F A C H H O C H S C H U L E W E D E L

S E M I N A R A R B E I T

in der Fachrichtung

Wirtschaftsinformatik

Thema:

Serviceorientierte Softwarearchitektur

Überblick über Web-Service Technologien

Eingereicht von:	Sebastian Reese (Mat.-Nr. 5585) Tinsdaler Weg 125 22880 Wedel Tel. (0 4103) 803 855 1 Mobil: (0179) 231 947 60 Email: wi5585@fh-wedel.de
Erarbeitet im:	7. Semester
Abgegeben am:	23. Oktober 2005
Referent (FH Wedel):	Prof. Dr. Sebastian Iwanowski Fachhochschule Wedel Feldstraße 143 22880 Wedel Tel. (04103) 8048- 63 Email: iw@fh-wedel.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

WS	Web-Service
RPC	Remote Procedure Call
o.V.	ohne Verfasser
http	Hypertext Transfer Protokoll
i.w.S	im weiteren Sinne
XML	Extensible Markup Language
URI	Uniform Resource Identifier

1. Grundlagen

1.1. Entfernte Aufrufe

Eine wichtige Grundlage für die Maschine-Maschine Kommunikation und damit auch für das Thema Web-Service Technologien bilden die so genannten entfernten Aufrufe oder auch „Remote Procedure Calls“ (RPCs). Hierbei handelt es sich um Methodenaufrufe, die ein Client auf einem Server über ein Transportnetz hinweg ausführen lässt.[1]

Hierfür muss der Client genaue Informationen über den Prozeduraufruf beim Server bereitgestellt bekommen haben. Somit ist dem aufrufenden Rechner bekannt, wie der Prozeduraufruf zu kodieren ist, in welcher Reihenfolge welche Parameter, wie verschlüsselt werden müssen, damit ein erfolgreicher Aufruf möglich ist.

Bei dem Prozeduraufruf wird von dem Client eine Nachricht erstellt, in der auf eine vom Server vorgegebene Art und Weise der Prozeduraufruf, sowie evtl. Parameter, kodiert sind. Diese Nachricht wird dann über ein Transportnetz (z.B. über das Internet per HTTP) an den Server geschickt. Dort wird die Nachricht dekodiert und der Methodenaufruf wird gemäß dieser Nachricht ausgeführt (siehe Abbildung 1: RPC)

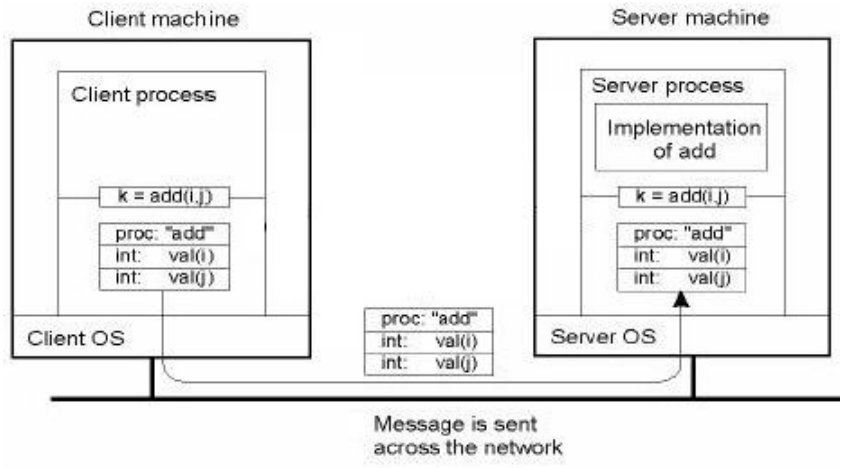

Abbildung 1: RPC[2]

[1] Vgl. o.V. Wikipedia, RPC

[2] Abbildung angelehnt an Iwanowski, Sebastian Vorlesung 5 Folie 5

Sowohl der Client, als auch der Server sind auf eine genaue Einhaltung der Prozedurdefinition angewiesen, so dass hier ein großer Wartungsaufwand entsteht. Die Technologie der Web-Services setzt genau hier an. Web-Services bieten eine definierte Schnittstelle, um Entfernte Aufrufe Möglich zu machen und minimieren den Wartungsaufwand.

1.2. Der Servicebegriff

Service

Im Rahmen der Web-Service Technologie wird ein Service als eine eigenständige Softwarekomponente bezeichnet. Ein Service repräsentiert eine eigene, klar abgegrenzte Geschäftsfunktion innerhalb einer Applikation. Verschiedene Services können dabei miteinander interagieren und sich gegenseitig verwenden. Dabei ist allerdings zu beachten, dass die Services nicht zu stark voneinander abhängig sind, sie müssen lose gekoppelt sein.[3]

XML Web-Service

Ein XML Web-Service (im Folgenden nur noch Web-Service genannt) ist ein Service im obigen Sinne und stellt den allgemeinen Standard von Services dar. Damit ein Service als Web-Service bezeichnet werden kann, muss er zwei Kriterien erfüllen.

- Die Kommunikation muss mit einem Internetprotokoll stattfinden (z.B. HTTP)
- Das ausgetauschte Nachrichtenformat ist XML

Hiermit ist eine Grundlage dafür gelegt, dass verschiedene Web-Services einfach miteinander interagieren können. Um im Rahmen der Serviceorientierten Softwarearchitektur von einem Web-Service sprechen zu können muss allerdings weiter gegangen werden. Für einen Web-Service werden hierbei weitere ergänzende De-facto-Standards[4] verwendet.

Es wird von Web-Services gefordert, dass eine Servicebeschreibungsdatei angeboten wird, die den Zweck und die Art der Verwendung eines Service formuliert. Diese Servicebeschreibungsdatei muss in Form eines WSDL

[3] Vgl. Erl, Thomas Seite 48f

[4] De-facto-Standard: keine gesetzliche Norm, „hat sich in der Praxis durchgesetzt“

Dokumentes vorliegen (WSDL siehe Kapitel 2.1) und mit einem allgemeinen Suchdienst aufgefunden werden können (siehe Kapitel 2.3). Weiter wird gefordert, dass das Nachrichtenformat, in dem die Nachrichten übermittelt werden über beliebiges XML hinausgeht und dem SOAP Standard entspricht und immer über HTTP realisiert wird. (SOAP siehe Kapitel 2.2).[5]

1.3. Rollen von Web-Services

Web-Services können in verschiedenen Rollen auftreten. Ein Web-Service kann dabei als nachfragender und ein zweiter als antwortender, oder ausführender Service fungieren. Hierbei sind zwar für eine konkrete Anfrage die Rollen klar verteilt, allerdings ist es einem Service Grundsätzlich möglich sowohl als nachfragender, wie auch antwortender Service zu handeln (siehe Abbildung 2: Rollen von Web-Service).[6]

