Vorlesung
Component Ware und Web-Services

- Webservices -

13. Web-Services I

Prof. Dr. Hans-Gert Gräbe, F. Schumacher
Wintersemester 2003/2004

Web-Services – Agenda

1. Web-Services als Komponententechnologie
2. Grundmodell und Definition
3. SOAP – Nachrichtenaustausch
4. WSDL – Beschreibung eines Webservices
5. UDDI – Registry für Webservices
6. Webservices und Sicherheit
7. Zukunft und Trends
8. Frameworks für Webservices

Schlagzeilen zu Webservices

Erst Kopplung von Web-Services bringt Erfolg (CZ)

Web-Services finden Eingang in die Produkte (CZ)

...Altanwendungen werden mit Web-Service-Komponenten versehen... (CZ)

...jede Funktion einer unternehmensweiten Anwendung soll als Web-Service zur Verfügung gestellt werden... (CZ)

*...Ziel einer solchen Web-Service-Architektur ist es, eine **flexible** und **effektive** Umgebung bereitzustellen... (CZ)*

Web-Services sollen schnell Kosten sparen (CZ)

Kostendruck zwingt Unternehmen zu aufwändiger Integration (CZ)

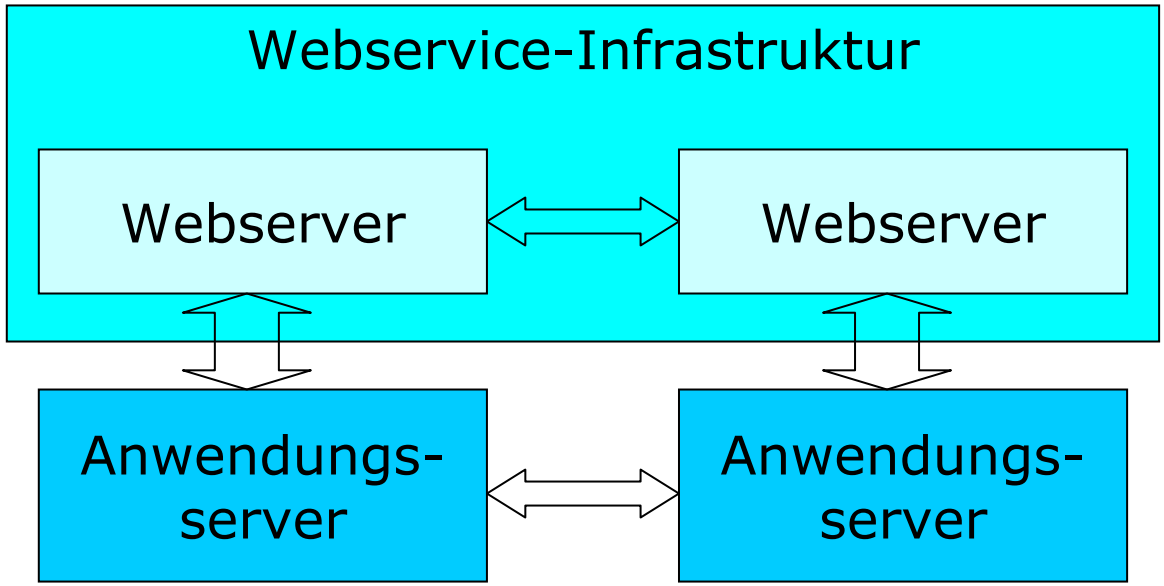
Was ist und kann XML ?

- **XML**: Start 1998, erster Standard 2000
- Durchbruch in vielen Bereichen, wo andere Zugänge bisher fehlschlugen
  - Grund: interessante Eigenschaften, Zeit war reif für Standards
- XML = Darstellung beliebiger strukturierter oder semistrukturierter Daten
- Einsatzgebiete heute: Nachrichtenaustausch, Webseiten, Mark-up traditioneller Dokumente, Konfigurationsmanagement
- syntaktische lingua franca zum Datenaustausch zwischen unabhängigen Anwendungen

XML erlaubt es, das Abstraktionsniveau der Kommunikations-infrastruktur („the wiring standard") verteilter Anwendungen von der Ebene der Protokolle auf die Ebene der Daten zu heben.

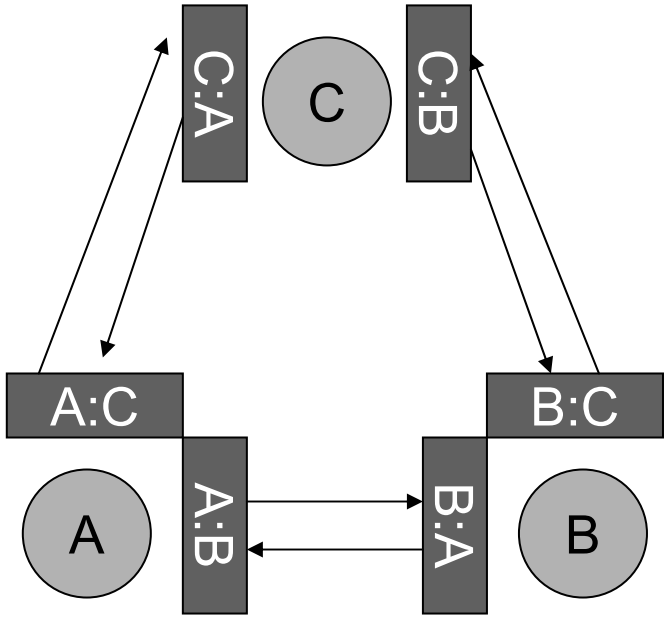
Web-Services als Komponententechnologie

- **Komponenten bisher** (COM, CORBA, EJB, CLR): starke Kopplung über Proxy-Stub-Konzept mit fester Kommunikationsinfrastruktur (RMI, RPC, DII)
- **Komponenten als Webservices**: lose gekoppelte Anwendungen, die Datenaustausch über Nachrichten betreiben



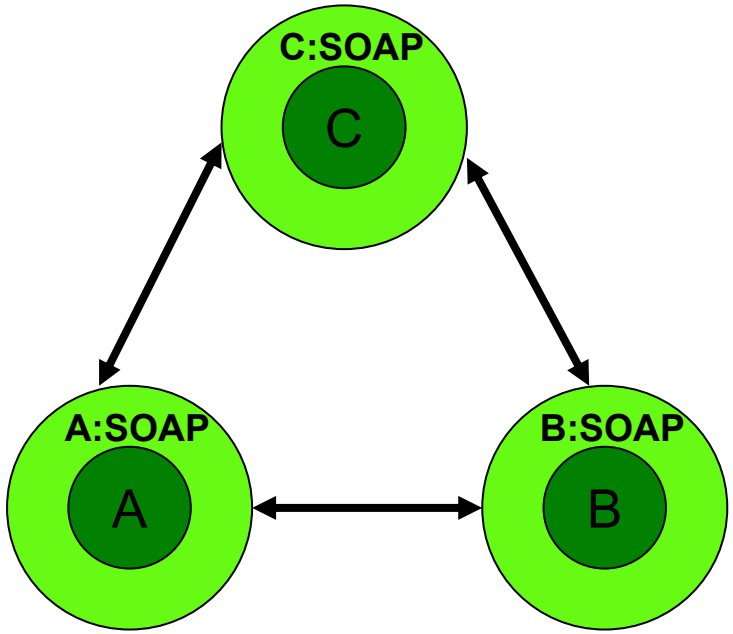
Webservices als EAI-Ansatz

Interface-Strukturen ohne Web-Services



Zwischen jedem System existieren spezielle Schnittstellen (z.B. A:C).

Interface-Strukturen mit Web-Services



Jedes System hat nur eine Schnittstelle zu SOAP, um via SOAP die Daten auszutauschen.

