Web Services in der Praxis

Uwe Hanisch und Ines Birkhahn

T-Systems Multimedia Solutions GmbH
Riesaer Str. 5, 01129 Dresden
uwe.hanisch@t-systems.com; ines.birkhahn@t-systems.com

Abstract. Web Services setzen wir in Informationsportalen ein Für diese Portale eignen sich Web Services zur einheitlichen und einfachen Integration externe Datenanbieter sowie zur Umsetzung von Single-Sign-On- und Clearing-Szenarien. Da wir in einem heterogenen, offenen Umfeld agieren, erwies es sich für einen effektiven Einsatz als notwendig, besondere Sorgfalt auf Toolauswahl, Schnittstellenentwurf, Interoperabilität, Dokumentation und Support zu legen.

1 Umfeld

Portale integrieren Daten und Geschäftsanwendungen. In unserem Falle handelt es sich um Informationsportale. Diese können mehrere Ausgabekanäle bedienen (Web, WAP, PDA). Sie können Teil externer Portale sein. Ein Portal kann Single-Sign-On und Clearing Funktionen anderen Portalen anbieten bzw. selber nutzen. Daten können aus der eigenen Datenbank kommen oder online bei Partnern abgefragt werden.

Wir arbeiten mit einer kleinen aber flexibler Anzahl von Partnern zusammen. Diese verfügen über heterogene Middleware (unterschiedliche Plattformen, Programmiersprachen, Betriebssysteme) inklusive eines Web-Servers. Dabei setzen sie verschiedene Skript- bzw. Programmiersprachen ein.

Die Entwicklung und Integration neuer Teilanwendungen für die Portale unterliegen bestimmten Randbedingungen:

- Die Ressourcen (Zeit, Kosten, Personal) sind beschränkt
- Neue Anwendungen müssen sich in bestehende Middleware integrieren lassen
- Neue Anwendungen müssen nach endlicher Zeit stabil laufen ➔ Kunst des Machbaren

Eine einheitliche, plattformübergreifende Kommunikationstechnologie zur Anbindung externer Partner, würde die Möglichkeit eröffnen, sich auf fachliche Aspekte zu konzentrieren.

Es bietet sich der Einsatz von Web Services an.

2 Kritische Betrachtung der Web Service Technologie

2.1 Kurzübersicht: Web Services

„Definition: A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML based messages exchanged via internet-based protocols." [1]

Folgende XML Schema [2] basierende Protokolle sind für Web Services von grundlegender Bedeutung:

- SOAP – ein Austauschprotokoll (wie packe ich meine Daten ein) [3]
- WSDL – eine Schnittstellenbeschreibung (wie sieht meine Schnittstelle aus) [4]
- UDDI – ein Verzeichnisdienst (wie finde ich einen Dienst) [5]

SOAP ermöglicht den Aufruf von Operationen mit komplexen Datenstrukturen als Ein-/Ausgabewerte.

Als Transportprotokoll definiert SOAP eine Bindung zu HTTP.

2.2 Die Verheißung der Web Services

„Web Services sind eine Infrastrukturtechnologie zur schnellen und einfachen Integration heterogener Anwendungen". [6]

Auf Grund der Nutzung verbreiteter Standards (Internet, XML) und einer simplen Kommunikationsstruktur (Operationen mit Datenstrukturen) lassen sich unter Nutzung von SOAP-Bibliotheken und Generatoren Web Services für alle gängigen Plattformen erstellen bzw. nutzen.

Web Services sind einfach zu verstehen und einzusetzen. Sie ermöglichen eine schnelle Entwickler-Lernkurve.

Web Services eignen sich für eine schnelle, standardisierte Vernetzung vorhandener Systeme im Backend-Bereich. Vorhandene Systeme können mittels entsprechender Generatoren in Web Services gekapselt und ins Netz gestellt werden. Über UDDI Registries können sich Kunden weltweit entsprechend ihren Anforderungen den passenden Web Service heraussuchen.

Firmen können mit Ihren Altsystemen nun auch im Internet leicht Geld verdienen, Kunden steht bald eine reichhaltige Auswahl an Backend Diensten zur Verfügung.

