Technischer Überblick zu Web Services

Stephan Karrer

Web Services: Was ist das?

??? Eine Dienstleistung, die über das Web genutzt werden kann ???

Ist **Google** ein Web Service ?

Ist mein **Online Banking Zugang** ein Web Service ?

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."

W3C's Web Services Architecture Working Group

„Web-Services sind selbstbeschreibende, gekapselte Software-Komponenten, die eine Schnittstelle anbieten, über die ihre Funktionen entfernt aufgerufen, und die lose durch den Austausch von Nachrichten miteinander gekoppelt werden können. Zur Erreichung universeller Interoperabilität werden für die Kommunikation die herkömmlichen Kanäle des Internets verwendet. Web-Services basieren auf den drei Standards WSDL, SOAP und UDDI: Mit WSDL wird die Schnittstelle eines Web-Services spezifiziert, via SOAP werden Prozedurfernaufrufe übermittelt und mit UDDI, einem zentralen Verzeichnisdienst für angebotene Web Services, können andere Web-Services aufgefunden werden."

Arbeitskreis Web Services der Gesellschaft für Informatik

Merkmale von Web Services

- Autonom
- Funktion ist gekapselt und über definierte Schnittstelle zugreifbar
- Lose gekoppelt
- Selbstbeschreibend
- Werden über das Web veröffentlicht, lokalisiert und aufgerufen
- Sprachunabhängig und interoperabel
- Basieren auf herstellerübergreifenden Standards (Offenheit)

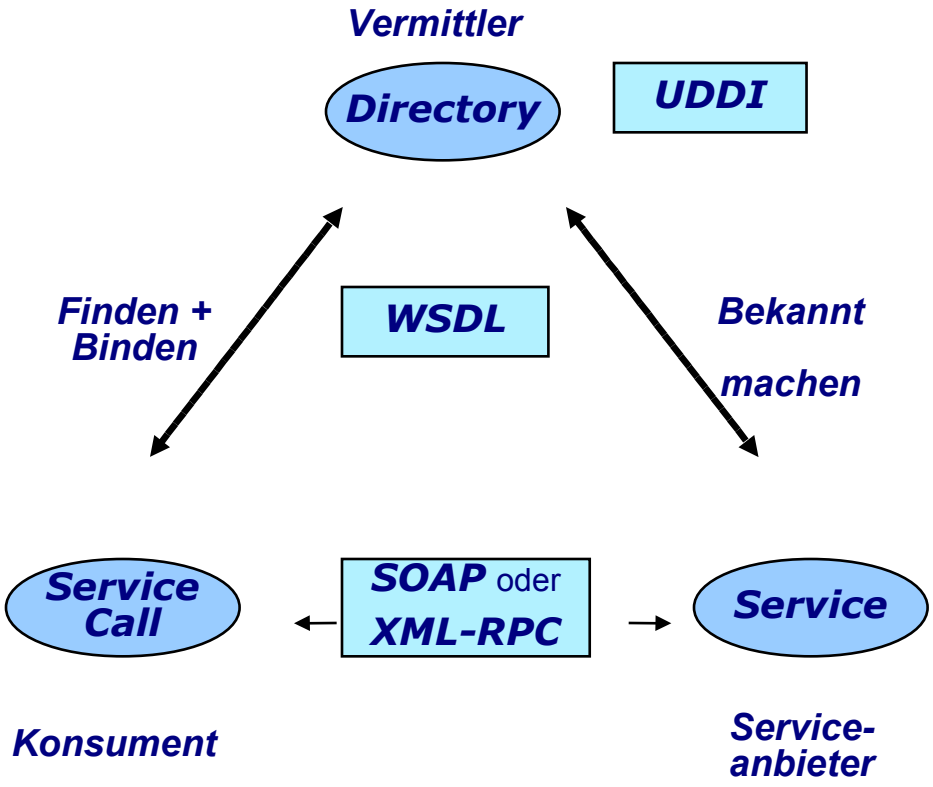
Bisherige Architektur-Ansätze für verteilte Applikationen und Infrastruktur:

- Proprietär, z.B. Datenbankhersteller wie Oracle, IBM, ...
- RPC-Konzepte: Sun RPC (ONC), DCE-RPC (Microsoft)
- CORBA (OMG)
- COM/DCOM/.NET (Microsoft)
- J2EE
  - aber letztendlich bis heute nicht vollständig Plattform und Hersteller-übergreifend

Die zu lösenden technischen Herausforderungen sind nach wie vor gegeben:

Web Services als kleinster gemeinsamer Nenner ?

Web Services: Basistechnologien



Web Services sind keine Revolution sondern eher Evolution durch Kombination verschiedener Technologien:

- HTTP, SHTTP, SMTP
- XML

und Konzepte

Auf dieser Basis entstehen:

SOAP, WSDL, UDDI, ...

HTTP als Transport-Protokoll

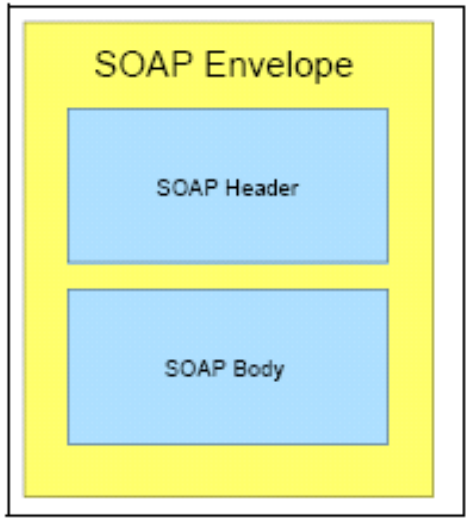
- **Vorteile:**
  - weitreichende Unterstützung und Verfügbarkeit
  - offenes Protokoll
  - HTTP-Daten passieren in der Regel vorhandene Firewalls
  - sichere Kommunikation kann via HTTPS erfolgen

- **Nachteile:**
  - HTTP bietet keine spezielle Servicequalität (QoS)
  - HTTP ist „stateless“
    - ☞ hier muß bei Bedarf auf der Applikationsebene nachgesteuert werden
    - ☞ man kann auch andere Transport-Protokolle verwenden