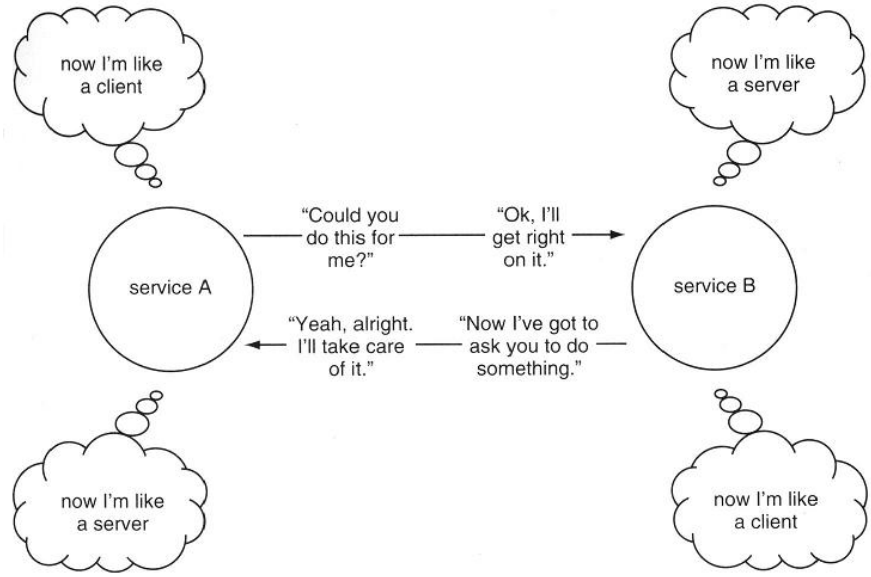

Abbildung 2: Rollen von Web-Service[7]

Der Standard der Web-Services unterscheidet hierbei 5 verschiedene Rollen, die ein Web-Service einnehmen kann (siehe Abbildung 3: WS Rollenverteilung).

[5] Vgl. Erl, Thomas Seite 49f
[6] Vgl. Erl, Thomas Seite 50
[7] Vgl. Erl, Thomas Seite 51

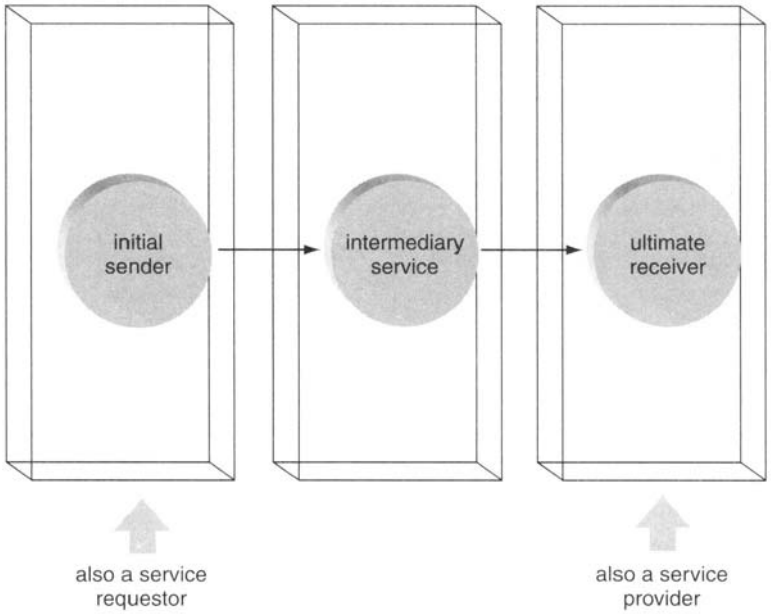

Abbildung 3: WS Rollenverteilung[8]

Service Provider[9]

Einen Web-Service, der anderen Web-Services einen bestimmten Dienst zur Verfügung stellt, bezeichnet man als *Service Provider*

Service Requestor[10]

Einen Web-Service, der Dienste von anderen Web-Services benutzt, bezeichnet man als *Service Requestor*

Intermediary[11]

Wenn ein Web-Service eine Nachricht erhält, die nur weitergeleitet und nicht von dem Web-Service selbst verarbeitet werden muss, so bezeichnet man diesen Web-Service als *Intermediary.* Wenn ein Intermediary eine Nachricht empfängt, agiert er als Service Provider, wenn er die Nachricht weiterleitet als Service Requestor.

[8] Abbildung Angelehnt an Erl, Thomas, Seite 59
[9] Vgl. Erl, Thomas Seite 55
[10] Vgl. Erl, Thomas Seite 56
[11] Vgl. Erl, Thomas Seite 56

Initial Sender[12]

Da durch die Verwendung von Intermediaries nicht immer eindeutig ist, welcher Web-Service nun tatsächlich eine Dienstanfrage gestellt hat (Service Provider und Requestor reichen hierfür nicht aus), wird der Web-Service der die erste Dienstanfrage stellt als *Initial Sender* bezeichnet und somit Eindeutig zuordenbar.

Ultimate Receiver[13]

Das Gegenstück zum Initial Sender wird *Ultimate Receiver* genannt und kennzeichnet den Web-Service, der tatsächlich die Dienstanfrage ausführt.

1.4. Serviceorientierte Softwarearchitektur

Unter serviceorientierter Softwarearchitektur versteht man eine Anwendung, die von Grund auf für Web-Services entworfen, bzw. entwickelt wurde. Eine Anwendung hingegen, die mit anderen Applikationen interagiert und dabei die Technologie der Web-Services verwendet, basiert nicht auf den Grundsätzen der serviceorientierten Softwarearchitektur. Für die serviceorientierte Softwarearchitektur ist es notwendig, dass sowohl die Prinzipien der serviceorientierten Softwarearchitektur eingehalten werden, als auch die Einführung eines Service Integration Layer.[14]

Forderungen an Serviceorientierung[15]

- *Wiederverwendbarkeit/ Modularisierung* – Wenn es Geschäftsfunktionalität gibt, die potentiell wieder verwendet werden kann, so ist diese als Service zu implementieren.

- *Formale Übereinstimmung* – Damit Web-Services miteinander kommunizieren können, müssen sie sich an ein Mindestmass an formaler Übereinstimmung (Standards) halten.

[12] Vgl. Erl, Thomas Seite 57
[13] Vgl. Erl, Thomas Seite 57
[14] Vgl. Erl, Thomas Seite 50ff
[15] Vgl. Thomas Erl, Seite 53f

- *Lose Kopplung* – Web-Services sollen in keiner engen Beziehung zueinander stehen. Im Rahmen der Autonomie und Abstraktion sollen sie lose gekoppelt sein und bleiben.
- *Abstraktion* – Nur die Service Beschreibung ist bekannt. Über die Art der Implementierung wird keine Auskunft gegeben.
- *Services sind kombinierbar* – Web-Services können aus anderen Web-Services zusammengesetzt werden und sind demnach miteinander kombinierbar
- *Autonomie* – Web-Services werden durch klar definierten Grenzen getrennt. Innerhalb dieser „Dienstgrenzen“ ist der Service autonom
- *Statuslosigkeit* – Web-Services speichern keine Informationen über Status. Evtl. notwendiges Kontextmanagement muss anderweitig implementiert werden.
- *Auffindbar über Suchdienst* – Damit ein Web-Service genutzt werden kann sollte er von Entwicklern im Rahmen eines Allgemeinen Such- oder Verzeichnisdienstes aufgefunden werden können.

