Web Services – Grundlagen und praktisches Beispiel

Ho Ngoc Duc
http://come.to/duc
duc@ifis.uni-luebeck.de

Gliederung

- Einführung
  - Was sind Web Services?
  - Warum Web Services?
- Spezifikationen und Standards
  - Beschreiben: WSDL
  - Auffinden: UDDI
  - Verwenden: SOAP
- Entwicklung von Web Service Anwendungen
- Ausblick

Einführung

- Dominierendes Thema in der IT-Branche
- Gesamte Software-Industrie hat das Thema aufgegriffen
- Viele Firmen bieten ihre Dienste als Web Services an: Google, Amazon...
- Auffällig: offene Standards von führenden Unternehmen unterstützt (Microsoft, Oracle, IBM, Sun , Hewlett-Packard...)

Motivation

- Um sich besser vorstellen zu können, was ein WS ist: ein Beispiel.
- Online-Shop:
  - Katalog anschauen, Ware bestellen (Angabe Kreditkartennummer, Adresse)
  - Nach Überprüfung: Bestellung bearbeitet; Versandkosten, Währungsumrechnungskurs, Steuersatz ermittelt
  - Kunde kann auf einer speziellen Webseite jederzeit den Status der Bestellung einsehen
- Online-Shop benötigt Daten anderer Firmen
  - Kreditkartennummer korrekt? Adresse korrekt?
  - Wann kann Ware geliefert werden?
  - Versandkosten? Status der Lieferung?

Warum Web Services?

- Daten manuell zu sammeln ist ineffizient
  - Traditionell: Brief, Fax, Telefon, E-Mail
  - Browser bedienen, bei Web-Seiten der Kreditkartenfirma, des Lieferanten, des Paketdienstes nachschauen
- Mögliche Automatisierung: 'Screen scraping'
  - HTML-Seiten mit Programmen laden und parsen, um relevante Informationen zu extrahieren
  - Nachteil: unzuverlässig, fehleranfällig
- Notwendig: programmatischer Zugriff auf Dienste bzw. Daten von Geschäftspartnern
  - Sendet Anfrage, erhält Daten, formatiert für Kunden
  - Web Services bilden Infrastruktur für solche Dienste!

Was sind Web Services?

- Keine allseits akzeptierte Definition
- Wichtige Charakteristika:
  - Software-Komponente (Stück Geschäftslogik)
  - Bereitstellung der Funktionalität im Internet/Intranet (Zugriff über einen URI)
  - Beschreibung, Entdeckung, Zugriff mit XML-Nachrichten
  - Datenaustausch über Standard-Protokolle des Internets (HTTP, SMTP, FTP...)

Was leisten Web Services?

- Computer-zu-Computer-Kommunikation, nicht Mensch-zu-Computer
- Austausch von Daten und Funktionalität zwischen heterogenen Anwendungen auf unterschiedlichsten Systemen
  - Dadurch: große, lose gekoppelte Systeme
  - Unabhängigkeit von Plattform / Implementierung der Komponenten
- Idee: Funktionalität eines Dienstes duch Austausch von XML-Nachrichten bekannt geben, verwenden

Nutzungsbeispiel

- Online-Shop
  - verifiziert durch Web Service A Gültigkeit der Kreditkarte
  - überprüft mit Web Service B die Gültigkeit der gegebenen Adresse
  - ermittelt (durch Web Service C) Versandkosten
  - fragt Web Service D nach Währungskurs
  - ...

Basis-Infrastruktur für Web Services

- Austausch zwischen heterogenen Systemen: maschinell lesbare Informationen bereitstellen → Standards definieren
- Grundlegend: 3 Standards
  - Dienst beschreiben: Wo? Protokolle? Welche Funktionalitäten? Parameter?... →WSDL (Web Service Description Language)
  - Format der Frage/Antwort festlegen: aufzurufende Funktion, Parameterwerte, Struktur der Antwort... →SOAP
  - Möglichkeit, Web Services zu finden →UDDI (Universal Description, Discovery and Integration)
- Datenrepräsentation in allen 3 Standards: XML

XML – Extensible Markup Language

- Standard zur Beschreibung von Daten
  - Universelles Format für strukturierte Dokumente und Daten im World Wide Web
- Wichtige Eigenschaften
  - Ganze Familie von Technologien zur Definition von Auszeichnungssprachen, um Daten zu strukturieren
  - Ähnlichkeit mit HTML: hierarchisch aus Elementen und Attributen aufgebaut; Elemente mit Start- und Endmarken („Tags“)
  - Für Menschen lesbar, aber primär für Programme
  - Modular, plattform-unabhängig, breite Unterstützung

Web Service Description Language (WSDL)

