Technische Universität Ilmenau
Informatik / Automatisierung

Juli / August, SS 2003

WSDL – Web Service Description Language

Hauptseminar: Webservices
Betreuer: Thorsten Strufe

Verfasser:

Karsten Schmidt
Matrikelnummer: 28833

E-mail: karsten.schmidt@in.stud.tu-ilmenau.de

Inhaltsverzeichnis

1. WSDL Einstieg

1.1 Einleitung

Das Webservices ein Teil der Zukunft gehört und das sie jetzt schon stark vertreten sind ist unbestritten. Doch was kann WSDL mehr oder neues und wie ? Aktuelle Systeme, wie z.B. EDI (Electronic Data Interchange), haben sich etabliert und sind bei den meisten Industriepartnern im Einsatz. Meistens sind diese Systeme mit hohem Aufwand einzeln angepasst worden. Das erfordert wiederum viel Arbeit diese Systeme zu erweitern und Konformitäten zu sichern. Auch die Interoperabilität verschiedener Systeme / Dienste zu eindander ist sehr mangelhaft. Bisher gibt es auch keine Möglichkeiten Webservices dynamisch zu suchen und zu benutzen. Die im vorrangegangenen Vortrag vorgestellte Technik SOAP (Simple Object Access Protocol) stellt die meistgenutzte Grundlage zur Übertragung von Daten im Internet für B2B Systeme da. Auch bei SOAP müssen die Schnittstellen des Webservices bekannt sein, um diesen benutzen zu können. Weiterhin sind Namespaces, Datentypen und Methodenaufrufe nicht, oder sehr umständlich beschreibbar.

Mit WSDL versucht man alle diese Eigenschaften zu etablieren. Dabei kommt XML zum Einsatz. WSDL ist eine Dienstebeschreibungssprache ähnlich wie die IDL in CORBA. Es wird nur die Nutzung des Dienstes beschrieben und nicht die Implementierung. Es gibt keine Nachrichten – und Formatvorschriften. Genauso wie es keine Netzwerkprotokollvorschriften gibt. Jeder Diensteanbieter beschreibt auch die Nutzung von SOAP, HTTP GET/POST und MIME. Dabei ist jedem selbst überlassen wieviele und welche Möglichkeiten er anbietet.

Allerdings müssen einige Vorraussetzungen erfüllt werden, im Bezug auf die Informationen die beschrieben werden müssen. In den WSDL Dokumenten muss vereinbart werden, welche Informationen der Dienst vom Benutzer erwartet und in welcher Form er diese erwartet. Auch für die eventuelle Dienstantwort wird das Format bekannt gegeben.

Zusammengefaßt bedeutet das :

WSDL – standardisiert und beschreibt Webservices, unabhängig vom Protokoll der Nachrichtenschicht (z.B. SOAP) und unabhängig von Datentypepräsentationen.

1.2 Webservicestack und die Einordnung von WSDL

Da die WSDL auf Techniken wie XML aufbaut ordnet sie sich in dem in Abbildung 1.2.1 gezeigten Webservicestack relativ weit oben ein. Die unterliegenden Sprachen und Techniken wurden in den vorrangegangenen Hauptseminaren ausgiebig erläutert (siehe Vortrag SOAP und XML). Die UDDI, welche oberhalb der WSDL im Webservicestack zu finden ist, wird später im Zusammenhang der Nutzung von Webservices näher betrachtet.

Service Negotiation	Trading Partner Agreement	oben ↑
Workflow, Discovery, Registries	UDDI, ebXML registries, IBM WSFL, MS XLANG	
Service Description Language	**WSDL** / WSCL	
Messaging	SOAP/XML Protocol	
Transport Protocols	HTTP, HTTPS, FTP, SMTP	

Abb. 1.2.1 - (Web Service Stack nach WebServices.Org)

Weitere Workflow Sprachen sollen hier nicht betrachtet werden. Es sei darauf verwiesen, das WSDL nicht die einzige Bemühung darstellt einen Standard für die WebService Beschreibung zu etablieren (siehe History und Zukunft von WSDL).

1.3 History von WSDL

- Stand Anfang 2000 :

  Auf dem Markt befanden sich zwei inkompatible Webservicebeschreibungssprachen von IBM und Microsoft. Die von IBM stammende NASSL (Network Accessibility Service Specification Lanquage) nutzte XML und SOAP. Einer der Nachteile

von NASSL war die zwingende Bindung des Dienstes an eine URL. Datentypen konnten schon in XML beschrieben werden.

Das Konkurenzprodukt von Microsoft. SCL (Service Contract Language), später SDL (Service Description Lanquage) gennant, war der NASSL sehr ähnlich. Allerdings konnten Datentypen nur in der eigenen XDR (XML Data Reduced) Sprache definiert werden.

- September 2000:
  Eine gemeinsame Spezifikation der WSDL von IBM und Microsoft wurde ausgearbeitet und angeboten.

- November 2000:
  Es folgte auch gleich die UDDI Spezifikation von IBM und Microsoft, um die Suche und Registrierung von Webservicen zu erleichtern.

- März 2001:
  Die von IBM, Microsoft und Ariba beim W3C Konsortium eingereichte Empfehlung der WSDL Version 1.1 wurde anerkannt. Mit dem Ziel : selbstbeschreibende Webservices zu erstellen, die unabhängig vom Betriebssystem, der Programmiersprache und dem verwendeten Objektmodell benutzt werden können.

- Januar 2002:
  Das W3C erkannte das Potential der WSDL und startete eine Working Group für WSDL, die sich mit den vorhandenen Inkompatibilitäten auseinandersetzen soll.

