

Web Services

Max Trense
uni@trense.info

17.01.2006

Inhaltsverzeichnis

1 Motivation	2
1.1 Verteilte Anwendungen	2
1.2 Möglichkeiten der Kommunikation	2
2 Serviceorientierte Architektur	3
2.1 Serviceorientierung	3
2.2 Merkmale eines Webservice	3
2.2.1 Provider	4
2.2.2 Broker	4
2.2.3 Consumer	4
3 Standards	5
3.1 WSDL – Web Service Description Language	5
3.2 UDDI – Universal Description, Discovery and Integration	6
3.3 SOAP	7
3.4 WS-Security	7
3.5 WSFL – Web Services Flow Language und WS-BPEL – Business Process Execution Language	8

1 Motivation

1.1 Verteilte Anwendungen

Software, die heute entwickelt wird, ist immer häufiger darauf angewiesen mit anderer Software über Netzwerke zu kommunizieren. Dieser Kommunikationsprozess gestaltet sich in der Regel schwierig, da die beteiligte Software nicht nur meist auf unterschiedlichen Rechnern läuft, sondern es sich in vielen Fällen sogar um unterschiedliche Software handelt. In einer solchen Anwendung ist es von entscheidender Bedeutung, dass jede beteiligte Software die Daten, die ausgetauscht werden, syntaktisch und semantisch korrekt verarbeiten kann.

Unternehmensanwendungen – also Anwendungen, die dazu dienen, die Prozesse innerhalb eines Unternehmens zu unterstützen und zu steuern – werden oft in

- In-Business
- Business-To-Business
- Business-To-Consumer

unterschieden.

Business-To-Business-Anwendungen verbinden Prozesse und Strukturen eines Unternehmens, mit denen eines anderen. Sie müssen daher oft unterschiedliche strukturierte Software und Datenbestände miteinander verbinden.

1.2 Möglichkeiten der Kommunikation

Verteilte Anwendungen müssen über Netzwerke miteinander kommunizieren. Um diese Kommunikation zu realisieren, verfolgt man verschiedene Ansätze:

Spezialisierte Kommunikationsprotokolle werden oft eingesetzt, wenn Performance eine überragende Rolle spielt oder die Kommunikation sehr spezifisch an die Anwendung angepasst werden muss.

Verteilte Methodenaufrufe abstrahieren von der zugrundeliegenden Kommunikationsform und stellen die Möglichkeit bereit, Methoden in einer entfernten Software aufzurufen.

CORBA wird oft im professionellen Umfeld eingesetzt. CORBA stellt bereits viele Merkmale zur Verfügung, die sich auch bei Webservices wiederfinden.

2 Serviceorientierte Architektur

2.1 Serviceorientierung

Dienste (Services) sind Sammlungen von zusammengehörigen Objekten und Methoden. Jeder Dienst stellt eine bestimmte Funktionalität zur Verfügung. Diese Funktionalität ist komplett gekapselt, so dass nach Aussen nur die Dienstschnittstelle sichtbar ist. Im Unterschied zu Objekten der Objektorientierung ist ein Dienst in der Serviceorientierung grob granular. Der Dienst stellt also alle Methoden bereit, die für eine bestimmte Funktionalität notwendig sind.

Dienste werden formal beschrieben. Diese Beschreibung erlaubt es, die zur Verfügung gestellten Fähigkeiten eines Dienstes zu erfassen. Ausserdem enthält sie die Aufrufkonvention für diesen Dienst. Damit ist ein Aufrufer in der Lage, seine Anfrage so zu formulieren, dass der Dienstanbieter in der Lage ist, sie zu verstehen. (vgl. [6, S. 33 ff])

Serviceorientierung zeichnet sich vor allem durch die folgenden Merkmale aus (vgl. [6, S. 38 ff]):

Dynamische Lokalisierung Es ist möglich, Dienste anhand verschiedener Merkmale zu lokalisieren. Es ist dazu nicht notwendig, den speziellen Dienst bereits vorher zu kennen.

Grob granular Im Unterschied zur Objektorientierung sind Dienste grob granular. Das bedeutet, dass ein Dienst sämtliche Funktionalität kapselt, die für seine Ausführung notwendig ist.

Lose Kopplung Dienste sind untereinander und mit den Dienstabnehmern lose gekoppelt. Es muss also zur Entwicklungszeit nicht bekannt sein, mit welchem Dienst eine Software später einmal kommunizieren soll.