2.3 Allgemeine Probleme beim Einsatz von Web Services

Im Bereich Web Service Technologie vollzieht sich z.Z. eine rasante Entwicklung Bei der Planung eines eigenen Web Services tun sich sofort Fragen auf:

- Welche Web Service Spezifikationen [7] und Versionen sind gültig und werden von Tools unterstützt?
- Welche Web Service Spezifikationen und Versionen werden derzeit platformübergreifend von Tools unterstützt?
- Setzen die verfügbaren SOAP/Web Service Implementierungen diese Spezifikationen zu 100% um? Kann ich diese bedenkenlos einsetzen?

Tests zeigen [8], [9], dass die existierenden Tools nicht 100 % interoperabel sind.

Orientierung bei Spezifikationen gibt die Anfang 2002 gegründete Web Services Interoperability Organization (WS-I). Unter anderem definiert sie Profile von Spezifikationen. Als erstes wurde das WS-I Basic Web Services Profile [10] festegelegt, welches folgende Spezifikationen umfasst:

- XML Schema 1.0 [2]
- SOAP 1.1 [3]
- WSDL 1.1 [4]
- UDDI 1.0 [5]

Somit besteht jetzt die Möglichkeit sowohl bei Tool Herstellern als auch bei Anwendern sich auf ein Profile zu beziehen.

Sicherheit bei den Anwendern von Web Service Tools gibt es jedoch erst, wenn die WS-I Testsuite steht und die ersten Web Service Implementierungen das WS-I Gütesiegel tragen.

Für viele Problemstellungen, die mit dem WS-I Basic Web Services Profile nicht abgedeckt werden, gibt es Spezifikationen und erste Tool-Unterstützung:

- Signierung und Verschlüsselung (XML Signature, XML Encryption)
- Beschreibung von Workflows (Business Process Execution Language for Web Services (BPEL4WS))

Diese sind zum Teil aber noch im Entwurfs-Stadium. Für den Einsatz im heterogenen Umfeld sollte man jedoch abwarten, bis:

- die entsprechende Spezifikation stabil sind und als Bestandteil von Web Services allgemein anerkannt ist (Teil eines WS-I Profiles),
- für alle verbreitetenden Plattformen / Programmiersprachen Implementierungen verfügbar sind und
- diese Implementierungen zusammenarbeiten.

In der Produktion kann der Einsatz von nicht fertigen Spezifaktionen zu Problemen führen. (Beispiel: Frühe SOAP Implementierungen setzten auf einen Entwurf des XML Schemas Part 2 / 2001, in denen der Typ für eine Zeitinstanz noch „timeInstant“ im Gegensatz zu „dateTime“ hieß.)

Zu beachten ist, dass Web Service softwaretechnisch eine „Prozeduale Programmierung übers Web“ darstellen und SOAP Aufrufe zustandslos sind.

Benötigt man ein Session Handling, muss man es selbst umsetzen. Man kann z.B. alle Funktionen um eine Session Variable erweitern. Sofern man ein bestehendes System über Web Service zugänglich machen will, bedeutet dies natürlich erhöhten Programmieraufwand des Dienste Anbieters. Mit anderen Methoden, z.B. Nutzung des SOAP Headers, braucht man den alten Code nicht anzufassen. Dafür lässt sich dies derzeit nicht in einer WSDL Datei beschreiben. Somit haben Partner einen erhöhten Programmieraufwand bei der Anbindung, sofern ihre Middleware/Tools das entsprechende Feature überhaupt unterstützen.

D.h. nicht alle bestehenden Anwendungen lassen sich ohne weiteren Programmieraufwand als Web Service im Internet verfügbar machen.

Ein Einsatz von neuen Features eines Web Services Werkzeuges sollte in einer offenen Umgebung sehr genau bedacht werden. Die Vorteile der besseren Abbildung der bestehenden Anwendung wird mit möglicher Inkompatibilität erkauft. Der Test des eigenen Web Service mit anderen gebräuchlichen Tools, eventuell sogar mit einer anderen Programmiersprache, ist hilfreich.

Für einen effektiven Einsatz von Web Services wären allgemein bekannte und anerkannte Sammlung von Best Practise, Does and Dont's, Style Guides und Patterns bei der Entwicklung/Einsatz von Web Services sehr hilfreich.