- „XML-Grammatik“, um eine allgemeine Schnittstelle für Web-Services zu definieren
- Hauptelemente:
  - Adress-Informationen (Service lokalisieren)
  - Informationen über öffentlich verfügbaren Funktionen
  - Datentyp-Informationen für XML-Messages
  - Informationen über das zu benutzende Transport-Protokoll
- Client-Programme (teilw.) aus WSDL erzeugen!
- Bsp: http://api.google.com/GoogleSearch.wsdl

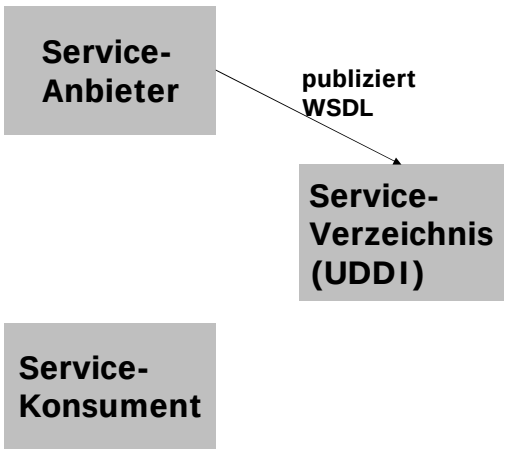
SOAP

- Ursprünglich: Simple Object Access Protocol
- Standardisiert den Austausch von XML-Nachrichten zwischen unterschiedlichen Plattformen / Programmiersprachen
- SOAP legt fest, wie ein XML-Dokument für den Versand verpackt wird
- SOAP legt Transport nicht fest! SOAP-Messages können mit HTTP, SMTP, FTP... gesendet werden
  - In der Praxis: überwiegend mit HTTP

Universal Description, Discovery and Integration (UDDI)

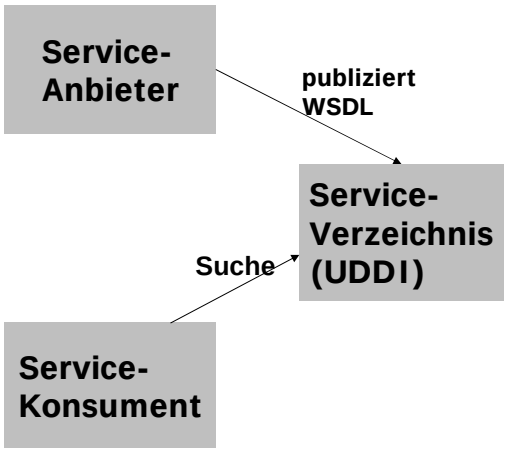
- Industrie-Initiative (Ariba, IBM, Microsoft)
- Durchsuchbares Verzeichnis von Diensten und ihren Beschreibungen
  - Besonders für Web-Services gedacht
  - Suche manuell oder über UDDI-API
- 3 Typen von Informationen:
  - Weiße Seiten: Suche nach Unternehmen
  - Gelbe Seiten: Suche in Kategorien
  - Grüne Seiten: technische Informationen über Web-Services (WSDL-Dokumente)

Basis-Architektur



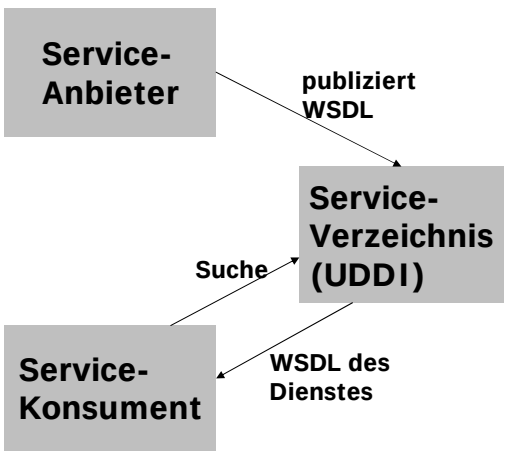
• Anbieter publiziert Beschreibung des Dienstes (WSDL-Dokument)

Basis-Architektur



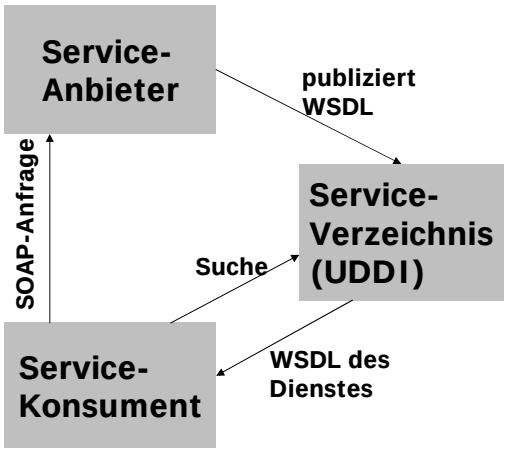
- Anbieter publiziert Beschreibung des Dienstes (WSDL-Dokument)
- Konsument sucht im (UDDI) Verzeichnis nach Anbieter

