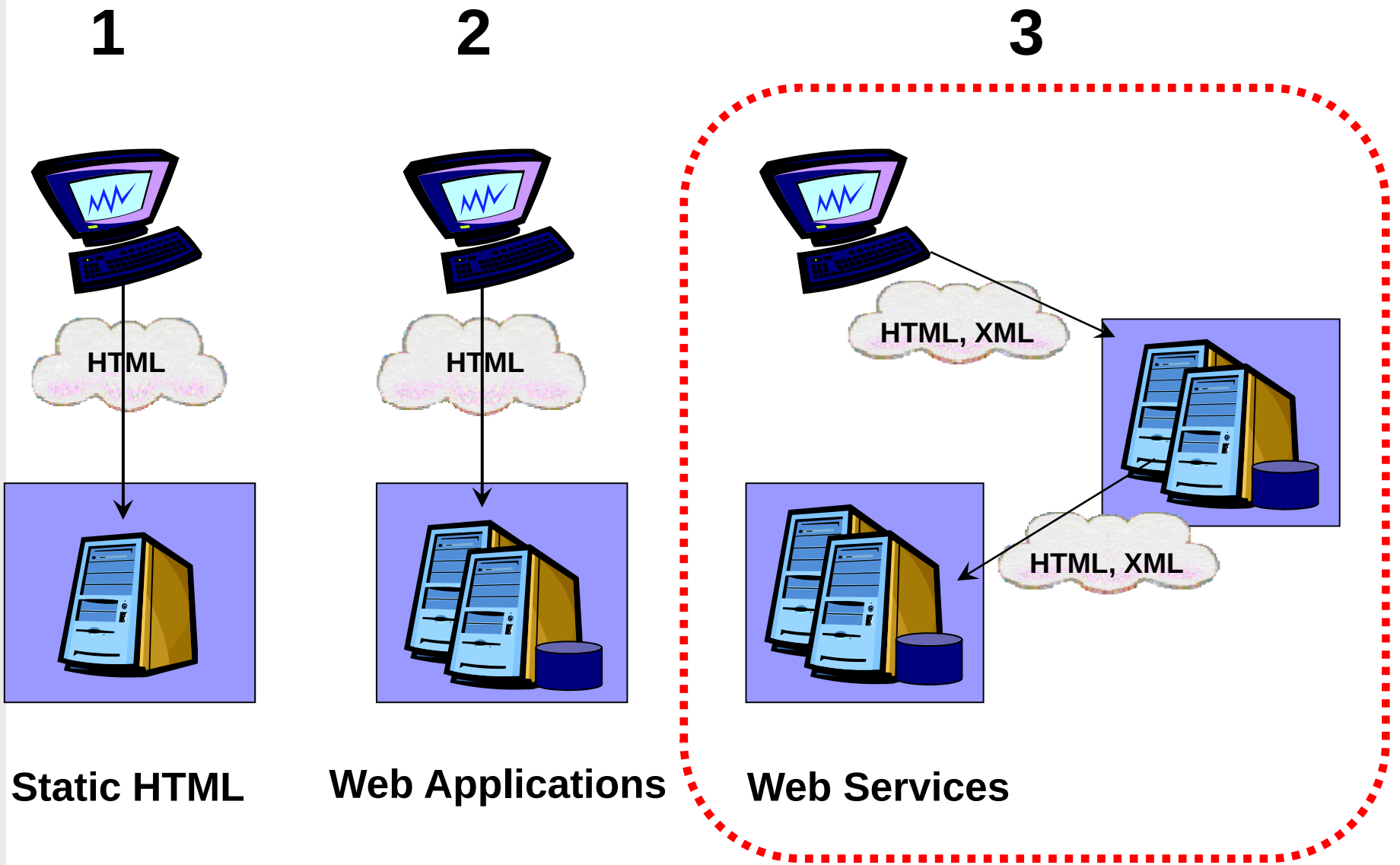
Vorlesung am 9. Juni 2009

Serviceorientiertes E-Government

Web Services und Verzeichnisdienste

Dr. Frank Sarre
Lehrbeauftragter der LMU München

Evolution der Internet-Technologie



Was ist ein Web Service?

W3C-Definition

„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.“

Hier ist nicht die Rede von Mensch-Maschine-Interaktion.
Benutzer (Menschen) nutzen Web Services nur mittelbar.

Grundlegende Eigenschaften von Web Services

Web Services ….

- … bieten für andere **Programme** (innerhalb und außerhalb des Unternehmens) **plattform- und programmiersprachenunabhängige Dienste** an
- Die (sich z.T. **selbst beschreibenden**) Dienste eines Web Services können im Web **publiziert**, **lokalisiert** und **aufgerufen** werden
- … basieren auf bekannten **Web-Technologie-Standards**
- … tauschen mit ihren Aufrufern **Nachrichten** (in einem Standardformat) aus

Web Services erleichtern den **Aufbau** und die **Integration** verteilter Web-Applikationen

Beispiele für den Einsatz von Web Services

- Internet-Reisebuchungssysteme
- Datenaustausch zwischen eBusiness-Partnern
- Supply Chain Management
- (Informations-) Portale
- E-Procurement, elektronische Marktplätze
- E-Payment, Billing

Aufgaben von Web Services (1)

Aufgaben

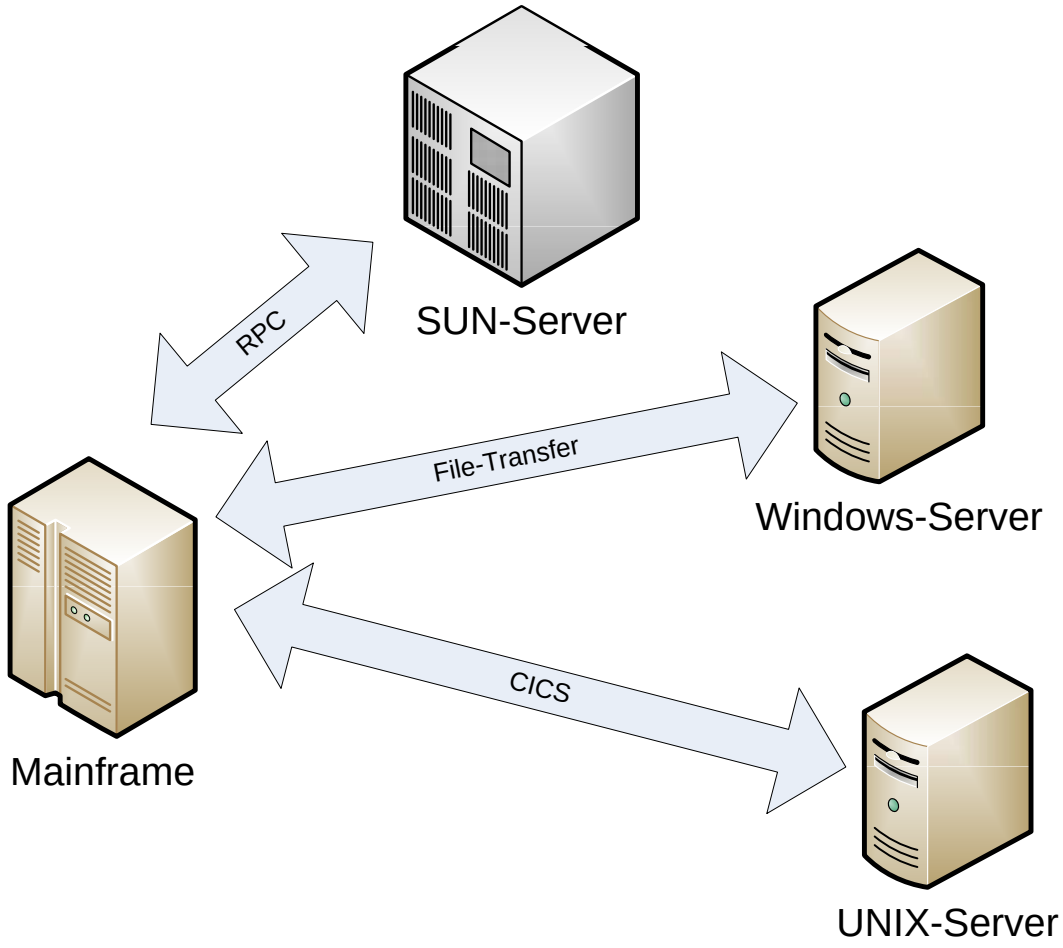
- Aufruf von entfernten Prozeduren
- Übermittlung von Daten

Aber auch:

- Beseitigung proprietärer Schnittstellen
- Beseitigung komplexer Kommunikationsprotokolle
- Herstellung einer losen Kopplung
- Reduktion der Abhängigkeit von einem oder mehreren Herstellern
- …