Weitere Transport-Protokolle (SOAP bindings)

- SOAP Over Email
  - im Body der Email
  - als Anhang (Verwendung von MIME)
- Direkte Verwendung des Transport-Protokolls
  - z.B. UDP, TCP, ...
- SOAP über JMS
  - z.B. Nutzung von Websphere MQ

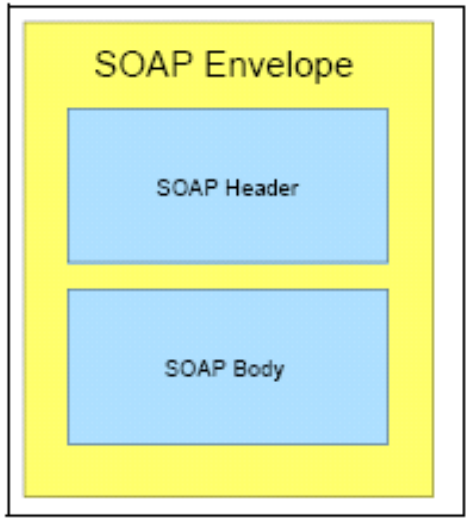
SOAP Nachricht (Simple Object Access Protocol)



- SOAP spezifiziert Nachrichten im XML-Format
- SOAP definiert einen Rahmen für die Inhalte der Nachricht und Regeln, wie diese zu verarbeiten sind
- SOAP definiert auch Regeln für die Bindung an darunterliegende Transport-Protokolle
- Es existieren Codier/Decodier-Schemata für grundlegende Datentypen

```
<?xml version="1.0">
<env:Envelope xmlns:env = http://www.w3c.org/soap-envelope>
        <env:Header> … </env:Header>*
        <env:Body>
                …
                <env:Fault> … </env:Fault>*
                …
        </env:Body>
</env:Envelope>
```

SOAP Nachricht: Bestandteile



- SOAP definiert nicht den Inhalt der einzelnen Elemente wie Header und Body
- Der optionale Header enthält Kontroll-Informationen aus der Anwendungssicht:
  - Welche Knoten müssen was bearbeiten
  - Zustellhinweise
  - Kontext-Informationen bzgl. des Inhalts
- Der Body enthält die eigentliche, applikationsspezifische Nachricht
- Im Falle einer Fehler-Signalisierung besteht der Body aus den Fehlerinformationen

SOAP: Merkmale

- Unterstützte Datentypen:
  Basistypen (string, int, date, ...), structs (records), arrays
  - es gibt Erweiterungen darüber hinaus, z.B. JAX-RPC
    (Java APIs for XML-based RPC)
  - die Kommunikationspartner müssen dies unterstützen

- Kommunikationsstil:
  - Document (Nachrichten-orientiert): Beliebiges Kommunikationsmuster
  - RPC als Spezialform von Request/Reply:
    synchrone Kommunikation, d.h. Kommunikationsschema RPC

- Prinzipiell nicht auf HTTP angewiesen

SOAP Document Beispiel:

Reisereservierung

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
</env:Header>
 <env:Body>
  <p:departure xmlns:p="http://travelcompany.example.org/reservation/travel">
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
  </p:departure>
  <q:lodging
   xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

Quelle: W3C

SOAP RPC-Beispiel Teil 1:

Request mit Input-Parametern

Quelle: W3C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
  <t:transaction
        xmlns:t="http://thirdparty.example.org/transaction"
        env:encodingStyle="http://example.com/encoding"
        env:mustUnderstand="true" >5</t:transaction>
 </env:Header>
 <env:Body>
  <m:chargeReservation
     env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
       xmlns:m="http://travelcompany.example.org/">
   <m:reservation xmlns:m="http://travelcompany.example.org/reservation">
    <m:code>FT35ZBQ</m:code>
   </m:reservation>
   <o:creditCard xmlns:o="http://mycompany.example.com/financial">
    <n:name xmlns:n="http://mycompany.example.com/employees">
        Åke Jógvan Øyvind
    </n:name>
    <o:number>123456789099999</o:number>
    <o:expiration>2005-02</o:expiration>
   </o:creditCard>
  </m:chargeReservation>
 </env:Body>
</env:Envelope>
```

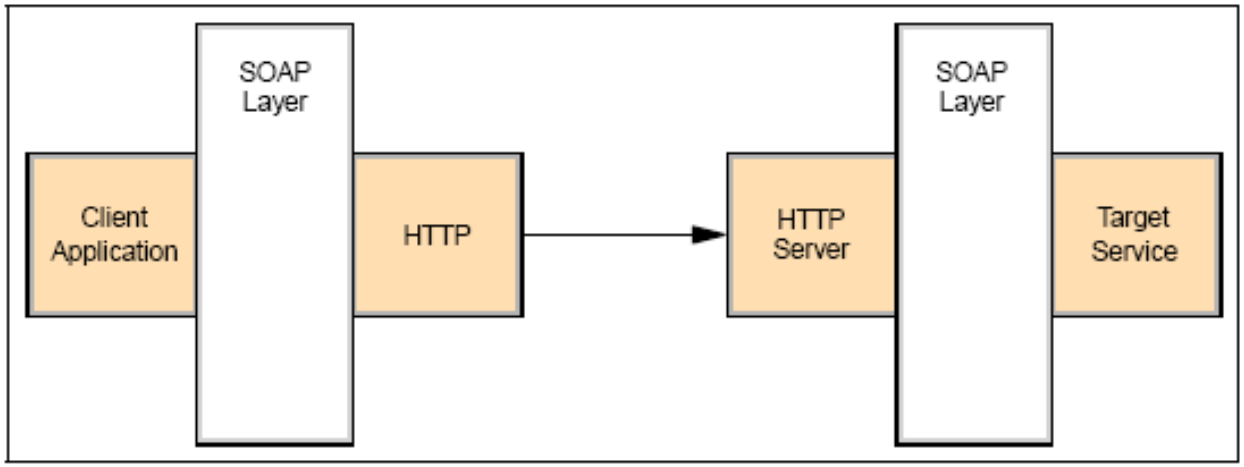
SOAP RPC-Beispiel
Teil 2:

Response mit Output-Parametern

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
    <t:transaction
      xmlns:t="http://thirdparty.example.org/transaction"
       env:encodingStyle="http://example.com/encoding"
        env:mustUnderstand="true">5</t:transaction>
 </env:Header>
 <env:Body>
    <m:chargeReservationResponse
       env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
          xmlns:m="http://travelcompany.example.org/">
      <m:code>FT35ZBQ</m:code>
      <m:viewAt>
       http://travelcompany.example.org/reservations?code=FT35ZBQ
      </m:viewAt>
    </m:chargeReservationResponse>
 </env:Body>
</env:Envelope>
```

