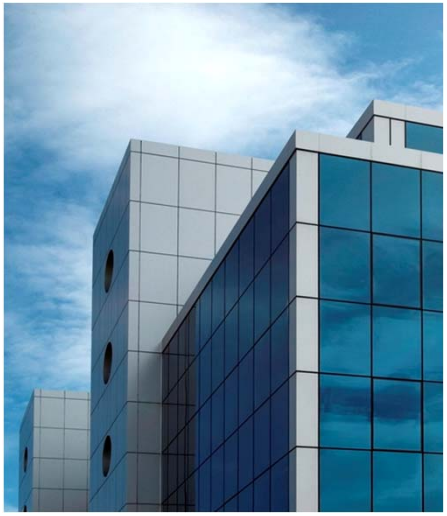
Web Services

Integration heterogener Systemlandschaften

Prof. Dr. Gregor Engels
Fabian Christ
08. Juni 2010

Technische Kooperation

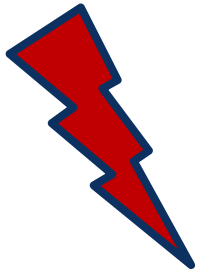


Datenaustausch /
Benutzung
technischer Dienste
über das Internet

Mein Unternehmen

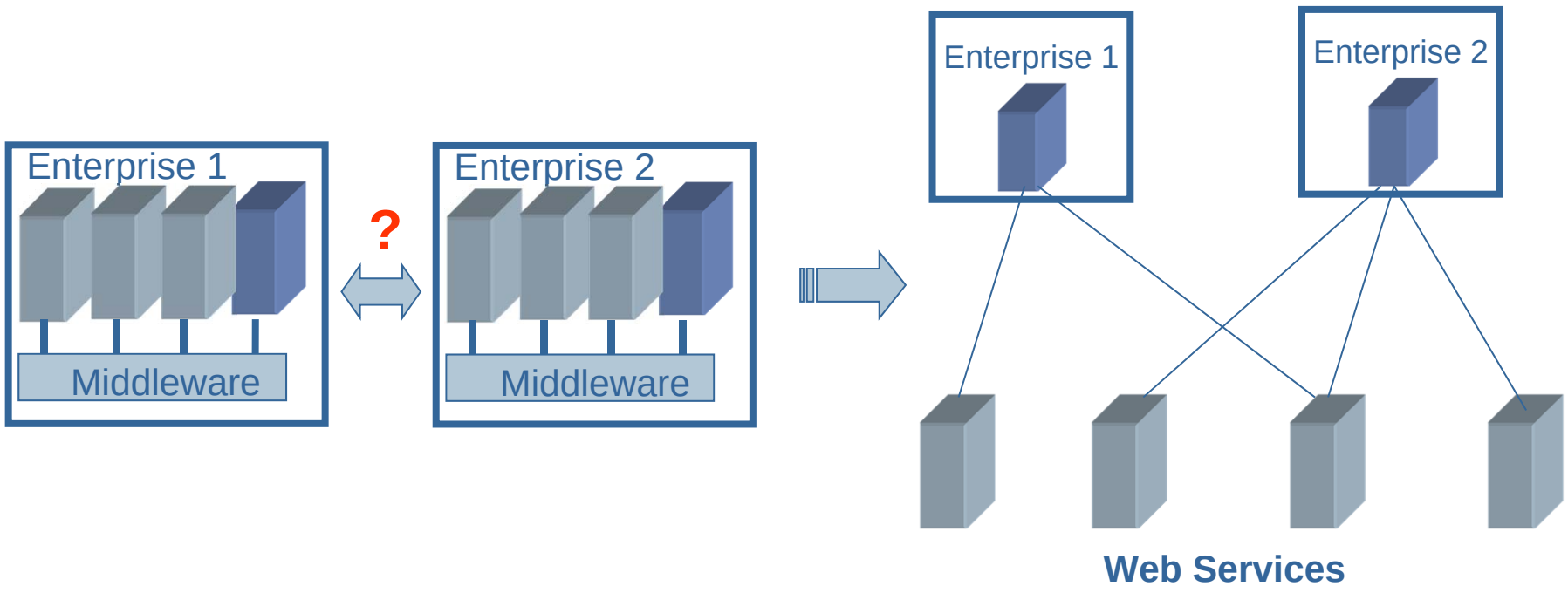
Problem:
Heterogene Systeme

Mein Partner



Integration via Web Services

- Inkompatibilität durch gemeinsamen Standard überwinden
- Web Services bilden einen systemunabhängigen Standard zur Systemintegration.