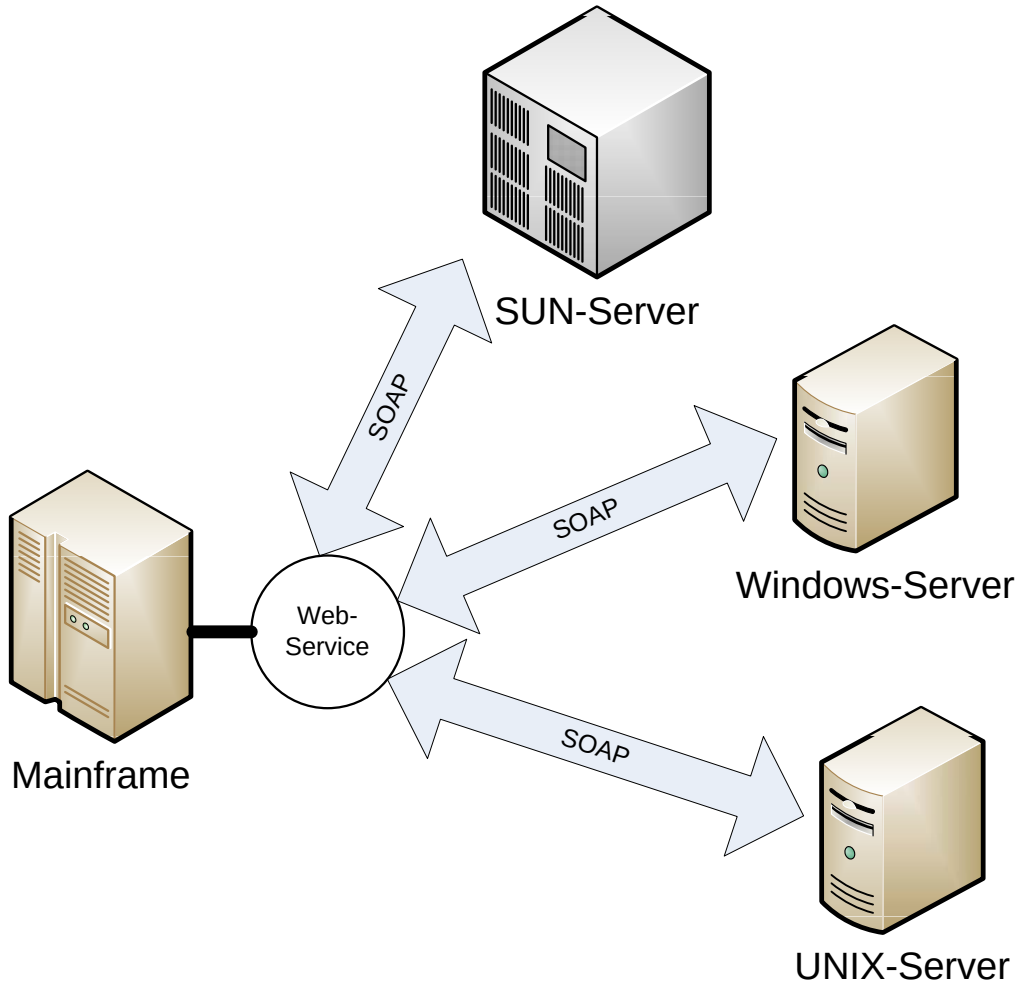
Aufgaben von Web Services (2)

Ausgangssituation



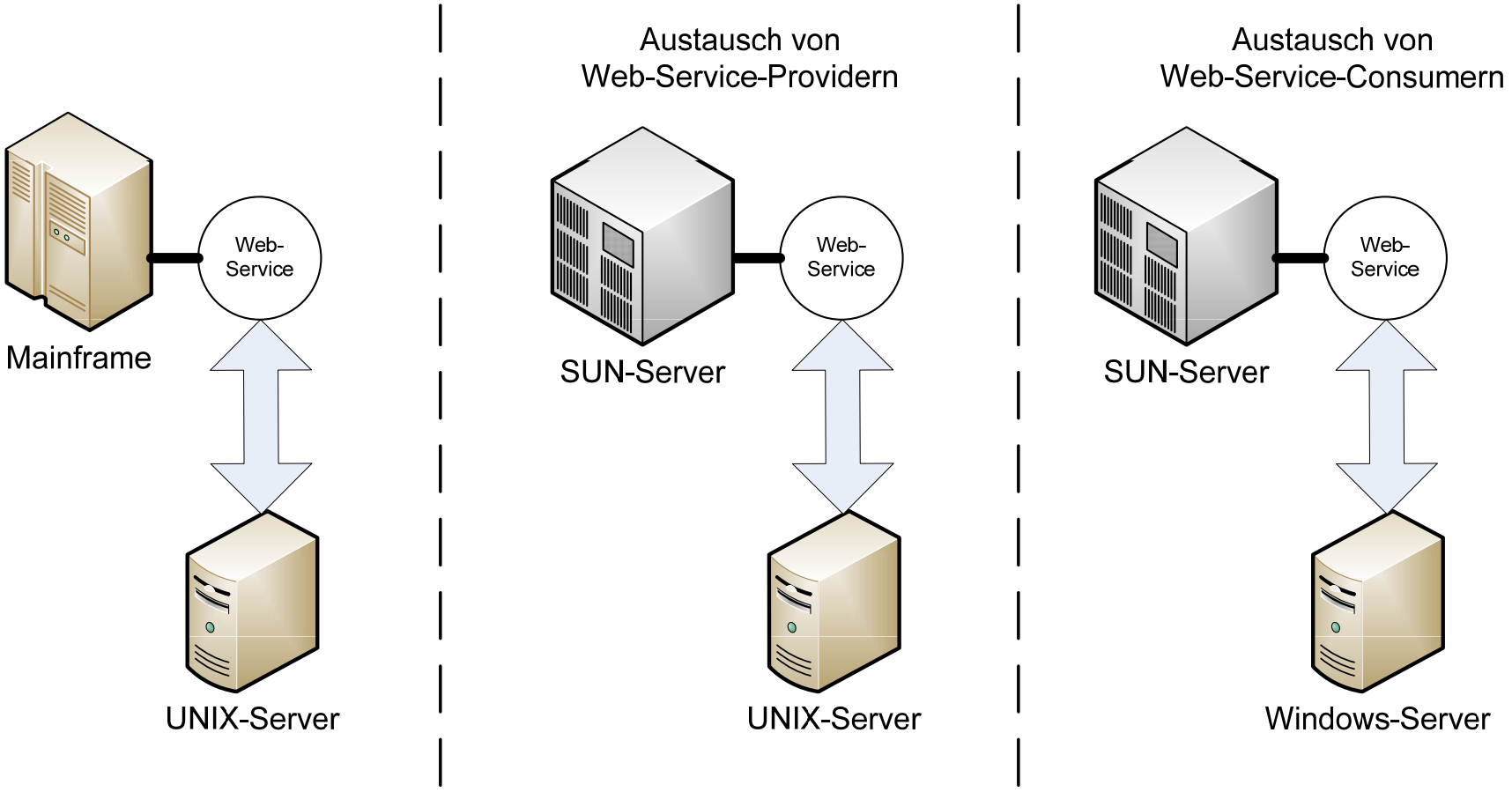
Aufgaben von Web Services (3)

Transparente Kommunikation unabhängig von Hardware, Betriebssystem und Netzwerk



Aufgaben von Web Services (4)

Verbesserte Austauschbarkeit von Systemen



Was bieten Web Services von Haus aus nicht?

- Sicherheit
- Transaktion
- Session Handling
- Autorisierung / Authentisierung
- Hohe Performance
- Hohe Verfügbarkeit
- Abrechnungsmodelle für die Nutzung
- Konzepte für die Anwendungsentwicklung im großen Stil
- …

Grundlegende Standards für Web Services

Standards

- SOAP (Kommunikationsprotokoll)
- WSDL (Service-Beschreibung)
- UDDI (Verzeichnisdienst)

Standardisierungsgremien

- W3C – World Wide Web Consortium (www.w3c.org)
  → Web-Standards, z.B. XML, WSDL
- OASIS - Organization for the Advancement of Structured Information Standards (www.oasis-open.org)
  → erweiterte WS-Standards, z.B. WS-Security
- WS-I – Web Services Interoperability Organization (www.ws-i.org)
  → Standardprofile, z.B. Basic Profile

Technische Basis von SOAP

- SOAP basiert auf **XML**
- Kommunikation in Form von „**SOAP-Messages“**

  Es gibt zwei Message-Typen:

  a) SOAP-**Request** (Aufruf eines Service)

  b) SOAP-**Response** (Antwort eines Service)

  Standardisierte Struktur für Fehlermeldungen, die in einer SOAP-Response übermittelt werden

Die aktuelle SOAP-Version ist **1.2**

SOAP und HTTP

Transport von SOAP-Nachrichten

- Grundsätzlich ist SOAP unabhängig vom Transportprotokoll
- In der Praxis wird jedoch fast immer HTTP verwendet

Warum HTTP?

- Weit verbreitetes Standard-Protokoll
- Auf allen Plattformen verfügbar
- Einfach zu implementieren
- Firewall-freundlich (aber: Sicherheitsrisiko!)