Quelle: W3C

Verarbeitung eines Service-Calls



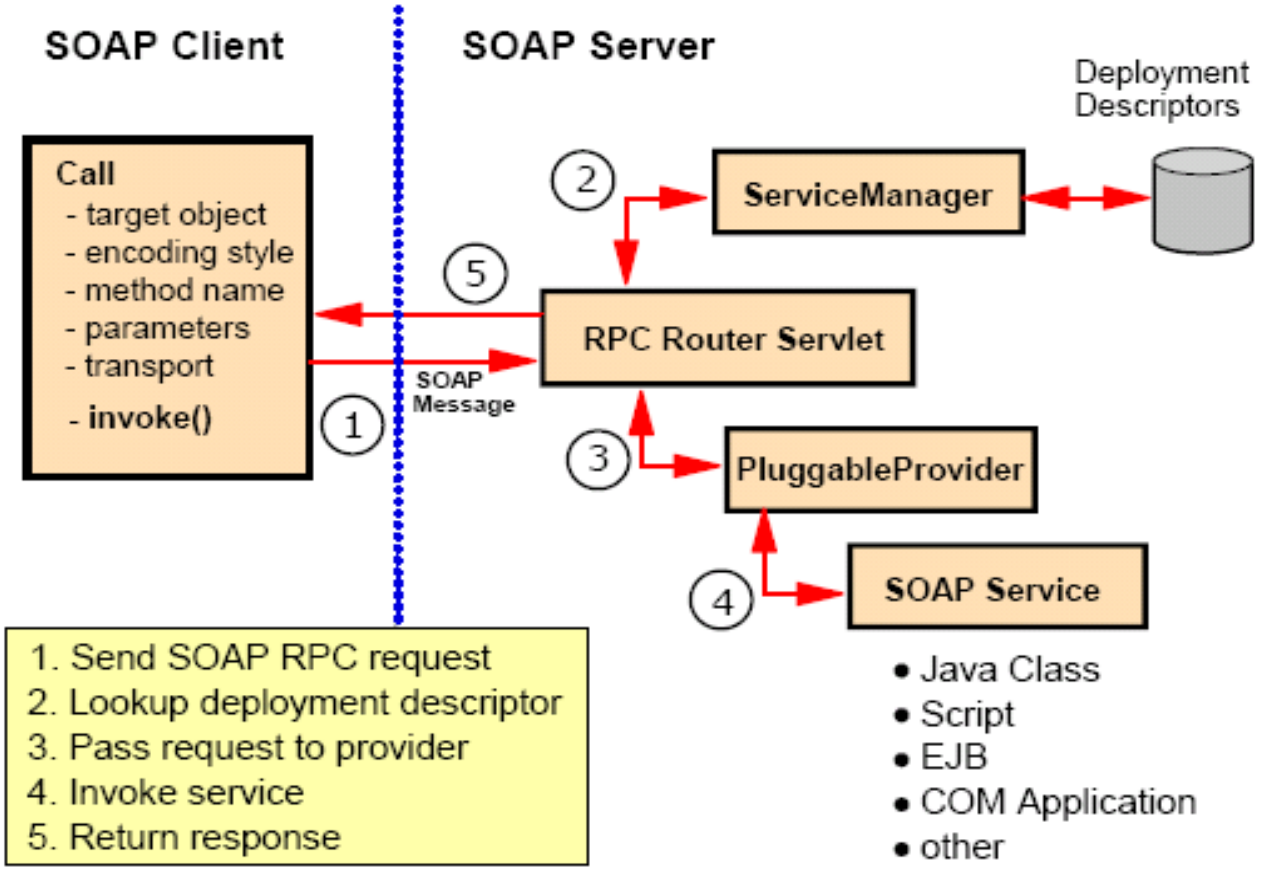
1. Client ruft Methode (Prozedur) via Proxy (Stub) auf
2. SOAP-Layer wandelt den Aufruf in XML-Format
3. SOAP-Layer packt das XML-Format in ein SOAP-Paket (ebenfalls XML-Format)
4. SOAP-Nachricht wird an URI verschickt

5. Web-Server empfängt SOAP-Nachricht
6. SOAP-Layer extrahiert den XML-Aufruf aus der SOAP-Nachricht
7. SOAP-Layer wandelt das XML-Format in einen Aufruf für die Zielumgebung
8. Der Service wird lokalisiert und aufgerufen

Generierung des SOAP-Layers

- Die Aufgaben des SOAP-Layers auf der Client- sowie Server-Seite können in hohem Maße automatisiert werden.
- Es existieren für viele Zielsprachen, Laufzeitumgebungen und Transportmechanismen entsprechende Werkzeuge für die Entwickler
- Häufig sind die Services schon vorhanden:
  Entsprechende Werkzeuge generieren dann anhand des vorhandenen Codes möglichst komplett den SOAP-Layer
  - Apache Axis
  - Microsoft .NET Tools