Einsatzziele für Webservices

- Web-Services machen die Funktionalität von Applikationen standardisiert über das Internet verfügbar
- Applikationen können unabhängig von Sprachen, Plattformen oder Protokollen miteinander kommunizieren
- Legacy-System können standardisiert, konsistent und wiederverwendbar gekapselt werden
- Implementierung eines personalisierbaren Informationsaustausches zwischen B2B-Partnern
- leistungsfähige neue Methode, um Software-Systeme aus verteilten Komponenten aufzubauen
  - Techniken heute oft noch nicht ausgereift
- Werden Web-Services alles revolutionieren?
  - Vielleicht, aber vermutlich nicht so glamourös, nicht so lukrativ und nicht so bald wie verheißen
  - Großer Vorteil ist die vollkommene Sprachunabhängigkeit, die durch eine Interoperabilitätsschicht erreicht wird

2. Grundmodell und Definition

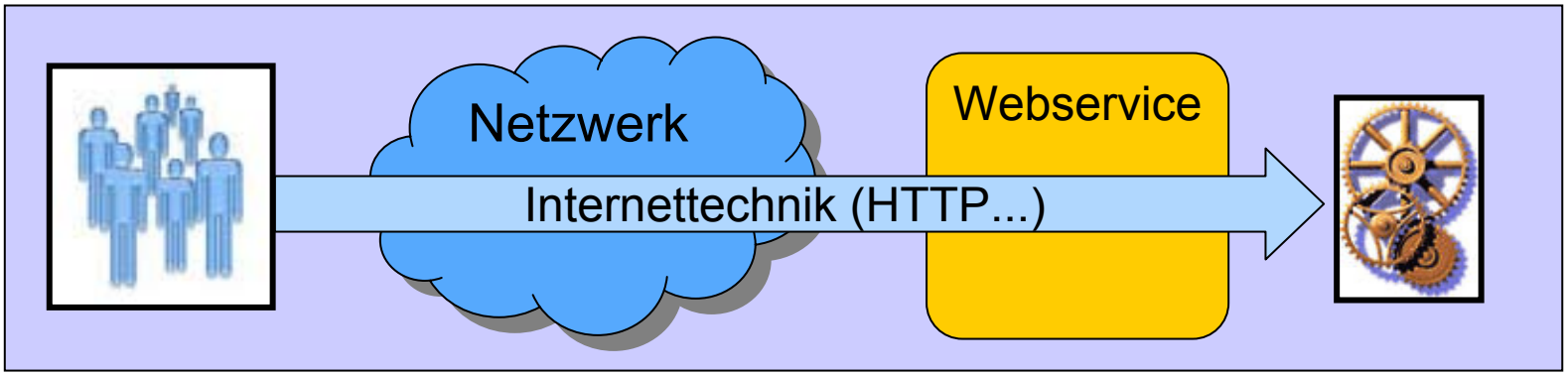
Definition „Webservice“

- „Web services extend the World Wide Web infrastructure to provide the means for software to connect to other software applications...“ (Microsoft)
- "*self-describing, self-contained, modular applications that can be mixed and matched with other Web services.*“ (IBM)
- "*a service **available via the Internet** that completes tasks, solves problems or conducts transactions.*“ (HP)
- “*accessibility via the Web, exposure of an XML interface, ability to be located via a registry, use of XML messages over standard Web protocols, and support of **loosely coupled connections** between systems*” (Sun)

Definition „Webservice“

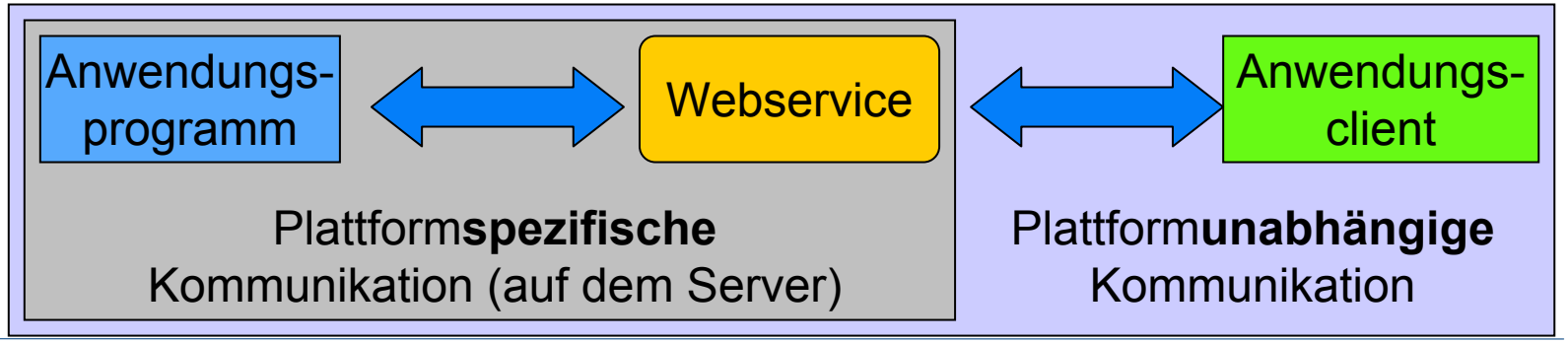
Ein Web-Service ist eine über ein Netzwerk zugängliche nachrichten-basiert gesteuerte Schnittstelle zu Anwendungsfunktionen:

- Standards des Internets (HTTP, SMTP, ...) kommen zum Einsatz
- Anwendungsfunktionen sind über das Internet ansprechbar
- Systeme sind lose koppelbar
- Nachrichten werden in XML ausgetauscht (SOAP)
- Die Schnittstelle der Anwendungsfunktionen wird in einer speziellen IDL in XML dargestellt (WSDL)
- Die Funktionen können lokalisierbar sein (UDDI)



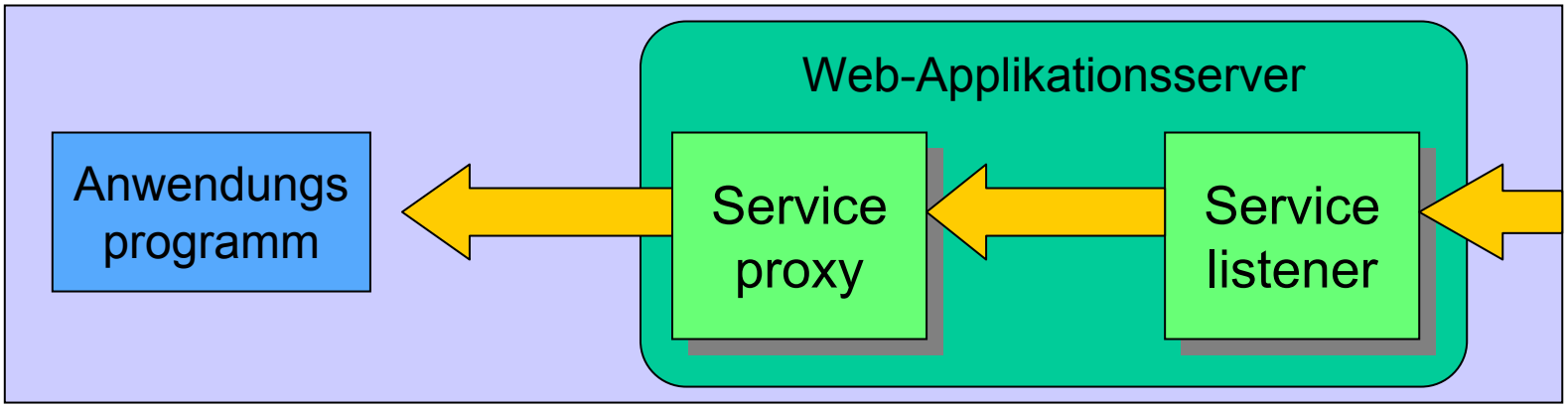
Definition „Webservice“

- Web-Services sind:
  - Eine Art Dokumentenaustausch zwischen Applikationen
  - Mechanismen zur Durchführung von verteilten Geschäftsprozessen zwischen Unternehmen (z.B. CRM, SCM...)
  - Schnittstellen zu Geschäften, Unternehmen und Systemdiensten
- Web-Services bieten:
  - Informationsaustausch mittels etablierter Standards
  - Hohe Verständlichkeit durch XML
  - Abstraktionsschicht von Plattformen und Programmiersprachen
  - Jede Sprache, die Webservices unterstützt, kann auf beliebige Webservice-Funktionen zugreifen (u.a.: Java kommuniziert mit C# über Webservices)



Aufbau von Webservices

- **Anwendungsprogramm:** enthält Logik und ausführbaren Code, z.B. für eine Temperaturabfrage, Bezahlfunktion etc.
- **Service-Listener:** nimmt die Anfragen (z.B. SOAP, HTTP) entgegen, läuft auf dem Web-Applikationsserver
- **Service-Proxy**: übersetzt die Anfragen (z.B. SOAP) in einen Funktionsaufruf des Anwendungsprogramms (z.B. int getTemperatur („Leipzig")) und codiert die Funktionsrückgabe wieder in das XML-Transportprotokoll