Web Service Kommunikation

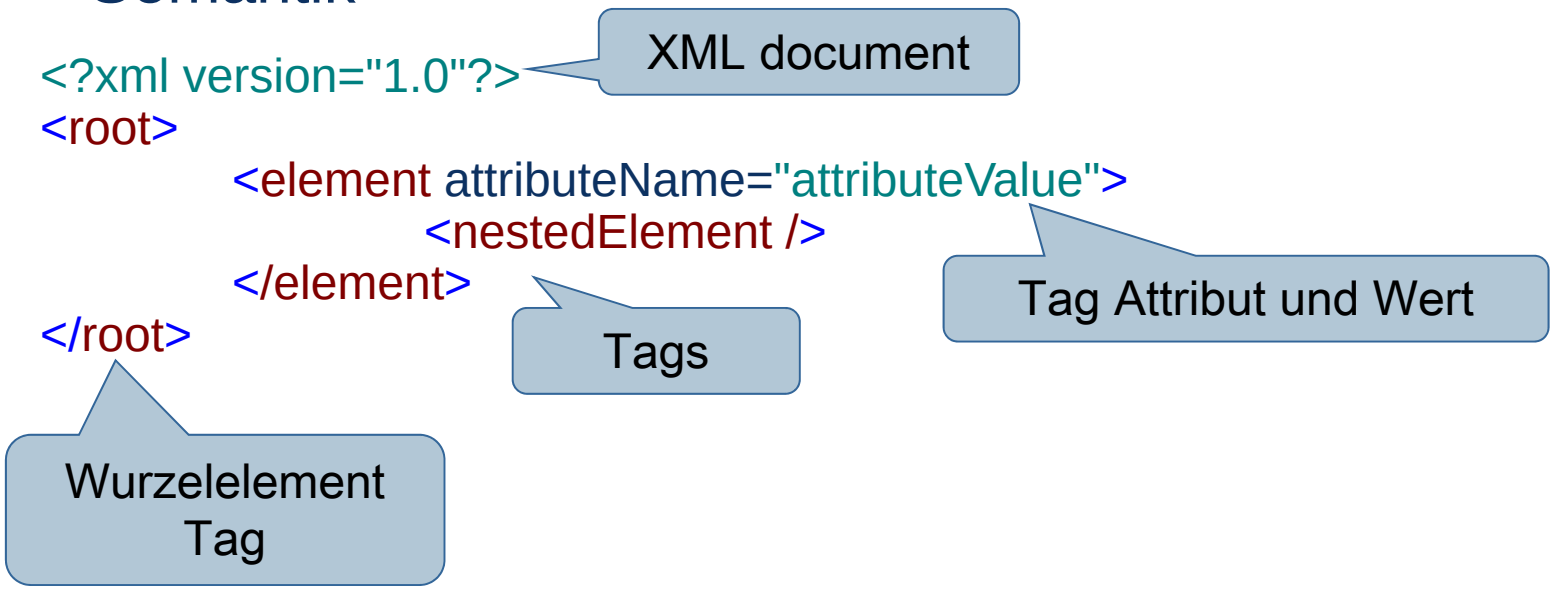
- Das Web gibt den Netzwerkstandard vor: Hypertext Transfer Protocol (HTTP)
- Kommunikationspartner werden über Web-Adressen (URI) identifiziert und angesprochen
  - URI = Universal Resource Identifier = Web Adresse
    Beispiel: http://www.spb.de/services/transfer

Definition: Web Service

- *Ein Web Service ist ein durch eine URI identifiziertes und adressierbares Softwaresystem.*
- Seine Schnittstellen inklusive ausgetauschter Daten werden mittels XML maschinenlesbar spezifiziert.
- Daten werden über die Schnittstellen im XML Format ausgetauscht.