Service Integration Layer

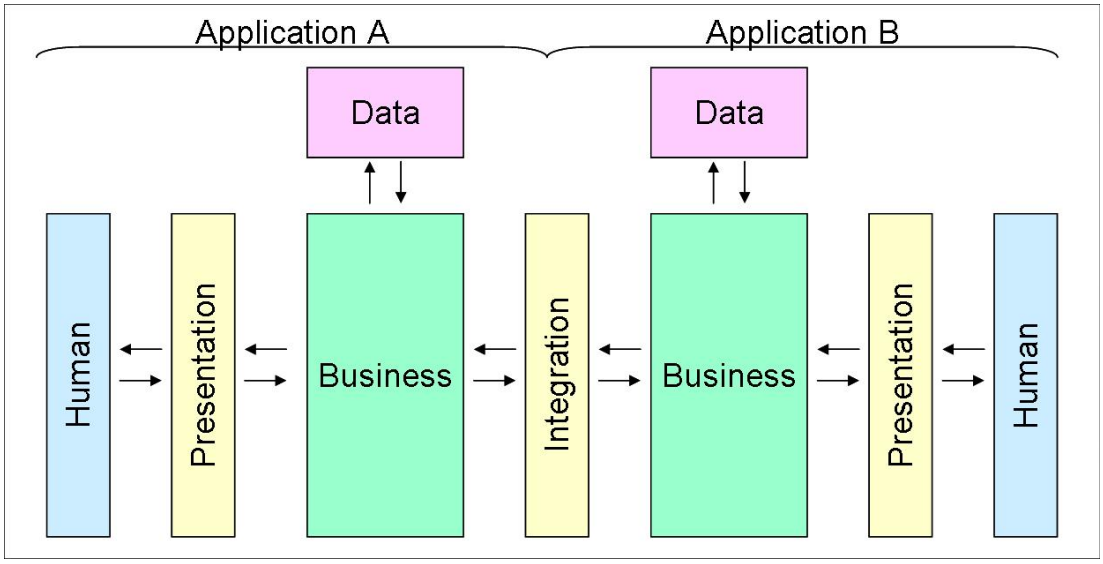

Abbildung 4: Service Integration Layer[16]

[16] Abbildung angelehnt an Erl, Thomas Seite 53

Der Service Integration Layer wird dort notwendig, wo Maschine – zu - Maschine Kommunikation betrieben wird. Also überall dort, wo automatische Prozesse ablaufen, auf die der Mensch keinen unmittelbaren Einfluss hat.

Bei der Mensch-Maschine Interaktion wird eine so genannte Präsentationsschicht eingesetzt, in der die Businessdaten in eine für den Menschen lesbare Form gebracht werden. Bei der Maschine-Maschine Kommunikation geschieht dies nun über den *Service Integration Layer* (siehe Abbildung 4: Service Integration Layer). Hier werden die Businessdaten von Applikation A in eine Form gebracht, die Applikation B versteht. Hierfür ist es unabdingbar, dass die Applikationen auf den Standards der Web-Services arbeiten, da hierdurch eine Störungsfreie Kommunikation gewährleistet wird. Anderenfalls müssten sich die Entwickler der Applikationen sich erst mit großem Aufwand auf einen Kommunikationsstandard einigen und diesen entwickeln.[17]

[17] Vgl. Erl, Thomas Seite 52

2. Web-Services der ersten Generation

Die Web-Services der ersten Generation werden durch vom W3 Konsortium entwickelten Standards gebildet. Die drei Standards für die Servicebeschreibung (siehe Kapitel 2.1), das Nachrichtenformat (siehe Kapitel 2.2) und den Suchdienst (siehe Kapitel 2.3) ergeben zusammen mit den Entwicklungsprinzipien eine grundlegende, auf XML basierende serviceorientierte Softwarearchitektur.[18] Abbildung 5 zeigt die Beziehungen zwischen den Standards auf.

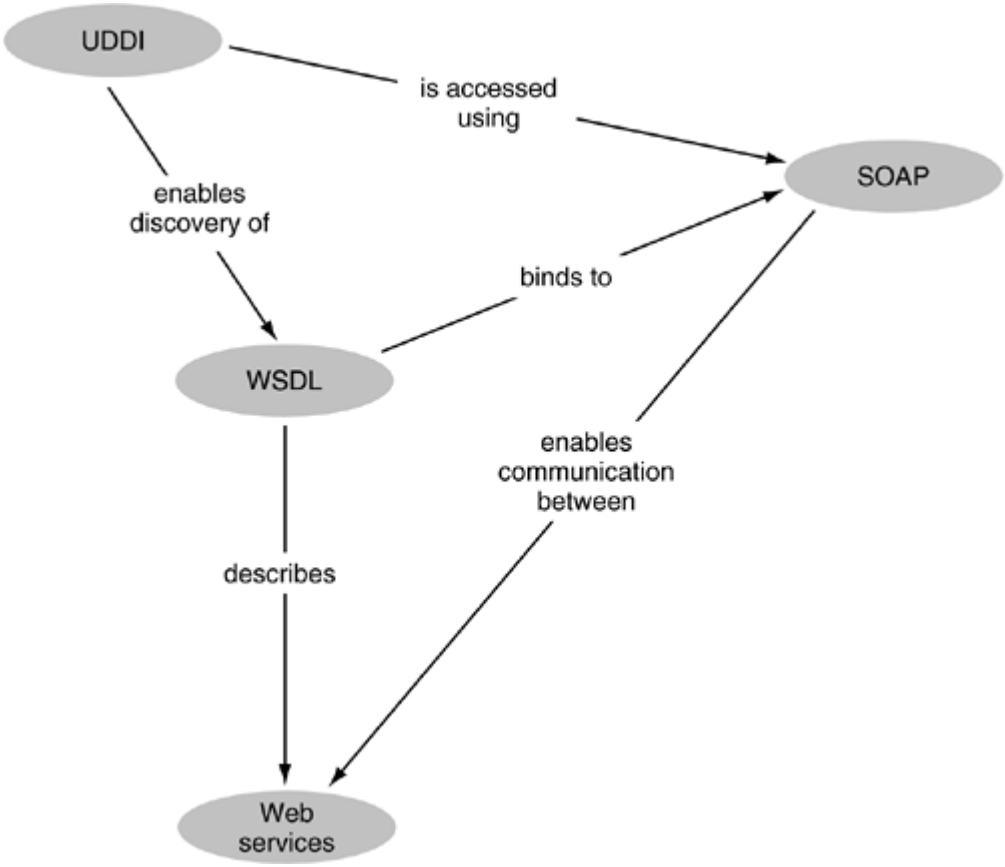

Abbildung 5: Zusammenspiel der WS der ersten Generation[19]

2.1. WSDL

WSDL ist die Abkürzung für *Web-Service Description Language* und repräsentiert die für Web-Service geforderte Servicebeschreibungsdatei (siehe Kapitel 2.1).