SOAP: Interaktion zwischen Client und Server am Bsp. von Axis

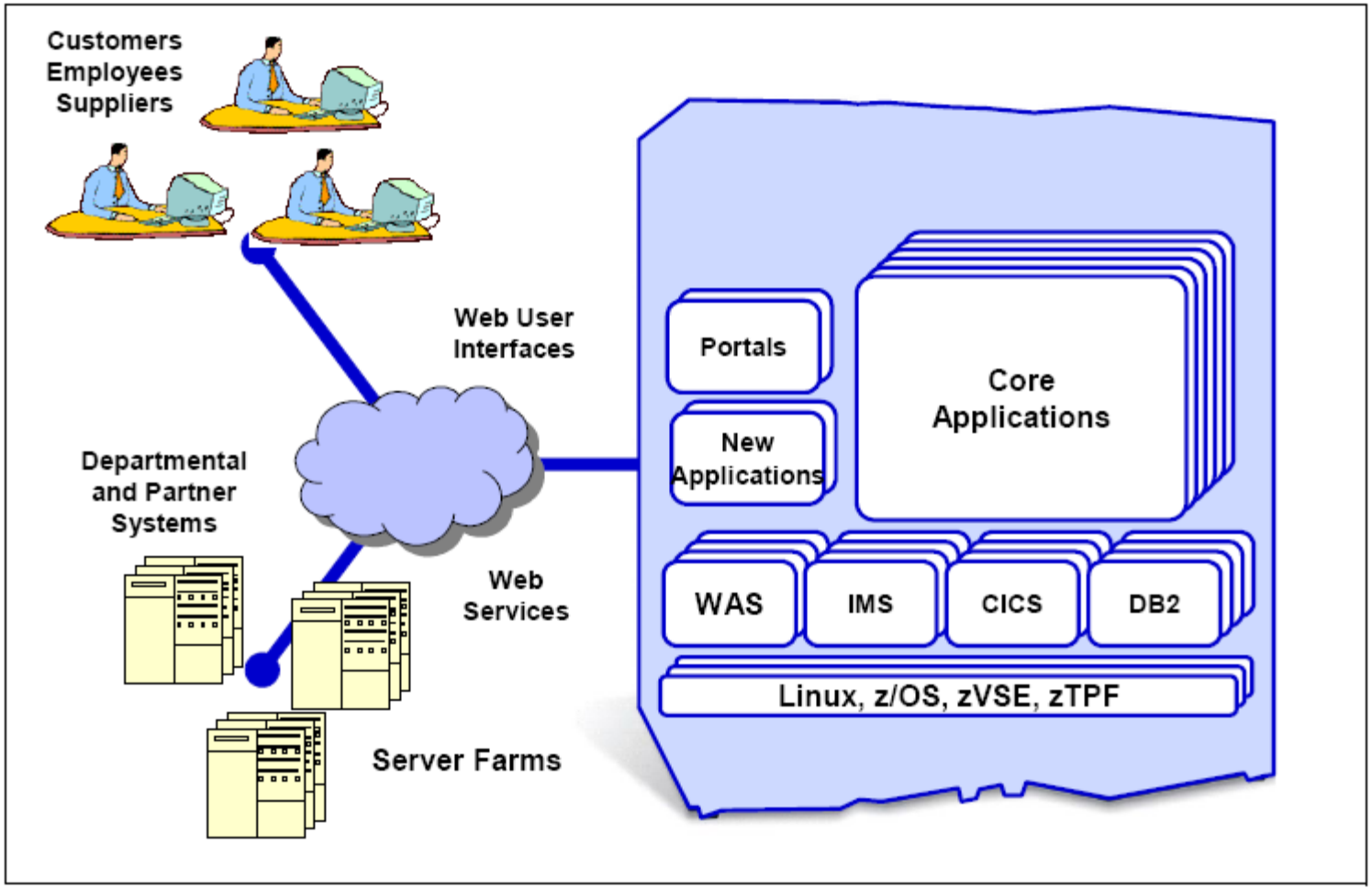


Quelle: IBM, 2005

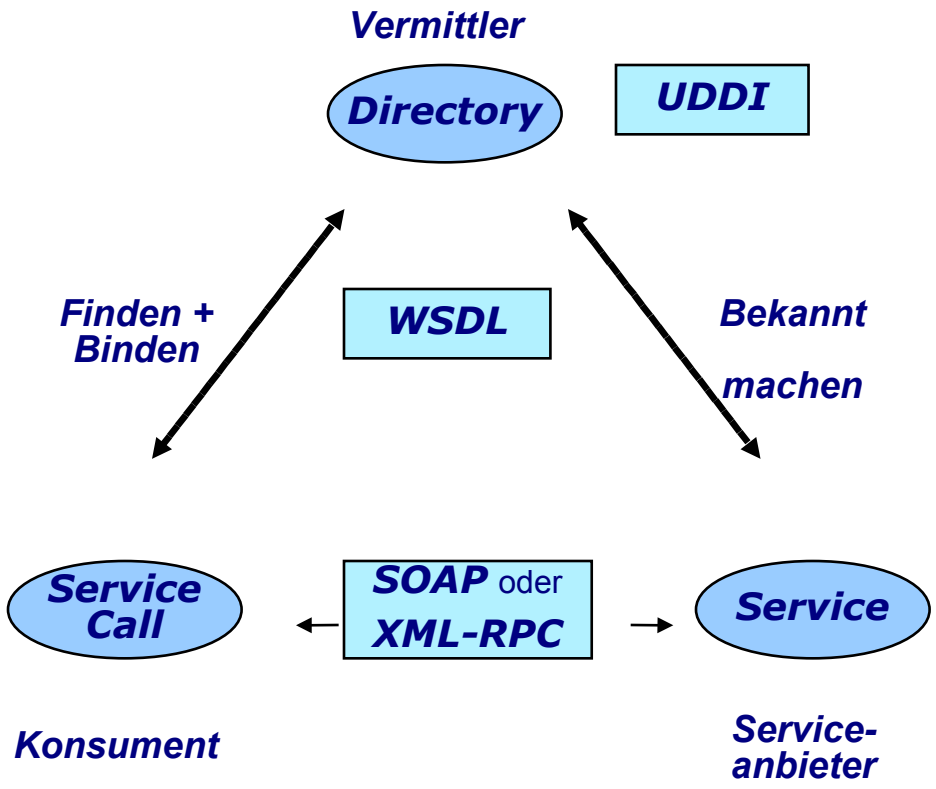
SOAP Implementierungen

- Apache Axis (Weiterentwicklung Apache SOAP, ursprüngliche Basis „SOAP4J“ von IBM)
- IBM Websphere (Apache SOAP 2.3)
- Microsoft SOAP Toolkit (Teil des .NET-Frameworks)
- Oracle Application Server
- Bea WebLogic
- sowie alle weiteren führenden (oder auch nicht) Hersteller von Applikationsservern
- Vielzahl von freien Implementierungen für bestimmte Zielsprachen wie PHP, Phyton, ...