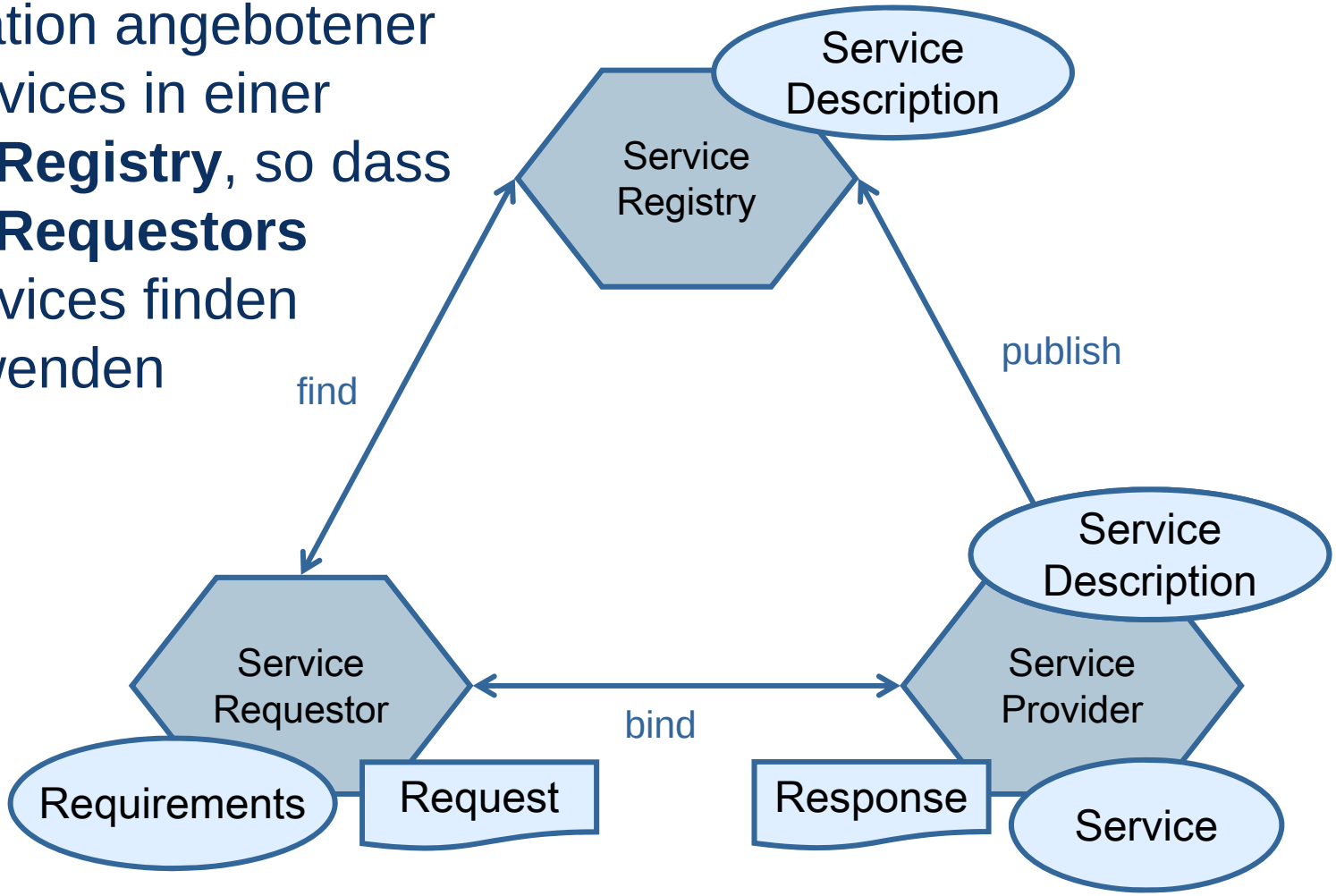
Kurze Einführung in XML

- XML ist eine Hypertext Sprache
- XML beschreibt Daten gemäß eines Schemas in einer Textdatei
- Daten werden in einer Baumstruktur abgelegt
- „tags“ definieren die Struktur und geben ihr eine gewisse Semantik