[18] Vgl. Thomas Erl, Seite 66

[19] Abbidlung aus Erl, Thomas Seite 49

Eine WSDL Datei dient dazu potentiellen Benutzern von Web-Services die Rahmenbedingungen für die Verwendung des Service bereitzustellen. Die WSDL Datei Gliedert sich dabei in einen abstrakten Definitionsteil und einen konkreten Implementationsteil auf. (siehe Abbildung 6: WSDL Grundstruktur).[20]

Abstract[21]

Wie in Abbildung 6 zu erkennen ist, besteht der abstrakte Teil aus den XML Elementen „<Types />“, „<Messages />“ und „<Interface />“.

Das *Types* Element ist hierbei für die Deklaration von Typen vorgesehen, die für einen Service evtl. notwendig sind. Mit dem *Messages* Element werden die für Operationen benötigten Parameter deklariert. Wobei eine *Message* einem Parameter entspricht. Im *Interface* Element werden die konkreten Operationen, die ein Service bereitstellt definiert.

Concrete[22]

Der Concrete Teil eines WSDL Dokumentes gibt die Implementierungsinformationen bekannt. Er besteht aus den beiden Elementen „<Binding />“ und „<Service />“.

Innerhalb des Binding Elementes werden den im Interface definierten Operationen Konkrete Kodierungs- und Protokollinformationen zugeordnet. Im Service Element wird dann letztendlich die konkrete Adresse in Form eines URI bekannt gegeben, unter der der Service erreichbar ist.

[20] Vgl. Erl, Thomas Seite 67
[21] Vgl. Erl, Thomas Seite 68
[22] Vgl. Erl, Thomas Seite 70

```
<definitions>
    <!-- abstract part -->
    <types>
        <!-- type definition -->
    </types>
    <messages>
        <!-- declaration of message (parameters) -->
    </messages>
    <interface>
        <!-- declare operations and associate with messages (parameters) -->
    </interface>

    <!-- Concrete part -->
    <binding>
        <!-- associate operations with message format and protocol -->
    </binding>
    <service>
        <!-- definition of endpoints with physically addresses of SOAP nodes -->
    </service>
</definitions>
```

Abbildung 6: WSDL Grundstruktur

Beispiel: GoogleSearch, WSDL Dokument

Der Suchdienst Google bietet Entwicklern die Möglichkeit an Suchanfragen per Web-Service zu stellen. Hierfür wurde die in Anhang I gezeigte Servicebeschreibungsdatei erstellt und veröffentlicht. Anhand dieser Beispieldatei soll die Funktionsweise von WSDL Dokumenten beschrieben werden.

Innerhalb des Types Elementes wird hier ein aus verschiedenen Elementen zusammengesetzter Typ mit der Bezeichnung *GoogleSearchResult* deklariert. Eine Antwort auf eine Suchanfrage, die mit einem Web-Service ausgeführt wurde liefert ein Element dieses Typs zurück.

Danach werden in dem WSDL Dokument zwei Messages (Parameter) definiert.

- doGoogleSearch – diese Message wird für die Anfrage einer Suche verwendet. Hier ist zu sehen, dass die erwarteten Elemente, aus denen die Message besteht auch innerhalb des Message Elementes deklariert werden können.

- *doGoogleSearchResponse* – die doGoogleSearchResponse Message stellt den Rückgabewert für eine Suchanfrage dar. Die als Rückgabe erwarteten Werte werden hierbei aus dem Type Element übernommen, bzw. mit *type="typens:GoogleSearchResult"* referenziert.

Im Interface Element des Google Dokumentes wird die Operation für die Suchanfrage definiert. Der Name dieser Operation ist *doGoogleSearch* und der erwartete Eingabeparameter ist die Message *doGoogleSearch*. Als Rückgabewert für die Operation wird hier die Message *doGoogleSearchResponse* festgelegt.

Der Bindingteil des WSDL Dokumentes von Google beschreibt für die Operation der Suchanfrage, wie diese Kodiert ist und über welches Protokoll die Nachricht verschickt werden soll. In diesem Fall wird für die Operation doGoogleSearch mit dem SOAP Protokoll kodiert und per HTTP verschickt.

Den letzten Teil des Dokumentes bildet das Service Element. Hier wird der eigentlich Service von Google mit dem Namen *GoogleSearchService* definiert. Hier findet die Bekanntgabe der Adresse, die den Service repräsentiert statt. In diesem Fall ist das *http://api.google.com/search/beta2*. Außerdem findet in dem Service Element die Verknüpfung von abstraktem und konkretem Beschreibungsteil statt. Hier werden Interface- und Bindingelement verbunden und referenziert.

2.2. SOAP

SOAP steht für Simple Object Access Protocol und ist ursprünglich für die eingangs erklärten RPCs entwickelt worden und hat sich als Standardnachrichtenformat für Web-Services durchgesetzt.[23]

Für den Nachrichtenaustausch via SOAP wurden neue Begriffe eingeführt. Ein Web-Service, der auf Anfragen wartet wird auch als *SOAP Node*, *SOAP Listener* oder auch *SOAP Server* bezeichnet. Die Adresse *http://api.google.com/search/beta2* aus dem obigen Beispiel ist demnach ein URI, unter der ein *SOAP Server* auf Anfragen wartet. Für SOAP nodes gibt es dieselben

[23] Vgl. Erl, Thomas Seite 72

Rollenverteilungen, wie für Web-Services (siehe Kapitel 1.3). Die Nachrichten, die mit SOAP versandt werden bezeichnet man, als *SOAP Envelope*.[24]

Ein solcher SOAP Envelope besteht aus zwei XML – Elmenten (siehe Abbildung 7: SOAP Envelope). Der Header Teil ist optional und dient dazu weitere Nachrichtenspezifikationen, die über die Standards der Web-Services der ersten Generation hinausgehen, zu spezifizieren (Siehe Kapitel 3). Der zweite Teil ist der SOAP Body und enthält den eigentlichen Aufruf.[25]

```
<env:Envelope xmlns:env:="http://…">
   <env:Header />
   <env:Body />
</env:Envelope>
```

