

Beschreibung aktueller technologischer Standards bei der Umsetzung von Web-Services

Alexander Rudolf

Fakultät für Informatik
Hochschule Mannheim
Paul-Wittsack-Straße 10
68163 Mannheim

`alexander.rudolf@hs-mannheim.de`

Zusammenfassung Sucht man heute nach einer eindeutigen Beschreibung des Begriffs 'Web-Service', so wird man eine Fülle von Definitionen finden, welche sich grob in zwei Bereiche untergliedern lassen: Einzelne kurze Definitionen[1], welche zwar treffend die Kernaufgabe von Web-Services beschreiben, jedoch in keiner Weise auf aktuelle Architekturen, Konzepte und Problemstellungen eingehen. Zum anderen findet man sehr konkrete Ausarbeitungen, welche aber zum Großteil nur einzelne Konzepte fokussieren und Alternativlösungen kaum berücksichtigen (Beispiel: [2]). Die hier beschriebenen Konzepte sollen einen Überblick über die Thematik vermitteln und so eine Brücke zwischen diesen beiden Bereichen schlagen. Es werden die am häufigsten verwendeten Konzepte zur Umsetzung von Web-Services erläutert, ohne jedoch auf konkrete programmiertechnische Einzelheiten einzugehen.

1 Einleitung

1.1 Motivation: Warum Web-Services?

Seit dem letzten Jahrzehnt hat sich der Zugang zu Informationen und Wissen stark verändert: Anstatt das Wissen in Bibliotheken und wissenschaftlichen Archiven zu lagern, ist es uns möglich über das Internet in Sekundenschnelle die benötigten Informationen zu erhalten. Dabei helfen uns Internetportale und Suchmaschinen wie zum Beispiel Google oder Amazon. Doch wie genau funktioniert der Zugriff auf Informationen? Was macht diese 'Dienstanbieter' zu dem, was sie sind? Die Bezeichnung 'Dienstanbieter' ist hier nicht zufällig gewählt, denn genau genommen ist es das, was Google usw. für uns tun: Sie bieten uns ihre Dienste an. Doch wie genau funktioniert der Zugriff auf einen solchen Dienst? Um diese Frage zu klären, unterscheiden wir zwischen zwei verschiedenen Arten, um auf die angebotenen Dienste zuzugreifen: Zum einen die uns Privatanwendern bekannte Art, als Mensch einen maschinellen Dienst wie zum Beispiel die Google-Suche zu nutzen. Zum anderen führte die zunehmende Automatisierung der Industrie- und Geschäftswelt dazu, dass auch der Zugriff auf durch Computer

unterstützte Dienste automatisiert von statten gehen musste. Eine Möglichkeit, diese Maschine-Maschine-Kommunikation zu realisieren, bietet der Einsatz von Web-Services.

Der Gedanke auf Funktionen und Programme über ein Netzwerk (zum Beispiel das Internet) zuzugreifen ist dabei nicht neu: Bisherige Lösungen sind unter anderem CORBA, DCOM und Java RMI. Jedoch wachsen die Anforderungen an Skalierbarkeit, Sicherheit und Plattformunabhängigkeit bei entfernten Funktionsaufrufen mit der Komplexität der Aufgaben. Problematisch hierbei ist, dass zum Beispiel die Anpassung der Firewalls an das von CORBA verwendete Protokoll in großen Firmen oftmals als sicherheitskritisch eingestuft wird. Ein weiteres Problem taucht bei der Verwendung von Java RMI auf: Dieser Ansatz basiert vollständig auf Java und kann daher nur in Java-Umgebungen verwendet werden.

Die hier beschriebenen Web-Services stellen eine alternative Lösung zu den genannten Ansätzen dar. Anhand der Erläuterungen der Basistechnologien wird klar, weshalb Web-Services die Probleme bezüglich Sicherheit und Unabhängigkeit beheben können, aber auch, wo die Grenzen von Web-Services liegen.

1.2 Definitionsansätze

Da mit wachsender Komplexität der Anforderungen an Web-Services auch die genutzten Technologien weiter ausgebaut werden, ist es nicht leicht, die Funktionsweise von Web-Services sowohl treffend, als auch allumfassend zu beschreiben. Das 'World-Wide-Web Consortium (W3C) Glossary [3]' gilt als Standard für Definitionen bezüglich Web-basierten Diensten und beschreibt Web-Services wie folgt: 'A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.' Diese Definition mag zwar treffend die Funktionsweise beschreiben, ist aber für den ersten Kontakt mit Web-Services schwer zu verstehen. Nach Meinung des Autors bilden die folgenden 3 Definitionsansätze in Summe eine verständliche Basis um Web-Services zu verstehen:

Ein Web-Service...