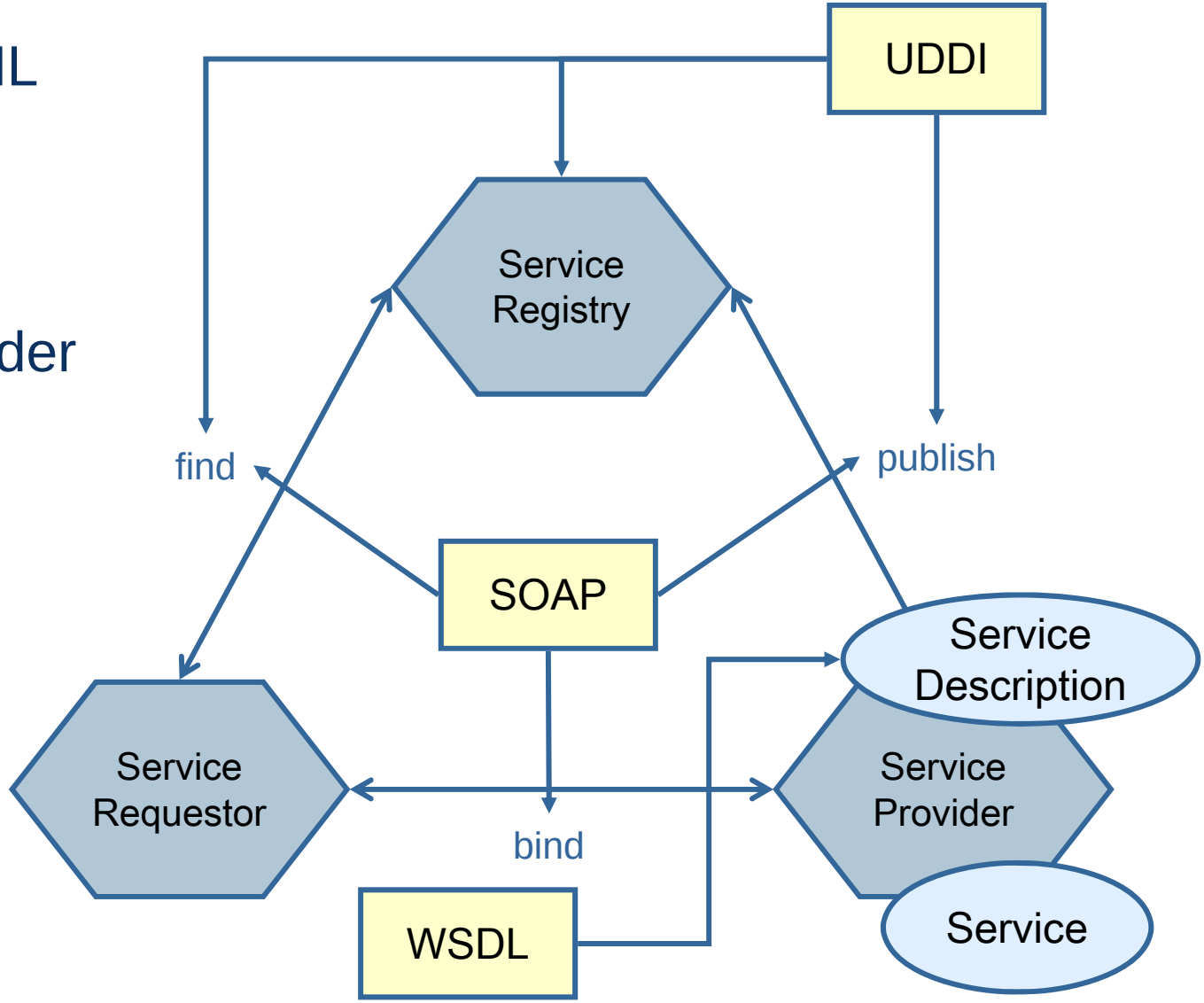
Web Service Architektur

- **Service Provider** veröffentlichen die Spezifikation angebotener Web Services in einer **Service Registry**, so dass **Service Requestors** Web Services finden und verwenden können.



Web Service Standards

- **SOAP** als XML basiertes Datenformat
- **WSDL** zur Spezifikation der Schnittstellen
- **UDDI** als universelle Registry

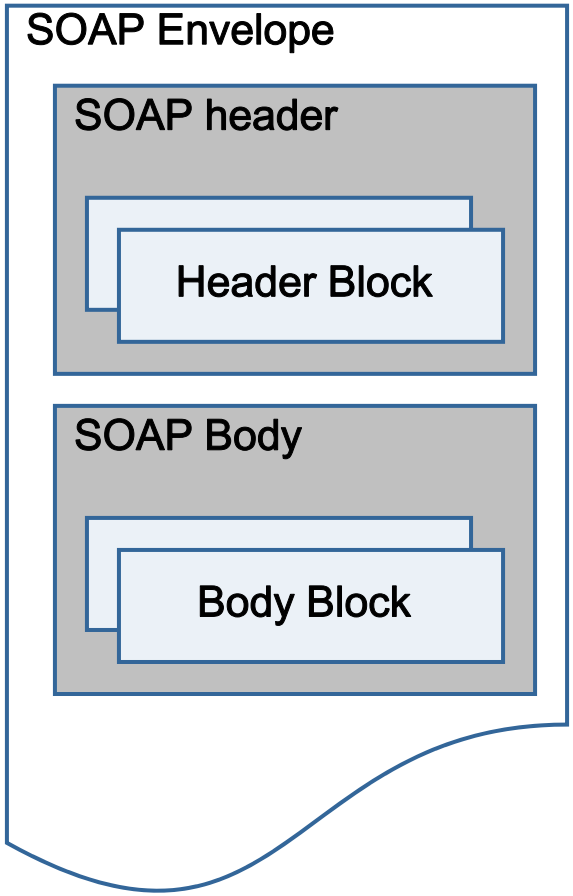


Struktur einer SOAP Nachricht

```
<?xml version="1.0"?>

<env:Envelope xmlns:env=
"http://www.w3.org/2003/05/soap-envelope">

        <env:Header>
        …
        </env:Header>

        <env:Body>
        …
        </env:Body>

</env:Envelope>
```