Quintessenz: Ganz so einfach, wie manchmal geschildert wird, ist der Einsatz von Web Services in einer heterogenen Umgebung nicht. Die Entwickler benötigen ein Grundverständnis der eingesetzten Protokolle. Bei der Toolauswahl und der Umsetzung ist Sorgfalt und eine gewisse Skepsis neusten Protokollen gegenüber hilfreich.

Wir wollen im folgendem zwei Einsatzbeispiele aus der Praxis darstellen, in denen wir Web Services sowohl als Dienste Nutzer als auch Dienste-Anbieter einsetzen. Schwerpunkt sollen nicht die technischen Details, sondern die aufgetretenen Probleme und offenen Fragen sein.

3 Praxisbeispiel WAP-Portal

Anfang 2001 waren wir an der initialen Entwicklung eines WAP-Portales eines Start-up Unternehmens beteiligt.

Ein Teil der Inhalte des Portales wird von Drittanbietern angeboten und in das Portal eingebunden. Ihren Kunden wollten die Drittanbieter die Möglichkeit einer Lokalisierung ihres Handset bieten, wofür unser WAP Portal den Drittanbietern eine entsprechende Schnittstelle anbieten musste.

Bei den Drittanbietern waren neben Webservern auch verschiedene Scriptsprachen zur Inhaltsgenerierung im Einsatz: PHP, Perl, JSP (Java), ASP (VBasic). Für all diese Sprachen gab es zu diesem Zeitpunkt existierende SOAP Implementierungen. Der Einsatz von Web Services (SOAP über HTTP(S)) zur Verbindung der Backends zwang sich geradezu auf. Zur Absicherung der Kommunikation (Authentifizierung, Verschlüsselung) standen uns mit Basic-Authentification von HTTP und SSL die entsprechenden Webserver tauglichen Protokolle zur Verfügung. Da eine direkte Beziehung zwischen dem WAP-Portal Betreiber und dem Drittanbietern bestand, wurde auf die Nutzung eines UDDI Verzeichnisses verzichtet.

In unserem Entwicklungsteam gab es kontroverse Meinung, wie der Web Service entwickelt werden und was den Drittanbietern zur Verfügung gestellt werden sollte.

Als erster Ansatz sollte aus bestehenden Java-Klassen (wir benutzten BEA Weblogic als Application Server und IdooXoap als SOAP Bibliothek) mit den Generatoren der SOAP Bibliothek eine WSDL Datei generiert werden. Dieses erwies sich jedoch als nicht brauchbar und es wurde ein Schritt zurückgegangen. Anhand der Erfordernisse der Drittanbieter und unserer eigenen Architektur wurde eine Schnittstelle definiert, die entsprechende WSDL Datei geschrieben und Aufruf Szenarien dokumentiert. Danach wurde anhand der WSDL Datei mit der serverseitigen Implementierung begonnen.

Als nächstes stellte sich die Frage, was wir den Drittanbietern zur Einbindung des Web Service liefern. Wir entwickelten Client-Referenz-Implementierungen des Dienstes in den verschiedenen Programmiersprachen. Diese erwiesen sich als gute Testumgebungen für unseren Web Service. Eine Idee war, diese Client-Referenz-Implementierungen den Drittanbietern zur Verfügung zu stellen. Doch damit hätten wir uns selbst in den Zugzwang gesetzt, diese Implementierung zu pflegen - ein unvertretbarer Aufwand. Außerdem war die IT-Landschaft (Programmiersprachen, und Version, Umgebung) bei den Drittanbietern zu heterogen, und wir hätten ihnen eine Implementierung aufgezwungen. Hinzu kommt noch, dass sich die SOAP Bibliotheken für die verschiedenen Programmiersprachen viel zu rasant weiterentwickeln, so dass es sehr schwer ist, über alle den Überblick zu behalten.

Am Ende wurde den Drittanbietern neben der WSDL Datei eine Dokumentation mit Beispielen, Workflow und API Beschreibung als auch eine Dokumentation mit Hinweisen zur Implementierung des Web Services in den einzelnen Programmiersprachen geliefert.

Ein Server mit dem Web Services stand zum Testen der eigenen Implementierung zur Verfügung. Für Rückfragen gab es einen Ansprechpartner. Vor der Aufnahme in das WAP Portal wurden die Web Services Implementierungen des Drittanbieters getestet und abgenommen.