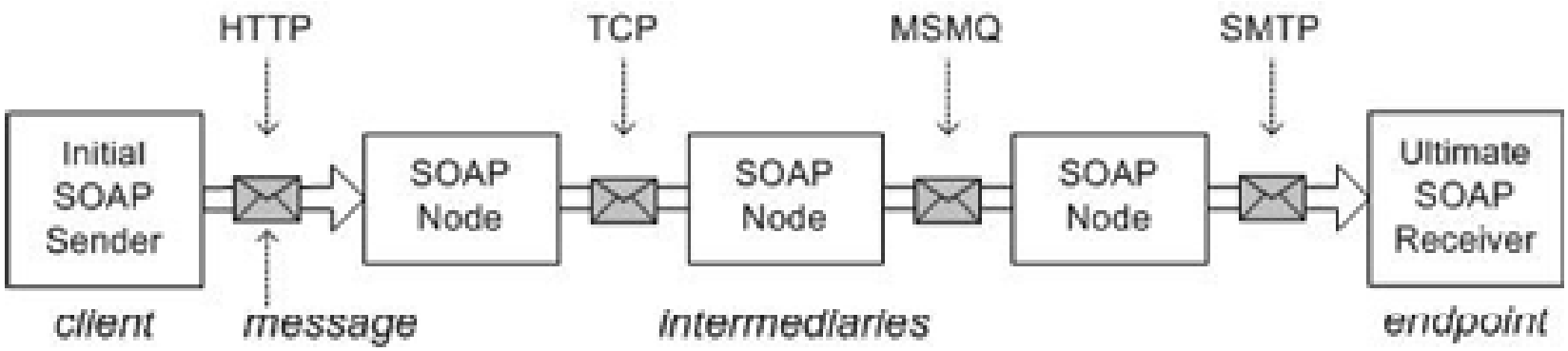
Nachteile von HTTP

- Synchrones Protokoll

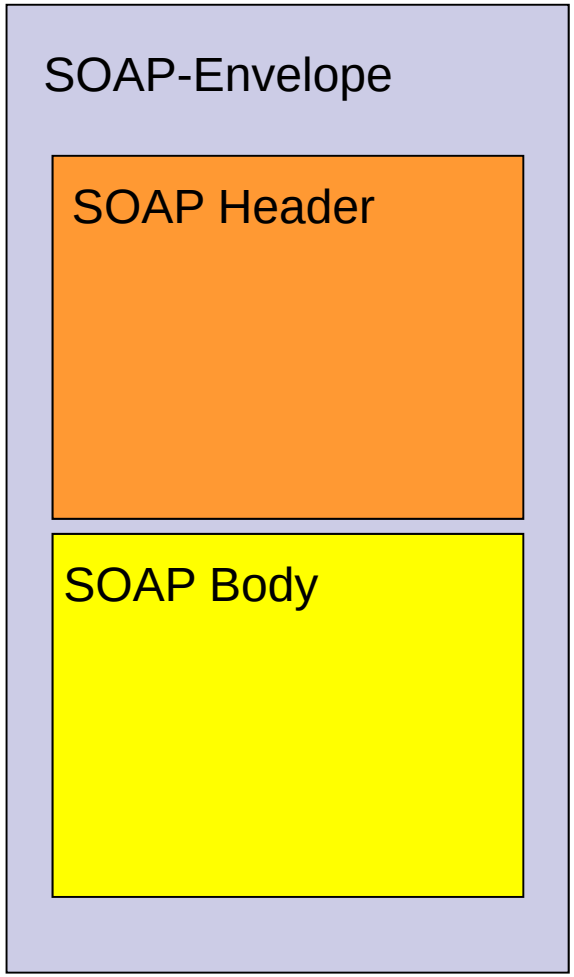
Beispiel einer SOAP-Kommunikationskette

Quelle: http://msdn.microsoft.com/en-us/library/ms995800.aspx

So könnte eine SOAP-Kommunikationskette **mit verschiedenen Transportprotokollen** aussehen:



Struktur einer SOAP-Nachricht



<SOAP-ENV:**Envelope**>

<SOAP-ENV:**Header**>

- Kein zwingender Bestandteil der Nachricht
- Parameter, die die Kommunikation steuern
- Gegenstand vieler Erweiterungen des Standards, wie z.B. WS-Addressing

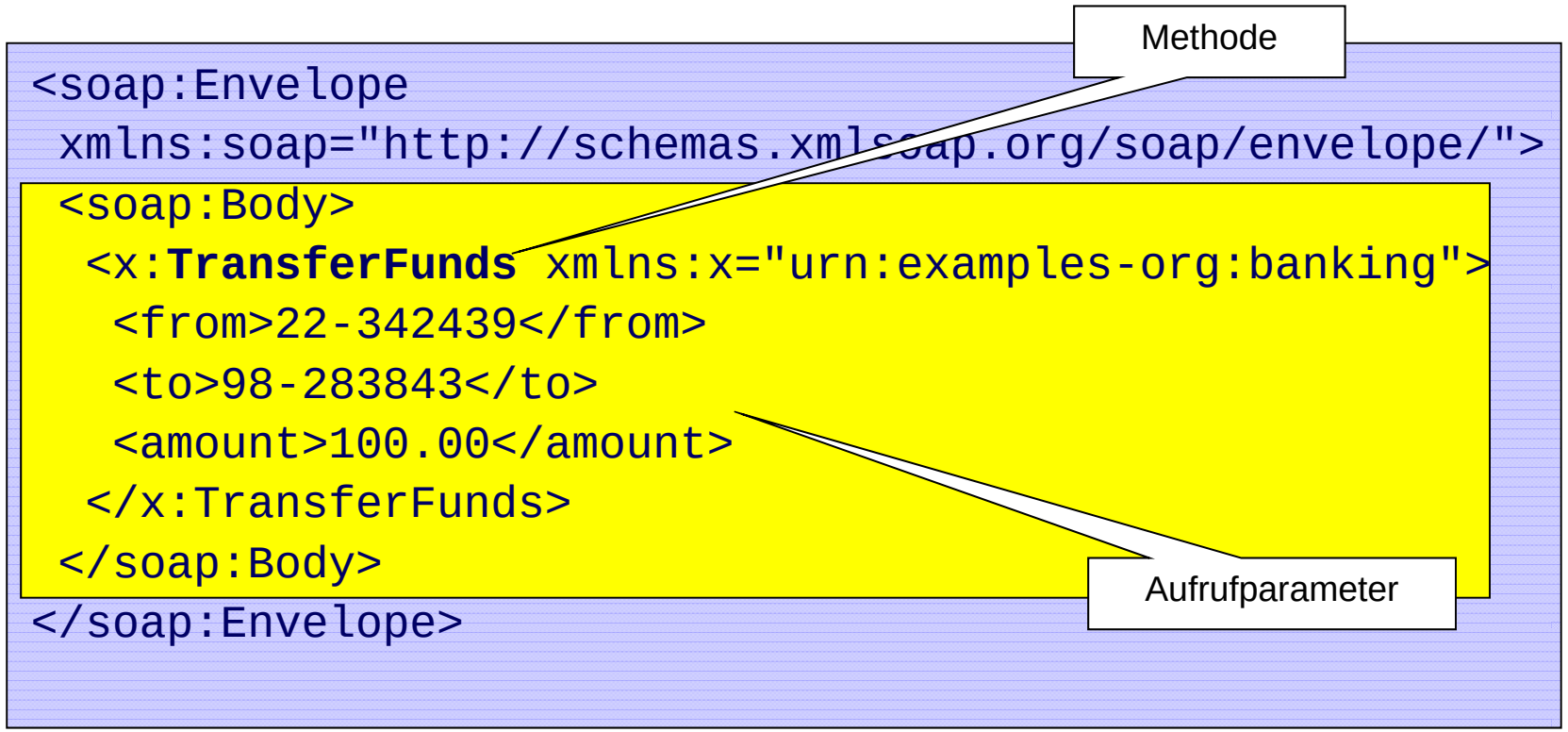
</SOAP-ENV:Header>

<SOAP-ENV:**Body**>

- Daten
- Funktionsaufrufe
- Funktionsparameter

</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

Struktur eines SOAP-Requests



Es handelt sich um einen „RPC-Style“-Request,
d.h. Aufruf einer Methode mit Parametern

SOAP im HTTP-Request

```
POST /soap/handlefunds HTTP/1.0
Host: localhost:80
Content-Type: text/xml; charset=utf-8
Content-Length: 265
SOAPAction: ""
```

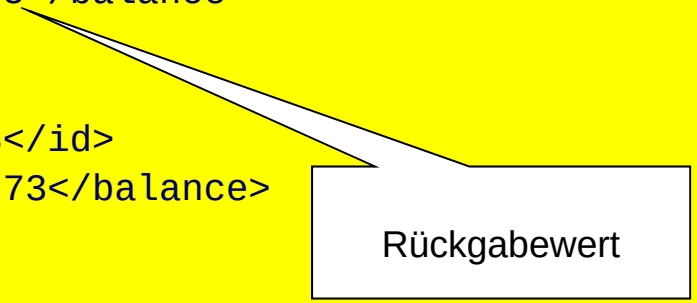
HTTP

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <x:TransferFunds xmlns:x="urn:examples-org:banking">
   <from>22-342439</from>
   <to>98-283843</to>
   <amount>100.00</amount>
  </x:TransferFunds>
 </soap:Body>
</soap:Envelope>
```

SOAP

Struktur einer SOAP-Response

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <x:TransferFundsResponse
   xmlns:x="urn:examples-org:banking">
   <balances>
    <account>
     <id>22-342439</id>
     <balance>33.45</balance>
    </account>
    <account>
     <id>98-283843</id>
     <balance>932.73</balance>
    </account>
   </balances>
  </x:TransferFundsResponse>
 </soap:Body>
</soap:Envelope>
```



SOAP-Styles (stark vereinfacht)

Document style

```
<soap:Body>
   <m:purchaseOrder xmlns:m="someURI">
   ...
   </m:purchaseOrder>
</soap:Body>
```

Inhalt der Nachricht ist frei strukturierbar und muss durch Sender und Empfänger validiert werden, z.B. über ein XML-Schema.

RPC style

Verwendung eines Methodennamens

```
<soap:Body>
   <m:placeOrder xmlns:m="someURI">
        <m:purchaseOrder>
        ...
        </m:purchaseOrder>
   </m:placeOrder>
</soap:Body>
```

Parameter können durch Standarddatentypen abgebildet und automatisch validiert werden.

Im Standardschema sind bereits 40 Datentypen definiert.