SOAP Envelope Tag

Namensraum SOAP envelope



Beispiel einer SOAP Nachricht

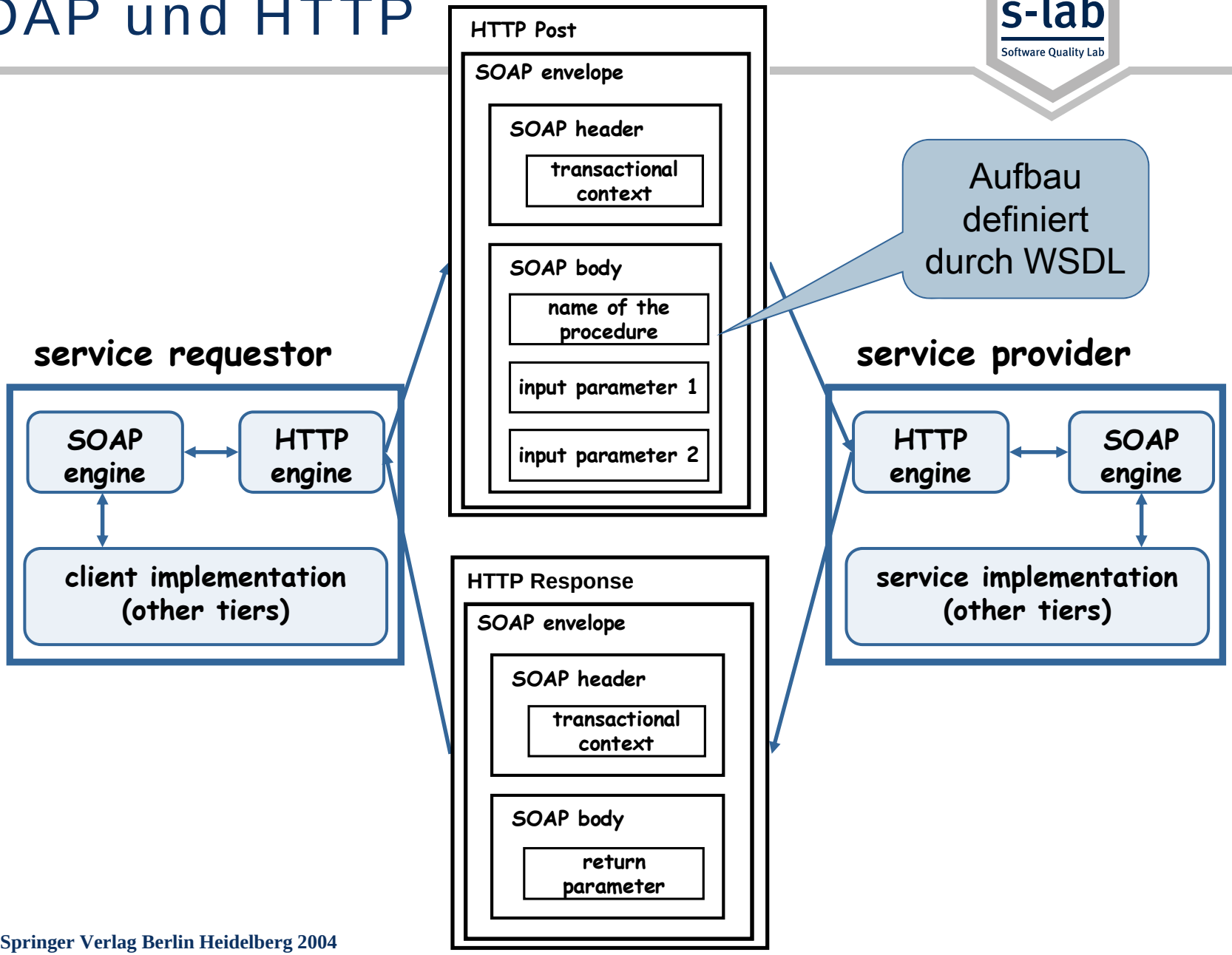
Namensraum
SOAP envelope

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
      <n:priority>1</n:priority>
      <n:expires>2001-06-22T14:00:00-05:00</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Pick up Mary at school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```

