboratories Palo Alto, 11.07.2002, Abruf am 14.05.2003.
[70] Schneier B.: *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2. Aufl., New York, John Wiley & Sons Inc., 1996.
[71] Shenker S., Partridge C., Guerin R.: *Specification of Guaranteed Quality of Service*, Request for Comments 2212, http://www.ietf.org/rfc/rfc2212.txt, The Internet Society, Sep. 1997, Abruf am 14.05.2005.
[72] Singh I., Brydon S., Murray G., Ramachandran V., Violleau T., Stearns B.: *Designing Web Services with the J2EE 1.4 Platform JAX-RPC, SOAP, and XML Technologies*, 1. Aufl., Boston, Addison-Wesley Professional, 2004.
[73] Sokolovsky Z., Löschenkohl S.: *Handbuch Industrialisierung der Finanzwirtschaft*, Wiesbaden, Gabler Verlag, 2005.
[74] Steinmetz R.: *Multimedia-Technolohgie: Grundlagen, Komponenten und Systeme*, 3. Aufl., Berlin, Heidelberg, New York, Barcelona, Hongkong, London, Mailand, Paris, Singapur, Tokio, Springer, 2000.
[75] Steinmetz R., Berbner R., Martinovic I.: *Web Services zur Unterstützung flexibler Geschäftsprozesse in der Finanzwirtschaft*, in: Handbuch Industrialisierung der Finanzwirtschaft, Wiesbaden, Gabler Verlag, 2004.
[76] Sturm R., Morris W.: *Foundations of Service Level Management*, Indianapolis, Sams Publishing, 2000.
[77] Tanenbaum A. S.: *Computernetzwerke*, 3 Aufl., München, Prentice Hall, 1998.
[78] Thompson H. S., Beech D., Maloney M., Mendelsohn N.: *XML Schema Part 1: Structures Second Edition*, W3C Recommendation, http://www.w3.org/TR/xmlschema-1/, W3C, 28.10.2004, Abruf am 13.05.2005.
[79] Tian M., Gramm A., Ritter H., Schiller J., Winter R.: *A Survey of current Approaches towards Specification and Management of Quality of Service for Web Services*, in: PIK - Praxis der Informationsverarbeitung und Kommunikation, 27 (2004) 3, München, K.G. Saur Verlag GmbH, S. 132-139.
[80] Tosic V.: *Service Offerings for XML Web Services and their Management Applications*, Dissertationsschrift, Carleton University, Department of Systems and Computer Engineering, Faculty of Engineering, Ottawa, Ontario, Canada, 2004.
[81] Ullenboom C.: *Java ist auch eine Insel*, 4. Aufl., Bonn, Galileo Press GmbH, 2004.
[82] Viescas J. L.: *Microsoft Office Access 2003 Inside Out*, Redmond, Microsoft Press, 2004.
[83] Wang D., Bayer T., Frotscher T., Teufel M.: *Java Web Services mit Apache Axis*, Frankfurt, Software & Support Verlag GmbH, 2004.
[84] Weerawarana S., Curbera F., Leymann F., Storey T., Ferguson D. F.: *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing,*

WS-BPEL, WS-Reliable Messaging, and More, Upper Saddle River, Prentice Hall PTR, 2005.

[85] Weitzel T., König W., Busse R.: *Beiträge zur Industrialisierung von Finanzprozessen durch Web Services – Externes Straight Through Processing für Master-KAGen*, in: Information Management & Consulting, 19 (2004) Sonderausgabe zum 50. Geburtstag von Helmut Krcmar, Saarbrücken, mc information multimedia communication AG, S. 64-72.

[86] Weitzel T., Martin S. V., König W.: *Straight Through Processing auf XML-Basis im Wertpapiergeschäft*, in: Wirtschaftsinformatik, 45 (2003) 4, Wiesbaden, Vieweg Verlag, S. 409-420.

[87] Wroclawski J.: *Specification of the Controlled-Load Network Element Service*, Request for Comments 2211, http://www.ietf.org/rfc/rfc2211.txt, The Internet Society, Sep. 1997, Abruf am 14.05.2005.

[88] Wroclawski J.: *The Use of RSVP with IETF Integrated Services*, Request for Comments 2210, http://www.ietf.org/rfc/rfc2210.txt, The Internet Society, Sep.1997, Abruf am 14.05.2005.

[89] Zeng L., Benatallah B., Dumas M., Kalagnanam J., Sheng Q. Z.: *Quality Driven Web Services Composition*, in: Proceedings of 12th International Conference on World Wide Web (WWW 2003), Budapest, Ungarn, 2003, S. 411-421.