Lokationstransparenz Ein Dienst ist unabhängig davon, an welchem Ort er angeboten wird. Es ist also möglich einen Dienst über ein Netzwerk aufzurufen.

Wiederverwendbarkeit Dienste können durch Komposition wiederverwendet werden. Es ist so möglich, Dienste nacheinander oder parallel auszuführen. Ausserdem können Dienste zur Erfüllung ihrer Aufgabe auf andere Dienste zurückgreifen.

2.2 Merkmale eines Webservice

Webservices sind durch die WS-I (Webservice Interoperability Organization) [1] standardisiert. Die Spezifikation sieht eine Implementation der Serviceorientierten Architektur vor.

Abbildung 1 zeigt den prinzipiellen Aufbau eines Webservice. Dabei gilt zu beachten, dass Provider, Broker und Consumer Rollen sind, die in einer Anwendung üblicherweise von unterschiedlicher Software wahrgenommen werden. (vgl. [6, S. 37 ff])

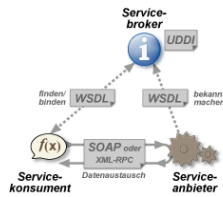


Abbildung 1: Aufbau eines Webservice (Quelle: wikipedia.de)

Zur Kommunikation zwischen diesen Peers dient das SOAP-Protokoll. Die Beschreibung der bereitgestellten Dienste wird durch WSDL-Dokumente ermöglicht. UDDI stellt im Idealfall eine Möglichkeit bereit, über Registrierungen nach Diensten zu suchen. Alle diese Protokolle werden im nächsten Abschnitt erläutert.

2.2.1 Provider

Der Provider stellt sämtliche Anwendungsfunktionalität zur Verfügung. Er veröffentlicht WSDL-Dokumente, die alle von ihm bereitgestellten Dienste beschreiben, über den Broker.

Der Provider verwaltet alle Abhängigkeiten, der von ihm zur Verfügung gestellten Dienste. Diese können sich wiederum auf andere Dienste beziehen oder auf sogenannte Legacy-Software. Im ersten Fall wird der Dienstanbieter so zum Dienstabnehmer. Der zweite Fall ist besonders in Umgebungen interessant, in denen Dienste als Schnittstelle zu schon vorhandener Funktionalität genutzt werden sollen. Oft kommt es zum Beispiel vor, dass bestehende Programme weiterhin die Daten verwalten, die aber über Webservices publiziert oder weiterverarbeitet werden sollen.

2.2.2 Broker

Der Broker stellt eine zentrale Datenbank zur Verfügung, um Dienste mit ihrer Beschreibung zu erfassen und auf Suchanfragen zu antworten. Der Broker stellt also gewissermaßen ein Kontaktverzeichnis für Webservices bereit.

Da die automatische Verarbeitung einer solchen Suchanfrage kompliziert ist, sind Broker noch nicht weit verbreitet. Die Funktionalität, die Broker bislang zur Verfügung stellen beschränkt sich auf die Suche von vordefinierten Namensräumen, die jeweils durch Vereinbarung eine bestimmte Art von Funktionalität benennen und auf die Suche von Schlüsselwörtern.

Nicht selten werden auch Broker als normale Webservices von einem Provider zur Verfügung gestellt.

2.2.3 Consumer

Der Consumer kann zuerst auf die Suchfunktionalität des Brokers zugreifen. Dieser gibt eine Liste von verfügbaren Webservices zurück, die der Consumer nutzt, um den pas-

senden Webservice zu Finden. Kennt der Consumer den Provider, der den gewünschten Webservice zur Verfügung stellt, wird er den Provider kontaktieren und eine Abfrage abschicken, die der Provider entweder beantwortet, oder eine Fehlermeldung zurückgibt.

Es ist Aufgabe des Consumers die zurückgelieferten WSDL-Dokumente zu verarbeiten, um eine korrekte Abfrage stellen zu können.

3 Standards

3.1 WSDL – Web Service Description Language

Die Web Service Description Language erlaubt eine formale Beschreibung von Diensten. Listing 1 enthält ein Beispiel für ein WSDL-Dokument, welches den Administrations-Dienst des Apache Axis Containers beschreibt.