Basis-Architektur



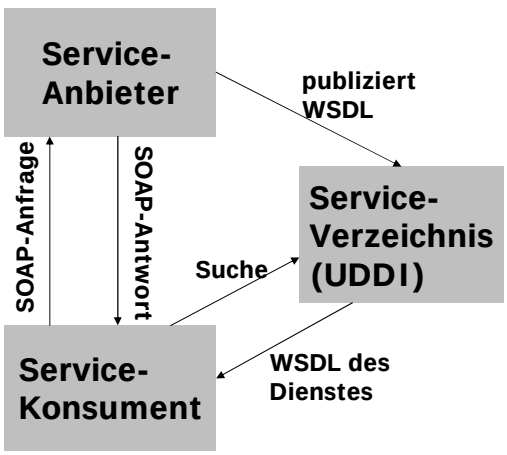
- Anbieter publiziert Beschreibung des Dienstes (WSDL-Dokument)
- Konsument sucht im (UDDI) Verzeichnis nach Anbieter
- Konsument erhält WSDL (Dienstbeschreibung)

Basis-Architektur



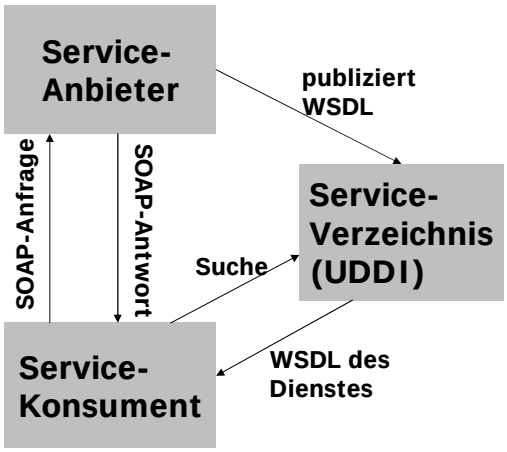
- Anbieter publiziert Beschreibung des Dienstes (WSDL-Dokument) im Verzeichnis
- Konsument sucht im (UDDI) Verzeichnis nach Anbieter
- Konsument erhält WSDL (Dienstbeschreibung)
- Konsument sendet SOAP-Anfrage an Anbieter

Basis-Architektur



- Anbieter publiziert Beschreibung des Dienstes (WSDL-Dokument) im Verzeichnis
- Konsument sucht im (UDDI) Verzeichnis nach Anbieter
- Konsument erhält WSDL (Dienstbeschreibung)
- Konsument sendet SOAP-Anfrage an Anbieter
- Konsument erhält SOAP-Antwort vom Anbieter, kann sie interpretieren / weiter verarbeiten

SOAP, WSDL und UDDI



- SOAP- und WSDL-Dokumente: relativ komplex
  - durch Tools generiert
- UDDI für automatische Suche nach Web-Services gedacht
  - in der Praxis kaum so benutzt: Auswahl schwer zu automatisieren
  - Web-Services / WSDL auf andere Weise veröffentlicht

Beispiel: Entwicklung von Web Services

- Nach den Grundlagen: am konkreten Beispiel demonstrieren
  - Web Service erstellen
  - Service-Konsument entwickeln, Dienst nutzen
- Technologie
  - Java Web Service mit Apache Axis
    - Open source Java Web-Services Toolkit
    - Entwicklung von Service-Anbieter und -Konsument
  - Clients in verschiedenen Sprachen (Java, PHP)

Standard-Vorgehensweise

- Service-Anbieter
  - Geschäftslogik implementieren
  - Beschreibung (WSDL) erstellen, veröffentlichen
  - Unter einem URI (am Web Service Server) registrieren
  - Wenn SOAP-Request ankommt: analysieren, Geschäftslogik ausführen, SOAP-Response senden
- Service-Konsument
  - Programmierschnittstelle aus Beschreibung des Anbieters extrahieren
  - Anfrage (auszuführende Funktion, Parameter) in SOAP-Request packen, an URL des Anbieters senden
  - Ergebnis aus SOAP-Response extrahieren, weiter verarbeiten

Web Services mit Apache Axis

- Axis von apache.org herunterladen
- Als Web-Applikation in JSP/Servlet Container (z.B. WebLogic, Apache Tomcat) installieren
- Funktionalitäten der Geschäftslogik, die man als Web Services zur Verfügung stellen will, mit Axis registrieren
- Axis kann Beschreibung des Services (WSDL-Dokument) aus Java-Klassen generieren
- SOAP-Nachrichten zur Nutzung der Methoden von AXIS automatisch erzeugt