- ist ein Programm, das über das Internet von einem anderen Programm plattform-, sprach- und objektmodellunabhängig aufgerufen werden kann.
- sind verteilte, lose gekoppelte und wiederverwendbare Software-Komponenten, auf die über Standard-Internetprotokolle programmatisch zugegriffen werden kann.
- ist ein Dienst, der mittels SOAP angesprochen werden kann und (meist) mittels WSDL beschreiben wird.

1.3 Historischer Überblick über die Entstehung von Web-Services

Abbildung 1 zeigt die Entwicklung der eingesetzten Technologien bei der Umsetzung von entfernten Funktionsaufrufen (Remote Procedure Call, RPC). Die x-Achse stellt dabei den zeitlichen Verlauf dar, die y-Achse signalisiert die wachsenden Anforderungen im Verhältnis zur Zeit. Den Grundstein für die Entwicklung entfernter Funktionsaufrufe legten die Einzelplatzsysteme, welche zu Beginn durch Assamblcode programmiert wurden. Zusammen mit der Vernetzung solcher Einzelplatz-Systeme kam der Wunsch auf, auf entfernten Rechner Funktionen auszulösen. Die ersten Überlegungen führten 1976 zum RPC. Dieser erlaubte es, Funktionen über das Netz auszulösen und einen erwarteten Wert zurückzubekommen. Im Zuge der aufkommenden objektorientierten Programmierung entstanden mit CORBA, DCOM und Java RMI Möglichkeiten, nicht nur Funktionen auszulösen, sondern auch ganze Objekte verteilt anzusprechen. Um die im Motivationskapitel genannten Probleme bezüglich Sicherheit und Plattformabhängigkeit zu lösen, entwickelte sich der zu diesen Architekturen alternative Ansatz: Die Web-Service Architektur.

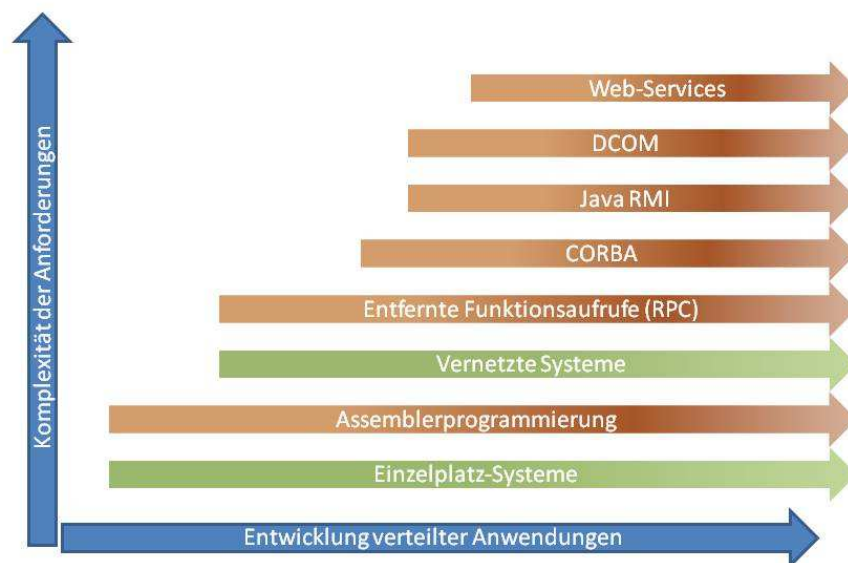


Abbildung 1. Historie von Web-Services

2 Beschreibung der Basistechnologien

2.1 Die Architektur von Web-Services

Die im Folgenden erläuterte Architekturbeschreibung bietet einen groben Überblick über die einzelnen Komponenten, Rollen und Technologien bei der Umsetzung von Web-Services, und enthält daher einige Fachbegriffe, welche erst im späteren Verlauf dieser Ausarbeitung detailliert beschreiben werden.

Die theoretische Umsetzung von Web-Services erfordert 3 unterschiedliche Rollen: Den Web-Service Anbieter (1), welcher den Web-Service erstellt und diesen im so genannten UDDI-Verzeichnis(2) publiziert. Der Web-Service Nutzer(3) hat anschließend die Möglichkeit, einen für ihn passenden Web-Service im UDDI-Verzeichnis zu suchen. Er erhält dann eine Referenz auf die Schnittstellenbeschreibung (WSDL) und kann anschließend den Web-Service aufrufen. Funktionsaufrufe und der Datenaustausch erfolgen über das für Web-Services übliche Protokoll SOAP. Das UDDI-Verzeichnis, die WSDL-Beschreibung und SOAP-Nachrichten basieren auf der Beschreibungssprache XML. Den Zusammenhang der hier beschriebenen Rollen zeigt Abbildung 2.