- Februar/August 2002:
  IBM und Microsoft dauerte der Prozess des W3C zu lange, und starteten das WS-I (Webservices Interoperability Organization) Projekt um eine BPEL (Business Process Execution Lanquage) zu definieren. BPEL setzt auf WSDL auf und beschreibt abstraktere Zusammenhänge von Webservices.

- März 2003:
  W3C gibt neue Version 1.2 der WSDL, mit überarbeiteten Binding Funktionen frei.

- aktuelle Situation:

  Die BPEL wird von IBM, Microsoft, SAP und BEA unter der OASIS unterstützt. Wohingegen Oracle und SUN das W3C unterstützen.

1.4 WSDL Architektur

Die WSDL Dokumente sollten auf 2 getrennte Dokumente aufgeteilt werden. Dem WSDL – Service Interface Document, in dem die Schnittstellen des Dienstes beschrieben werden. Und in das WSDL – Service Implementation Document, in dem der Webservice beschrieben, und das Interface Document eingebunden wird. Diese Dokumente werden mit Hilfe der UDDI in Registries abgelegt. In den Verzeichnissdiensten kann per UDDI ein WSDL Dokument gesucht werden, um den Dienst dahinter zu nutzen. Bei der Nutzung der WSDL (Kapitel 3) wird auf dieses Szenario weiter eingegangen.

Die Inhalte der einzelnen Dokumente werden im nächsten Abschnitt genauer erläutert. Hier nur kurz das jeweilige XML – Tag Schema, welches nötig ist um einen Dienst zu beschreiben :

WSDL – Implementation Document	WSDL – Interface Document
<definitions ...>	
<types ...>	<definitions ...>
<import ...>	<import ...>
<message ...>	<service ...>
<portType ...>	<port ...>
<binding ...>	</service>
</definitions>	</definitions>

2. WSDL Sprache und Elemente

2.1 Hauptelemente der WSDL

2.1.1 Tag <definitions>

Das Wurzelelement jeder XML-formbasierten WSDL Implementation beinhaltet vorwiegend 2 verschiedene Elemente. Zum einem den Namen des Webservice und als zweites alle verwendeten Namespaces werden hier referenziert.

```
<definitions name = "MyService"
 targetNamespace = "http://www.mydomain.net/wsdl/MyService.wsdl"
 xmlns = "http://schemas.xmlsoap.org/wsdl/"
 xmlns:soap = "http://schemas.smlsoap.org/wsdl/soap/"
 xmlns:tns = "http://www.mydomain.net/wsdl/MyService.wsdl"
 xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
 ...
</definitions>
```

In diesem Beispiel wurde ein Webservice mit dem Namen „MyService“ definiert. Mehrere verwendete Namespaces für XML, SOAP und ein eigener wurden angegeben. Damit kann der XML Parser alle Schemas und Namespaces verstehen und interpretieren.

2.1.2 Tag <types>

Hier können eigene und zusätzliche Datentypen, die beim benutzen des Service nötig sind deklariert werden. Das standardmäßig verwendete Deklarationsschema ist das W3C XML Schema. Einfache Datentypen, wie Strings und Integer sind bereits eingebaut, und brauchen nicht neu definiert werden. Es können beliebig viele frei strukturierte und komplexe Datentypen deklariert werden. Im folgenden Beispiel wird ein komplexer Datentyp mit dem Namen „artikel“ definiert. Dieser bekommt Artikeltypische Attribute, wie Preis und einen Namen.

```
<types>
 <xsd:schema targetNamespace="http://www.mydomain.net/schema"
  xmlns="http://www.w3c.org/2001/XMLSchema">
  <xsd:complexType name="artikel">
   <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="beschreibung" type="xsd:string"/>
    <xsd:element name="preis" type="xsd:double"/>
    <xsd:element name="id" type="xsd:string"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:schema>
</types>
```

2.1.3 Tag <message>

Als nächstes müssen noch Nachrichten definiert werden. Diese Nachrichten werden ausgetauscht zwischen Service Benutzer und Anbieter. Jede Nachricht wird XML-typisch mit einem Namen und den optionalen Parametern sowie Rückgabewerten spezifiziert. Es gibt verschiedene Möglichkeiten die Nachrichten zu nutzen, One-Way Messages werden, wie der Name sagt nur in eine Richtung geschickt und erwarten keine Antwort. Im Request Response Schema gelten die Nachrichten als Anfrage oder Antwort. Das ist nur eine semantische Bindung des Kontextes in dem die Nachricht verwendet werden soll, und keine technische Bindung.

```
<message name="RequestService">
  <part name="Parameter1" type="xsd:string"/>
</message>
<message name="ResponseService">
  <part name="ReturnValue" type="xsd:string"/>
</message>
```

Für jede message muss ein eindeutiger Name vergeben werden. Mit dem <part> Tag können Parameter angegeben werden. Hier werden dann auch die selbstdefinierten Datentypen verwendet.

2.1.4 Tag <portType>

Der nächste Schritt, um eine Anfrage oder Antwort senden zu können ist, den Aufbau, der zu kommunizierenden Nachrichten zu bestimmen. Mit dem portType Tag verbindet man Nachrichten, um Funktionen für den Dienst zu definieren. Eine Funktion beinhaltet ein oder mehrere Operationen (gekennzeichnet durch den <operation> Tag) und in diesen ein oder mehrere Messages. Die Messages werden an dieser Stelle in Ein – und Ausgehende Nachrichten unterschieden. Für den Fehlerfall, wenn er vom Dienst erkannt wird, gibt es die fault message.