SOAP-Header

- Wichtige Standardattribute
  - **encodingStyle**

    Informationen, wie Daten in einer SOAP-Message serialisiert werden sollen
  - **role**

    Spezifiziert Empfänger bzw. Zwischenstation, die dieses Header-Element verarbeiten darf
  - **mustUnderstand**

    Legt fest, ob ein Header-Element bei der Weiterleitung der SOAP-Message ausgewertet werden muss
- Individuelle Erweiterungen sind möglich
- Zahlreiche standardisierte Erweiterungen verfügbar, z.B. WS-Addressing, WS-Security

Beispiel für einen SOAP-Header

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Header>
  <!-- security credentials -->
  <s:credentials xmlns:s="urn:examples-org:security"
   soap:mustUnderstand="1">
   <username>dave</username>
   <password>evad</password>
  </s:credentials>
 </soap:Header>
 …
</soap:Envelope>
```

Das Beispiel stellt dar, dass der Empfänger dieser Nachricht die credentials <username> und <password> verstehen und auswerten **muss**.

SOAP-Fault

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <soap:Fault>
   <faultcode>soap:Receiver</faultcode>
   <faultstring>Insufficient funds</faultstring>
   <detail>
    <x:TransferError xmlns:x="urn:examples-org:banking">
     <sourceAccount>22-342439</sourceAccount>
     <transferAmount>100.00</transferAmount>
     <currentBalance>89.23</currentBalance>
    </x:TransferError>
   </detail>
  </x:TransferFunds>
 </soap:Body>
</soap:Envelope>
```

<faultcode> und <faultstring> sind Pflichtinformationen

<detail> ist optional

SOAP-Fault-Codes

- **VersionMissmatch**

  Ein Knoten (ein Kommunikationspartner in der gesamten Kommunikationskette) erwartet eine andere SOAP-Version

- **MustUnderstand**

  Ein Knoten kann eine Pflichtinformation im Header nicht auswerten

- **DataEncodingUnknown**

  Es sind Datentypen verwendet worden, die nicht in eine SOAP-Nachricht übersetzt werden konnten

- **Sender**

  Die Nachricht konnte vom Sender nicht verarbeitet werden

- **Receiver**

  Die Nachricht konnte vom Empfänger nicht verarbeitet werden

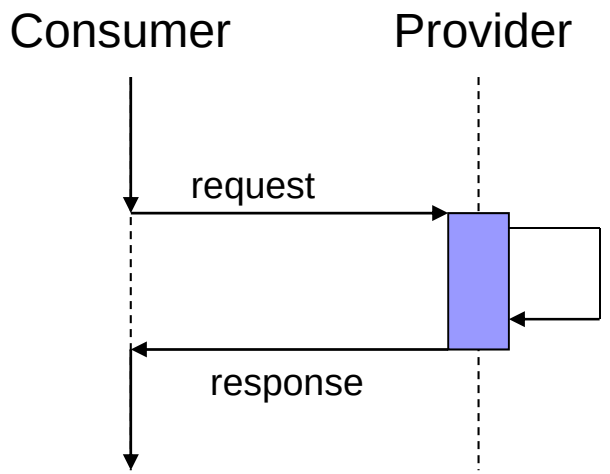
Message Exchange Patterns (1)

- Nachrichten werden in einer festgelegten Abfolge verschickt, sog. **Message Exchange Patterns** (MEP)
- Die wichtigsten Patterns sind:
  - Request / Response
  - One-Way
  - Request / Callback
  - Publish / Subscibe

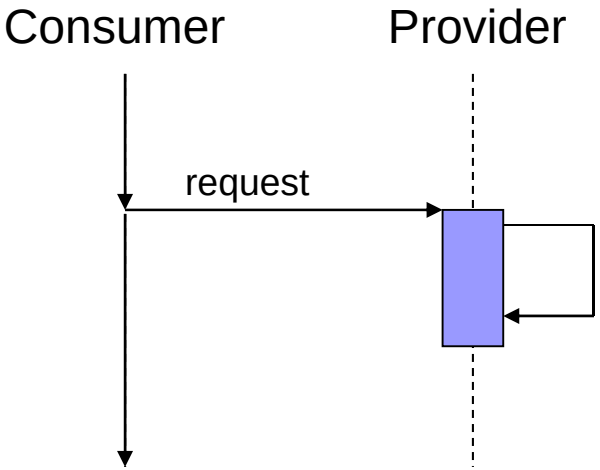
Message Exchange Patterns (2)

Request / Response

„synchron“

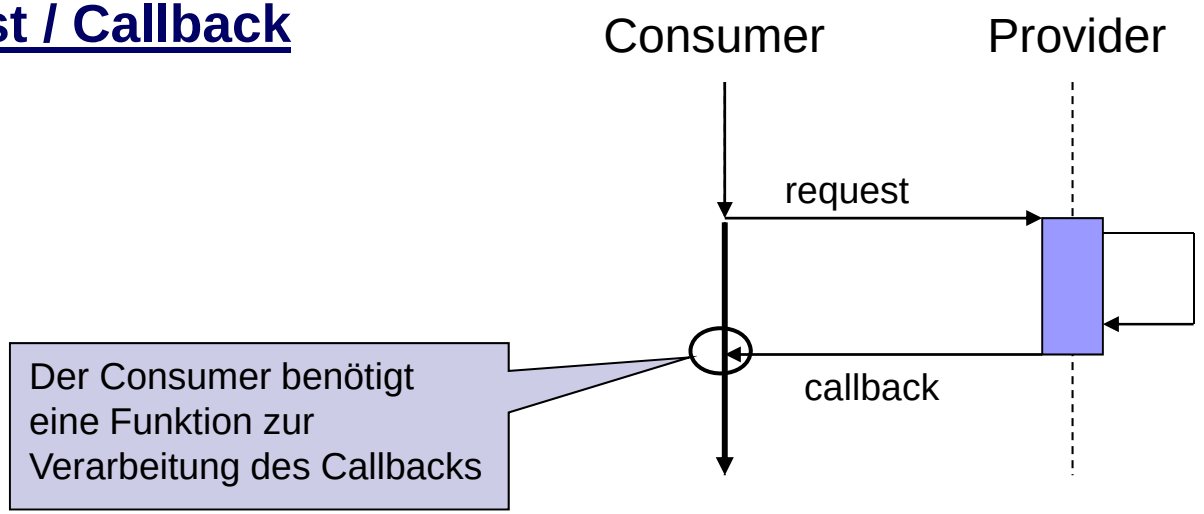


One-Way

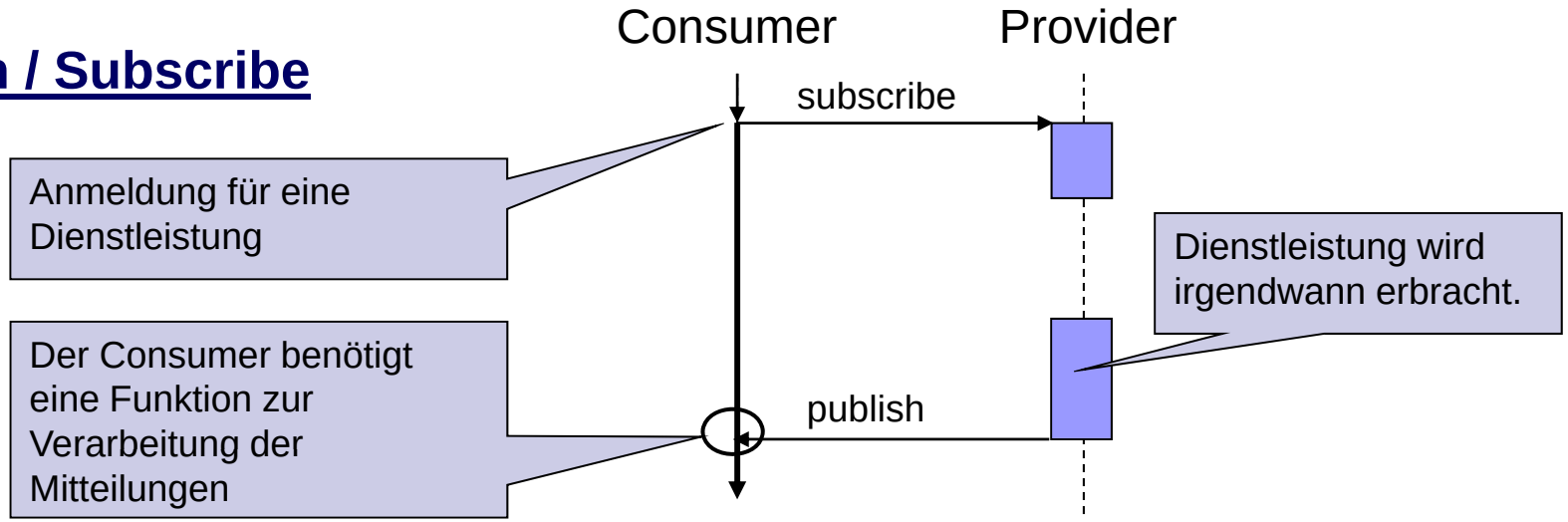


Message Exchange Patterns (3)

Request / Callback



Publish / Subscribe



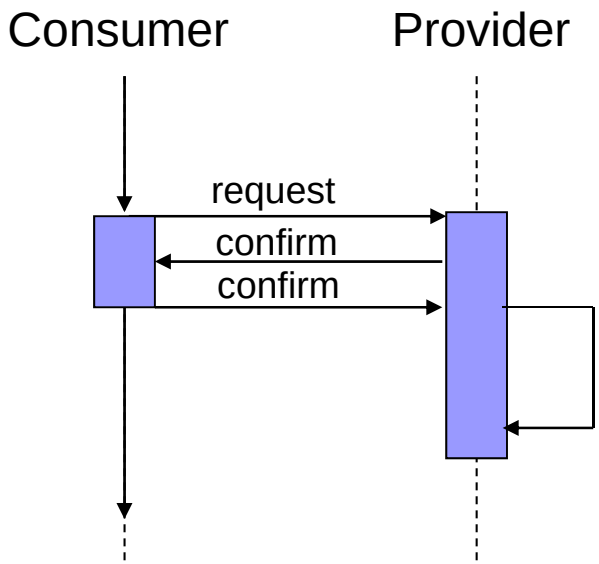
Message Exchange Patterns (4)