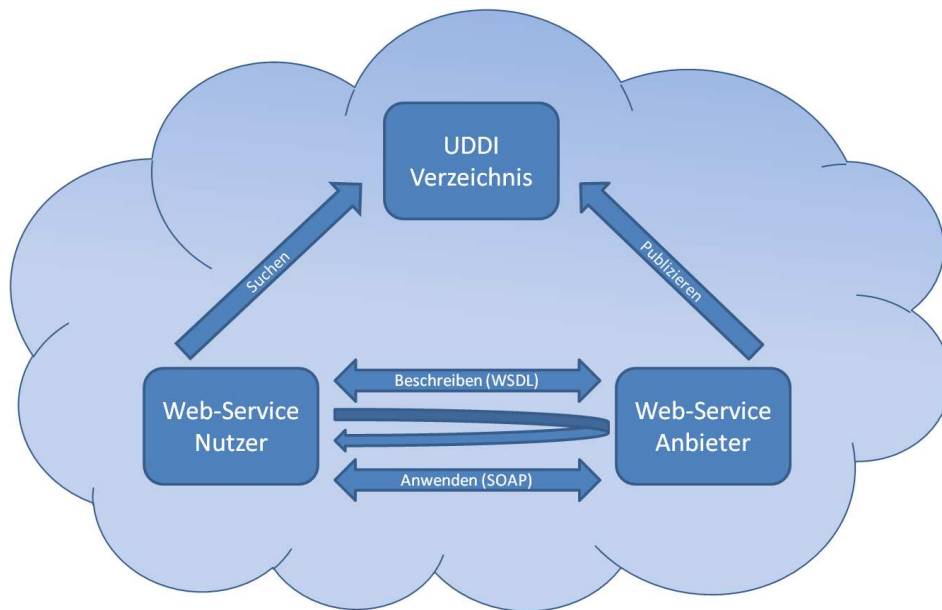
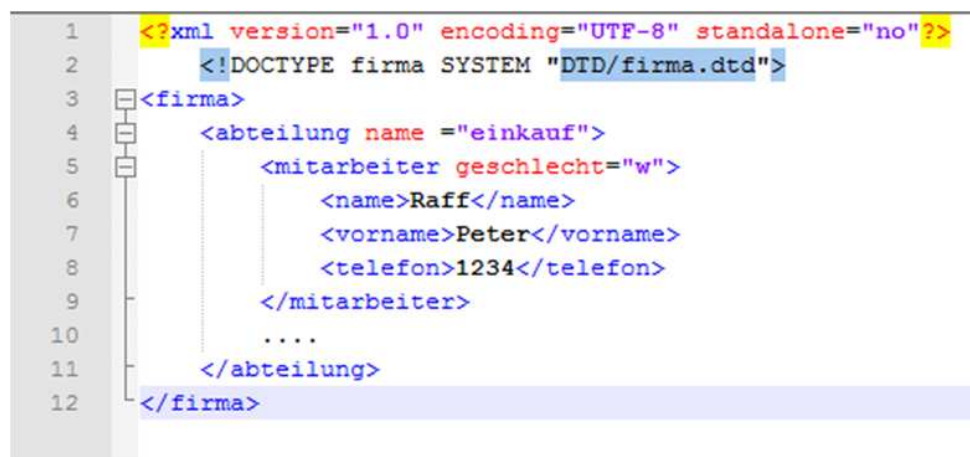


Abbildung 2. Architektur von Web-Services

2.2 Basistechnologie: XML

XML gilt als standardisierte Strukturierungs- und Auszeichnungssprache und wird von nahezu allen Plattformen als Sprache zur Strukturierung und zum Transport von Daten unterstützt. Der Aufbau ist dabei simpel: In einer Baumstruktur angeordnete Elemente (sog. Tags) dienen als Strukturierungsform. Die Einzelnen Elemente können dabei durch Attribute angepasst werden, welche wiederum durch eine Wertzuweisung definiert werden. Als Wurzelement dient ein so genanntes XML-Tag, welches die XML-Version, sowie den verwendeten Zeichensatz festlegt. Gefolgt wird dieses Tag von einer Referenz auf eine XML-Sprachdefinition (Document Type Definition, oder XML-Schema), in welcher die verwendeten Auszeichnungselemente definiert sind. Durch diese Sprachdefinition lässt sich die mächtige XML-Sprache einschränken und somit eine neue, eigene Sprache definieren. Weitere Möglichkeiten in XML bieten zum Beispiel



```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE firma SYSTEM "DTD/firma.dtd">
3 <firma>
4   <abteilung name="einkauf">
5     <mitarbeiter geschlecht="w">
6       <name>Raff</name>
7       <vorname>Peter</vorname>
8       <telefon>1234</telefon>
9     </mitarbeiter>
10    ...
11  </abteilung>
12 </firma>
```