Eine Request und eine Response Nachricht in einer Operation bezeichnet man als Single Operation. Am häufigsten sind die folgenden 4 Standardtypen :

1. One Way : die Nachricht wird ohne eine Antwort zu erwarten geschickt
2. Request + Response : Frage und Antwort, die wohl häufigste Nutzung
3. Solicit Response : der Dienst erhält eine Antwortnachricht vom Nutzer
4. Notification : der Dienst sendet eine Nachricht (z.B. Info's)

```
<portType name="My_PortType">
  <operation name="My_Operation">
    <input message="tns:RequestService"/>
    <output message="tns:ResponseService"/>
      <fault message="tns:ErrorService"/>
  </operation>
</portType>
```

In diesem Beispiel wurde ein PortType mit einer Operation definiert. Die Funktion des Dienstes bzw. Operation „My_Operation“ erwartet eine Anfrage definiert unter Messages als RequestService und gibt standardmäßig eine Antwortnachricht ResponseService aus.

2.1.5 Tag <binding>

Mit den Bindings wird die Nutzung der PortTypes genauer spezifiziert. Hier wird das zu verwendende Protokoll festgelegt. Am häufigsten kommt hier SOAP zum Einsatz, doch per HTTP/GET, HTTP/POST und MIME kann man die Nachrichten genauso austauschen. Allerdings bietet SOAP schon einigen Komfort, der in dem Vortrag „SOAP und HTTP Framework“ genauer gezeigt wurde. Um wieder Unabhängigkeit und Vielseitigkeit zu beweisen, kann man für einen PortType mehre-

re Bindings definieren, um dem Klienten nicht ein einziges Protokoll vorschreiben zu müssen. Ein Nachteil besteht darin, das bei Änderungen eventuell mehrere Bindings angepasst werden müssen. (Update Fehler bei Redundanter Implementierung)

```
<binding name="My_Binding" type="tns:My_PortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="MyOperation">
    <soap:operation soapAction="MyOperation"/>
    <input>
      <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:examples:myservice"
        use="encoded"/>
    </input>
    <output>
      ...
    </output>
  </operation>
</binding>
```

Bei diesem Binding Beispiel kam SOAP zum Einsatz. Die Funktion „MyOperation“ kann als SOAP Nachricht aufgerufen werden. Man muss die Input und Output Nachrichten im einzelnen beschreiben.

2.1.6 Tag <service>

Im Service Tag werden den Bindings noch Aufrufadressen hinzugefügt. Hierbei handelt es sich meistens um eine URL, an welche die Anfragen geschickt werden müssen.

```
<service name="My_Service">
  <documentation>WSDL File for MyService</documentation>
  <port binding="tns:My_Binding" name="My_Port">
    <soap:address location="http://irgendwo.de:8080/soap/servlet/rpcrouter"/>
  </port>
</service>
```

Es ist wichtig für jedes Binding, welches der Webservice anbieten soll, ein “<port binding>“ anzugeben. Hier anhand einer URL, wohin die SOAP Nachricht geschickt werden muss. Der optionale <documentation> Tag ist für die freie Verwendung, sollte aber dazu genutzt werden, den Service nochmal genauer Se-

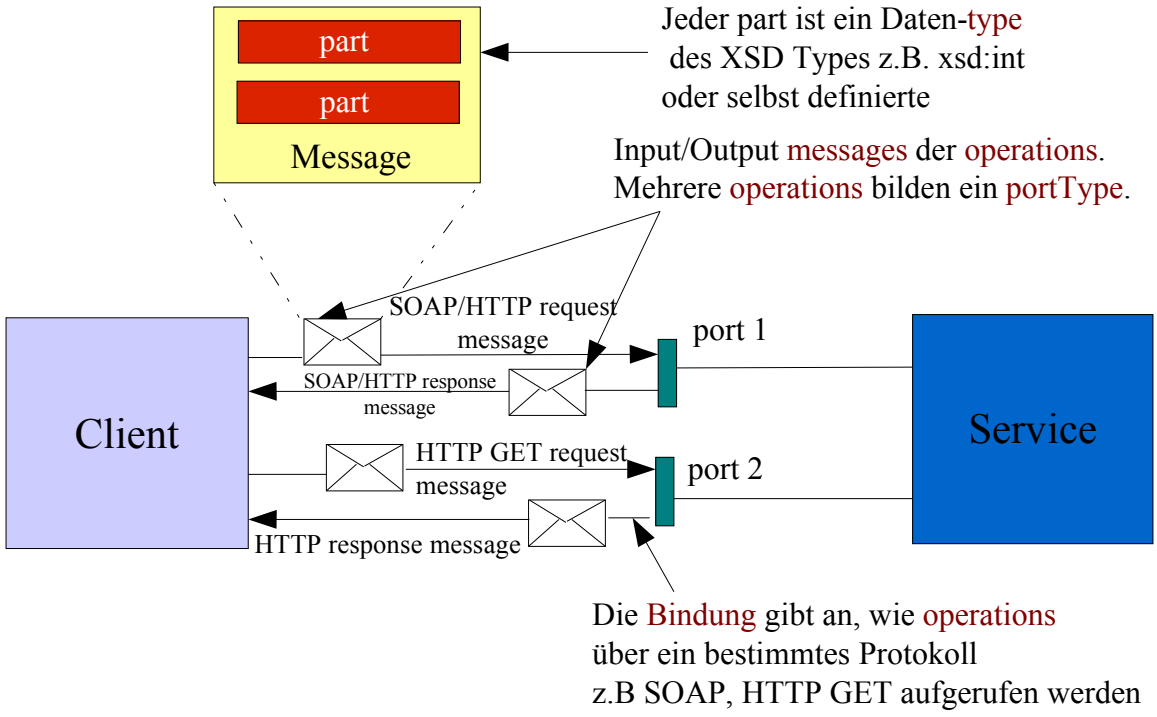
mantisch zu beschreiben. Durch ihn können evtl. Registries weitere Suchkriterien für Webservices bestimmen.