4 Praxisbeispiel Informations-Portal

Seit ca. zwei Jahren arbeiten wir an der Entwicklung / Pflege eines Auskunfts-Portales. Das Entwicklungs-Team inklusive Design, Spezifikation, Projektleitung und Betrieb umfasst ca. 30 Mitarbeiter. Des weiteren werden auch Inhalte von Drittanbietern in das Portal integriert, die ihre Seiten selbst anpassen.

Die Informationen werden für die drei Ausgabemedien WEB, PDA und WAP aufbereitet. Dafür nutzen wir den BEA Weblogic Server 6.1 als Application Server und somit Java als Programmiersprache. Serverseitige Dienste werden intern als Stateless Session Beans gekapselt. Informationen kommen aus verschiedenen Quellen. Zum einen halten wir diese in einer Oracle Datenbank, zum anderen werden diese bei Drittanbietern synchron abgefragt. Dabei wird ein Sammelsurium an Kommunikationsprotokollen verwendet, fast jeder Drittanbieter hat sein eigenes. Eine Auswahl:

- ➔ Anfrage: HTTP GET Antwort ➔: Stück HTML Codes
- ➔ Anfrage: HTTP GET Antwort ➔: XML gekapselt in HTTP POST
- ➔ Anfrage: XML gekapselt in HTTP POST Antwort➔: XML gekapselt in HTTP POST
- Web Service mit SOAP über HTTPS

Die zu nutzenden Protokolle werden von den Drittanbietern vorgegeben. Derzeit versuchen wir, den Drittanbietern von Web Services zu überzeugen.

In einer neuen Version unseres Auskunfts-Portales wollen wir Partnern einen „Gelbe Seiten“ Dienst zur Verfügung stellen. Für die Implementierung ist von Anfang an die Nutzung von Web Services (SOAP 1.1 über HTTPS mit Basic Authentification, WSDL 1.1) vorgesehen. Für eine Nutzung eines UDDI Verzeichnisdienstes sehen wir keine Notwendigkeit. Der Service soll exklusiv ausgewählten Partnern zur Verfügung stehen. Nach einer Pilotphase soll dieser Dienst kostenpflichtig sein. Web Service stellen z.Z. noch kein allgemein anerkanntes Framework für die Anmeldung bzw. Abrechnung von kostenpflichtigen Diensten zur Verfügung. Ein öffentlicher Zugang zu diesem Dienst ist somit ausgeschlossen, eine Nutzung eines UDDI Verzeichnisdienstes ist nicht unbedingt notwendig.

4.1 Erfahrung – Anbindung von Web Service dritter

4.1.1 WAP / PDA Portal

Unsere WAP und PDA Anwendung ist Partner eines anderen Portales. Dieses bietet einen Web Service zur Lokalisierung mobiler Geräte, Single-Sign-On und Abrechnung von Mehrwertdiensten an. Dabei wird SOAP 1.1 über HTTPS mit Basic Authentification und WSDL 1.1 als Schnittstellenbeschreibung genutzt.

Vom Dienste Anbieters bekamen wir einer sehr klaren WSDL Datei und eine Dokumentation geliefert. Die Dokumentation beschrieb:

- den Zwecks der angebotenen Dienste
- den Workflows für sinnvollen Einsatz
- die Methoden, insbesondere
    - Welcher Wertebereich die Ein- und Ausgabedaten haben
    - Welche Fehlermeldungen wann auftreten können

Wir selber benutzten für die Implementierung eine Open Source Bibliothek (Apache SOAP [12]) und kapselten die Schnittstelle mit einer Abstract Factory. Somit ist es möglich, die SOAP Bibliothek auszutauschen, ohne die Teile unserer Anwendung, die diesen Lokalisierungs-Dienst nutzen, anzufassen. Erst dieses Jahr wurden mit JAXM und JAX-RPC ([13],[14]) Java Schnittstellen geschaffen, auf die man hin programmieren kann, ohne sich an eine bestimmte SOAP Bibliothek zu binden. Z.Z. sind die Anbieter von Java SOAP Bibliotheken dabei diese umzusetzen (Apache Axis, Wasp).