Abbildung 7: SOAP Envelope

Beispiel: GoogleSearch, SOAP Nachricht

Anhand des Beispiels aus Kapital 2.1 lässt sich sehr gut die einfache Funktionsweise von SOAP erklären. Abbildung 8 zeigt ein Beispiel dafür, wie eine SOAP Nachricht verfasst ist, die die im WSDL Dokument beschriebene Operation *doGoogleSearch* ausführt. Die im „<ns1 />“ Bereich enthaltenen Elemente stellen den Inputparameter für die Operation dar und entsprechen der Definition der Message *doGoogleSearch*.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http:…" …>
  <SOAP-ENV:Body>
    <ns1:doGoogleSearch xmlns:ns1="urn:GoogleSearch"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <key xsi:type="xsd:string">00000000000000000000000000000000</key>
      <q xsi:type="xsd:string">shrdlu winograd maclisp teletype</q>
      <start xsi:type="xsd:int">0</start>
      <maxResults xsi:type="xsd:int">10</maxResults>
      <filter xsi:type="xsd:boolean">true</filter>
      <restrict xsi:type="xsd:string"></restrict>
      <safeSearch xsi:type="xsd:boolean">false</safeSearch>
      <lr xsi:type="xsd:string"></lr>
      <ie xsi:type="xsd:string">latin1</ie>
      <oe xsi:type="xsd:string">latin1</oe>
    </ns1:doGoogleSearch>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Abbildung 8: SOAP Envelope, doGoogleSearch

[24] Vgl. Erl, Thomas Seite 75
[25] Vgl. Erl, Thomas Seite 78

2.3. UDDI

UDDI steht für *Universal Description, Discovery and Integration* und stellt einen Verzeichnisdienst für Unternehmen dar. Es geht hierbei darum, dass potentiellen Benutzern von Services ein einfacher und bequemer Weg geboten wird um Dienste aufzufinden. Im Rahmen der Web-Services ist es den Unternehmen möglich über diesen Verzeichnisdienst ihre WSDL Dokumente mit einer genauen Beschreibung ihres Web-Service zu veröffentlichen.[26]

Hierfür existiert eine *Universal Business Registry (UBR)* in der die Unternehmen geführt werden. Die UBR besteht aus einzelnen *UDDI Operatoren*, die die einzelnen Daten der Unternehmen zur Verfügung stellen, aber auch untereinander austauschen und abgleichen. Als UDDI Operator agieren meist nur große Konzerne wie etwa Microsoft, SAP und IBM, da diese die hierfür notwendigen Ressourcen aufbringen können. Ein Serviceanbieter, bzw. Unternehmen kann sich dabei direkt bei einem UDDI Operator anmelden, oder den Weg über einen Registrar wählen, der die Anmeldung für sie übernimmt (siehe Abbildung 9: UDDI Übersicht).[27]

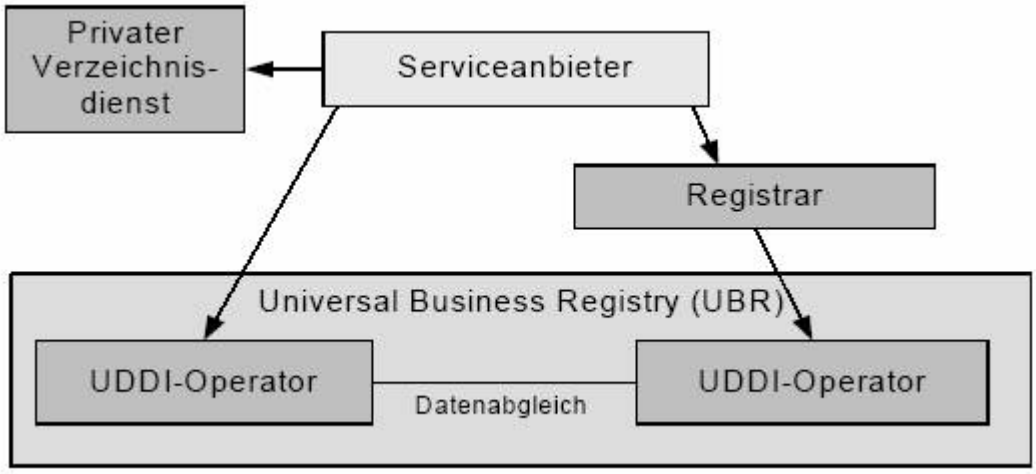

Abbildung 9: UDDI Übersicht[28]

Neben den öffentlich zugänglichen UDDI Operatoren gibt es noch weitere Möglichkeiten für Verzeichnisdienste.

[26] Vgl. o.V, Wikipedia, UDDI
[27] Vgl. Iwanowski, Sebastian, Vorlesung 11 Folie 19f
[28] Abbildung aus Iwanowski, Sebastian, Vorlesung 11 Folie 19

Service Marketplaces[29]

Ein *Service Marketplace* ist ein Verzeichnisdienst, der nicht wie die öffentlichen UDDI Operatoren kostenlose Services anbietet, sondern nur solche Web-Services die gegen Bezahlung nutzbar sind.

Private Registry[30]

Private Registries sind Organisationsinterne Verzeichnisse, die nach außen hin nur für ausgewählte Geschäftskunden oder Berater zugänglich sind.

Internal Registry[31]

Internal Registries sind rein Organisationsinterne Verzeichnisse, die nicht von außen zugänglich sind.

[29] Vgl. Erl, Thomas Seite 81
[30] Vgl. Erl, Thomas Seite 81
[31] Vgl. Erl, Thomas Seite 81

3. Web-Services der zweiten Generation

Die Web-Services der ersten Generation stellen ein sehr gutes Grundgerüst dar, mit dem es möglich ist serviceorientierte Software komfortabel zu entwerfen. Mit der zweiten Generation (WS-*) kommen nun noch weitere Standardisierungen für Web-Services auf, die die bisherigen Möglichkeiten der serviceorientierten Softwarearchitektur erweitern.[32]

3.1. Neuerungen

Die Grundsätzlichen Neuerungen der WS-* bestehen in der unten aufgeführten Liste von Techniken. Die einzelnen Punkte werden jeweils in den nebenstehenden Kapiteln beschrieben.[33]

- Kontext und Transaktionen (3.2 Koordination und Transaktion)
- Geschäftsprozesse (3.3 BPEL4WS)
- Sicherheit (3.4 Web-Service Sicherheit)
- gesicherte Nachrichtenversendung (3.5 Reliable Messaging)
- Richtlinien (3.6 WS Policy)
- Anhänge (3.7 Attachments)

3.2. Koordination und Transaktion

Die in Kapitel 1.4 geforderten Prinzipien der losen Kopplung und Statuslosigkeit der einzelnen Web-Services birgt die Schwierigkeit, dass bei der Interaktion mehrerer Services zusammen kein gemeinsamer Kontext existiert, bzw. dass die Verwaltung eines solchen Koordinationsprozesses selbstständig implementiert werden muss. Die WS-* stellen nun ein Werkzeug bereit, mit dem ein solcher Kontextprozess erzeugt werden kann und somit ein Standard dafür existiert. Es ist möglich für einen solchen Kontextprozess verschiedene Kontexttypen zu implementieren, wobei die