2.1.7 Tag <import>

Damit die empfohlene Aufteilung der WSDL Dokumente wieder verbunden werden kann, ermöglicht der Import Tag die Einbindung weiterer WSDL Dokumente. Desweiteren kann man es für die Wiederverwendbarkeit nutzen, wobei das nicht zu vergleichen ist, mit der Wiederverwendbarkeit von Sourcecode in herkömmlichen Programmiersprachen.

2.2. Überblick & Zusammenfassung

2.2.1. Überblick



So kann man sich den Weg und die Art & Weise einer SOAP oder HTTP Nachricht zwischen einem Webservice und einem Klienten vorstellen. Dabei sieht man, das der Service unterschiedliche Ports für den SOAP und den HTTP Aufruf anbietet.

2.2.2. Zusammenfassung

WSDL **Interface** Dokument :

- ➔ definitions — Wurzelelement + Namespaces
- ➔ types — Datentyp Definitionen – Type Systems like XSD
- ➔ import — weitere WSDL Dokumente einbinden
- ➔ message — Syntax der Nachricht mit den definierten Datentypen
- ➔ portType — Menge von Operationen
- ➔ binding — bindet Operationen eines PortTypes an ein Protokoll

WSDL **Implementation** Dokument :

- ➔ definitions — Wurzelelement
- ➔ import — weitere WSDL Dokumente einbinden
- ➔ port — Endpunkt, Netzwerkadresse + Binding
- ➔ service — Zusammengehörige Ports bilden ein Webservice

3. Nutzung von WSDL

3.1. Im Webservice Szenario

Die WSDL Dokumente werden im Allgemeinen in Zusammenhang mit Registries genutzt. Diese Registries stellen eine Art Gelbe Seiten für Webservices da. In welchen, der Dienstanbieter seine Beschreibung hinterlegen kann. Ein Standard für solche „Gelben Seiten“ stellt das System der UDDI (Universal Discovery and Discription Language) da. Die UDDI wurde von den Branchengrößen Microsoft, IBM und SAP entwickelt und im Moment kostenlos zur Verfügung gestellt. Das bedeutet, das sie ein Netz von Registries hosten und somit jedem die WSDL Dokumente zur Verfügung stellen. Ein Dienstbenutzer kann sich so seinen Webservice suchen, indem er anhand seiner Kriterien in der UDDI sucht. Bei erfolgreicher Suche, kann er sich die WSDL Dokumente herunterladen und somit den Webservice nutzen.

3.2. Development & Invocation

3.2.1. Service Anbieter

	Neues Service Interface	Vorhandenes Interface
Neuer Webservice	green field	top-down
Vorhandener Webservice	bottom-up	meet-in-the-middle

1. Green Field
   Neuer Webservice und neues Interface !
   Um einen neuen Webservice zu entwickeln benötigt man einen Webserver, bei SOAP Nutung noch den SOAP RPC-Router und den SOAP Message Router. Als nächstes kann der Service, die Webserviceapplikation, entwickelt werden. Ein neues Service Interface muss definiert werden. Um die Operationen des Webservices abstrakt zu beschreiben (type, message und portType). Es folgt das Publishing, indem die Interface Definition bekannt gemacht wird. Entweder in die UDDI oder einer anderen Registry eintragen. Danach soll die WSDL Implementation Definition erstellt und veröffentlicht werden (binding,service und port). Ab jetzt kann der Webservice und/oder die WSDL Beschreibung von anderen genutzt werden.

2. Top-Down
   Zu einem exisitierenden Interface kommt ein neuer Webservice !
   Da die WSDL Beschreibungen existieren, bedeutet der erste Schritt die Suche danach, wiederum in der UDDI oder einer anderen Registry. Vor der Erstellung des Webservices soll eine Implementation Vorlage, als eine Art Prototyp erstellt werden. Und nach dem Webservice wird das endgültige WSDL Implementation Dokument veröffentlicht.
   Sinn macht die Top-Down Methode beim Entwickeln von Webservices, wenn man die Interoperabilität der Dienste fördern und gewährleisten will, falls es schon Dienstebeschreibungen zu der Art von Diensten gibt.

3. Bottom-Up
   Interface exisitiert nicht, aber der Webservice exisitiert bereits !
   Zuerst wird die Interface Beschreibung, genau wie bei der Green-Field Methode erstellt. Danach sollte die Implementierungsbeschreibung folgen, und beide WSDL Dokumente veröffentlicht werden.

4. Meet-In-The-Middle
   Interface Beschreibung und Webservice existieren !
   Bei Meet-In-The-Middle gilt es zu dem Webservice eine passende Interfacebeschreibung zu finden. Durch einen sogenannten „Service-Wrapper“ wird die Verbindung zwischen der exisitierenden Anwendung und der Interfacebeschreibung hergestellt. Für die Veröffentlichung ist nur noch eine Implementierungsbeschreibung zu erstellen.

3.2.2. Service Nutzer

	Statisches Binden	Dynamisches Binden
Erstellen	static binding	build-time dynamic binding
Laufzeit	--	runtime dynamic binding

1. Static Binding
   Die statische Bindung gilt als die unflexibelste Möglichkeit einen Webservice zu benutzen. Während der Erstellung der Klientapplikation sucht sich der Nutzer die passende Implementationdefinition. Hierbei kann ein Service Proxy benutzt werden, welcher die implementierten Methoden mit den Interface Definitionen verbindet. Änderungen am Dienst oder deren Interfaces können später zu Problemen führen, wenn die statischen Aufrufe nicht mehr dem Schema entsprechen.