Listing 1: Beschreibung eines Webservice

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <wsdl:definitions [...] >
3   <wsdl:types>
4     <schema targetNamespace="http://xml.apache.org/axis/wsdd/" xmlns="http://www.w3.org/2001/XMLSchema">
5       <element name="AdminService" type="xsd:anyType"/>
6       <element name="AdminServiceReturn" type="xsd:anyType"/>
7     </schema>
8   </wsdl:types>
9
10  <wsdl:message name="AdminServiceRequest">
11    <wsdl:part element="impl:AdminService" name="part"/>
12  </wsdl:message>
13
14  <wsdl:message name="AdminServiceResponse">
15    <wsdl:part element="impl:AdminServiceReturn" name="AdminServiceReturn"/>
16  </wsdl:message>
17
18  <wsdl:portType name="Admin">
19    <wsdl:operation name="AdminService">
20      <wsdl:input message="impl:AdminServiceRequest" name="AdminServiceRequest"/>
21      <wsdl:output message="impl:AdminServiceResponse" name="AdminServiceResponse"/>
22    </wsdl:operation>
23  </wsdl:portType>
24
25  <wsdl:binding name="AdminServiceSoapBinding" type="impl:Admin">
26    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
27    <wsdl:operation name="AdminService">
28      <wsdlsoap:operation soapAction=""/>
29      <wsdl:input name="AdminServiceRequest">
30        <wsdlsoap:body use="literal"/>
31      </wsdl:input>
32      <wsdl:output name="AdminServiceResponse">
33        <wsdlsoap:body use="literal"/>
34      </wsdl:output>
35    </wsdl:operation>
36  </wsdl:binding>
37
38  <wsdl:service name="AdminService">
39    <wsdl:port binding="impl:AdminServiceSoapBinding" name="AdminService">
40      <wsdlsoap:address location="http://localhost.localdomain:8080/axis/services/AdminService"/>
41    </wsdl:port>
42  </wsdl:service>
43
44 </wsdl:definitions >
```

Eine Beschreibung gemäß dem WSDL-Standard enthält folgende Definitionen:

Datentypen (Zeilen 3-8) Enthält die Beschreibung aller verwendeter Datentypen. Diese Definition folgt den gleichen Regeln wie sie bei XML-Schema Anwendung finden.

Nachrichten (Zeilen 10-16) Beschreibt alle Nachrichten, die verwendet werden.

Porttypen (Zeilen 18-23) Definiert den Typ jedes einzelnen Ports. Folgende Typen sind möglich:

One-way Der Service bekommt eine Nachricht vom Consumer, beantwortet diese jedoch nicht.

Request-response Der Service bekommt eine Nachricht vom Consumer und beantwortet diese mit einer weiteren Nachricht.

Solicit-response Der Server sendet eine Nachricht zum Consumer und wartet auf eine Antwort.

Notification Der Server sendet eine Nachricht zum Consumer und erwartet keine Antwort.

Bindung (Zeilen 25-36) Beschreibt das Protokoll und alle formalen Aspekte (z.B: Zeichensatz, etc.) der Kommunikation.

Port (Zeilen 39-41) Definiert eine URI unter der diese Methode aufrufbar ist.

Service (Zeilen 38-42) Fasst eine Menge von Ports zu einem Dienst zusammen.

Hier sind Ports die Webservice-spezifischen Methoden, die von einem Consumer im Laufe einer Sitzung aufgerufen werden können.

3.2 UDDI – Universal Description, Discovery and Integration

UDDI ist ein Standard, um Beschreibungen von Webservices in einer Registrierung zu erfassen und anhand bestimmter Kriterien danach zu suchen. Der Eintrag in einem UDDI-Register enthält neben den notwendigen Daten über die Art des Zugriffs (beispielsweise über eine Referenz auf ein zugehöriges WSDL-Dokument) auch nicht-operationelle Eigenschaften des Dienstes. So ist es beispielsweise möglich, Daten über das Unternehmen bereitzustellen, welches den Dienst anbietet. Auch textuelle Beschreibungen sind möglich.

UDDI ist darauf ausgelegt, neben den operationellen Attributen eines Webservice (die durch das WSDL-Dokument abgebildet werden) auch weitere Attribute, wie beispielsweise Verfügbarkeit, Kosten eines Aufrufs oder ähnliches bereitzustellen.

Leider wird UDDI noch recht wenig genutzt. Die wenigen Anbieter von UDDI-Registries (siehe z.B: [4], [5]) haben fast alle ihre Dienstleistung eingestellt. Es bleibt zu hoffen, dass sich Registries in der Art von UDDI in Zukunft durchsetzen werden, da sie eine wichtige Grundlage für die Verbreitung und Nutzung von Webservices bilden.

3.3 SOAP

SOAP ist ein Protokoll, das verwendet wird, um Nachrichten zwischen einzelnen Peers auszutauschen. SOAP-Nachrichten sind XML-Dokumente, die einem vorgegebenen Format genügen. Sie bestehen in der Regel aus:

Envelope Kapselt Header und Body der Nachricht und enthält Daten über den Sender und den Empfänger

Header Beschreibt die Daten, die in dieser Nachricht enthalten sind

Body Enthält die eigentlichen Nutzdaten

Im Konzept der Webservices werden SOAP-Nachrichten benutzt, um Anfragen an einen Dienst zu stellen.

Listing 2 und 3 zeigen Beispiele für eine Anfrage und eine Antwort über das SOAP-Protokoll.

Listing 2: Beispiel einer SOAP-Anfrage

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
3   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
5   <soapenv:Body>
6     <ns1:validate soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
7       xmlns:ns1="urn:Calculator">
8       <arg0 xsi:type="xsd:integer">123</arg0>
9       <arg1 xsi:type="xsd:integer">321</arg1>
10    </ns1:validate>
11  </soapenv:Body>
12 </soapenv:Envelope>
```