[32] Vgl. Erl, Thomas Seite 90
[33] Vgl. Erl, Thomas Seite 92f

Atomare Transaktion und die *Geschäftsaktivität* bereits als solche Kontexttypen vordefiniert sind.[34]

Atomare Transaktion

Die Atomare Transaktion bietet Entwicklern von Web-Services die Möglichkeit bestimmte Dienste nach dem ACID Prinzip ausführen zu lassen. Das bedeutet, dass nun die Funktionalität des „Commit“ und „Rollback“, sowie die exklusive Nutzung von Ressourcen möglich wird.[35]

Geschäftsaktivität

Als Geschäftsaktivität bezeichnet man eine Transaktion, die sich über einen langen Zeitraum hinweg erstreckt. Da bei einer atomaren Transaktion im obigen Sinne die Ressourcen für den gesamten Zeitraum gesperrt blieben und eine Geschäftsaktivität sehr lange dauern kann (bis zu mehreren Tagen) würde dies zu Insuffizienzen führen. Bei der Geschäftsaktivität werden daher die Ressourcen nicht gesperrt, sondern es bleibt nur der Kontext erhalten, in dem eine Aktivität steht. So bleiben die Ressourcen für andere Transaktionen oder Aktivitäten zugreifbar und der Verlauf der bisherigen Aktivität geht nicht verloren.[36]

Fehlerbehandlung

Tritt bei einer atomaren Transaktion oder bei einer Geschäftsaktivität ein Fehler auf, so müssen verschiedene Fehlerbehandlungen für die beiden Kontexttypen angewandt werden. Während bei der atomaren Transaktion ein einfaches „Rollback“ ausreichend ist, um die bisher ausgeführten Aktionen rückgängig zu machen, ist dies bei Geschäftsaktivitäten ungleich komplexer. Im Fehlerfall bei der Geschäftsaktivität wird ein eigener Kompensationsprozess gestartet, der anhand des Kontextes den Fehler analysiert und korrigiert, bzw. bisherige Aktionen rückgängig macht.[37]

[34] Vgl. Erl, Thomas Seite 96f
[35] Vgl. Erl, Thomas Seite 98
[36] Vgl. Erl, Thomas Seite 98
[37] Vgl. Erl, Thomas Seite 98f

3.3. BPEL4WS

BPEL4WS ist die Abkürzung für *Business Process Execution Language for Web-Services* (im folgenden BPEL genannt) und ist eine Skriptsprache für den koordinierten Einsatz von Web-Services. Mit BPEL ist es möglich komplexe Geschäftsprozesse zu steuern und in eine Reihenfolge zu bringen. Hierfür gibt es verschiedene Partnerservices und Sprachelemente. Wobei es atomare, nicht weiter aufteilbare und zusammengesetzte, komplexe Elemente gibt, die aus atomaren oder komplexen Elementen bestehen können.[38]

Partner Service

Als Partnerservice werden Services bezeichnet, die an einem Geschäftsprozess beteiligt sind. Ein *external Partnerservice* ist dabei ein Service, der die Ausführung eines BPEL Dokumentes fordert. Der *process Partnerservice* ist der ausführende Service und ein *invoked Partnerservice* ist dadurch definiert, dass er vom *process Partnerservice* aufgerufen und in den Geschäftsprozess miteinbezogen wird (siehe Abbildung 10: Partnerservice).[39]

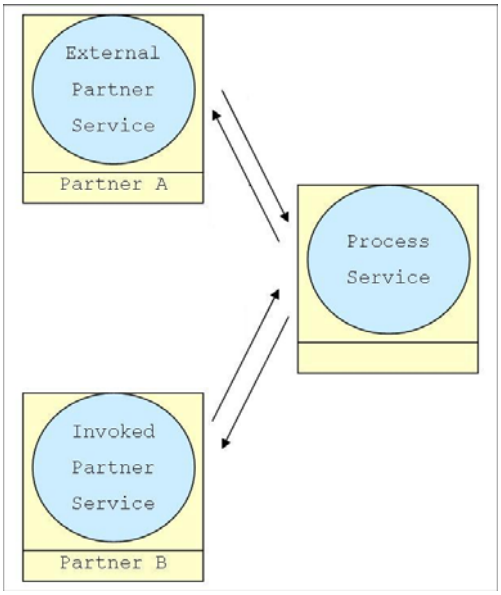

Abbildung 10: Partnerservice[40]

[38] Vgl. Erl, Thomas Seite 100
[39] Vgl. Erl, Thomas Seite 102ff
[40] Abbildung angelehnt an Erl, Thomas Seite 105

Atomare Elemente[41]

- *Receive* – mit Receive wird ein BPEL Prozess gestartet. Der process Partnerservice startet daraufhin den restlichen Teil des BPEL Skriptes
- *Invoke* – mit Invoke wird vom process Partnerservice der invoked Partnerservice angesprochen
- *Reply* – mit dem Reply Element wird einem Partnerservice geantwortet
- *Throw* – mit dem Throw Element kann festgelegt werden, wie der process Partnerservice im Fehlerfall zu reagieren hat
- *Wait* – mit Wait ist es möglich ein BPEL Skript für eine bestimmte Zeit zu stoppen oder ab einem bestimmten Zeitpunkt weiterlaufen zu lassen

Komplexe Elemente[42]

- *Sequence* – Sequence ist ein Grundlegendes BPEL Element, in dem eine Reihenfolge der Ausführung von atomaren oder komplexen Elementen festgelegt werden kann
- *Flow* – Dient dazu mehrere Elemente gleichzeitig ausführen zu lassen.
- *Switch* – Mit Switch ist es möglich bestimmte Aktionen nur unter einer bestimmten Bedingung ausführen zu lassen (entspricht If bei konventionellen Programmiersprachen)
- *While* – While stellt eine einfache Schleife dar

3.4. Web-Service Sicherheit

Hinter dem Begriff der Web-Service Sicherheit verbergen sich viele einzelne Formen der Sicherung von Web-Services. Der Gedanke der Web-Service Sicherheit ist relativ neu, allerdings wird für die Lösung von spezifischen Sicherheitsproblemen auf bereits vorhandene Techniken zurückgegriffen. Für die Web-Services sind konkret folgende Sicherheitsprobleme definiert worden.[43]

[41] Vgl. Erl, Thomas Seite 104
[42] Vgl. Erl, Thomas Seite 104f
[43] Vgl. Erl, Thomas Seite 109f

- *Identifikation* – „Wer bist du?“
- *Authentifizierung* - „Woher weiß ich, dass du der bist, der du vorgibst zu sein?“
- *Autorisation* – „Darfst du das ausführen, was du ausführen möchtest?“
- *Integrität* – „Wurde die Nachricht auf dem Weg verändert?“
- *Diskretion* – „Wer hat die Nachricht unerlaubt auf dem Weg gelesen?“[44]

Wie in Tabelle 1 zu erkennen ist, gibt es für die jeweiligen Problemstellungen verschiedene Lösungen. Bis auf das *WS-Security Framework* wird hier nicht näher auf die einzelnen Techniken eingegangen.