Anwendungsspezifische
Namensräume

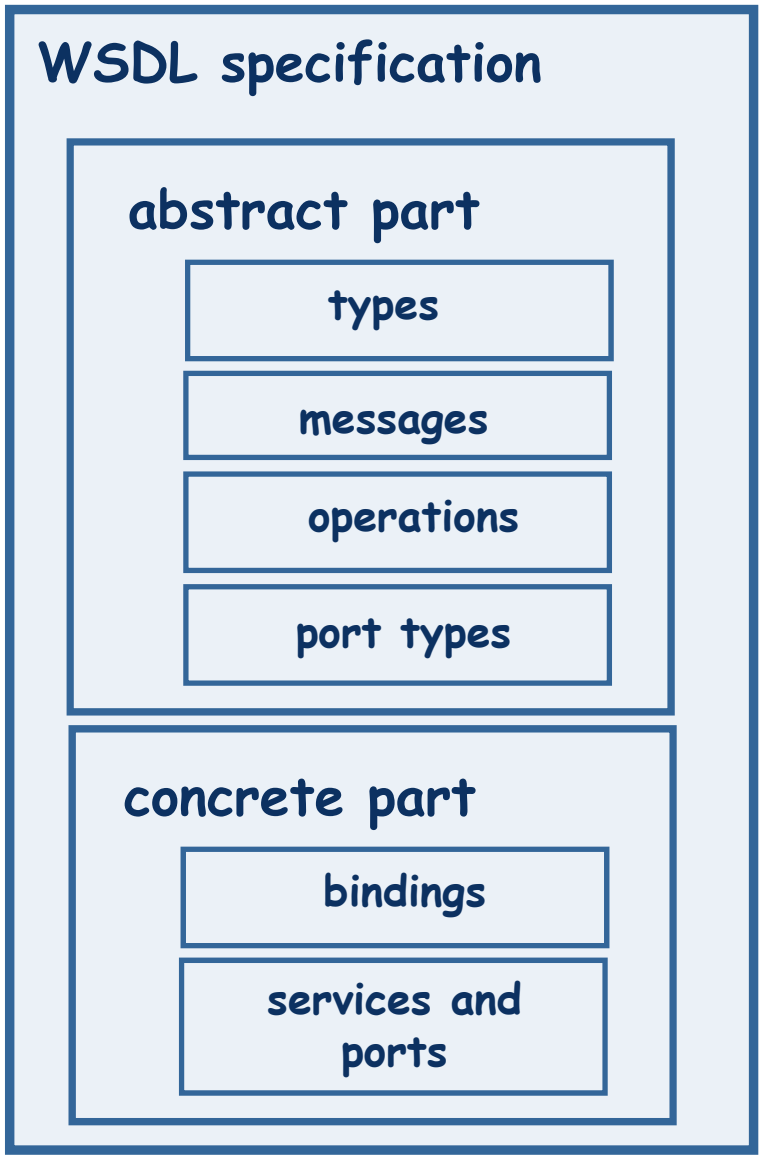
From:
SOAP Version 1.2 Part 1: Messaging Framework
W3C Recommendation 24 June 2003