Listing 3: Beispiel einer SOAP-Antwort

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
3   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
5   <soapenv:Body>
6     <ns1:validateResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
7       xmlns:ns1="urn:sampleDomain">
8       <result xsi:type="xsd:integer">444</result>
9     </ns1:validateResponse>
10  </soapenv:Body>
11 </soapenv:Envelope>
```

In der Antwort auf eine SOAP-Anfrage kann auch eine Fehlermeldung enthalten sein. Diese gibt Aufschluss darüber, welcher Teil der Anfrage nicht bearbeitet werden konnte.

3.4 WS-Security

Diese Erweiterung erlaubt es, SOAP-Nachrichten zu signieren und zu verschlüsseln. Ausserdem ist es möglich den Absender und den Empfänger einer Nachricht zu authentifizieren. Gewöhnlich kommen X.509 Zertifikate oder Kerberos zum Einsatz.

Diese Erweiterung findet vor allem bei kostenpflichtigen oder sensiblen Diensten Anwendung.

3.5 WSFL – Web Services Flow Language und WS-BPEL – Business Process Execution Language

Die Web Services Flow Language erlaubt es, Geschäftsprozesse durch Verkettung von Diensten auszudrücken. WSFL bedient sich eines gerichteten Graphen als Modell für die Abbildung von Prozessen. Durch die Standardisierung bietet WSFL eine portable Möglichkeit, Geschäftsprozesse zu Modellieren.

Die Business Process Execution Language ist eine Weiterentwicklung von WSFL. Über die Fähigkeiten von WSFL hinaus ist es mit WS-BPEL möglich, konsistent innerhalb der Programmiersprache Prozesse zu kaskadieren. WS-BPEL unterstützt ausserdem Laufzeitbereiche, in denen es zum Beispiel möglich ist, lokale Variablen oder Funktionsblöcke zu definieren.

Es existieren bereits einige Workflow-Engines, die in der Lage sind WSFL und WS-BPEL zu interpretieren. (z.B: [7])

Literatur

- [1] Webservice Interoperability Organization: <http://www.ws-i.org>
- [2] UDDI Homepage des OASIS Consortium: <http://www.uddi.org>
- [3] Spezifikation des SOAP 1.2 Standards: <http://www.w3.org/TR/soap12-part1/> und <http://www.w3.org/TR/soap12-part2/>
- [4] UDDI-Testregistrierung von IBM: <https://uddi.ibm.com/testregistry/publish>
- [5] UDDI-Suche: <http://www.soapclient.com/uddisearch.html>
- [6] Wang, Dapeng; et.al: Java Web Services mit Apache Axis. Software Support Verlag, 2004
- [7] Oracle BPEL Process Manager: <http://www.oracle.com/technology/products/ias/bpel/index.html>