Identifikation	WS-Security Framework
Authentifizierung	Extensible Access Control Markup Language (XACML)
Autorisation	Extensible Rights Markup Language (XrML) XML Key Management (XKMS) Security Assertion Markup Language (SAML) .NET Passport
Integrität	WS-Security Framework XML-Encryption Secure Socket Layer (SSL)
Diskretion	WS-Security Framework

Tabelle 1: Web-Service Sicherheit[45]

[44] Vgl. Erl, Thomas Seite 110f

[45] Tabelle Übersetzt aus Erl, Thomas Seite 112

Web-Service Security Framework

Das Web-Service Security Framework (im Folgenden WSSF genannt) bildet aus verschiedensten Sicherheitswerkzeugen eine auf Web-Services zugeschnittene Sicherheitsplattform. Das WSSF umfasst zum einen die Sicherung von Ende-Zu-Ende Verbindungen für das Versenden von SOAP Envelopes und zum anderen beinhaltet das WSSF einige der schon genannten Sicherheitslösungen (siehe Abbildung 11: WS-Security Framework)[46]

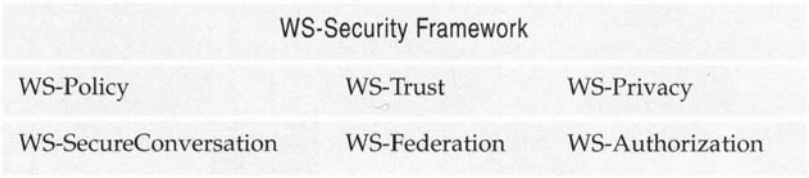

Abbildung 11: WS-Security Framework[47]

Die erste Schicht wird als *Policy Layer*, die zweite als *Federation Layer* bezeichnet.

- *WS-Policy* – WS-Policy stellt einen Schlüsselteil des WSSF dar (genaueres siehe Kapitel 3.6)
- *WS-Trust* – Authentifizierungsteil, stellt einen Kommunikationsprozess zur Verfügung, über den dritte Sicherheitsstellen zur Prüfung von Identitäten erreicht werden können.
- *WS-Privacy* – Web-Services können hier Datenschutzrichtlinien übermitteln und feststellen, ob ein Nachfragender Service diesen Richtlinien entspricht.
- *WS-SecureConversation* – SecureConversation bietet formale Definitionen für Sicherheitskontexte und session Keys an, weitere Protokolle zur sicheren Kommunikation können hier eingebettet werden.
- *WS-Federation* – Mit WS-Policy und WS-Trust können *trust domains* erstellt werden. WS-Federation bietet nun die Möglichkeit an eine Föderation, einen Bund zwischen solchen bereits bestehenden *trust domains* zu schließen.

[46] Vgl. Erl, Thomas Seite 115
[47] Abbildung aus Erl, Thomas Seite 116

- *WS-Authorization* – Biete einen Standard für die Zugriffskontrolle und Autorisation für Web-Services.[48]

3.5. Reliable Messaging

Reliable Messaging steht für die zuverlässige Nachrichtenauslieferung und stellt sicher, dass ein verschickter SOAP Envelope ankommt. Hierfür wird vom empfangenden Web-Service eine Empfangsbestätigung an den absendenden Web-Service zurückgeschickt. Diese Antwort wird als Acknowledgement bezeichnet. Das Acknowledgement kann sowohl als eingeständige SOAP Nachricht versandt werden, als auch als Teil eines Reply. Außerdem ist es mithilfe des Reliable Messaging möglich eine Folge von zusammenhängenden Nachrichten zu verschicken. Bei dem Versandt von einer Nachrichtenfolge kann das Acknowledgement als eine einzige Nachricht zurückgeschickt werden. Es ist allerdings auch ein Acknowledgement pro empfangener Nachricht einer Folge möglich. Wird die Variante mit nur einem Acknowledgement gewählt, kann angegeben werden welche Nachrichten angekommen sind (siehe Abbildung 13: Acknowledgement einer Nachrichtenfolge).[49]

Sequence

Um einen SOAP Envelope zuverlässig zu versenden wird in den Header Teil ein „<sequence />“ Element eingebaut, in dem die Informationen für das Reliable Messaging zu finden sind (siehe Abbildung 12: Sequence Element).

```
<wsrm:Sequence>
    <wsu:Identifier>http://www.examples.ws/</wsu:Identifier>
    <wsrm:MessageNumber>6</wsrm:MessageNumber>
    <wsrm:LastMessage />
</wsrm:Sequence>
```

Abbildung 12: Sequence Element

Das Sequence Element besteht aus drei Sub-Elementen.

[48] Vgl. Erl, Thomas Seite 116f
[49] Vgl. Erl, Thomas Seite 118ff

- *Identifier* – Hier wird der SOAP Nachricht eine Eindeutige Identifizierung gegeben. Alle Nachrichten mit demselben Identifier gehören zusammen.
- *MessageNumber* – Gibt an, um die wievielte Nachricht es sich handelt
- *LastMessage* – Optional, kennzeichnet die letzte Nachricht in einer Folge von Nachrichten[50]

Acknowledgement[51]

Für ein *Acknowledgement* von Nachrichten wird ein *SequenceAcknowledgement* Element in den SOAP Header eingefügt (siehe Abbildung 13: Acknowledgement einer Nachrichtenfolge).

```
<wsrm:SequenceAcknowledgement>
    <wsu:Identifier>http://www.examples.ws/</wsu:Identifier>
    <wsrm:AcknowledgementRange Upper="3" Lower="1" />
    <wsrm:AcknowledgementRange Upper="6" Lower="5" />
</wsrm:SequenceAcknowledgement>
```

Abbildung 13: Acknowledgement einer Nachrichtenfolge

Das SequenceAcknowledgement besteht hier aus zwei Sub-Elementen.

- *Identifier* – Hierdurch wird gekennzeichnet auf welche Nachrichten sich das Acknowledgement bezieht
- *AcknowledgementRange* – Angabe, welche Nachrichten Empfangen wurden. Im Beispiel handelt es sich um das Acknowledgement einer Nachrichtenfolge, wobei die Nachrichten 1 bis 3 und 5 bis 6 empfangen wurden. Die vierte Nachricht ist nicht übermittelt worden und muss erneut verschickt werden.

[50] Vgl. Erl, Thomas Seite 119
[51] Vgl. Erl, Thomas Seite 121

Auslieferungszusicherungen[52]

Es gibt im Rahmen der zuverlässigen Nachrichtenübermittlung vier verschiedene Zusicherung, wie eine Nachricht ausgeliefert werden soll.