SOAP und HTTP

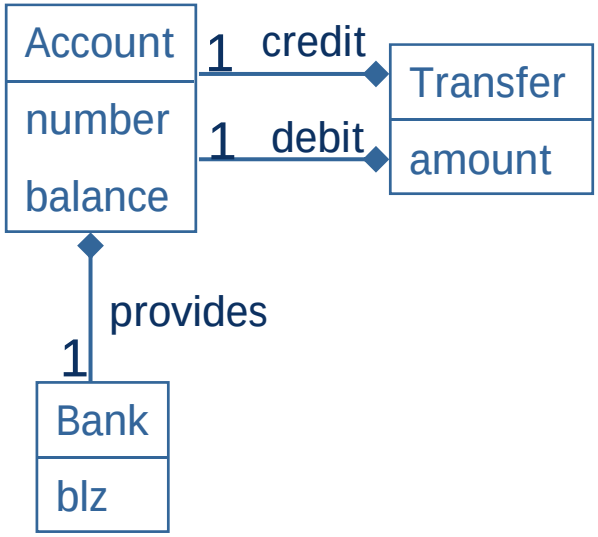


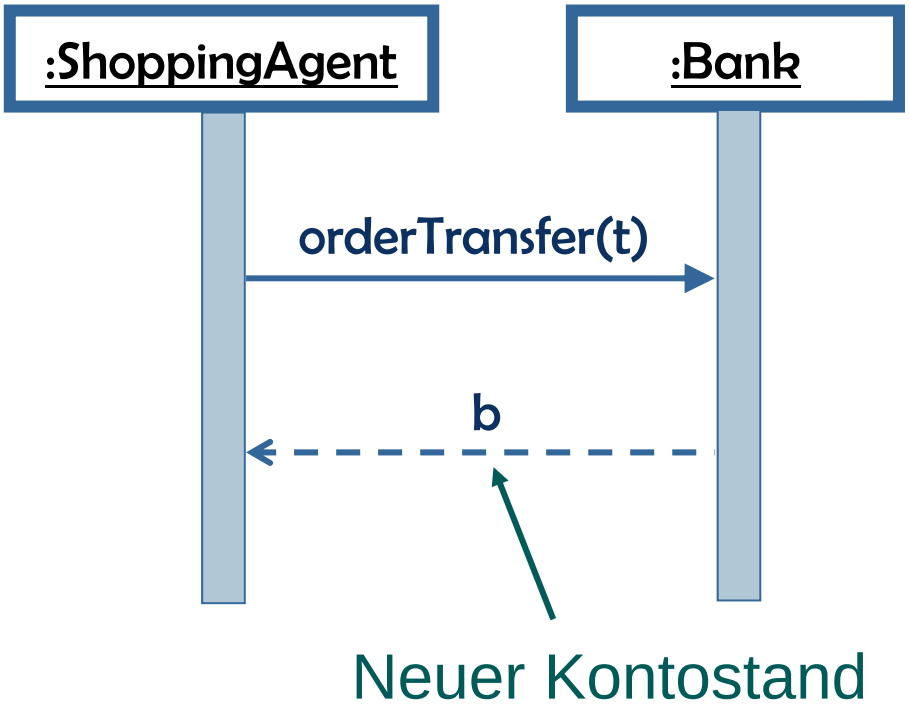
Schnittstellendefinition

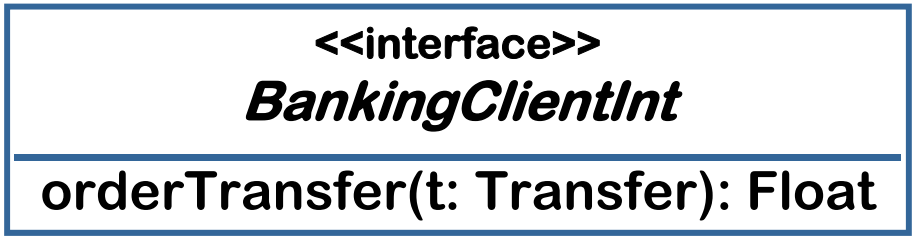
- **W**eb **S**ervice **D**escription **L**anguage



Remote Procedure Calls







SOAP Request

```
<SOAP-ENV:Envelope
SOAP-ENV:encodingStyle = "http://www.upb.de/personalEncodingStyle"
            xmlns:bs="http://www.sparkasse-pb.de/ebanking/bs“ >
  <SOAP-ENV:Header>
    <bs:banking>
            <bs:reference>uuid:093a2da1-q345-739bqa5djh895</bs:reference>
            <bs:date>2006-01-17</bs:date>
            <bs:client>engels@upb.de</bs:client>
    </bs:banking>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
            <bs:orderTransfer>
               <bs:Transfer>
                  <bs:credit> ... </bs:credit> <bs:debit> ... </bs:debit>
                  <bs:Transfer.amount>-100</bs:Transfer.amount>
               </bs:Transfer>
            </bs:orderTransfer>
   </SOAP-ENV:Body>
</ SOAP-ENV: Envelope>
```

Kontext

Operation

Parameter

SOAP Response

```
<SOAP-ENV:Envelope SOAP-ENV:
encodingStyle = "http://www.upb.de/personalEncodingStyle"
            xmlns:bs="http://www.sparkasse-pb.de/ebanking/bs“ >
    <SOAP-ENV:Header>
       <bs:banking>
          <bs:reference>uuid:093a2da1-q345-39bqa5djh895</bs:reference>
          <bs:date>2006-01-17</bs:date>
          <bs:client>engels@upb.de</bs:client>
       </bs:banking>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
            <bs:orderTransferResponse>
                        <bs:Account.balance> -500 </bs:Account.balance>
            </bs:orderTransferResponse>
    </SOAP-ENV:Body>
</ SOAP-ENV: Envelope>
```

Kontext

Response

Rückgabewert

Einsatz in der Praxis

- Service Registries werden in der Praxis selten eingesetzt. Meist direkte Service Requestor – Service Provider Kommunikation
- XML SOAP Verarbeitung erzeugt Overhead, der sich negativ auf die Performanz auswirken kann
  - Mehr Daten
  - Zusätzliche Rechenzeit für XML Serialisierung / Deserialisierung
- HTTP wird nur eingeschränkt verwendet
  - HTTP wird nur als Transportmittel gebraucht (nur POST)
  - Weitere HTTP-Operationen (z.B. PUT, DELETE) werden nicht genutzt

Einsatz in der Praxis

Web Services als Geschäftsmodell

- Online Zugriff auf Dienste:
  - Speicherung & Datenbanken
  - E-Commerce, Zahlung- und Rechnungswesen
  - Messaging

- Alle Google Dienste sind über Web Services benutzbar
- Aber, Google setzt nicht mehr auf SOAP Web Services, sondern bevorzugt schlankere Alternative.