Abbildung 3. XML Beispiel

die Namespaces. Diese erlauben unter Angabe von weiteren Sprachdefinitionen das Mischen von verschiedenen XML-Sprachen innerhalb einer einzigen XML-Datei. Außerdem existiert durch die oben beschriebenen Sprachdefinition mittels XML-Schema die Möglichkeit komplexe Datenstrukturen in XML zu beschreiben, welche selbst wiederum unter Angabe von nativen Datentypen wie zum Beispiel Zeichenketten (type='String') beschreiben werden können.

2.3 Basistechnologie: SOAP

Das 1999 aus dem Übertragungsstandard 'XML-RPC' entstandene SOAP (ursprünglich: Simple Object Access Protocol) wird im Bereich der Web-Services zur

Übertragung von Daten zwischen dem Web-Service Nutzer und dem Web-Service anbieter verwendet. Das Protokoll basiert, wie sämtliche Web-Service Technologien, auf dem XML-Standard. Die Struktur von SOAP-Nachrichten besteht aus einem SOAP-Envelope, welcher die 2 Grundelemente - den SOAP Header und den SOAP-Body - einschließt. Der optionale SOAP-Header beschreibt hierbei Authentifizierungselemente, bzw. Metadaten über die im SOAP-Body enthaltenen eigentlichen Datenformate. Man unterscheidet 3 SOAP-Nachrichtentypen:

- SOAP-Request: Anfrage vom Web-Service Nutzer
- SOAP-Response: Antwort vom Webservice Anbieter
- SOAP-Fault: Fehlermeldung vom Web-Service Anbieter



Abbildung 4. Beispiel: SOAP-Request

Das in Abbildung 4 skizzierte Beispiel zeigt den Aufbau eines einfachen SOAP-Requests, welches die Funktion 'sayHello' auf der Serverseite anspricht. Nach einer serverseitigen Bearbeitung erfolgt die Rückgabe eines SOAP-Responses bei erfolgreicher Ausführung der Funktion (inklusive dem Rückgabewert der Funktion), bzw. eines SOAP-Faults, bei fehlerbehafteter Ausführung der Funktionsanweisung.

2.4 Basistechnologie: WSDL

Die ebenfalls auf XML basierende Beschreibungssprache für Web-Services WSDL (Web Service Description Language) dient dazu, die Schnittstellen und damit die Zugriffspunkte von Web-Services eindeutig zu definieren, damit Web-Service Nutzer darauf zugreifen können. Der Aufbau einer WSDL-Beschreibung gestaltet sich daher wie folgt:

- types-Elemente: In den types-Elementen werden die vom Web-Service verwendeten Datentypen definiert.
- message-Elemente: Die message-Elemente beschreiben die einzelnen Funktionen und Operationen, welche der Web-Service unterstützt.
- portType-Elemente: PortType-Elemente dienen dazu, Schnittstellen für zusammengehörige Operationen zusammenzufassen.
- binding-Elemente: Implementierungsdetails inklusive der Protokollbeschreibungen.
- service-Elemente: Technische Zugriffspunkte (Schnittstellen und Ports).

2.5 Basistechnologie: UDDI

Um auf Web-Services zugreifen zu können, entstand im Jahr 2000 der Begriff UDDI (Universal Description, Discovery and Integration). UDDI stellt einen - ebenfalls auf XML basierenden - Verzeichnisdienst für Web-Services dar. Anhand von verschiedenen Indikatoren sollten Web-Service-Suchende (also die späteren Web-Service Nutzer) automatisiert nach geeigneten Web-Services suchen können. Als Gegenstück sollte es möglich sein, dass Web-Service Anbieter ihre Services im Verzeichnisdienst veröffentlichen können. UDDI unterscheidet hierbei folgende Komponenten, welche nach den vielgenutzten Telefonbüchern benannt sind:

- Die 'White Pages' enthalten Informationen über die Service-Anbieter inklusive der Kontaktinformationen.
- Die 'Yellow Pages' enthalten die Services, kategorisiert nach dem jeweiligen Anwendungsbereich, bzw. Geschäftsfeld.
- Die 'Green Pages' enthalten eine 'Menschen-lesbare' Schnittstellenbeschreibung.
- Die 'Service Type Registration' enthält angelehnt an die WSDL-Beschreibung eine 'Maschinen-lesbare' Schnittstellenbeschreibung