Zwischenzeitlich gab es Probleme beim Aufruf des Web Services, weil zwischen zwei API Versionen das eingesetzte XML Schema von Stand 1999 auf 2001 geändert wurde. Die eingesetzte Apache SOAP 2.2 Bibliothek unterstützte XML Schema des Jahres 2001 aber nur in der Entwurfs Version („timeInstant“ statt „dateTime“). Durch einen eigenen Patch konnte das Problem schnell gelöst werden – ein Vorteil von Open Source Software.

Ein Fehler trat auf unsere Seite auf, als wir von Bea Weblogic 6.0 auf 6.1 umstellten. Dies lag an einem fehlerhaften XML Parser (Xerces 1.3.1), welcher das SOAP Fault Element nicht akzeptierte. Nach dem wir den XML Parser durch eine neuere Version ersetzt hatten, funktionierte unser Web Service Client wieder.

In der Produktion läuft unser Web Service Client stabil. Probleme gibt es mit der Erreichbarkeit des Web Service, was aber an Schwierigkeiten mit automatischen Updates auf der Dienste Anbieter Seite liegt, nicht aber am Web Service selber.

Insgesamt gestaltete sich die Zusammenarbeit sehr effektiv. Für kurzfristige Rückfragen stand uns ein direkter Ansprechpartner zur Verfügung. Die Implementierung des Web Service Clients und das Verständnis des Workflows stellten aufgrund der guten Dokumentation kein Problem da.

4.1.2 Web Portal

Unser Web Anwendung ist Partner eines anderen Portales. Seit Mitte dieses Jahres stellt dieses ein Web Services für Singl-Sign-On und Abrechnung zum Test bereit. Dabei kommt SOAP 1.1 über HTTPS und WSDL 1.1 zur Schnittstellenbeschreibung zum Einsatz.

Neben einer nicht mehr aktuellen und nicht vollständigen Dokumentation stand uns online die WSDL Datei bereit. Dieses zeichnete sich durch einige Eigentümlichkeiten

aus. Für jede Ein- und Ausgabenachricht wurde ein eigener Datentyp definiert. Anstatt das SOAP Fault Konstrukt zu nutzen, wird in jeder Ausgabenachricht ein selbst auszuwertender Fehlercode mitgeben. Die Schreibweise von Variablen ist sehr uneinheitlich. Einmal werden diese in Kleinbuchstaben dargestellt, ein anderes mal wiederum in Großbuchstaben.

Die vom WSDL2Java Generator erzeugten Java Klassen sahen sehr unübersichtlich aus. Um den Programmierern, die das Frontend erstellen, die Arbeit zu erleichtern, wurden die erzeugten Klassen in eigene Klassen/Schnittstellen gekapselt. Datentypen mit gleichen Signaturen wurden zusammenfasst und die Methoden vereinfacht. Zeigt die Fehlercode Variable in der Ausgabenachricht einen Fehler an, wird der Wert auf eine eigene Exception gemappt.

Als SOAP Bibliothek kam Apache Axis Beta3 [15] zum Einsatz. Um die Bibliothek ersetzten zu können, wurde wieder das Abstract Factory Pattern eingesetzt.

Mit erhöhtem Programmieraufwand konnte unseren Frontend-Entwicklern eine einigermaßen zufriedenstellende API angeboten werden.

Im Moment befinden sich die auf den Web Service aufbauenden Anwendungen in der Testphase. Ende September beziehungsweise Anfang Oktober sollen sie in Produktion gehen.

4.2 Erfahrung – Implementierung eines eigenen Web Services

Auf Nachfrage von potentiellen Nutzern wurde Anfang des Jahres die Bereitstellung unser internen Routenplaner Schnittestelle als externer Dienst geplant.

Als Mittel zum Zweck wurden Web Services gewählt. Ein erster Ansatz war die Web Service Implementierung von BEA Weblogic 6.1. BEA liefert einen Generator mit, der aus einem EJB die Web Service Klassen, das entsprechende WSDL File und die Deployment Descriptoren generiert.

Die vom Generator erzeugte WSDL Datei war schlichtweg falsch. (Variablen von Datenklassen wurden in „attribute“ statt „element“ Tags umgewandelt.) .

Die weitere Entwicklung wurde jedoch aus projektbedingten Gründen zurückgestellt.