Vergleichbare MEPs in der SOAP-Spezifikation

- **In-Only**: One-Way-Nachricht vom Consumer zum Provider
- **Robust-In-Only**: One-Way-Nachricht mit Eingangsbestätigung
- **In-Out**: Request mit erwarteter Response
- **In Optional-Out**: Request mit optionaler Response

Beispiel: **Robust-In-Only**

- Stellt sicher, dass eine One-Way-Nachricht tatsächlich ankommt
- Doppelte Bestätigung erforderlich



Servicebeschreibung (1)

Beschreibung der Schnittstelle und Funktion eines Services

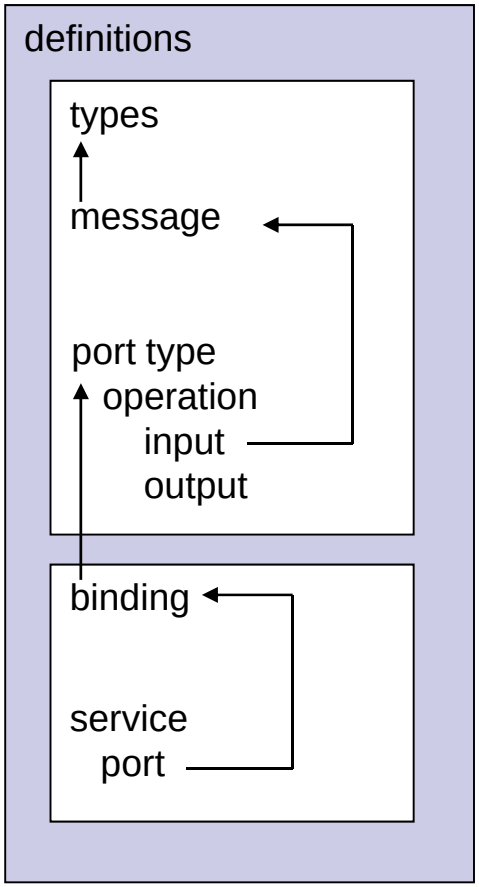
- Um einen Web Service nutzen zu können, muss eine Beschreibung der Service-Schnittstelle vorliegen
- Beschreibung geschieht mit **WSDL** (Web Service Definition Language)
- Weit verbreitet ist die Version 1.1, aktuell ist die Version 2.0
- Eine WSDL-Datei enthält:
  - Beschreibung der angebotenen **Operationen**
  - **Protokoll**, mit dem der Service aufgerufen werden kann
  - **Adresse,** über die der Service erreichbar ist
- Anhand einer WSDL-Beschreibung kann automatisch das Gerüst einer SOAP-Nachricht generiert werden

Servicebeschreibung (2)

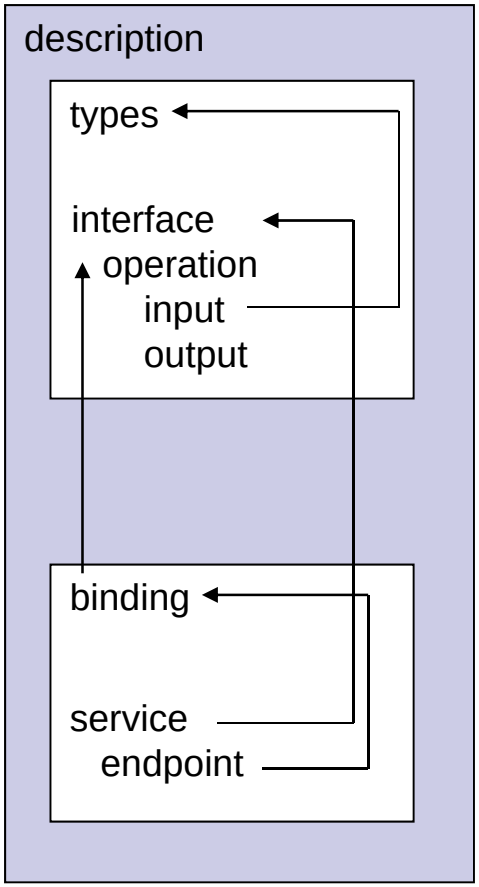
Struktur einer WSDL-Datei

verwendet →

WSDL 1.1



WSDL 2.0

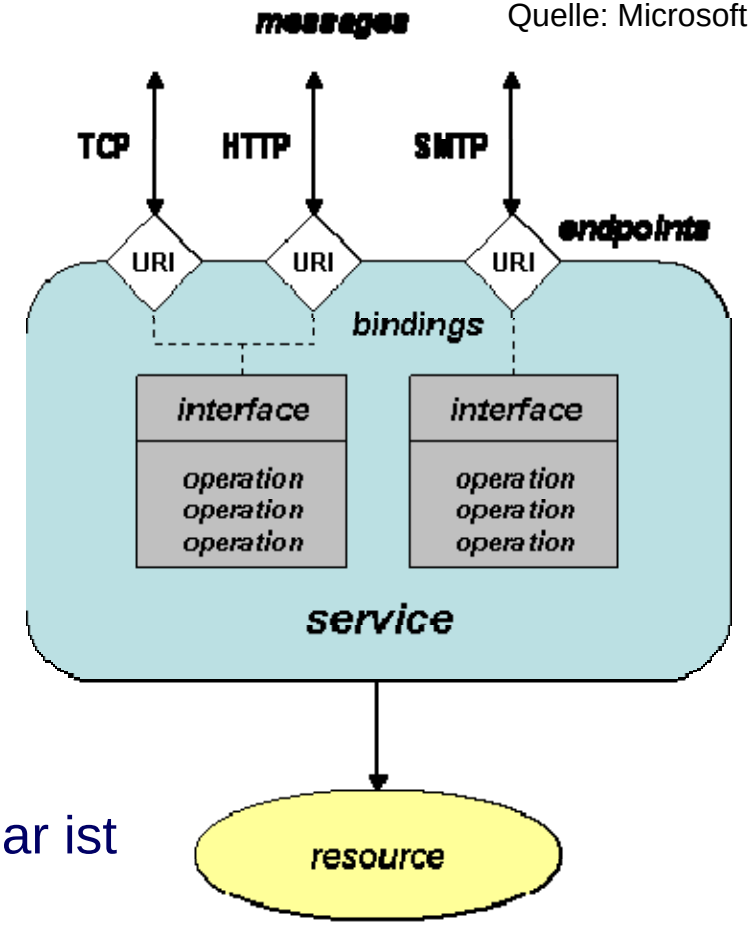


Servicebeschreibung (3)

Bedeutung der einzelnen Elemente

- **types**
  Definition verwendeter Datentypen
- **portType** bzw. **interface**
  Beschreibung der Operationen einer Schnittstelle
- **binding**
  Informationen über das zu verwendende Protokoll
- **service** / **port** bzw.
  service / **endpoint**
  Adresse, unter der der Service erreichbar ist

Quelle: Microsoft



Nicht alle Informationen liegen zum gleichen Zeitpunkt vor.

Servicebeschreibung (4)

WSDL-“Lebenszyklus“

Teil der WSDL-Datei	Zeitpunkt der Integration
Service-Schnittstelle	Design, Entwurf
Service-Detaillierung z.B. Types	Desing, Implementierung
Protokollinformationen	Implementierung, Systemkonfiguration
Adressinformationen	Deployment

Servicebeschreibung (5)

Quelle: Josuttis, SOA in der Praxis

```
<types>
  <xsd:schema
      targetNamespace="http://soa-in-der-praxis.de/xsd"
      xmlns="http://soa-in-der-praxis.de/xsd">

    <xsd:element name="lieferAdresse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="kundennummer" type="xsd:long"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="lieferAdresseResponse" type="adresse"/>
    <xsd:complexType name="adresse">
      <xsd:sequence>
        <xsd:element name="strasse" type="xsd:string"/>
        <xsd:element name="ort" type="xsd:string"/>
        <xsd:element name="plz" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>

  </xsd:schema>
</types>
```

Beschreibung eines Typs für den Aufruf

Beschreibung eines Typs für das Ergebnis

Servicebeschreibung (6)

```
<interface name="KundeInterface">

  <operation name="lieferAdresse"
             pattern="http://www.w3.org/2006/01/wsdl/in-out"
             style="http://www.w3.org/2006/01/wsdl/style/iri"
             wsdlx:safe="true">
    <input messageLabel="In"
           element="xsd1:lieferAdresse"/>
    <output messageLabel="Out"
            element="xsd1:lieferAdresseResponse"/>
  </operation>

</interface>
```

Aufzurufende Methode

Aufruf-Parameter

Ergebnis-Parameter

Servicebeschreibung (7)

```
<binding name="KundeSOAPBinding"
    interface="tns:KundeInterface"
    type="http://www.w3.org/2006/01/wsdl/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">

  <operation ref="tns:lieferAdresse"
     wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

</binding>

<service name="KundeService"
         interface="tns:KundeInterface">

  <endpoint name="KundeEndpoint"
            binding="tns:KundeSOAPBinding"
            address="http://soa-in-der-praxis.de/kunde20"/>

</service>
```