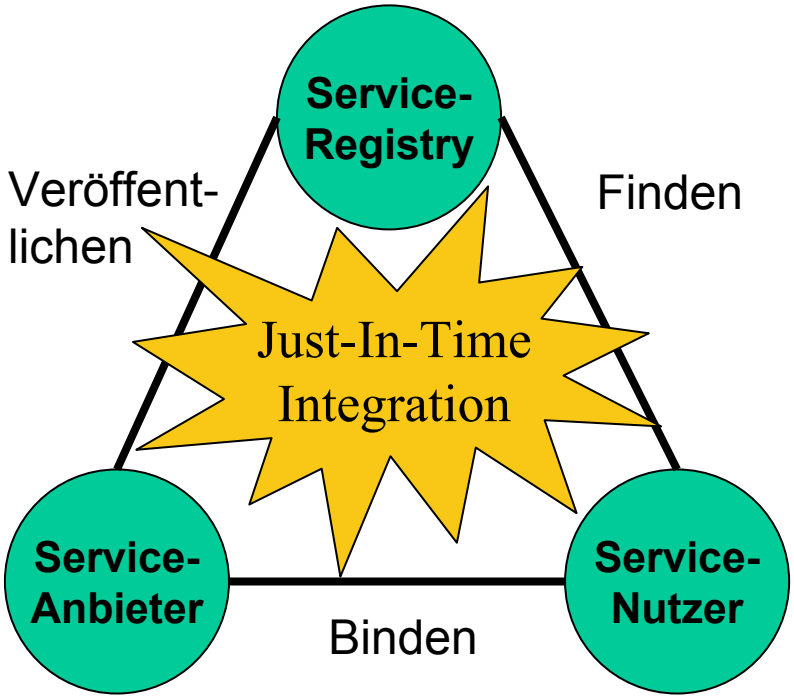
Aufbau von Webservices (2)

- **Service-Proxy** und **Service-Listener** können eigenständige Anwendungen (z.B. TCP-Server, HTTP-Daemon) sein oder in einem Applikationsserver laufen (z.B. IBM Websphere).
- Webservices können überall dort laufen, wo Standardtechnologien des Internets (HTTP, SMTP) verfügbar sind (z.B. auch auf Pocket PCs und Palms).
- Die **Nutzung** von Webservices setzt **nicht** zwingend eine Serverumgebung voraus.
- Webservices können auf **unterschiedlichen Servermodellen** wie Client-Server (traditionell), Mehrschicht-Modellen (Datenspeicherung, Logik und GUI sind getrennt) oder Peer-To-Peer-Systemen aufsetzen.

2. Grundmodell und Definition

Webservice-Architektur

Veröffentlichen:
Service-Anbieter stellt in die Service-Registry eine Beschreibung seiner Dienste



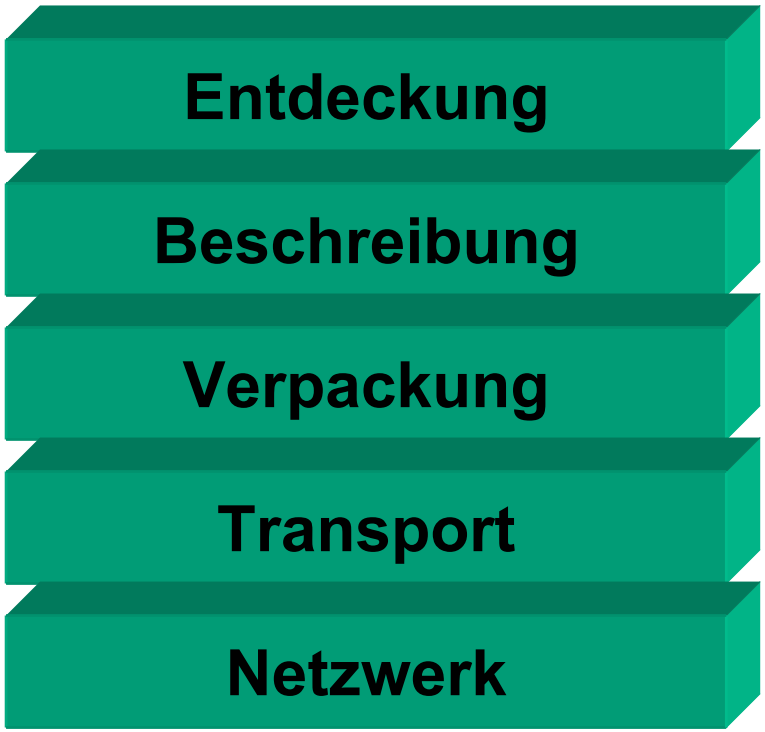
Finden:
Der Service-Nutzer (Mensch, Programm) sucht in der Service-Registry nach einem Service

Binden:
Service-Nutzer nutzt einen angebotenen Service des Service-Anbieters

Schichtenmodell der Webservices

- Fünf Techniken bilden die Schichten der Webservice-Architektur
- Die Schichten
  - Verpackung
  - Beschreibung
  - Entdeckung

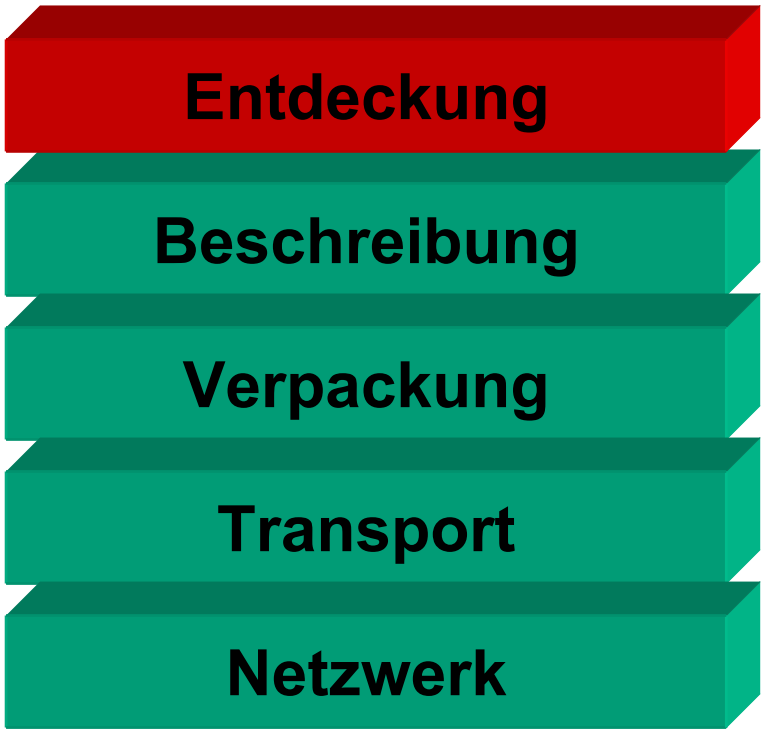
  ermöglichen die **Unabhängigkeit** von der jeweiligen Plattform (wie ISO/OSI-Schichtenmodell)
- Die Schichten decken unabhängige Sachverhalte ab, so dass der Entwickler wählen kann, welche Schichten er implementiert
- Schichten dienen der Modularisierung



Schichtenmodell der Webservices (2)