Einbindung der klassischen Host-Welt

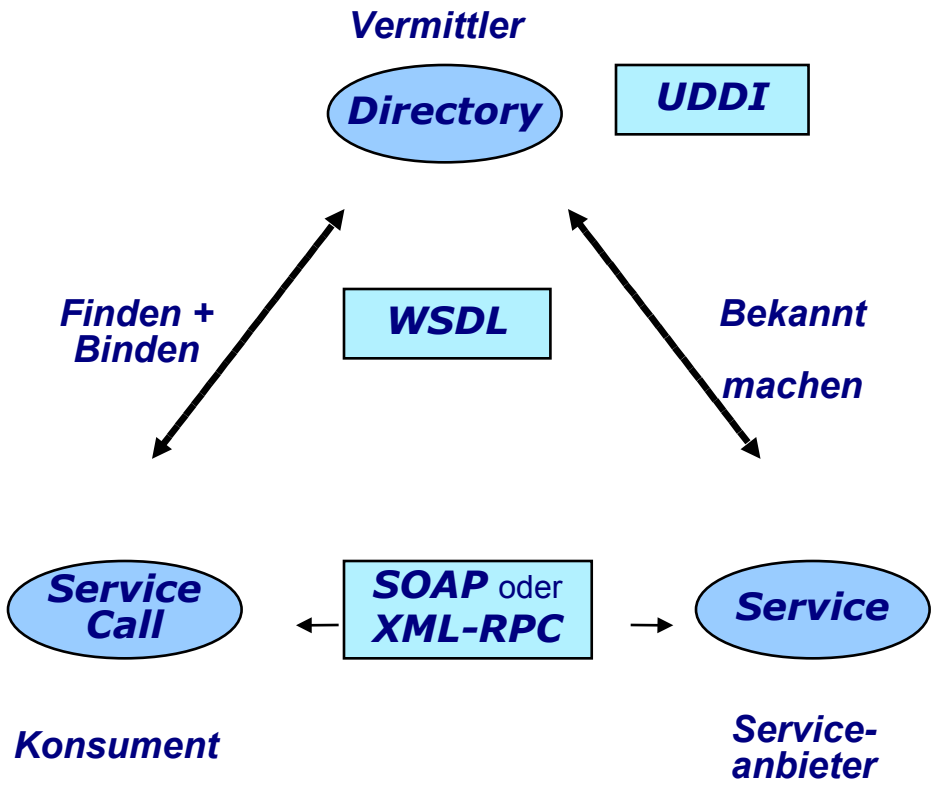


WSDL (Web Service Description Language)



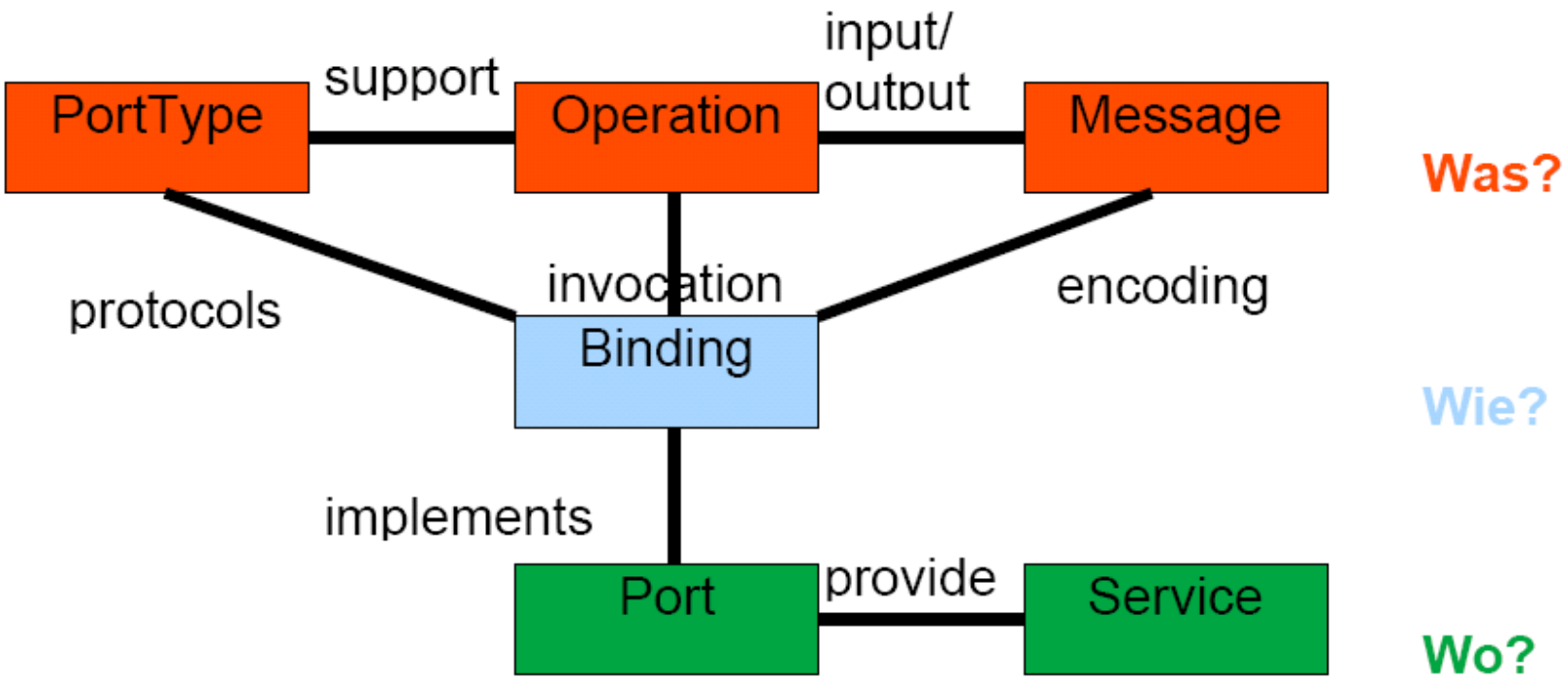
- Wie bei früheren Konzepten kommt auch hier der Schnittstelle zwischen Aufrufer und Aufgerufenem überragende Bedeutung zu
- Der Service wird mittels eines XML-Dialekts, der WSDL, beschrieben
- WSDL entspricht den IDLs anderer Konzepte
- Damit ist quasi die Selbstbeschreibung gegeben (self description)
- Auch hier kann mittels entsprechender Werkzeuge ein erheblicher Anteil der Codierungen auf der Client- bzw. Server-Seite generiert werden
- Die neueste Version ist WSDL Version 2.0 (W3C Recommendation 26 June 2007), am Markt etabliert ist WSDL Version 1.1 (Mitte 2007)