Beispiel: Web Service für Paketverfolgung

- Paketdienst bietet Kunden Funktionalität der Paketverfolgung als Web Service an
- Java-Klasse mit Methode getStatus()

```
package com.parcelservice.ws;
public class PacketTracker {
   public String getStatus(String code) {
            Packet pk = Packet.getPacket(code);
            if (pk != null) {return pk.getStatus();} else return null;
   }
}
```


Web Service veröffentlichen

- Web Service Deployment Descriptor (Datei deploy.wsdd in XML-Format) schreiben
- WS Server mitteilen: „Klasse PacketTracker ist WS“

  java org.apache.axis.client.AdminClient deploy.wsdd

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
        xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
 <service name="PacketTrackingService" provider="java:RPC">
  <parameter name="className"
value="com.parcelservice.ws.PacketTracker"/>
  <parameter name="allowedMethods" value="*"/>
 </service>
</deployment>
```

WSDL erzeugen

- Der neue WS nun verfügbar am URI:

http://ws.parcelservice.com:8080/axis/services/PacketTrackingService

- Beschreibung des WS (WSDL) mit Axis-Tool Java2WSDL erzeugen und veröffentlichen
- WSDL auch dynamisch generiert:

http://ws.parcelservice.com:8080/axis/services/PacketTrackingService?wsdl

- WSDL komplexes XML-Dokument mit Angaben zum Web Service: URI, Methoden, Parameter, Transportprotokoll, Datentypen...
  - Web Service ist selbstbeschreibend

```
- <wsdl:message name="getStatusRequest">
    <wsdl:part name="code" type="xsd:string" />
  </wsdl:message>
- <wsdl:message name="getStatusResponse">
    <wsdl:part name="getStatusReturn" type="xsd:string" />
  </wsdl:message>
- <wsdl:portType name="PacketTracker">
  - <wsdl:operation name="getStatus" parameterOrder="code">
      <wsdl:input message="impl:getStatusRequest" name="getStatusRequest" />
      <wsdl:output message="impl:getStatusResponse" name="getStatusResponse" />
    </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="PacketTrackingServiceSoapBinding" type="impl:PacketTracker">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  - <wsdl:operation name="getStatus">
      <wsdlsoap:operation soapAction="" />
    - <wsdl:input name="getStatusRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://ws.parcelservice.com" use="encoded" />
      </wsdl:input>
    - <wsdl:output name="getStatusResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://ws.parcelservice.com:8080/axis/services/PacketTrackingService"
          use="encoded" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="PacketTrackerService">
```

Service-Konsument in Java mit Axis

```
static String getPacketStatus(String packetCode) throws Exception {
    String endpoint = "http://ws.parcelservice.com:8080"
        + "/axis/services/PacketTrackingService";
    Service service = new Service();
    Call call = (Call) service.createCall();
    call.setTargetEndpointAddress(new URL(endpoint));
    call.setOperationName("getStatus");
    return (String) call.invoke(new Object[] { packetCode } );
}
```

- Aufruf der Operation mit Hilfe von org.apache.axis.client.Call
- Alternativ (insbes. wenn Interface des Web Service komplex): Proxy-Klassen (Stubs) mit WSDL2Java generieren

Service-Konsument in PHP

```
require_once('./lib/nusoap.php');
function askWSParcelService($packetcode) {
  $uri =
'http://ws.parcelservice.com:8080/axis/services/PacketTrackingService';
  $client = new soapclient($uri);
  return $client->call('getStatus', array($packetcode));
}
```

- Bibliothek nusoap.php verwenden
- Namen der Operationen und Typen der Parameter aus der Beschreibung
- Funktion call der PHP-Klasse soapclient aufrufen mit Namen einer Operation und Parameterwerten

Web Services nutzen

```
function getOrderStatus($ordercode) {
  $status = queryDatabase($ordercode);
  if ($status == 'PARTS_ORDERED') {
        /*Fragt den Web Service des Lieferanten nach Liefertermin*/
        $expectedDelivery = askWSSupplier($ordercode);
        return '... within ' . $expectedDelivery . ' days';
  } else if ($status == 'SENT') {
        /*Fragt den Web Service des Paketdienstes nach Status*/
        $packetcode = getPacketCode($ordercode);
        return ' ...' . askWSParcelService($packetcode);
  } else {
        return 'Your order is beeing processed';
  }
}
```

Status der Bestellung verfolgen: http://web.hndshop.com/status/

Zusammenfassung / Ausblick

- Diskutiert was Web Services sind und warum sie eine wichtige Entwicklung darstellen
- Basistechnologien für Web Services vorgestellt: SOAP, WSDL, UDDI
- Um Entwicklung komplexer WS-Anwendungen zu unterstützen: weitere Technologien nötig
  - Unterstützung für Sicherheit, Authentifizierung, Transaktion, Workflow, Dienstqualität (QoS), Mobilen Zugriff...
- Verschiedende Spezifikationen z.Z. entwickelt