Entdeckungsschicht

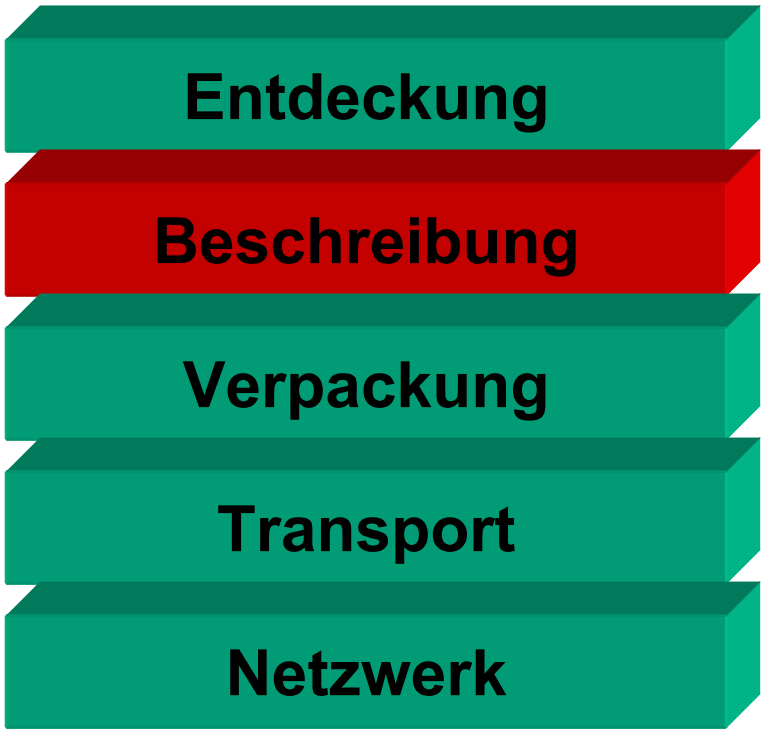
- Mechanismen für Service-Nutzer, um die Beschreibung über den Dienst und den Service-Anbieter zu bekommen
- Ausprägungen:
  - UDDI
  - WS-Inspection



Schichtenmodell der Webservices (3)

Beschreibungsschicht

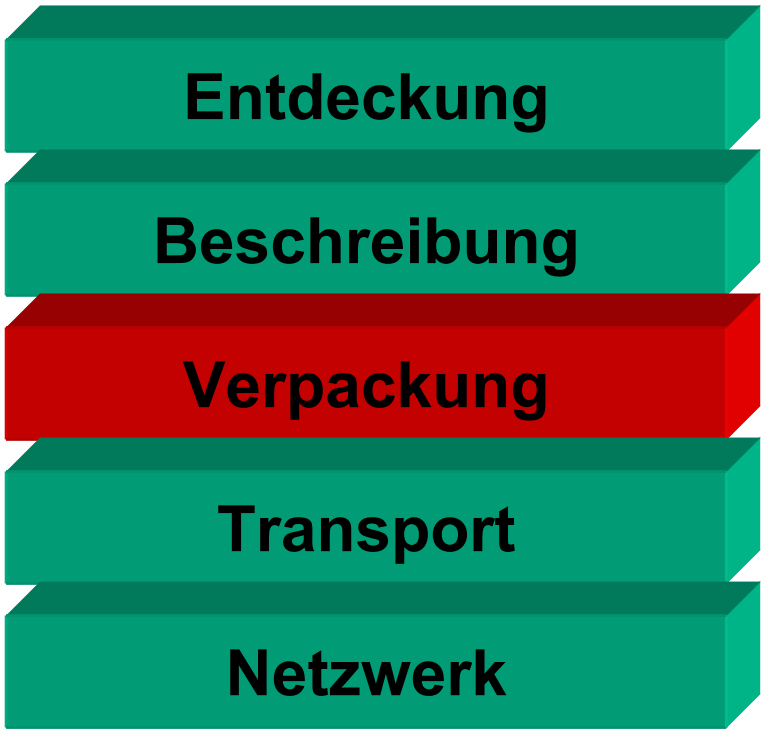
- Bereitstellung von Informationen über die Protokolle für Netzwerk, Transport und Verpackung
- Hilft dem Service-Nutzer, den Webservice zu kontaktieren und zu verwenden
- **Ausprägungen:**
  - WSDL (Web Service Description Language)
  - RDF (Resource Description Framework)
  - DAML (DARPA Agent Markup Language)



Schichtenmodell der Webservices (4)

Verpackungsschicht

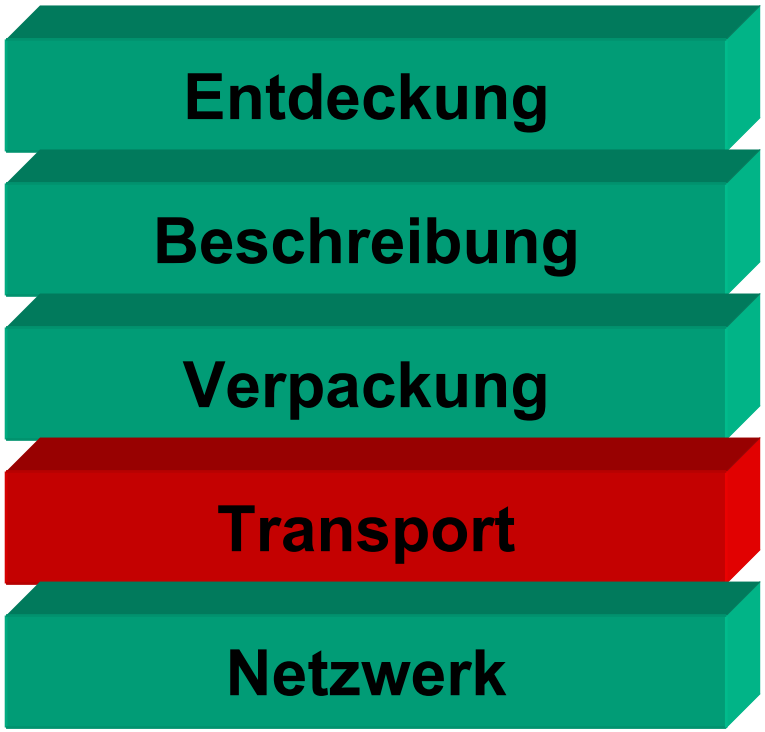
- Verpackt die Anwendungsdaten in XML, damit sie über die Transportschicht übertragen werden können (Serialisierung)
- Ausprägungen:
  - SOAP (Simple Object Access Protocol)
  - XML-RPC



Schichtenmodell der Webservices (5)

Transportschicht

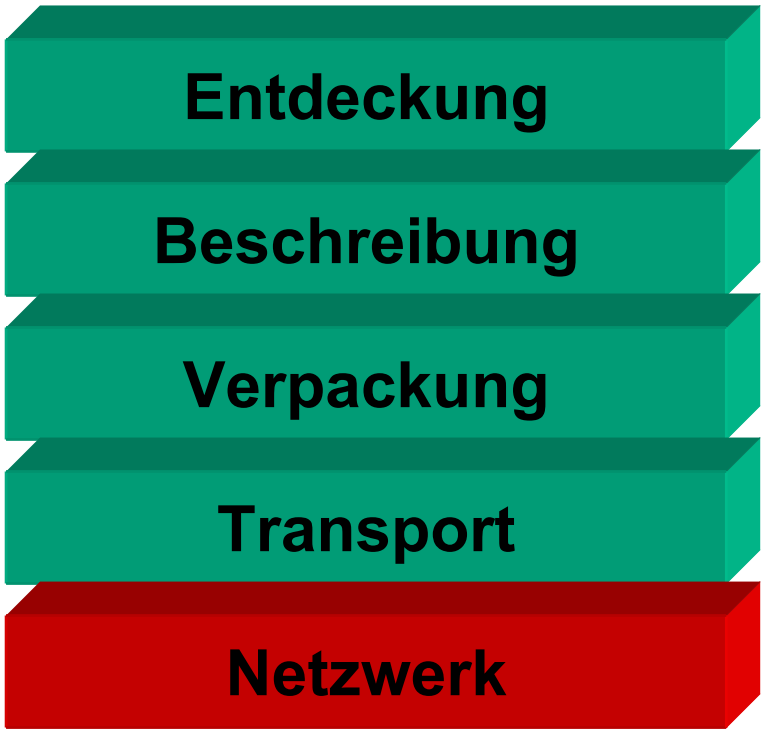
- Ermöglicht auf Basis der Netzwerkschicht die Kommunikation der Anwendungen
- Verwendete Techniken:
  - TCP
  - HTTP
  - SMTP
- Webservices können auf beliebigen Transportprotokollen aufsetzen
- HTTP ist am meisten verbreitet



Schichtenmodell der Webservices (5)