WSDL (Web Service Description Language)



- Wie bei früheren Konzepten kommt auch hier der Schnittstelle zwischen Aufrufer und Aufgerufenem überragende Bedeutung zu
- Der Service wird mittels eines XML-Dialekts, der WSDL, beschrieben
- WSDL entspricht den IDLs anderer Konzepte
- Damit ist quasi die Selbstbeschreibung gegeben (self description)
- Auch hier kann mittels entsprechender Werkzeuge ein erheblicher Anteil der Codierungen auf der Client- bzw. Server-Seite generiert werden
- Die neueste Version ist WSDL Version 2.0 (W3C Recommendation 26 June 2007), am Markt etabliert ist WSDL Version 1.1 (Mitte 2007)

WSDL: Was wird beschrieben



WSDL: Struktur der Beschreibung

```
<wsdl:definitions  xmlns:wsdl = „http://w3.org/...“>
  <wsdl:documentation ... />
  <wsdl:types>  Schema Imports  </wsdl:types>
  <wsdl:message>  Nachrichten  </wsdl:message>
  <wsdl:portType>  Operationen </wsdl:portType>
  <wsdl:binding> Protokolle und Formate </wsdl:binding>
  <wsdl:service> Service Definition </wsdl:service>
</wsdl:definitions>
```

WSDL: Beispiel Teil 1

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://address.jaxrpc.samples"
   xmlns:apachesoap="http://xml.apache.org/xml-soap"
   xmlns:impl="http://address.jaxrpc.samples"
   xmlns:intf="http://address.jaxrpc.samples"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <wsdl:types>
   <schema targetNamespace="http://address.jaxrpc.samples"
         xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="AddressBean">
    <sequence>
     <element name="street" nillable="true" type="xsd:string"/>
     <element name="zipcode" type="xsd:int"/>
    </sequence>
   </complexType>
   <element name="AddressBean" nillable="true" type="impl:AddressBean"/>
  </schema>
  <import namespace="http://www.w3.org/2001/XMLSchema"/>
 </wsdl:types>
```

Quelle: IBM, 2005

WSDL: Beispiel Teil 2

```
<wsdl:message name="updateAddressRequest">
     <wsdl:part name="in0" type="intf:AddressBean"/>
     <wsdl:part name="in1" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="updateAddressResponse">
     <wsdl:part name="return" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="updateAddressFaultInfo">
     <wsdl:part name="fault" type="xsd:string"/>"
</wsdl:message>

<wsdl:portType name="AddressService">
     <wsdl:operation name="updateAddress" parameterOrder="in0 in1">
        <wsdl:input message="intf:updateAddressRequest"
                    name="updateAddressRequest"/>
        <wsdl:output message="intf:updateAddressResponse"
                    name="updateAddressResponse"/>
        <wsdl:fault message="intf:updateAddressFaultInfo"
                    name="updateAddressFaultInfo"/>
     </wsdl:operation>
</wsdl:portType>
```

WSDL: Beispiel Teil 3

```
<wsdl:binding name="AddressSoapBinding" type="intf:AddressService">
     <wsdlsoap:binding style="rpc"
                      transport="http://schemas.xmlsoap.org/soap/http"/>
     <wsdl:operation name="updateAddress">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="updateAddressRequest">
           <wsdlsoap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="http://address.jaxrpc.samples" use="encoded"/>
        </wsdl:input>
        <wsdl:output name="updateAddressResponse">
           <wsdlsoap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="http://address.jaxrpc.samples" use="encoded"/>
        </wsdl:output>
        <wsdl:fault name="updateAddressFaultInfo">
           <wsdlsoap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="http://address.jaxrpc.samples" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
</wsdl:binding>
```

Quelle: IBM, 2005

WSDL: Beispiel Teil 4

```
 <wsdl:service name="AddressServiceService">
      <wsdl:port binding="intf:AddressSoapBinding" name="Address">
         <wsdlsoap:address
                location="http://localhost:8080/axis/services/Address"/>
      </wsdl:port>
 </wsdl:service>