Zur Zeit sind wir dabei, unsere interne Schnittstelle zur Suche in unserem Gelben Seiten Datenbestand als Web Service (SOAP 1.1 über HTTPS mit Basic Authentification, WSDL 1.1) zu kapseln. Entgegen dem ersten Versuch eines eigenen Web Services, schrieben wir als erstes eine WSDL Datei mit den Schnittellen, die wir nach außen anbieten wollen und eine Dokumentation. Dieses bietet zugleich die Möglichkeit unsere intern genutzten Schnittstellen und Klassen zu überarbeiten. Zum Beispiel wurden die Fehlermeldungen überarbeitet und Eingabefehler abgefangen.

Angefangen haben wir mit dem Test der verschiedene SOAP Bibliotheken. Zum Vergleich der erzeugten WSDL Datei dient unsere selbst geschriebene.

Erste Tests mit Partnern sind für Mitte September geplant.

5 Fazit

Web Services ermöglichen die Verknüpfung von heterogenen Systemen im Backend Bereich von eng zusammenarbeitenden Partnern. Aufwand und Zeit hängen wesentlich von der Stabilität und Akzeptanz der benutzten Web Service Spezifikationen, der Interoperabilität der eingesetzten Tools und von der Schnittstelle selber ab.

Das Aufsetzen eines Web Services geschieht auch mit der Nutzung von Generatoren nicht von selbst. Da die Schnittelle (WSDL Datei) eines Web Services eine Art öffentlicher Vertrag ist, sollte auf deren Entwicklung besondere Sorgfalt gelegt werden. Eine überfrachtete, unverständliche Schnittestelle nutzt man selbst auch nur ungern („Was ich nicht gerne mag, sollte ich auch keinem anderen anbieten!“). Dies hat aber weniger mit der Entwicklung eines Web Services zu tun, als mit guter Softwareentwicklung und der Frage, welche Schnittstellen ich Dritten zumute.

Mit der Weiterentwicklung der verfügbaren Tools, sollte sich in absehbarer Zeit das Schreiben einer WSDL Datei per Hand erübrigen.

Eine WSDL Datei stellt nur eine technische Definition der Schnittstelle dar. Ohne

- eine ausreichende Dokumentation
- Einsatzbeispiele
- Beschreibung des notwendigen Workflows
- Testen des eigenen Webservices auf Kompatibilität mit anderen gebräuchlichen SOAP Bibliotheken
- Server mit einer Implementierung des Web Services zum Testen
- schnelle Erreichbarkeit der Partner

kostet der Einsatz von Web Services genauso viel, wie jedes andere Kommunikationsprotokoll (insbesondere Kopfschmerzen bei den Entwicklern des Dienste Nutzers ☺).

Literatur

[1] Web Services Architecture Requirements, W3C Working Draft 19 August 2002, http://www.w3.org/TR/wsa-reqs

[2] XML Schema 1.0 Specification, W3C, http://www.w3.org/XML/Schema

[3] SOAP v1.1 Protocol Specifications (W3C Note), http://www.w3.org/TR/SOAP/

[4] WSDL1.1 Specification (W3C Note), http://www.w3.org/TR/wsdl

[5] UDDI Specifications, UDDI.org, http://uddi.org/specification.html

[6] Dirk Slama, Interview Shinka Technologie in XML & Web Services Magazin, 1.02

[7] soaprpc.com, Web Service - Specifications, http://www.soaprpc.com/specifications/

[8] SOAPBuilders Interoperability Lab "Round 2", http://www.whitemesa.com/interop.htm

[9] SOAPBuilders Interoperability Lab "Round 3". http://www.whitemesa.com/r3/interop3.html

[10] Web Services Interoperability Organization (WS-I), Web Service Profiles – An Introduction ,http://www.ws-i.org/docs/WS-I_Profiles.pdf

[11] Business Process Execution Language for Web Services (BPEL4WS), Version 1.0, http://www.ibm.com/developerworks/library/ws-bpel/

[12] Apache SOAP, http://xml.apache.org/soap

[13] Java API for XML Messaging (JAXM) , http://java.sun.com/xml/jaxm/

[14] Java API for XML-Based RPC (JAX-RPC) , http://java.sun.com/xml/jarpc/

[15] Apache Axis, http://xml.apache.org/axis