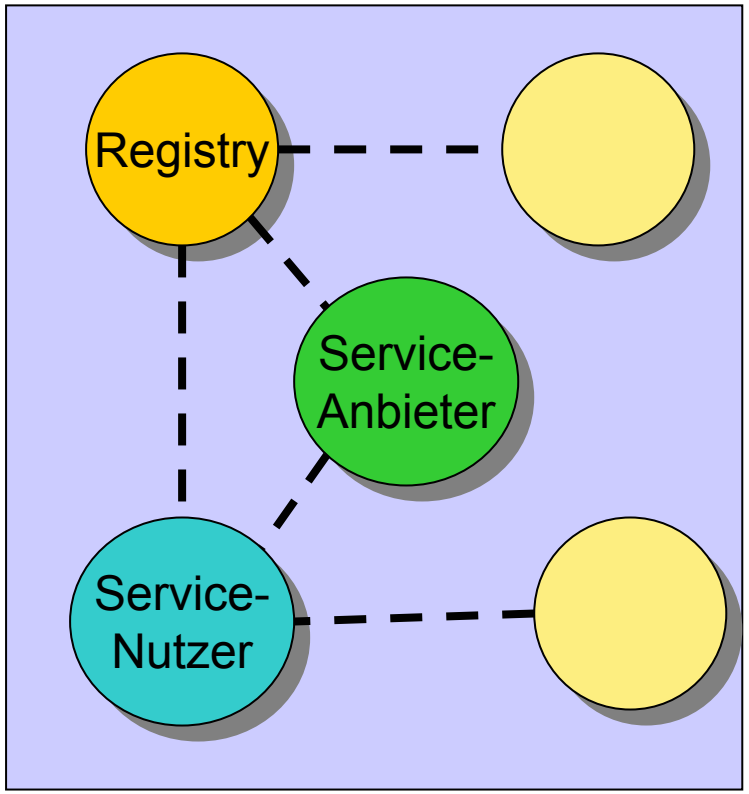
Netzwerkschicht

- Entspricht der Netzwerkschicht im Stapelmodell des TCP/IP-Netzwerkmodells
- Bereitstellung von Funktionalitäten für:
  - Kommunikation
  - Adressierung
  - Routing
- Im Netzwerkmodell sind die oberen drei Schichten zur Anwendungsschicht zusammengefasst.



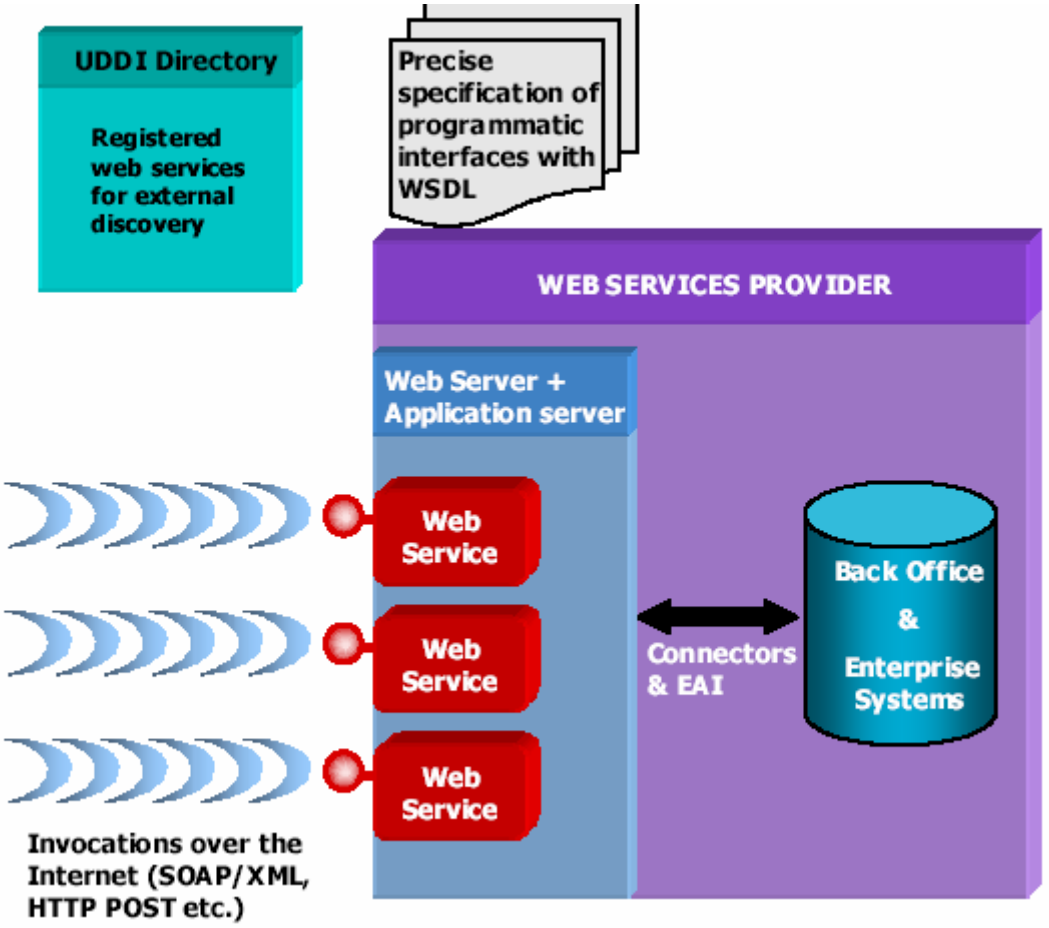
Das Peer-Service-Modell

- Webservice-Infrastruktur erfordert nicht zwingend eine zentrale Registry
- Beispiel: Peer-To-Peer-Netze
  - Peers (Mensch, Anwendung, Peergruppe) agieren gleichberechtigt als:
    - Service-Anbieter
    - Service-Nutzer
    - Service-Registry-Funktionalität
- Peers können dynamisch die Rollen tauschen
- Praxiseinsatz: Instant Messaging



2. Grundmodell und Definition

Überblick über Webservice-Komponenten



XML-Protokolle

- **SOAP**
  Datenaustausch-format
- **WSDL**
  Beschreibungs-format
- **UDDI**
  Gelbe Seiten, Repositoryformat

3. SOAP - Nachrichtenaustausch

Definition: SOAP

- **S**imple **O**bject **A**ccess **P**rotocol
- Entwickelt von Microsoft, DevelopMentor, UserLand
- Seit 2000 W3C-Spezifikation
- Standardisiertes Verpackungsprotokoll auf XML-Basis für Nachrichtenaustausch zwischen Anwendungen
- Stellt die Message-Spezifikation zur Kommunikation von Web-Services dar
  - einfacher XML-Umschlag um die zu übertragende Information + Regeln zur Darstellung anwendungs- und plattform-spezifischer Datentypen in XML
- Applikationen können so über das Internet Daten und Dokumente austauschen

Nachrichtenaustausch

- Anwendungen können Nachrichten (z.B. Aktienkurse, Warenbestellungen) in XML codieren und via SOAP-Dokumenten austauschen
- Durch XML liefert SOAP eine **plattformunabhängiges** Protokoll für den Nachrichtenaustausch
- SOAP stellt Konventionen für eine **standardisierte Darstellungsweise** der Informationen in XML für den Datenaustausch in heterogenen Systemen zur Verfügung
- SOAP liefert zwei Ansätze für den Nachrichtenaustausch:
  - ***Remote Procedure Calls***
    Funktionsaufrufe entfernter Prozeduren im Sinne verteilter Architekturen
  - ***Electronic Document Interchange***
    Dokument-basiertes SOAP, bei dem fachliche Dokumente z.B. Steuererklärung, Warenbestellungen ausgetauscht werden

SOAP-Nachrichten

- SOAP-Nachrichten bestehen aus einem Umschlag (envelope) mit den Unterelementen:
  - Kopf (header, optional)
    - kann mehrere Header-Blöcke enthalten
  - Rumpf (body, erforderlich)
    - Jede Nachricht enthält **genau einen** Rumpf
- Der Kopf enthält Informationen:
  - Wie die Nachricht verarbeitet werden soll
  - Routinginformationen
  - Authentifizierung, Autorisierung und Transaktionskonzepte
  - Sonstige Kontextinformation
- Der Rumpf enthält die Nachricht bzw. Funktionsaufrufe in XML-Syntax (<tag>...</tag>)
- Die XML-Syntax von SOAP 1.2 basiert auf dem Namensraum http://www.w3.org/2001/06/soap-envelope

SOAP-Nachrichten (2)