2. Build-Time Dynamic Binding
   Entweder sucht man eine Service Interface Definition per Registry oder einem anderen Verzeichnissdienst. Danach erstellt man einen generischen Service Proxy. Nun kann man die Requestor Applikation erstellen und den Service Proxy testen. Beim Start des Klienten wird als erstes die Dienstimplementation per Registry gesucht und danach an der angegebenen Netzwerkadresse der Dienst aufgerufen.

3. Runtime Dynamic Binding
   Bei der flexibelsten Möglichkeit, dem Runtime Dynamic Binding wird die Requestor Applikation mit einer „API“ erstellt. Der Service Proxy wird erst zur Laufzeit der Anwendung erstellt, um den Dienst zu binden. Somit wird die Suche nach einem Dienst per Registry, und der dazugehörigen Interface und Implementation Beschreibung, sowie die Bindung und der Aufruf an die angegebene Netzwerkadresse während der Laufzeit des Klienten durchgeführt.

3.3. Nutzung in der Wirtschaft

Momentan gelten Webservices als eine der Zukunftstechnologien. Deshalb sind Webservices etabliert und werden auch weiterhin den größten Teil des Internet Handels bestimmen. Dagegen ist die WSDL relativ neu und meist ungenutzt. Im Moment gibt es nur einige „große“ wie Amazon und Ebay die rudimentäre Ansätze von WSDL benutzen. Die meisten Service Anbieter werden evtl. ihre Aktiväten in Richtung plattformunabhängiger und interoperabler Anwendungen erweitern, ob mit WSDL bleibt abzuwarten. Was es jetzt schon gibt sind firmeninterne WSDL Verzeichnisse mit entsprechenden Anwendungen. Für die Entwickler könnten Kostensenkungen durchaus realistisch sein, da es genügend Tools für die Erstellung von Webservices mit WSDL

Beschreibung gibt. Die Softwarehersteller überschlagen sich mit integrierten Entwicklungsumgebungen, sei es IBM, Microsoft oder Sun, die es dem Entwickler sehr einfach machen, indem viele Prozesse per Tastendruck automatisch vorkonfiguriert werden.

4. Beispiele für die Einsatzmöglichkeiten von WSDL

4.1. In der Domäne – Versicherungen

4.1.1. Beteiligte / Rollen

1. Versicherungsgesellschaft

   Zum einen tritt die Versicherung als Dienstanbieter auf, doch in vielen Fällen ist eine Nutzung weiterer Dienste sinnvoll. Dabei können zusätzlich versicherungsfremde Wirtschaftszweige beteiligt sein.

2. Kunden

   Die Kunden selber nutzen hauptsächlich die Dienste der Versicherungen. Dabei werden, für den Kunden transparent, weitere Dienste involviert. Als Service Anbieter tretten die Kunden gegenüber den Versicherungen im Normalfall nicht auf.