3 Die Anwendung von Web-Services in der Praxis

Das in Abbildung 2 dargestellte Architekturmodell zeigt den theoretischen Aufbau einen Web-Service-Umgebung. Doch die reale Umsetzung folgt in der Regel einem anderen Ansatz: Die meisten öffentlich zugänglichen Web-Services werden ohne die Verwendung von UDDI vermarktet. Dies liegt zu einem großen Teil an der Tatsache, dass die einzelnen Web-Services schon bekannt sind und sie daher nicht mehr über einen Verzeichnisdienst gesucht werden müssen, sondern man direkt beim Anbieter sämtliche Zugriffsdaten erhält. Das bedeutet, die Web-Service suche folgt eher einem manuellen als einem automatisierten automatisierten Ansatz. Dieser Umstand führte dazu, dass das damals größte UDDI Verzeichnis (UDDI Business Registry (UBR)), gegründet von Microsoft, IBM, SAP, et al.) im Jahr 2005 heruntergefahren wurde.

4 RESTful Web-Services

Neben der in dieser Ausarbeitung beschriebenen Umsetzung von Web-Services (auch 'Big Web-Services' genannt) etablierte sich in den letzten Jahren eine abgewandelte Form der Umsetzung: So genannte RESTful Web-Services. REST steht hierbei für 'Representational State Transfer' und beschreibt keinesfalls ein unterschiedliches Protokoll, sondern vielmehr eine andere Architektur welche heutzutage gerade im WWW großen Anklang findet.

Anstatt einen Request über das etwas umständliche XML-Protokoll SOAP anzusprechen, macht sich REST die Eigenschaft zu nutze, dass sämtliche Objekte im Internet über URI (Unified Resource Identifier) ansprechbar sind. Die die 4 standartisierten HTTP-Methoden aus, um sämtliche nötigen Funktionen auf den Objekten auszulösen. Diese Methoden erinnern dabei an die aus dem Datenbanken-Umfeld bekannten CRUD-Aktionen.

HTTP Method	CRUD Action	Description
PUT	CREATE	Create a new resource
GET	RETRIEVE	Retrieve a representation of a resource
POST	UPDATE	Update a resource
DELETE	DELETE	Delete a resource

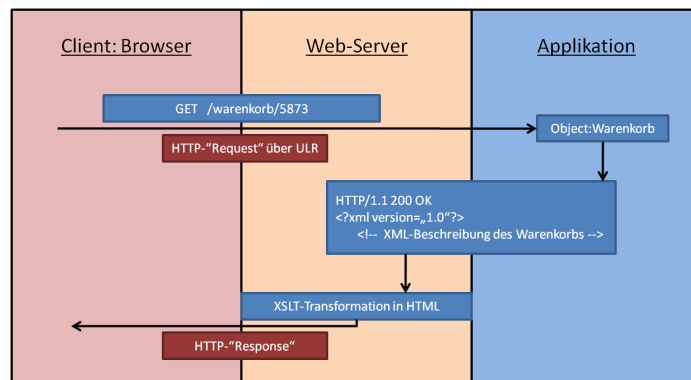


Abbildung 5. Beispiel: RESTful Anfrage

Abbildung 5 zeigt ein vereinfachtes Beispiel eines RESTful-Aufrufs über die GET-Methode bei Amazon. Der Web-Browser schickt eine GET-Anfrage an einen als Resource dargestellten Warenkorb. Die Anfrage wird vom Server verarbeitet und ein XML-Response wird generiert. Dieser wird anschließend über eine XSLT-Transformation in HTML umgewandelt und an den Browser zurückgeschickt.

Literatur

1. Sammlung von grundlegenden Informationen bezüglich Web-Services. Mario Jeckle. Juni 2004.
<http://www.jeckle.de/webServices/>
2. Java Web Services mit Apache Axis2. Thilo Frotscher et al. entwickler.press. April 2007.
3. The W3C Web-Services Glossary. Hugo Haas et al. Februar 2004.
<http://www.w3.org/TR/ws-gloss/>
4. Web Services Kompakt. Michael Kuschke et al. Spektrum Verlag. 2002.
5. Service-orientierte Architekturen mit Web Services. Ingo Melzer et al. Spektrum Verlag. 2008.
6. Definitionssammlung. Gesellschaft für Informatik e.V.
<http://gi-ev.de/>