Beispiel für dokumentbasiertes SOAP-Dokument:

```
<s:Envelope xmlns:s=„http://www.w3.org/2001/06/soap-envelope“>
  <s:Header>
   <m:transaction xmlns:m=„soap-transaction“ s:mustUnderstand=„true“>
    <transactionID>1221</transactionID>
   </m:transaction>
  </s:Header>
  <s:Body>
   <n:purchaseOrder xmlns:n=„urn:OrderService“
    n:encodingStyle=„http://www.w3.org/2001/06/soap-encoding“>
     <to>Hans Schulze</to>
     <order>
      <book>SOAP in a nutshell</book>
     </order>
   </n:purchaseOrder>
  </s:Body>
</s:Envelope>
```

Umschlag

Kopf (**optional**)

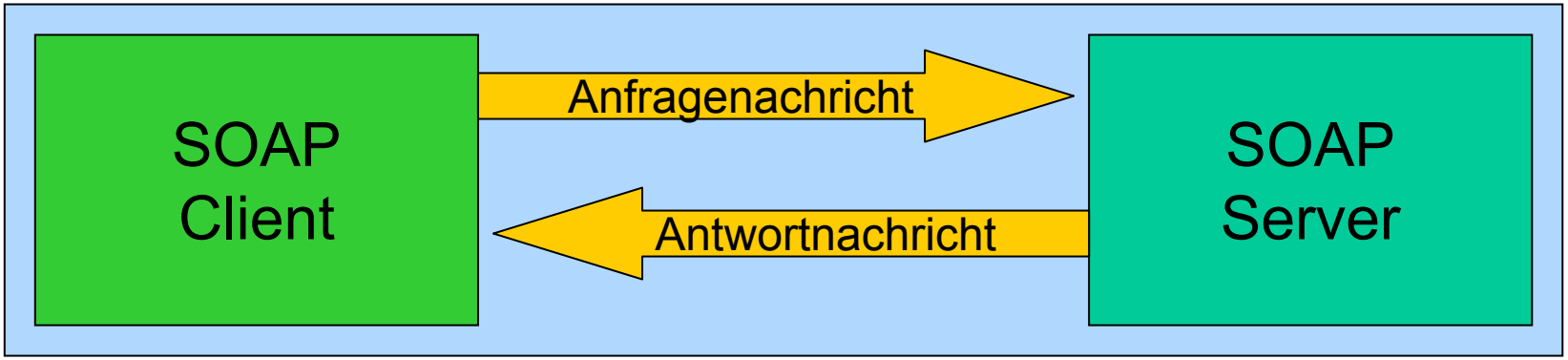
Rumpf

Inhalt

SOAP-RPC

- SOAP-Nachrichten im RPC-Stil (Remote Procedure Calls):
  - Nachrichten treten paarweise auf:
    - Nachricht mit dem Funktionsaufruf (request)
    - Nachricht mit dem Rückgabewert (response)
  - Asynchrones
  - Methodenaufruf entspricht einer einzelnen Struktur, wobei jeder Eingabe-/Rückgabeparameter als Feld zu modellieren ist.

- Modellbeispiel für SOAP-RPC:



SOAP-RPC (2)

Ein RPC-Request in SOAP:

```
<S:Envelope xmlns:S= „...“ >

  <S:Header>
    <m:transaction xmlns:m=„...“>
      <transactionID>1256</transactionID>
    <m:transaction>
  </S:Header>

  <S:Body>
    <m:GetTemperature xmlns:m=„...“>
      <ort xsi:type=„String“>Leipzig</ort>
    </m:GetTemperature>
  </S:Body>

</S:Envelope>
```

Aufruf folgender Java-Methode in SOAP:

```
int GetTemperature(String ort)
```

Mit dem Parameter:

```
Ort = „Leipzig“
```

Mit einer Transaktions-ID im Header: `1256`

SOAP-RPC (3)

Ein RPC-Response in SOAP:

```
<S:Envelope xmlns:S=„...“>

  <S:Body>
    <m:GetTemperatureResponse xmlns=„...“>
      <value xsi:type“int“>- 10</value>
    </m:GetTemperatureResponse>
  </S:Body>

</S:Envelope>
```

Rückgabewert der aufgerufenen Java-Methode in SOAP:

```
int GetTemperature(String ort)
```

Mit dem Integerwert:

```
-10
```

Namenskonvention des Response:

Es wird einfach nur „Response“ an den Methodennamen angehängt.

SOAP-Attribute

- Attribut **mustUnderstand** im SOAP-Header
  - Markiert Headerblöcke, die vom Empfänger verstanden werden müssen.
  - Versteht der Empfänger einen mit mustUnderstand=„true“ markierten Header-Block nicht, muss er die gesamte SOAP-Nachricht zurückweisen
  - Dient der Realisierung von Transaktionen

- Attribut **encodingStyle** für beliebige XML-Elemente
  - Legt fest, wie die Datentypen im Element und Kindelementen allgemein verbindlich zu codieren sind
  - Eine Art Typensystem für die Daten, um dynamischen Informationsaustausch zu gewährleisten
  - mehr dazu später

SOAP-Attribute (2)

- Attribut **xmlns** in XML-Elementen:
  - Verweist auf den benutzten Namensraum des XML-Dokumentes
  - Namensräume beschreiben in einem XML-Dokument (ähnlich wie DTDs), welche Elemente und Attribute im Dokument wie verwendet werden (müssen)
  - xmlns:m=„http://mynamespace.org“ bindet alle Elemente, die mit m beginnen, an den angegebenen Namensraum
  - **urn** (Uniform Resource Name)
    - Beispiel: **xmlns:m=„urn:OrderService“**
    - Gibt den Name einer Internet-Ressource an
    - Für Suchende ist **nur** der Name der Ressource wichtig, Dienste liefern dann bei Anfrage diese Ressource

Datencodierung in SOAP

- Prinzipiell sind SOAP-Envelopes so konzipiert, dass sie jeden beliebigen wohlgeformten XML-Code aufnehmen, die Verwendung des SOAP-Encoding-Styles ist also **optional**.
- Es gibt für SOAP einen speziellen Codierungsstil (encoding style), der als **übergreifendes Typensystem** entworfen wurde auf Grundlage von Programmiersprachen und Datenbanken.
- Für RPC-Funktionsaufrufen per SOAP bietet sich der SOAP-Codierungsstil an, bei Dokumentenaustausch ist er teilweise nutzlos.
- Der SOAP-Codierungsstil bietet sich an, um Anwendungen einen Informationsaustausch zu ermöglichen, ohne vorher Typinformationen ausgetauscht zu haben. (**late binding**)
- Beispiel für den SOAP-Codierungsstil:

```
<person xsi:type=„xsd:string“>Hans Müller</person>
```

Datencodierung in SOAP (2)

- Codierung eines einfachen Elementes

  Vorname = Niko

```
<vorname>Niko</vorname>
```

- Codierung eines zusammengesetzten Elementes (Struct)

```
<person>
  <vorname>Niko</vorname>
  <nachname>Klaus</nachname>
</person>
```

- Codierung eines zusammengesetzten Elementes (Array)

```
<gruppe>
  <person name=„Niko Klaus“/>
  <person name=„Knud Knirps“/>
</gruppe>
```

Datencodierung in SOAP (3)

- Mittels den Spezialattributen **id** und **href** können Elemente auf andere Elemente Bezug nehmen.
- Referenzierte Elemente verwenden **id**, um ihrem Wert eine Identität zu geben.
- Durch **href** kann ein Element auf den mit einer **id** versehenen Wert eines anderen Elementes zugreifen.

```
<gruppe>
  <person name=„Willi Wald“>
   <adresse href=„#adresse01“/>
  </person>
  <person name=„Hans Hirsch“>
   <adresse href=„#adresse01“/>
  </person>
</gruppe>
```

```
<adresse id=„adresse01“>
  <ort>leipzig</ort>
</adresse>
```

href = #adresse01
referenziert
Id = adresse01

Datencodierung in SOAP (4)

SOAP verfügt über mehrere Möglichkeiten, den Datentyp anzugeben:

- Verwendung des Attributs xsi:type

```
<person>
 <name xsi:type=„xsd:string“>Hans Hirsch</name>
</person>
```

- Verweis auf ein XML-Schema, welches den Datentyp jedes verwendeten Elementes festlegt

```
<person xmlns=„a_schema.xsd“>
 <name>Hans Hirsch</name>
</person>
```

- Verweis auf ein XML-Schema, welches als Namensraum definiert ist

```
<person xmlns=„http://namespace.de“>
 <name>Hans Hirsch</name>
</person>
```

Datencodierung in SOAP (5)