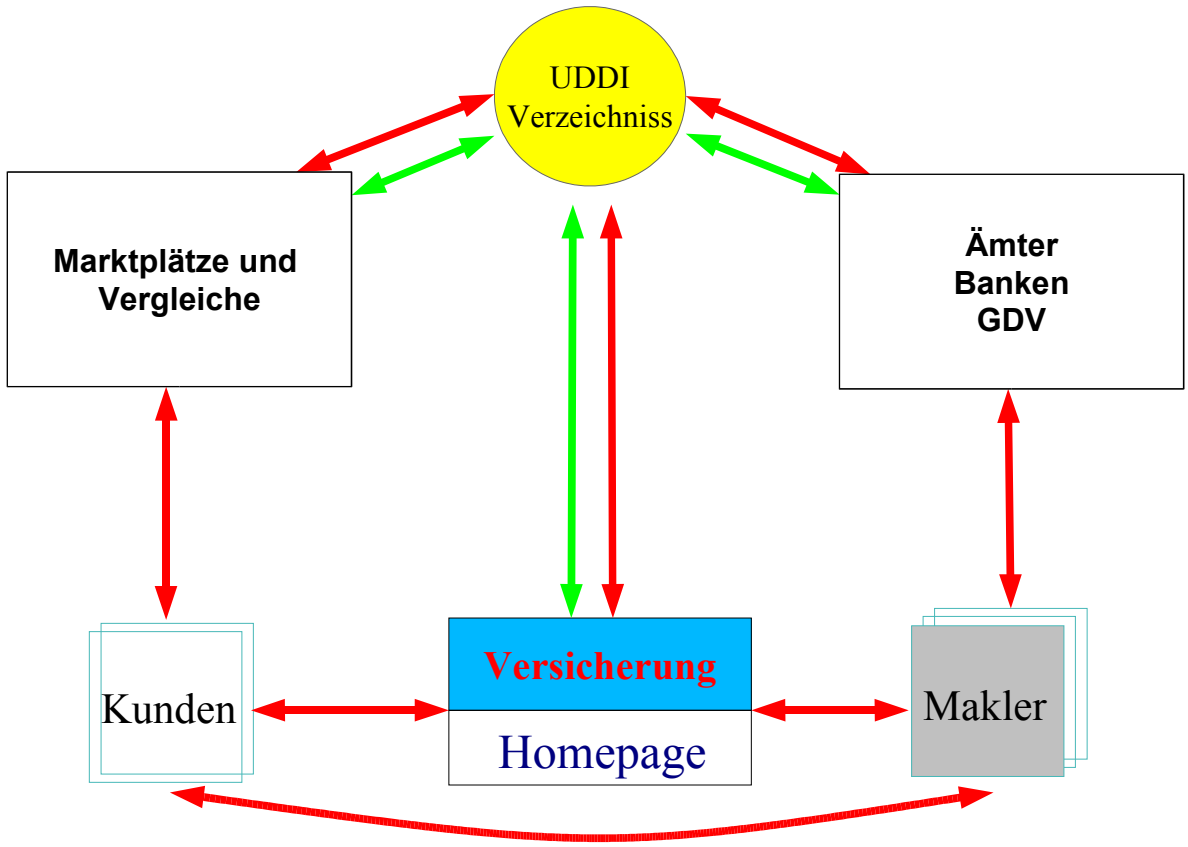
3. Makler & Partner

   Auch Makler und Partner der Versicherungen treten vorrangig als Dienstbenutzer auf. Aber im Gegensatz zu den Kunden bieten sie den Versicherungen auch eigene (Web)- Dienste an. (z.B. Kreditauskunft)

4. Portale & Marktplätze

   Die Portale und Marktplätze stehen oft zwischen den Versicherungen und den Kunden, da sie die Dienste der Versicherungen nutzen und selber den Kunden eigene Dienste anbieten.

4.1.2. Informationsflüsse



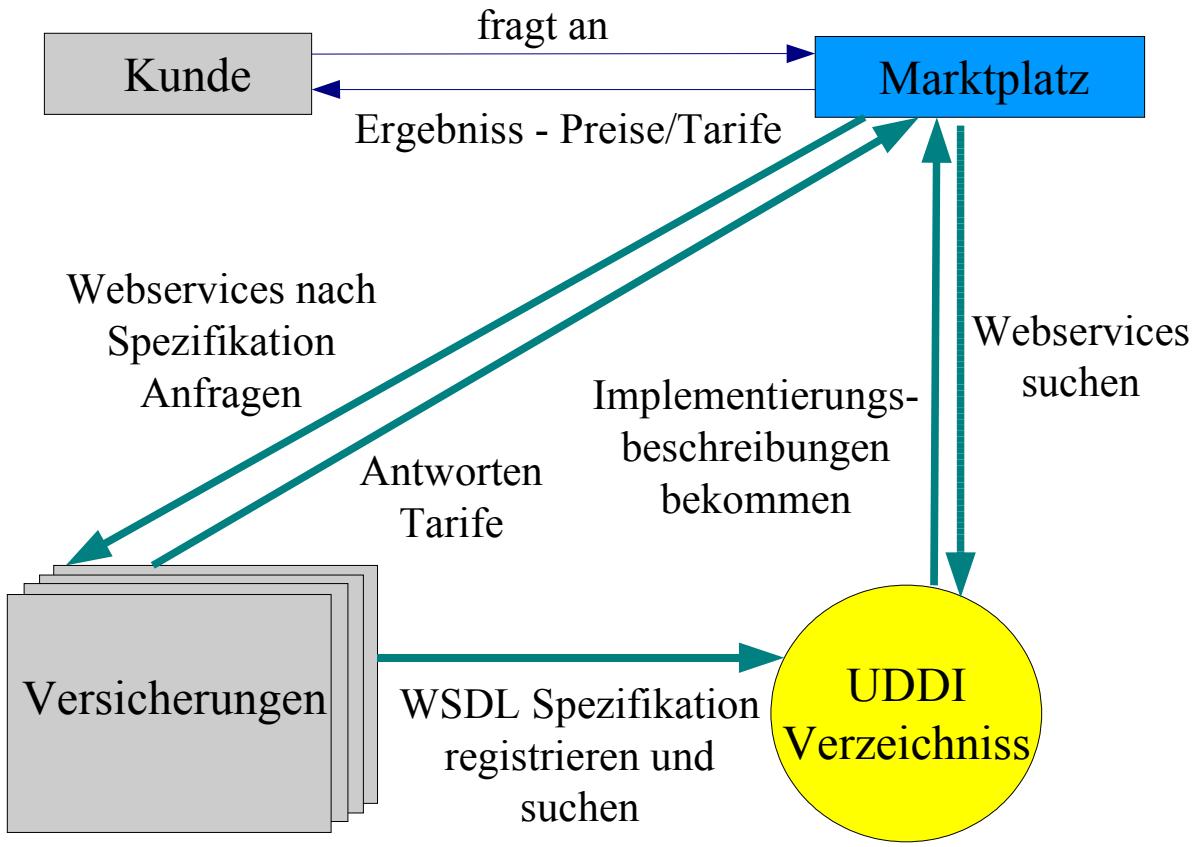
Die WSDL wird hauptsächlich zwischen den verschiedenen Service Anbietern benutzt. Hier kann durch die gewünschte Interoperabilität ein Zusammenarbeiten konkurierender Unternehmen und verschiedener Wirtschaftszweige erreicht werden. Im Mittelpunkt der WSDL unterstützten Informationsflüsse steht eine Registry, auf die alle beteiligten gleichberechtigt zugreifen können.

4.1.3. Dienste und Prozesse

Die typischen Kunden-, Marktplatz- und Partnerorientierten Prozesse sollen hier nur im Rahmen der WSDL Möglichkeiten betrachtet werden. Zu diesen Prozessen gehört der Vertragsabschluß zwischen Versicherung und Kunde, oder dem Makler und der Versicherung. Genauso die Beratung und die Bestandspflege können mit Webser-

vices absolviert werden. Auch Kündigungen, Schadensregulierungen und Vergleiche sind alltägliche Prozesse. Den Kunden ist es Möglich Informationen abzufragen, sowie Statistiken sich erstellen zu lassen. Alle diese Dienste können und werden bisher ohne WSDL als Webservices bereits angeboten. Doch hier wäre der Einsatz in Bezug auf verschiedene Gesellschaften und transparente Nutzung für Kunden und Vergleiche sinnvoll.