Protokoll:
SOAP mit HTTP

MEP:
Request / Response

Adresse für den Aufruf

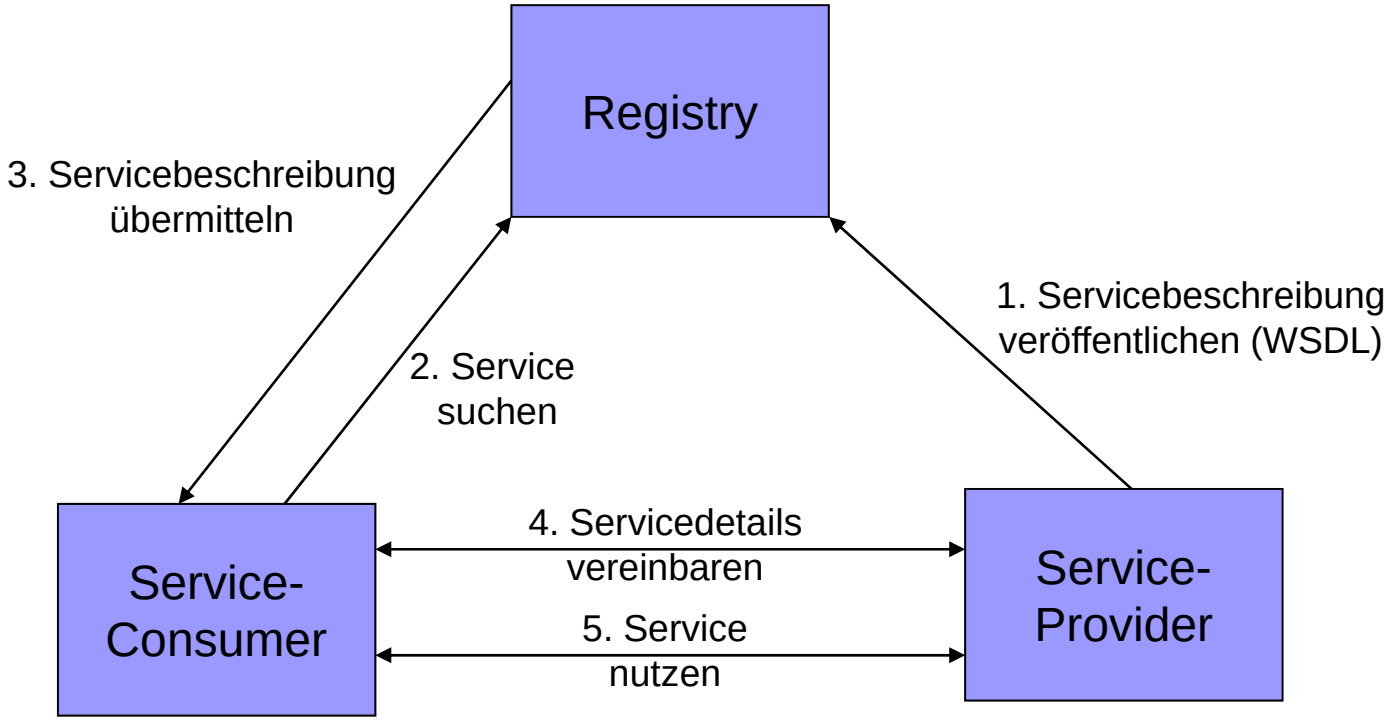
Schwächen von WSDL

- WSDL ist primär für die technische Beschreibung des Services geeignet (Signaturen, Parameter, Adressen).
- Darüber hinausgehende Eigenschaften wie z.B. Verfügbarkeit, Performance, Vor- und Nachbedingungen sowie Zugriffsrechte sind nicht darstellbar.
- Verbesserungen in der Version 2.0 ermöglichen die semantische Erweiterung von WSDL-Dateien.
- Derzeit werden weitere Dokumente (DOC, PDF, EXCEL, …) für die Servicebeschreibung zur Verfügung gestellt.

→ Ohne Erweiterungen von WSDL-Dateien kommt man in der Praxis nicht aus.

Verzeichnisdienste (1)

Veröffentlichung von Service-Informationen mit Hilfe einer Registry



Verzeichnisdienste (2)

UDDI

- Verbreiteter Standard, derzeit Version 3.0
- Speicherung von WSDL-Informationen und zusätzlicher Service-Beschreibungen
- Größter Schwachpunkt:
  Detailinformationen zu Services werden nur unzureichend repräsentiert, z.B. Service Level Agreements (SLAs) oder Sicherheitsanforderungen

  → für komplexe SOA nicht geeignet
- Stattdessen:
  Verwendung von sog. **Registry / Repositorys** wie z.B. ebXML, in dem detaillierte Informationen abgelegt werden können.

Ein Verzeichnisdienst wird im Anfangsstadium einer SOA nicht zwingend benötigt, da die Services noch überschaubar sind

Allgemeine Problemfelder von Web-Services

- **Performance**

  Durch Verwendung von XML viel Kommunikationsoverhead
  → nicht für Kleinstoperationen geeignet
  → nur eingeschränkt für performance-kritische Aufgaben geeignet

- **Sicherheit**

  Die Standardtechnologie beinhaltet keine Sicherheitsmechanismen. Solche Mechanismen müssen durch zusätzliche Maßnahmen realisiert werden.

- **Management**

  Regelung der Nutzung eines Services noch nicht vollständig gelöst

- **Art der technischen Lösung**

  Aufgrund von technischen Restriktionen können die Vorteile von Web Services nicht immer erreicht werden

Alternative Web Service Technologien (1)

XML-RPC

- Basiert ebenfalls auf XML
- Verwendet einfachere Strukturen
- Kennt nur wenige Datentypen
- Jedoch nicht standardisiert

→ Nur für einfache Aufgaben geeignet

Alternative Web Service Technologien (2)

Spezialisierte Standards

- **ebXML** (enterprise business XML)
  → als Erweiterung bzw. Ersatz für EDI im Industriebereich
- **ACORD XML**
  (Association for Cooperative Operations Research and Development)
  → Speziell für die Versicherungsbranche
- **RosettaNet**
  → Speziell für Supply Chain Management

Alternative Web Service Technologien (3)

REST (REpresentational State Transfer)

- Aufruf des Services über eine URL bzw. über die HTTP-Standardbefehle GET, POST, PUT, DELETE
- Alle Parameter sind in der URL enthalten
- Keine Datentypisierung
- Format der Antwort ist beliebig, in der Regel aber XML

→ Sehr einfach aufzurufen und zu nutzen

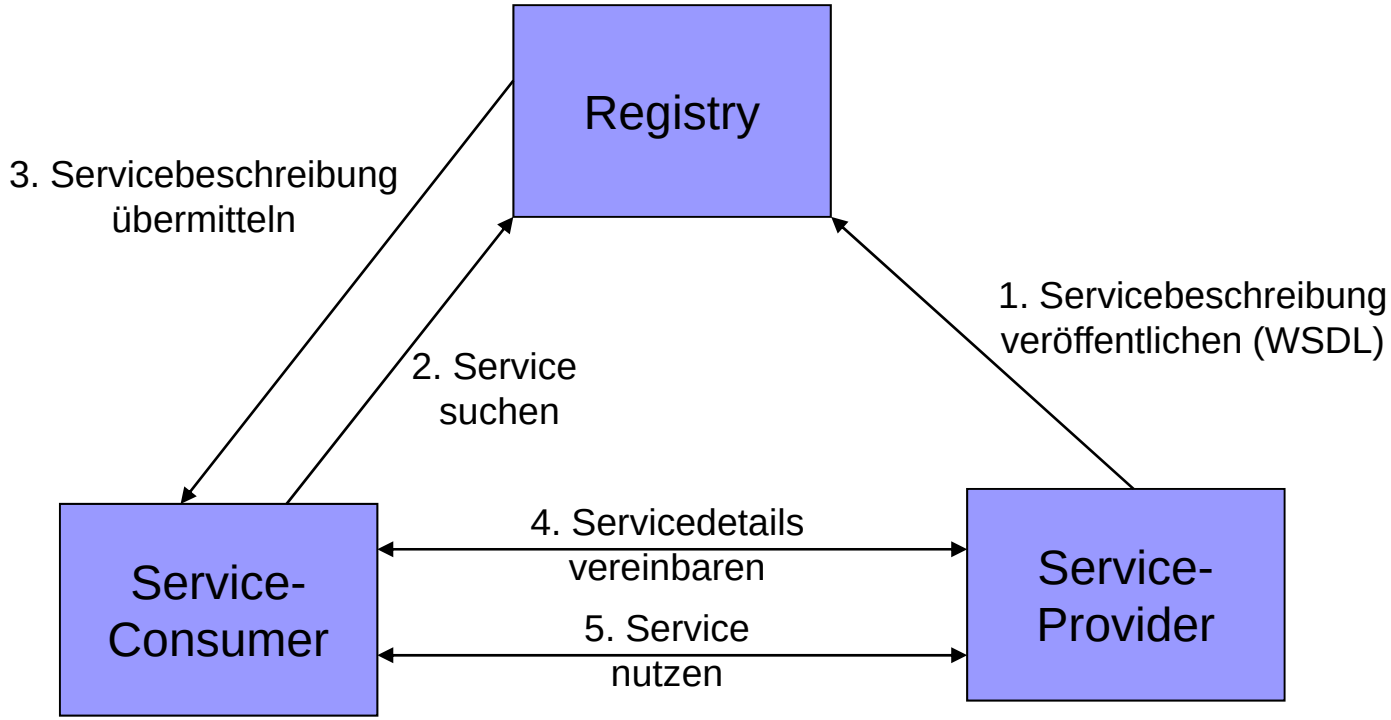
→ Auswertung des zurückgelieferten Ergebnisses muss vollständig manuell codiert werden

Beispiel:

Amazon Web Services (aws.amazon.com)

Verzeichnisdienste (1)

Service-Veröffentlichung und Service-Discovery mit Hilfe einer Registry



Verzeichnisdienste (2)

UDDI (Universal Description, Discovery & Integration)

- Verbreiteter Verzeichnisstandard, derzeit Version 3.0
- Vorläuferversionen sind zum Teil noch im Einsatz

Aufgaben

- Registrierung von WSDL-Informationen und Metadaten zu Services
- Recherchefunktionen für die Abfrage der WSDL- und Meta-Informationen