Beispiel für eine XML-Schema-Datei:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:annotation>
  <xsd:documentation xml:lang="en">
    das Schema für die Person
  </xsd:documentation>
 </xsd:annotation>

 <xsd:element name="person" type="personType"/>

 <xsd:complexType name="personType">
  <xsd:sequence>
   <xsd:element name="name" type="xsd:string"/>
   <xsd:element name="adress" type=„xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="personID" type="xsd:decimal"/>
 </xsd:complexType>
</xsd:schema>
```

Datencodierung in SOAP (6)

- Zur Darstellung von Arrays gibt es weiterhin den speziellen Typ **SOAP-ENC:Array**
- Der Elementtyp, den das Array enthält, wird mit dem Attribut **SOAP-ENC:arrayType** festgelegt
- Beispiel:

```
<gruppe
  xsi:type=„SOAP-ENC:Array“
  SOAP-ENC:arrayType=„xsd:string[2]>
  <person>Hans Müller</person>
  <person>Peter Schmidt</person>
</gruppe>
```

- Eckige Klammern [ ] geben die Dimension des Arrays an, die Zahlen innerhalb geben die Anzahl der Elemente des Arrays an
- Es sind auch mehrdimensionale Arrays möglich

SOAP-Faults

- SOAP-Faults geben Fehlermeldungen zurück, die beim fehlerhaften Verarbeiten von SOAP-Nachrichten auftreten
- Fault-Bestandteile:
  - **<faultcode/>** = qualifizierender Name im XML
    - Standardcodes für SOAP:
      - *VersionMismatch* = ungültiger Namensraum für SOAP-Envelope
      - *MustUnderstand* = Headerblock mit mustUnderstand wurde nicht verstanden
      - *Server* = Problem außerhalb des Nachrichten-Rumpfes
      - *Client* = Problem beim Verarbeiten des Nachrichten-Rumpfs aufgetreten
    - Lassen sich beliebig fein gliedern und von den Standardcodes ableiten z.B. Client.Authentification ist abgeleitet von Client
  - **<faultstring/>** = lesbare Erklärung des Fehlers
  - **<faultactor/>** = Bezeichner des Nachrichtenverarbeitungsknotens
  - **<details/>**= anwendungsspezifische Einzelheiten

SOAP-Faults (2)

Beispiel für einen SOAP-Fault (abgeleitet vom Typ Client):

```
<s:Envelope xmlns:s=„...“>
 <s:Body>
  <s:Fault>
   <faultcode>Client.Authentification</faultcode>
   <faultstring>Passwort falsch!</faultstring>
   <faultactor>http://no-identity.com</faultactor>
   <details> <!-- hier steht was --> </details>
  </s:Fault>
 </s:Body>
</s:Envelope>
```

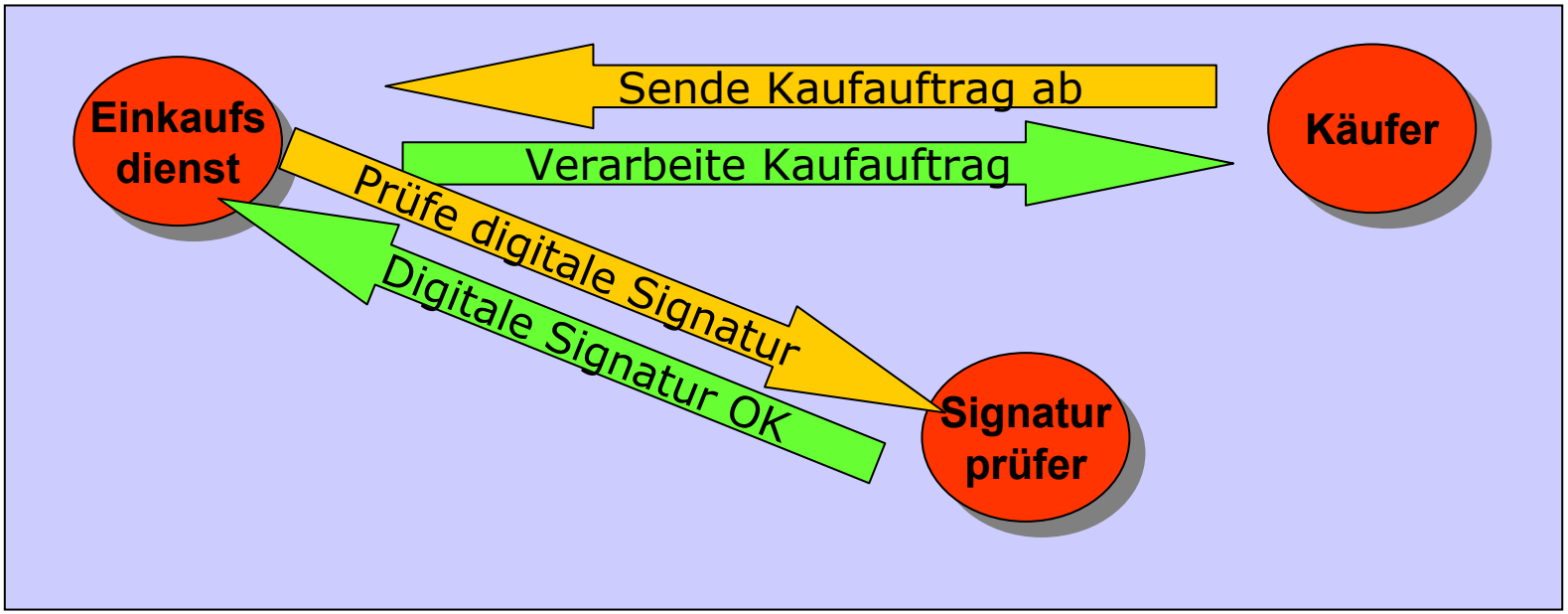
Nachrichtenaustauschmodell

- Eine SOAP-Nachricht kann über mehrere Intermediärdienste vom Sender zum Empfänger gelangen
- Jeder Intermediärdienst kann die SOAP-Nachricht „entfalten“, verarbeiten und verändern
- Webservices, die zwischen Sender und Empfänger liegen und die Nachrichten verarbeiten und weiterleiten, heißen Akteure (actors).
- Der Pfad vom Sender über diverse Akteure zum Empfänger heißt **Nachrichtenpfad**.
- Beschreibung des Nachrichtenpfads kann über SOAP Routing Protokoll (WS Routing) erfolgen
- SOAP kann über das Attribut **actor** in Headerblöcken festlegen, welche Teile der Nachricht von welchem Dienst verarbeitet werden.
- Der Wert von **actor** kann eine URL oder ein anderer Bezeichner sein.

Nachrichtenaustauschmodell (2)

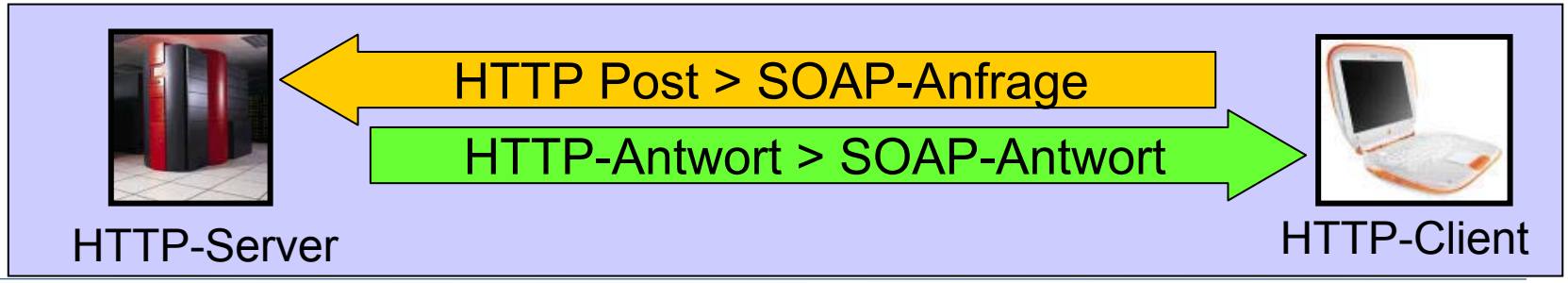
Beispiel für das Attribute **actor** im Headerblock **signature:**

- Der Intermediär „Signaturprüfdienst“ verarbeitet den für ihn bestimmten Headerblock **signature**
- Andere Dienste, welche die SOAP-Nachricht bekommen, ignorieren den Block **signature**, z.B. der Käufer



Transport der Nachrichten

- SOAP ist als Verpackungsprotokoll konzipiert und ist deswegen entkoppelt von der Netzwerk- oder Transportschicht
- SOAP kann so flexibel, je nach Anspruch und Einsatzanforderung, durch beliebige Transportprotokolle, wie HTTP, FTP, TCP, SMTP, POP3, MQSeries oder Jabber verschickt werden.
- Das am häufigsten verwendete Transportprotokoll ist HTTP
- Die SOAP-Spezifikation geht gesondert auf SOAP-über-HTTP ein und beschreibt wie der SOAP-Nachrichtenaustausch mittels HTTP realisiert werden kann
- SOAP eignet sich besonders in der RPC-Form für HTTP, da HTTP ebenso mit Requests und Responses arbeitet



Transport der Nachrichten (2)

Beispiel eines SOAP-Requests über HTTP:

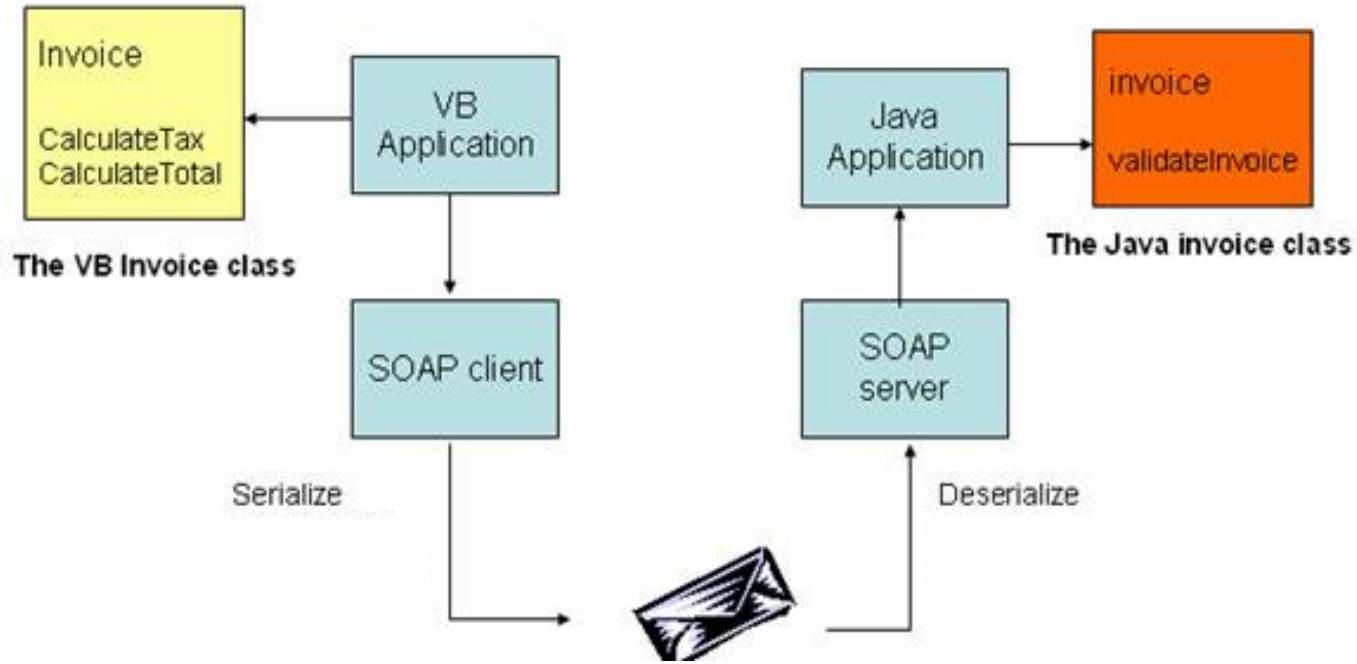
```
POST /soapworkshop/services/id/id.asp HTTP/1.1
Host: xxx.xxx.xxx.xxx
Content-Type: text/xml
Content-Length: nnn
SOAPAction: „urn:myDatabase#GetPersonID"