4.1.4. WSDL Dokumenten Einsatz am Marktplatz Beispiel



Am Anfang müssen die Versicherungsgesellschaften ihre WSDL Spezifikationen in der UDDI hinterlegen. Den Marktplätzen ist es somit möglich Dienste in der Registry zu suchen, zu nutzen und aufbereitet seinen Kunden anzubieten. Die Kunden selber bekommen von den transparenten Aufrufen der Marktplätze an die Versicherungen nichts mit. Für sie stellt der Marktplatz einen einzelnen Webservice da, den sie nutzen können.

4.1.5. WSDL Interface im Beispielszenario

Hier ein Beispiel Interface Grundgerüst für die Abfrage der Tarife einer Lebensversicherung.

```
<wsdl:definitions name="TarifeAbfrageLeben"
                  targetNamespace="http://www.insuranceX.com/wsdl/tarifleben">
<import weitere Namespaces und Schma Defintionen/>
<message name="getAngebotRequest">
  <part name="Beruf" type="xsd:string"/>
  <part name="Versicherungstyp" type="xsd:int"/>
weitere Attribute : Datum, Beitrag, Umfang, usw
</message>
<message name="getAngebotResponse">
  <part name="result" type="myxsd:InsuranceX.Angebot"/>
</message>
<portType name="LebenportType">
  <operation name="getAngebot">
    <input name="getAngebotRequest" message="tns:getAngebotRequest"/>
    <output name="getAngebotResponse" message="tns:getAngebotResponse"/>
  </operation>
</portType>
<binding name="LebenBindung" type="tns:LebenportType">
  <soap:binding ...>
  <operation name="getAngebot">
    <soap:operation ...>
     <input name="getAngebotRequest">
       <soap:body ...>
     </input>
     <output name="getAngebotResponse">
       <soap:body ...>
     </output>
  </operation>
</binding>
<binding ...>
...
</binding>
</wsdl:definitions>
```

Im Beispiel wurden zwei Nachrichten definiert, eine für die Anfrage und eine für die Antwort. In dem PortType „LebenportType“ werden die Nachrichten als „input“ und „output“ Messages definiert. Es folgt ein SOAP-binding und das Grundgerüst für das Interface Dokument ist fertig.

4.1.6. WSDL Implementation im Beispielszenario

Zu dem Interface Dokument fehlt noch das Implementation Dokument für das Beispiel, der Tarifabfrage. Der Webservice „LebenService“ wird mit dem binding „LebenBindung“ an eine URL gebunden, an welche die SOAP Anfragen geschickt werden müssen.

```
<wsdl:definitions name="Leben" targetnamespace="http://www....Leben">
<import weitere Namespaces und Schma Defintionen/>
  <service name="LebenService">
    <port name="LebenPort" binding="LebenBindung">
      <soap:address location="http://www.insuranceX.com/service/rpcrouter"/>
    </port>
  </service>
</wsdl:definitions>
```

4.1.7. Weitere Möglichkeiten

Es gibt viele Szenarien in welchen Webservices eingesetzt werden können, und somit auch die WSDL. Seien es, um Informationsdienste wie KFZ-Zulassung, Banken und Zinssätze abzufragen. Oder falls eine Gesellschaft den Vorversicherer erfahren muss.

Bei dem Marktplatzbeispiel könnte man noch Verträge abschliesen, dynamische Tarifabfragen oder sogar dynamischen Finden neuer Versicherungen einbauen. Der nächste Schritt wäre die Diensteintegration. Um zum Beispiel eine Reise zu buchen, oder ein Auto im Internet zu kaufen, wird während der Online Suche die passende Versicherungsprämie angezeigt. Hier gibt es noch etliche weitere Beispiele, inwieweit dabei die WSDL eine Rolle spielt ist Momentan nicht zu bestimmen, da es an Erfahrungen mangelt.

4.2. In der Domäne – Auktionen

4.2.1. Beteiligte und Dienste

Ähnlich wie bei den Versicherungen bieten die Auktionshäuser im Internet ihre Webservices den Kunden an. Das Auktionshaus bietet dabei eine Handelsplattform an,

und nutzt selber Fremddienste, wie z.B. Finnanzdienstleistungen. Den Kunden stehen mehrere Zugangsmöglichkeiten zur Auswahl. Die Nutzung übers Internet dominiert gegenüber den WAP und SMS Möglichkeiten. Neben den privaten Nutzern, gibt es viele professionelle Reseller die diese Plattformen nutzen. Zwischen den Finannzdienstleistern, den professionellen Resellern und den Auktionshäusern bestehen oft Kooperationen.
Zu den typischen Diensten eines Online Auktionshauses gehören das Anbieten von Waren und das Ersteigern dieser. Bilderdienste und Benachrichtigungsdienste erweitern den Umfang. Das Einbinden von Auktionen in andere Webseiten, um über den aktuellen Stand zu informieren oder Auktionen mit größten Interesse zu überwachen oder nach andere Kriterien, stellen ein weiteren Dienst da.
Dazu kommen die fremden Dienste, welche vom Auktionshaus genutzt werden können. Für die sichere Abwicklung eines Geschäfts besteht die Möglichkeit der Nutzung eines Treuhand Service. Hierbei soll das Risiko eines Betrugs, durch einen neutralen Dritten gemindert werden. Zusätzlich machen Versicherungen ab einen bestimmten Warenwert Sinn. Für die Bezahlung stehen zusätzliche Dienste bereit, wie PayPal. Und selbst der Transport der Waren, über einen Paketdienst, kann Online über einen Webservice veranlaßt werden.
Das war nur ein kurzer Überblick über die wichtigsten Teilnehmer und deren Dienste, aber man kann sich noch viele weitere Vorstellen und Umsetzen.