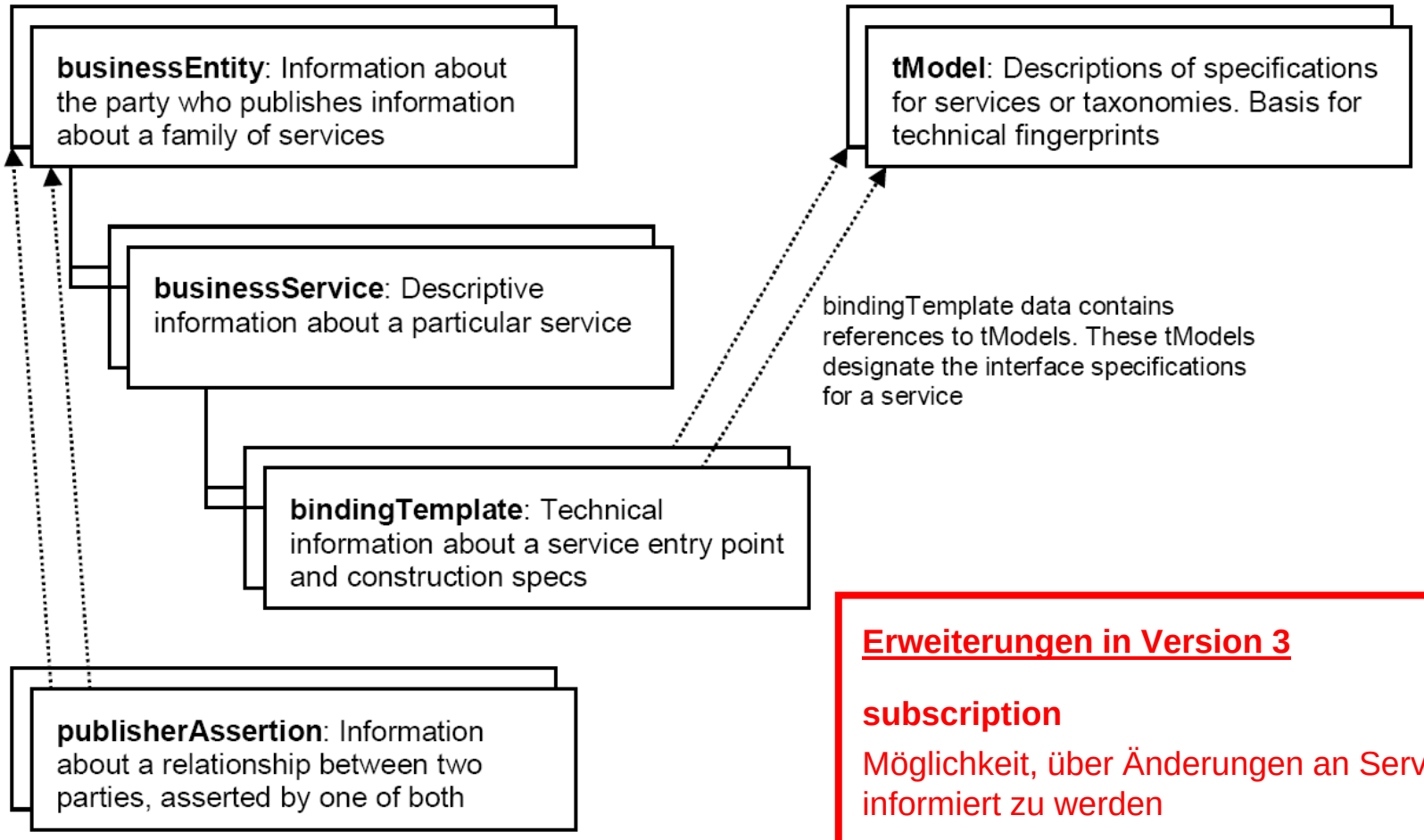
Umsetzung

- Definierte Datenstrukturen
- Definierte API-Calls und die dazugehörenden SOAP-Messages

Verzeichnisdienste (3)

UDDI-Strukturen

Quelle: http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.pdf



Erweiterungen in Version 3

subscription

Möglichkeit, über Änderungen an Services informiert zu werden

Verzeichnisdienste (4)

Struktur der businessEntity

Die businessEntity liefert Informationen über den Anbieter und dessen Angebot und ist evtl. der erste Einstieg, um einen Service zu finden.

```
<element name="businessEntity" type="uddi:businessEntity" />
<complexType name="businessEntity">
  <sequence>
    <element ref="uddi:discoveryURLs" minOccurs="0" />
    <element ref="uddi:name" maxOccurs="unbounded" />
    <element ref="uddi:description" minOccurs="0" maxOccurs="unbounded" />
    <element ref="uddi:contacts" minOccurs="0" />
    <element ref="uddi:businessServices" minOccurs="0" />
    <element ref="uddi:identifierBag" minOccurs="0" />
    <element ref="uddi:categoryBag" minOccurs="0" />
  </sequence>
  <attribute name="businessKey" type="uddi:businessKey" use="required" />
  <attribute name="operator" type="string" use="optional" />
  <attribute name="authorizedName" type="string" use="optional" />
</complexType>
```

Wo gibt es weitere Informationen?

Angebotene Services

Suchbegriffe / Kategorien

Quelle: UDDI Version 2.03 Data Structure Reference

Verzeichnisdienste (5)

Struktur des businessService

Der businessService liefert allgemeine Informationen über einen Service und den Verweis auf das bindingTemplate.

```
<element name="businessService" type="uddi:businessService" />
<complexType name="businessService">
  <sequence>
    <element ref="uddi:name" minOccurs="0" maxOccurs="unbounded" />
    <element ref="uddi:description" minOccurs="0" maxOccurs="unbounded" />
    <element ref="uddi:bindingTemplates" minOccurs="0" />
    <element ref="uddi:categoryBag" minOccurs="0" />
  </sequence>
  <attribute name="serviceKey" type="uddi:serviceKey" use="required" />
  <attribute name="businessKey" type="uddi:businessKey" use="optional" />
</complexType>
```

Allgemeine Information

Verweis auf das binding

Quelle: UDDI Version 2.03 Data Structure Reference

Struktur des bindingTemplate

Das bindingTemplate liefert Informationen über die technische Erreichbarkeit des Services

```
<element name="bindingTemplate" type="uddi:bindingTemplate" />
<complexType name="bindingTemplate">
  <sequence>
    <element ref="uddi:description" minOccurs="0" maxOccurs="unbounded" />
    <choice>
      <element ref="uddi:accessPoint" />
      <element ref="uddi:hostingRedirector" />
    </choice>
    <element ref="uddi:tModelInstanceDetails" />
  </sequence>
  <attribute name="serviceKey" type="uddi:serviceKey" use="optional" />
  <attribute name="bindingKey" type="uddi:bindingKey" use="required" />
</complexType>
```

Adresse für den Service-Aufruf

Detailinformationen über den Service

Quelle: UDDI Version 2.03 Data Structure Reference

Beispiel für accessPoint:

http://services.irgendwo.de/orderservice

Verzeichnisdienste (7)

Struktur des tModels

Das tModel liefert inhaltliche Informationen über Funktion des Services, z.B. die WSDL-Information.

```
<element name="tModel" type="uddi:tModel" />
<complexType name="tModel">
  <sequence>
    <element ref="uddi:name" />
    <element ref="uddi:description" minOccurs="0" maxOccurs="unbounded" />
    <element ref="uddi:overviewDoc" minOccurs="0" />
    <element ref="uddi:identifierBag" minOccurs="0" />
    <element ref="uddi:categoryBag" minOccurs="0" />
  </sequence>
  <attribute name="tModelKey" type="uddi:tModelKey" use="required" />
  <attribute name="operator" type="string" use="optional" />
  <attribute name="authorizedName" type="string" use="optional" />
</complexType>
```

Kurz-beschreibung

Überblicksinformationen über den Service

Quelle: UDDI Version 2.03 Data Structure Reference

Beispiel für overviewDoc:

http://services.irgendwo.de/orderservice.wsdl

Verzeichnisdienste (8)

Anmelden von Services

- Publishers API
- Erfordert Authentifzierung durch den Publisher
- Verwendung von SOAP über HTTPS