</wsdl:definitions>
```

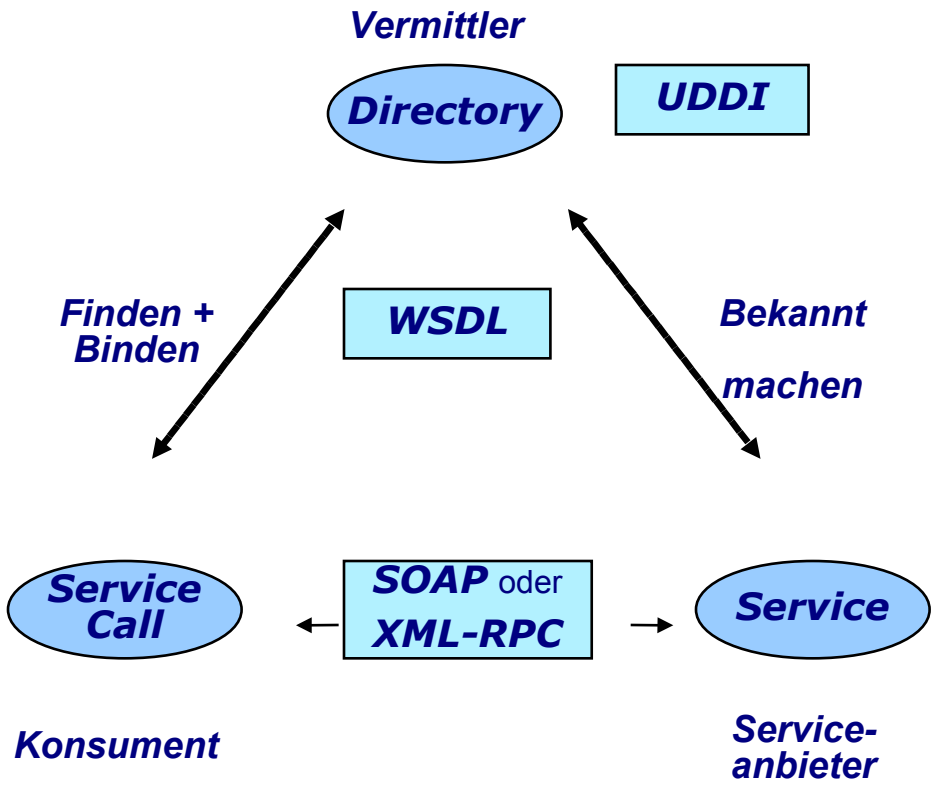
Quelle: IBM, 2005

WSDL: Operationen, die ein Port unterstützen kann

One-way	The port receives a message. There is an *input message* only.
Request-response	The port receives a message and sends a correlated message. There is an input message followed by an *output message*.
Solicit-response	The port sends a message and receives a correlated message. There is an output message followed by an input message.
Notification	The port sends a message. There is an output message only. This type of operation could be used in a publish/subscribe scenario.

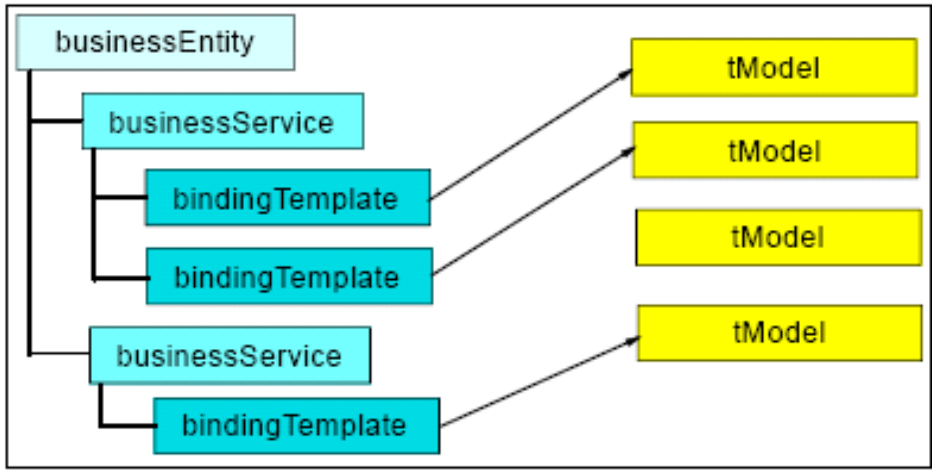
Quelle: IBM, 2005

Was ist UDDI



- Universal Description, Discovery, and Integration
- UDDI spezifiziert Protokole für:
  - Veröffentlichung von Services in Verzeichnissen
  - Suchmöglichkeiten nach Services in Verzeichnissen
  - Zugriffskontrolle auf Verzeichnisse
  - Verteilung auf verschiedene Verzeichnisse
- Bietet Lokations- und Aufruf-Metadaten für die dynamische Bindung von Konsumenten an Services zur Laufzeit
- Wird durch das OASIS Standardisierungs-Gremium verwaltet

UDDI: Struktur eines Informationseintrags



- **businessEntity** - Informationen (Metadaten) über den Anbieter (z.B. Unternehmen, Name, Beschreibung), Auflistung über die Angebotenen Dienste