4.2.2. Beispiel Szenarien

Ein Händler der gerade Online etwas ersteigert hat, kann es im gleichen Moment über seinen Online Shop verkaufen, ohne im Besitz der Ware zu sein. Oder mit Hilfe von Werbebannern auf Webseiten kann direkt auf den Webservice des Auktionshauses zugegriffen werden. Und mit Auktionsüberwachung werden automatisch Preise abgefragt und bei Bedarf versendet.
Für registrierte Kunden gibt es User Profile, anhand denen angepasste Angebote empfohlen werden. Dazu kommt eine interaktive Versankostenberechnung anhand der Maße und dem Gewicht der Waren, sowie dem Land des Käufers.
In all diesen Webservice Szenarien, neben dem „normalen“ Versteigern und Ersteigern kann WSDL eingesetzt werden. Dadurch werden verschiedene Auktions-

häuser und Fremddienste, wie Paketdienst einfacher und schneller angebunden. Man braucht nicht für jeden und alles neue Schnittstellen bauen, sondern nuzt vorhandene Standards.

5. Zukunft der WSDL

Das die WSDL eine Zukunft hat, dafür sorgen allein schon die Softwarehersteller, wie Microsoft, IBM und SUN. Durch integrierte Tools und Plattformen zur Entwicklung von Webservices mit automatischer WSDL Integration wird es den Entwicklern auch sehr leicht gemacht. Allerdings sind Webservices keine neue Technologie, und die WSDL als IDL betrachtet ist auch keine Neu-Entwicklung. Es gab schon viele IDL's und auch schon mehrere alternative Architekturmodelle für plattformübergreifende verteilte Anwendungen, doch alleinig durchgesetzt hat sich bis jetzt noch kein System. So gibt es auch für die WSDL schwerwiegende Nachteile. Größter Kritikpunkt stellt der Overhead auf Grund der verwendeten XML Basis da. Dazu kommt die schwache bis nicht vorhandene Unterstützung von SLA's (service level agreements). Hier gibt es schon einige Workflow Sprachansätze, die das unterstützten. Dazu kommt der technisch bedingte Nachteil beim Übertragen von binären Anhängen. Es gibt auch keine Routingmöglichkeiten für die Nachrichten. Für Businessanwendungen fehlen Vorbereitungen für Garantien, Signaturen, Verschlüsselungen und Transaktionen. Man könnte versuchen das alles in die WSDL Dokumente einzubauen, doch würde da wieder der Standard fehlen und die Interoperabilität leiden. Hierfür gibt es Empfehlungen und Entwürfe von der OASIS und dem WS-I. Doch durch die verschiedenen Ansätze gibt es erneut gegensätzliche Entwicklungen, die nicht in allen Teilen kompatibel sein werden.
Trotzdem wird die WSDL den Erfolg von Webservices verstärken, denn ob mit oder ohne der wachsenden Begeisterung für Webservices steht nichts im Weg.
Für spezielle Bereiche und etablierte Systeme werden die alten Punkt-zu-Punkt Projekte bestehen bleiben und auch neue wird es immer wieder geben. Sei es aus performance Gründen oder fehlendem Nutzen einer weiten plattformübergreifenden Interoperabilität. Mehr Vorraussagen kann man im Moment nicht treffen, da die WSDL kaum Einsatz findet.

6. Stichwortverzeichnis

Dienst :
Ein Dienst ist die Instanziierung eines Webservices. Er ist als SW-Modul durch den Service-Provider realisiert und über ein Netzwerk aufrufbar. Der Webservice ist durch seine Interfaces beschrieben. Der Dienst zusätzlich durch seine Netzwerkadresse.

Dienst-Beschreibung :
Die Dienstbeschreibung enthält die Interface-Beschreibung, sowie die Beschreibung der Implementation. Hierzu zählen z.B. die Methoden-Definitionen, Datentypem oder die Netzwerkadresse. Es können auch Metadaten, wie Dienstkategorie enthalten sein, um die Suche durch den Service-Requestor zu erleichtern

SOAP :
Ist ein Protokoll für den Einsatz von XML-Dokumenten als Kommunikations-Nachrichten.

WSDL :
Eine in XML-format gehaltene Sprache um Webservices zu standardisieren und beschreiben, unabhängig vom Protokoll der Nachrichtenschicht (z.B. SOAP) und unabhängig von Datentypenrepräsentation.

UDDI :
Universal Description, Discovery and Integration

URL :
Uniform Resource Locator.

XML :
extensible Markup Lanquage

7. Literaturverzeichnis

1. W3C - World Wide Web Consortium
   www.w3c.org
   http://www.w3c.org/2002/ws/
   http://www.w3c.org/TR/wsdl12/

2. WebServices.ORG
   www.webservices.org
   http://www.webservices.org/index.php/article/articleview/596/1/24/
   http://webservices.org/index.php/article/archive/15/

3. Web Services Architect
   www.webservicesarchitect.com

4. OASIS – Cover Pages
   http://xml.coverpages.org

5. Web Service Interoperability Organization
   www.ws-i.org

6. IBM developerWorks
   http://www-106.ibm.com/developerworks/webservices/library/ws-intwsdl/
   http://www.research.ibm.com/journal/sj41-2.html

7. WebMethods.COM
   www.webmethods.com

8. Ebay
   www.ebay.de

9. Diplomarbeit „ Einsatz des UDDI-Standards für B2B-Systeme“ 2002
   Christian Weber, TU Ilmenau