- *AtMostOnce* – eine Nachricht wird gar nicht oder einmal ausgeliefert
- *AtLeastOnce* – eine Nachricht wird mindestens einmal ausgeliefert
- *ExactlyOnce* – eine Nachricht wird genau einmal ausgeliefert
- *InOrder* – eine Nachrichtenfolge muss genau in der abgeschickten Reihenfolge empfangen werden

Diese Zusicherungen werden mithilfe der Web-Service Policy definiert und angewandt (siehe Kapitel 3.6).

3.6. WS Policy

Mit Web-Service Policy können Richtlinien für Web-Services erstellt und angewandt werden. Hierfür wird zunächst eine Richtlinie definiert und ein Objekt ausgewählt, auf das die Richtlinie angewandt werden soll.

- *Policy Assertions* – die Definition einer Richtlinie wird als *policy Assertion* bezeichnet
- *Policy Subjects* – das Objekt, auf das eine Richtlinie angewendet werden soll wird *policy Subject* genannt.
- *Policy Attachments* – Mithilfe des *policy Attachments* werden *Assertion* und *Subject* miteinander verknüpft

Abbildung 14 zeigt ein Beispiel dafür, wie eine Policy angewendet wird. Hier wird eine Auslieferungsversicherung (siehe Kapitel 3.5) für die Nachrichtensequenz aus Abbildung 12 hinzugefügt. Die Nachrichtensequenz wird über das Identifier Element

[52] Vgl. Erl, Thomas Seite 120

referenziert und die konkrete Zusicherung wird über das Policy Element definiert (hier AtLeastOnce).[53]

```
<wsp:PolicyAttachment>
    <wsp:AppliesTo>
        <wsrm:SequenceRef>
            <wsu:identifier>
                http://www.examples.ws/
            </wsu:identifier>
        </wsrm:SequenceRef>
    </wsp:AppliesTo>
    <wsp:policy>
        <wsrm:DeliveryMessageAssurance
            Value="wasrm:AtLeastOnce"
            wsp:Usage="wsp:Required" />
    </wsp:policy>
</wsp:PolicyAttachment>
```

Abbildung 14: Policy Attachments[54]

3.7. Attachments

Für die Web-Services der zweiten Generation gibt es die Möglichkeit Dateien an einen SOAP Envelope anzuhängen. Dies können etwa binäre Dateien wie Bilder oder auch anders kodierte XML Dateien sein. Es gibt für die Dateianhänge an SOAP Nachrichten zwei konkurrierende Standards.

- *WS Attachment*
- *SOAP Message with Attachments (SwA)*

Beide Standards haben gemeinsam, dass die SOAP Nachricht aus zwei Teilen besteht. In dem *primary* Teil wird die eigentliche SOAP Nachricht versendet und in dem zweiten *secondary* Teil wird dann der Anhang versandt. Die beiden Standards unterschieden sich lediglich in der Kodierung der Dateianhänge. Bei WS Attachments wird der Anhang als DIME Type und bei SwA als MIME Type kodiert.[55]

[53] Vgl. Erl, Thomas Seite 125f
[54] Abbildung angelehnt an Erl, Thomas Seite 126f
[55] Vgl. Erl, Thomas Seite 127ff

Anhang I: GoogleSearch WSDL Dokument

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

<definitions>

  <!-- Types for search - result elements, directory categories -->

  <types>

      <xsd:complexType name="GoogleSearchResult">

        <xsd:all>

          <xsd:element name="documentFiltering"           type="xsd:boolean"/>

          <xsd:element name="searchComments"              type="xsd:string"/>

          <xsd:element name="estimatedTotalResultsCount"  type="xsd:int"/>

          <xsd:element name="estimateIsExact"             type="xsd:boolean"/>

          <xsd:element name="resultElements"              type="typens:ResultElementArray"/>

          <xsd:element name="searchQuery"                 type="xsd:string"/>

          <xsd:element name="startIndex"                  type="xsd:int"/>

          <xsd:element name="endIndex"                    type="xsd:int"/>

          <xsd:element name="searchTips"                  type="xsd:string"/>

          <xsd:element name="searchTime"                  type="xsd:double"/>

        </xsd:all>

      </xsd:complexType>

    </xsd:schema>

  </types>

  <message name="doGoogleSearch">

    <part name="key"            type="xsd:string"/>

    <part name="q"              type="xsd:string"/>

    <part name="start"          type="xsd:int"/>

    <part name="maxResults"     type="xsd:int"/>

    <part name="filter"         type="xsd:boolean"/>

    <part name="restrict"       type="xsd:string"/>

    <part name="safeSearch"     type="xsd:boolean"/>

    <part name="lr"             type="xsd:string"/>

    <part name="ie"             type="xsd:string"/>

    <part name="oe"             type="xsd:string"/>

  </message>
```

```
  <message name="doGoogleSearchResponse">
    <part name="return"         type="typens:GoogleSearchResult"/>
  </message>
  <interface name="GoogleSearchPort">
    <operation name="doGoogleSearch">
      <input message="typens:doGoogleSearch"/>
      <output message="typens:doGoogleSearchResponse"/>
    </operation>
  </ interface >
  <binding name="GoogleSearchBinding" type="typens:GoogleSearchPort"    >
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="doGoogleSearch">
      <soap:operation soapAction="urn:GoogleSearchAction"/>
      <input>
        <soap:body use="encoded" namespace="urn:GoogleSearch"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded" namespace="urn:GoogleSearch"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>
  <service name="GoogleSearchService">
    <port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">
      <soap:address location="http://api.google.com/search/beta2"/>
    </port>
  </service>
</definitions>
```

Das WSDL – Dokument wurde zur Vereinfachung gekürzt und verändert. Das Original kann hier heruntergeladen werden: http://www.google.com/apis/download.html

Literaturverzeichnis

Literaturverzeichnis

Erl, Thomas (2004) „Service-Oriented Architecture – A Field Guide to Integrating XML and Web Services“ Upper Saddle River (2004): Pearson Education, Inc. (New Jersey, USA), Seite 47 - 129

Quellen im Internet

Erl, Thomas (2004) „Service-Oriented Architecture – A Field Guide to Integrating XML and Web Services“ Upper Saddle River (2004): Pearson Education, Inc. (New Jersey, USA), Seite 47 – 129

Iwanowski, Sebastian (2005) “Vorlesung Verteilte Systeme” Wedel: Fachhochschule Wedel – University of Applied Science (Deutschland), Internet: http://www.fh-wedel.de/~iw/Lehrveranstaltungen/SS2005/VS.html

o.V. (2005) Wikipedia, freie Enzyklopädie: “Remote Procedure Calls, [RPC]”; Internet: http://de.wikipedia.org/wiki/Remote_Procedure_Call, Stand 19.11.2005

o.V. (2005) Wikipedia, freie Enzyklopädie: “[UDDI] (Universal Description, Discovery and Integration)”; Internet: http http://de.wikipedia.org/wiki/UDDI, Stand 19.11.2005