- **businessService** - allgemeine Beschreibung einer Serviceklasse (Metadaten über den Dienst), repräsentiert eine logische Dienst-Klassifikation und ist Kind einer businessEntity Struktur
- **bindingTemplate** - beschreibt technische Eigenschaften des Services
- **t-Modell** - Referenzen auf technische Anforderungen, Speicherung anhand generischer Daten

UDDI Implementierungen

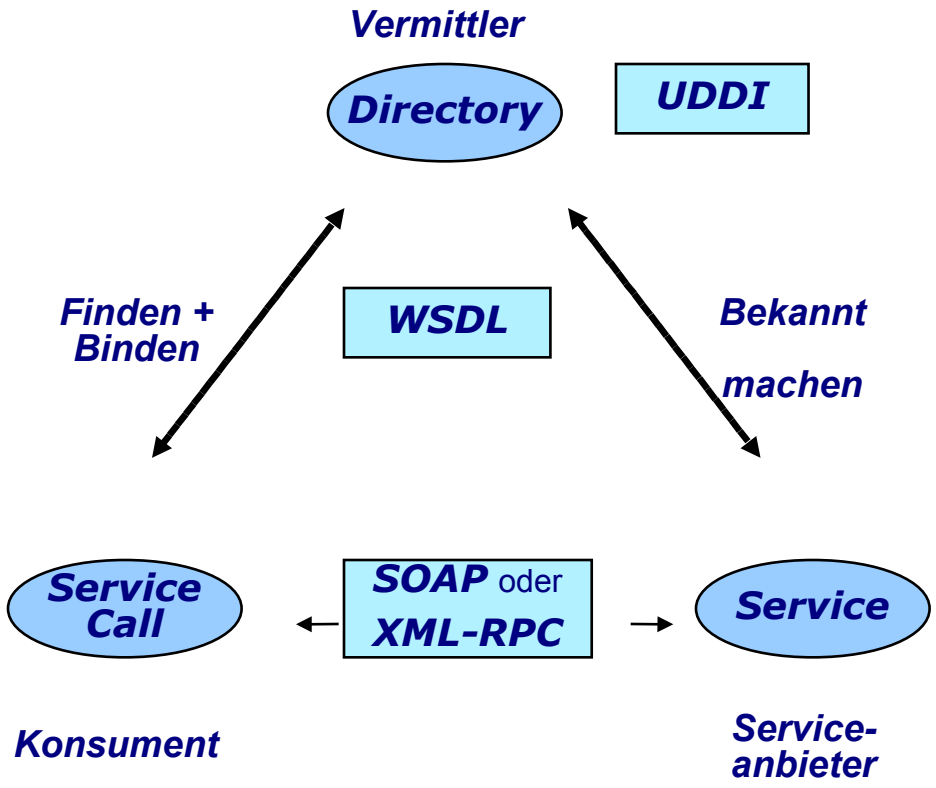
UDDI Clients

- uddi4j: UDDI for Java
- UDDI.NET SDK: UDDI for Microsoft .NET
- uddi4r: UDDI for Ruby
- uddi4py: UDDI for Python
- UDDI::Lite: UDDI for Perl
- ...

UDDI Server

- Apache jUDDI: Open-Source UDDI Server
- BEA Aqualogic Service Registry
- Novell nSure UDDI Server: Open-Source UDDI Server
- Microsoft Enterprise UDDI Server: Part of Windows 2003 Server
- Systinet Registry
- Oracle Service Registry
- Software AG CentraSite
- SAP Enterprise SOA PI7.1
- ...

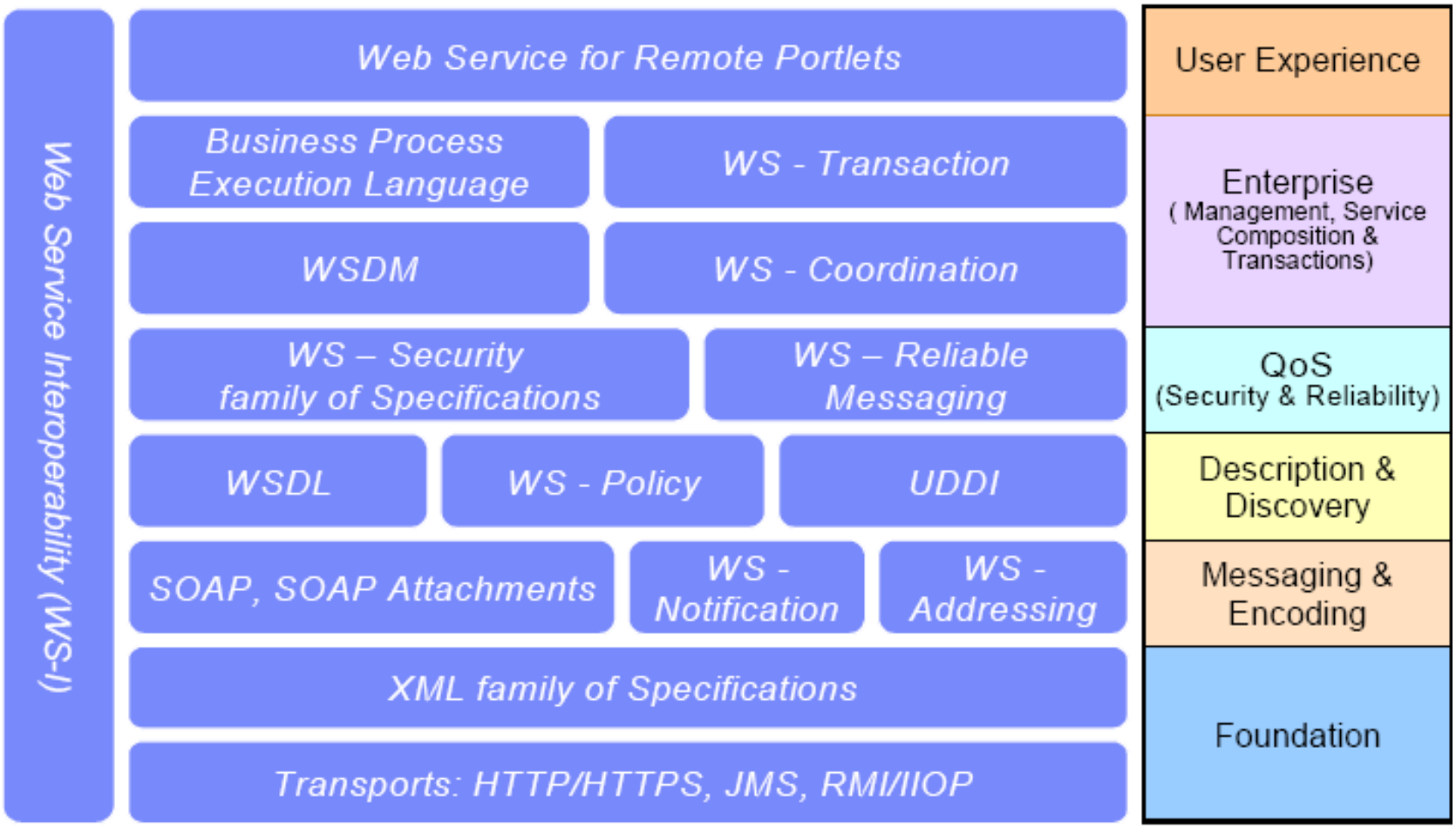
Wenn wir das mittels Java-Applikationen zugreifen:



Programmier-Schnittstellen

- Web Services for J2EE
- Java APIs for XML Messaging
- Java APIs for XML Registry
- Java APIs for XML-Based RPC
- Java APIs for WSDL

Web Services: Entwicklungen/Standards



Quelle: IBM

Diskussion

- Vorteile/Nachteile
- Best Practices

Vielen Dank für Ihre Aufmerksamkeit!

Noch Fragen…?

Ihre Ansprechpartner für die IT Trainings Roadmap:

Projektleitung:

Annette Heinemann
06083/9130-27
annette.heinemann@
freund-dirks.de

IT Fachberatung:

Stephan Karrer
06083/9130-0
stephan.karrer@
freund-dirks.de