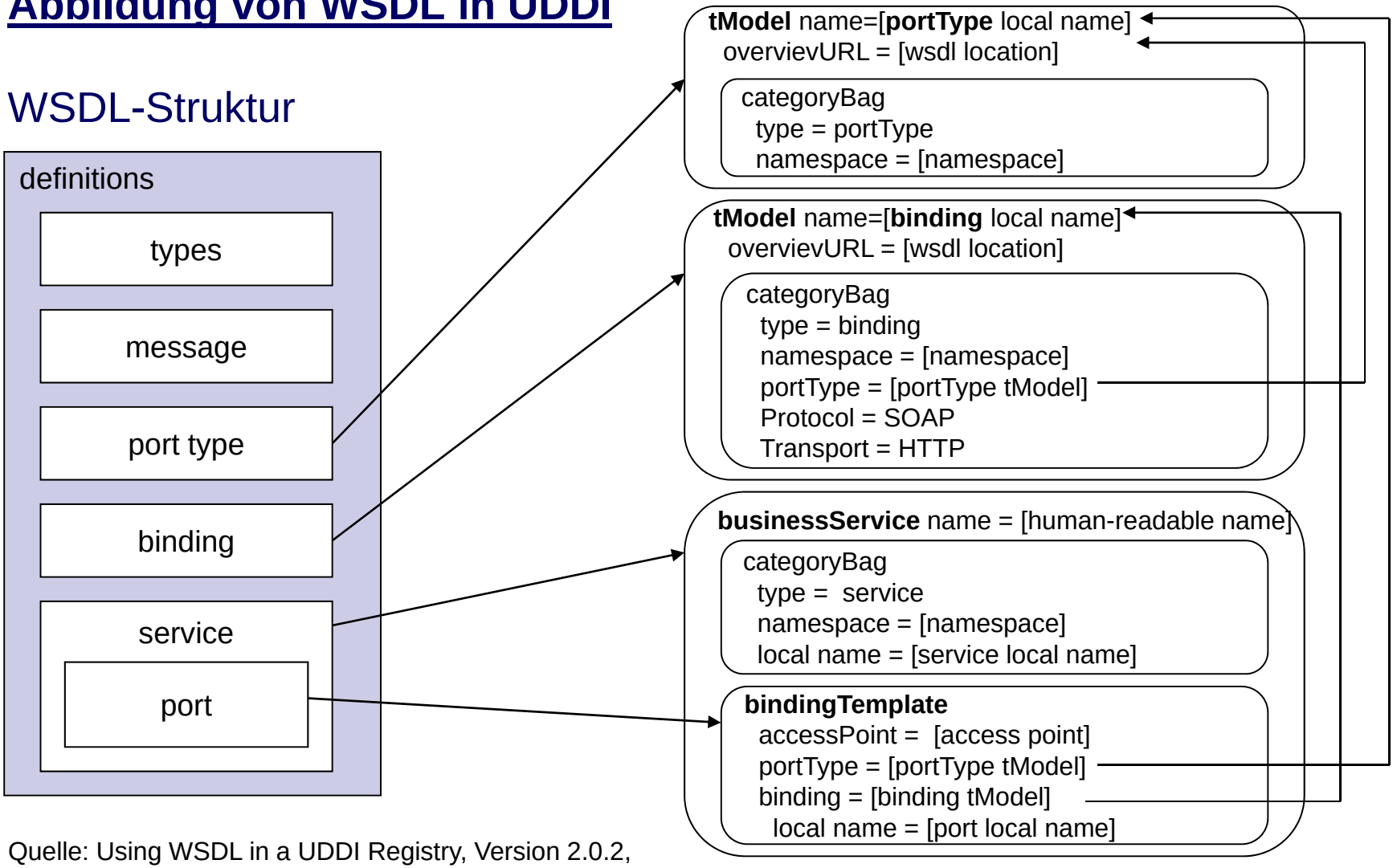
Beispielnachricht für Registrierung eines Bindings (SOAP-Body)

```
<save binding generic="1.0" xnkbs="urn:uddi-org:api">
  <authInfo>Microsoft Corporation</authInfo>
  <bindingTemplate>
    <description xml:lang="en">Production UDDI server,
       Publishing Interface</description>
    <accessPoint URLType="https">https://uddi.microsoft.com/publish</accessPoint>
    <tModelInstanceDetails>
      <tModelInstanceInfo tModelKey="UUID:64C756D1-3374-4E00-AE83-EE12E38FAE63"/>
    </tModelInstanceDetails>
  </bindingTemplate>
</save_binding>
```

Neben dem binding muss auch noch das tModel veröffentlicht werden, auf das sich dieser Schlüssel bezieht

Abbildung von WSDL in UDDI

WSDL-Struktur



Quelle: Using WSDL in a UDDI Registry, Version 2.0.2,
http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm

Verzeichnisdienste (10)

Abfragen des Verzeichnisses

- **Browse Pattern**

  Suche nach globalen Suchbegriffen und Verarbeitung der Suchergebnislisten
  (Zeitpunkt: Entwurf oder Implementierung einer Anwendung)

- **Drill Down Pattern**

  Suche mit exaktem Datenstrukturschlüssel, z.B. ServiceKey
  (Zeitpunkt: Implementierung einer Anwendung)

- **Invocation Pattern**

  Suche des bindings und des tModels eines bestimmten Services

  (Zeitpunkt: Implementierung einer Anwendung oder zur Laufzeit)

Verzeichnisdienste (11)

Nachricht, um Daten zu einer Firma zu finden (SOAP-Body)

Beispiele von http://msdn.microsoft.com/de-de/library/ms950813.aspx

```
<find_business generic="1.0" xmlns="urn:uddi-org:api">
   <name>Microsoft</name>
</find_business>
```

Antwort

```
<businessList generic="1.0" operator="Microsoft Corporation"
   truncated="false" xmlns="urn:uddi-org:api">
   <businessInfos>
      <businessInfo businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3">
      <name>Microsoft Corporation</name>
      <description xml:lang="en">Empowering people through great software
      - any time, any place and on any device is Microsofts vision. As
      the worldwide leader …</description>
      <serviceInfos>
        <serviceInfo businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
                     serviceKey="D2BC296A-723B-4C45-9ED4-494F9E53F1D1">
          <name>UDDI Web Services</name>
        </serviceInfo>
      ...
```

businessKey identifiziert den Anbieter von Services

Liste angebotener Services

Verzeichnisdienste (12)

Nachricht, um einen Service einer Firma zu finden (SOAP-Body)

```
<find_service generic='1.0' xmlns='urn:uddi-org:api'
    businessKey='0076B468-EB27-42E5-AC09-9955CFF462A3'>
    <name>UDDI Web Services</name>
</find_service>
```

businessKey in der Abfrage identifiziert den Anbieter

Antwort

```
<serviceList generic="1.0" operator="Microsoft Corporation"
        truncated="false" xmlns="urn:uddi-org:api">
  <serviceInfos>
    <serviceInfo businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
                 serviceKey="D2BC296A-723B-4C45-9ED4-494F9E53F1D1">
      <name>UDDI Web Services</name>
    </serviceInfo>
  </serviceInfos>
</serviceList>
```

serviceKey identifiziert eindeutig einen Service

Verzeichnisdienste (13)

Nachricht, um Servicedetails abzufragen (SOAP-Body)

```
<get_serviceDetail generic='1.0' xmlns='urn:uddi-org:api'>
    <serviceKey>D2BC296A-723B-4C45-9ED4-494F9E53F1D1</serviceKey>
</get_serviceDetail>
```

serviceKey in der Abfrage identifiziert den Service

Antwort

```
<serviceDetail generic="1.0" operator="Microsoft Corporation"
    truncated="false" xmlns="urn:uddi-org:api">
   ...
  <bindingTemplates>
    <bindingTemplate bindingKey="313C2BF0-021D-4050-...
                     serviceKey="D2BC296A-723B-4C45-9ED4-494F...
      <description xml:lang="en">Production UDDI server,
      Publishing interface</description>
    <accessPoint URLType="https">https://uddi.microsoft.com/publish</accessPoint>
    <tModelInstanceDetails>
      <tModelInstanceInfo tModelKey="uuid:64C756D1-3374-4E00-AE83-EE12E38FAE63">
        <description xml:lang="en">UDDI SOAP Publication Interface</description>
      </tModelInstanceInfo>
      </tModelInstanceDetails>
    </bindingTemplate>
   ...
```

Liefert ein oder mehrere bindingTemplates, da ein Service mehrere Operationen anbieten kann

Verzeichnisdienste (14)

Defizite von UDDI V3

- Die Meta-Informationen zu den Services werden nicht im Verzeichnis selbst abgelegt. Es gibt immer nur Verweise auf diese Informationen.
- Zugriffsrechte sind nur sehr grob definierbar, feingranulare Einstellungen sind teilweise sehr aufwendig
- Eine Standardisierung der Taxonomie (categories) ist nicht durchsetzbar bzw. eine falsche Kategorisierung wird nicht unterbunden

UDDI ist für eine einfache IT-Landschaft ausreichend.

Bei komplexen IT-Landschaften hat das Verfahren Grenzen.

Stattdessen können dann Registry Repositorys verwendet werden.

Verzeichnisdienste (15)

Registry Repositorys

- Verwaltung von Services aus fachlicher Sicht
- Meta-Informationen werden im Gegensatz zu einer reinen Registry im Repository gespeichert.
- Neuere Standards: Z.B. ebXML-Registry (OASIS/ISO-Standard)
- Herstellerspezifische Lösungen, z.B. WSSR von IBM

Verzeichnisdienste (16)

Alternative Lösungen

- Definiertes Verzeichnis im Filesystem, das Servicebeschreibungen enthält
- Web-Seiten, auf denen die Services dargestellt werden; dort wird dann auf die WSDL verwiesen
  - → http://seekda.com
  - → http://www.xmethods.com/ve2/index.po

Zusammenfassung zum Thema „Web Services“ (1)

Wesentliche Eigenschaften von Web Services

- Nutzung von Web Services geschieht überwiegend über SOAP over HTTP
- Beschreibungen von Services werden mit WSDL gemacht, die automatisiert (weiter-)verarbeitet werden können
- Hinterlegung der Servicebeschreibungen in Verzeichnisdiensten zur Abfrage zur Designzeit und zur Laufzeit

Ist damit das ursprüngliche Schnittstellenproblem gelöst?

Nur TEILWEISE!

Der Vorteil ist lediglich die Verwendung eines standardisierten Kommunikationsprotokolls

Zusammenfassung zum Thema „Web Services“ (2)

Wie sieht es mit der losen Kopplung aus?

→ Keine lose Kopplung, da HTTP ein „synchrones“ Protokoll ist

Was ist mit der Reduzierung der Schnittstellen?

→ Kaum Reduzierung der Schnittstellen, da nach wie vor Punkt-zu-Punkt-Verbindungen erforderlich sind

Was ist mit Systemen, die man nicht direkt SOAPen kann?

→ Adapter und Konvertierungsprogramme erforderlich

Wie steht es mit der Sicherheit?

→ Nur rudimentär bis überhaupt nicht gelöst

Für eine komplexe IT-Landschaft braucht man eine Kommunikationsinfrastruktur, die diese Probleme löst oder zumindest verringert!