<?xml version="1.0"?>
<S:Envelope xmlns:S=,...'>
  <S:Body>
    <vb:GetPersonID xmlns:vb=,...'>
      <person>Baron Münchhausen</person>
    </vb:GetPersonID >
  </S:Body>
</S:Envelope>
```

Der HTTP-Header **SOAPAction** wird in der SOAP-Spezifikation definiert.

SOAP-Action zeigt dem HTTP-Server, **was die SOAP-Nachricht will**, bevor der XML-Code entschlüsselt wird.

SOAP – ein Beispielablauf



- Plattformübergreifend und lose gekoppelt wird via SOAP der Nachrichtenaustausch möglich
- SOAP-Client und SOAP-Server stellen die Beziehung zwischen den XML-Daten und den originären Objekten her und leiten die konkreten Anfragen / Antworten weiter

Quellen

- **Hauptquellen**
  - Snell, J. et al: Webservice-Programmierung mit SOAP, O'reilly, 2002
  - Liberty, J.: Programmieren mit C#, O'reilly, 2002

- **Ziele, Motivatoren**
  [1a] Why We Need Web Services Networks
  http://e-serv.ebizq.net/wbs/spicer_1.html

  [1c] The true nature of Web Services
  http://www.perfectxml.com/articles/xml/wsnature.asp

- **Allgemein, Überblick, Modell**
  [2a] Web Services and Distributed Objects: Competing or Complementary?
  http://e-serv.ebizq.net/wbs/conway_1.html

  [2b] Web Services: The Next Generation of Distributed Computing
  http://e-serv.ebizq.net/obj/hildreth_1.html

  [2c] Web Services Spotlight Report (Triple Tree – Investment Bank)
  http://e-serv.ebizq.net/shared/goldclub.jsp?tripletree_5b.html

  [2d] Web Services: Standardizing EAI
  http://www.eaijournal.com/PDF/WebSeryvicesKuzyk.pdf

  [2e] Web Services, Business Objects and Component Models
  http://www.orchestranetworks.com/us/pdf/tmxwp09.pdf

Quellen (2)

Bausteine, Architektur

IIIa SOAP

[3aa] Introduction to Web Service, SOAP & WSDL
http://www.aspalliance.com/yusuf/Article11.asp

[3ab] Introducing SOAP
http://www.vbxml.com/soapworkshop/articles/intro/default.asp

[3ac] SOAP: The Internet Is Not Just for Browsing Anymore
http://e-serv.ebizq.net/shared/goldclub.jsp?newcomer_1.pdf

[3ad] SOAP: RPC or Messaging?
http://www.learnxmlws.com/tutors/rpcmsg/rpcmsg.aspx

[3ae] A Gentle Introduction to SOAP
http://radio.weblogs.com/0101679/stories/2002/03/16/aGentleIntroductionToSoap.html

[3af] SOAP-W3C-Specification
http://www.w3.org/TR/SOAP/

[3ag] Busy Developers Guide to SOAP 1.1
http://www.w3.org/TR/SOAP/

Quellen (3)

Bausteine, Architektur

IIIb WSDL

[3ba] WSDL: W3C Note 15 March 2001
http://www.w3.org/TR/wsdl

[3bb] WSDL: an Introduction
http://imatch.lcs.mit.edu/docs/seminar_wsdl_files/frame.htm

[3bc] Introduction to WSDL
http://www.learnxmlws.com/tutors/wsdl/wsdl.aspx

IIIc UDDI

[3ca] UDDI Registries and Reuse
http://e-serv.ebizq.net/wbs/meyer_1.html

[3cb] Introduction to UDDI
http://www.learnxmlws.com/tutors/wsdl/wsdl.aspx

[3cc] UDDI.org Whitepapers
http://uddi.org/pubs/ProgrammersAPI-V2.00-Open-20010608.pdf

Quellen (4)

Trends, Technologien, Zukunft

[4b] ***Web Services and Application Frameworks***
http://www.webservicesarchitect.com/content/articles/samtani04.asp

[5a] ***Why Web Services Will Gradually Dominate e-Business Development***
http://e-serv.ebizq.net/wbs/thomas_1.html

[5b] ***Asynchronous Web Services and the Enterprise Service Bus***
http://www.webservices.org/index.php/article/articleview/352/1/1/

[5c] Webservice Architect
http://www.webservice-architect.com