ROA + REST API als Alternative

- ROA = **Resource Oriented Architecture**
- Web Services mit ihrer URI sind Ressourcen
- Zusätzlich wird jede Entität eine Ressource mit einer URI
- Jede Ressource ist über eine URI abrufbar

- REST = **Representational State Transfer**
- Zustandslose Kommunikation über HTTP
- Direkte Nutzung von HTTP – kein SOAP Overhead
- HTTP Operationen wie GET, POST, PUT, DELETE zur Arbeit mit Ressourcen

Ressourcen am Beispiel

- Bank
  - http://www.spb.de/
- Alle Kunden der Bank
  - http://www.spb.de/clients
- Spezifischer Kunde über angehängte Kundennummer
  - http://www.spb.de/clients/0815
- Alle Konten eines Kunden
  - http://www.spb.de/clients/0815/accounts
- Spezifisches Konto eines Kunden über Kontonummer
  - http://www.spb.de/clients/0815/accounts/4538734
- Kontostand des Kontos
  - http://www.spb.de/clients/0815/accounts/4538734/balance

Repräsentation von Ressourcen

- Ressourcen können je nach gewünschtem Format unterschiedlich repräsentiert und abgefragt werden.

- GET text/html http://www.spb.de
  gibt die HTML Web-Seite der Sparkasse zurück

- GET application/xml http://www.spb.de
  gibt Daten über Sparkasse PB im XML Format zurück

```
<?xml version="1.0" encoding="UTF-8"?>
<bank>
        <name>Sparkasse Paderborn</name>
        <blz>47250101</blz>
        <city>Paderborn</city>
</bank>
```

JSON als Alternative zu XML

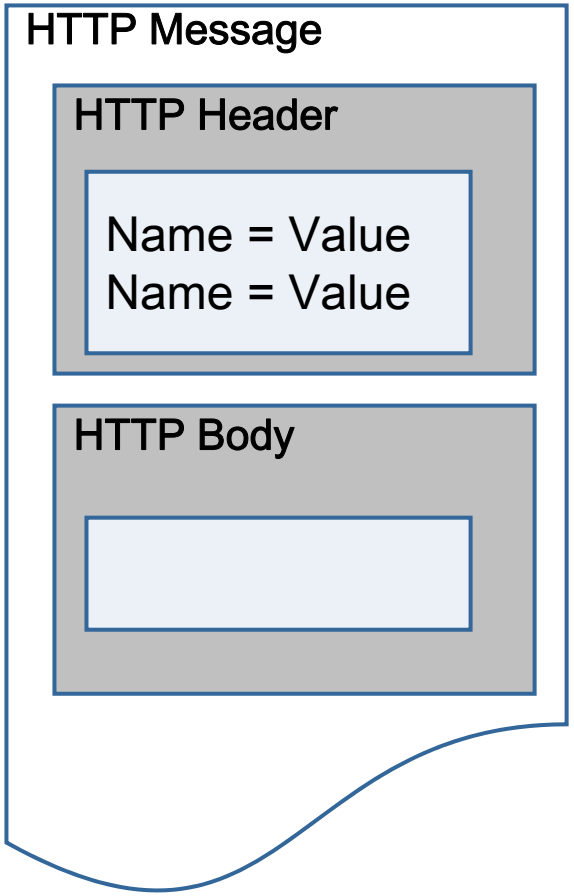
- Situation: Vermehrter Einsatz von JavaScript Clients
- XML zu aufwendig für Clients – Wunsch nach schlankerem Format: **JavaScript Object Notation**
- Beispiel Daten über Sparkasse PB im JSON Format

```
{
        “name” : “Sparkasse Paderborn”,
        “blz” : “47250101”,
        “city” : “Paderborn”
}
```

- Nicht so mächtig wie XML mit einem wohl definierten Schema, aber in vielen Anwendungsfällen ausreichend.

REST API

- Programmierschnittstelle auf Ressourcen nach REST Konzept
- Web Service orderTransfer wird zur POST Operation auf der Ressource Kontostand
- Transaktionelle Informationen werden im HTTP Header abgelegt
- Gewünschtes Datenformat (JSON, XML) wird im Header definiert
- Nutzdaten werden im HTTP Body mitgeschickt



Beispiel REST Aufruf

- Request

Operation

Ressource

```
POST http://www.spb.de/clients/0815/accounts/4538734/balance
HTTP-Header:
        accept: application/json
        reference: uuid:093a2da1-q345-39bqa5djh895
HTTP-Body:    {
                    “amount” : -100
                }
```

Datenformat

Kontext

Parameter

- Response

```
HTTP-Header:
        reference: uuid:093a2da1-q345-39bqa5djh895
HTTP-Body:    {
                    “balance” : -500
                }
```

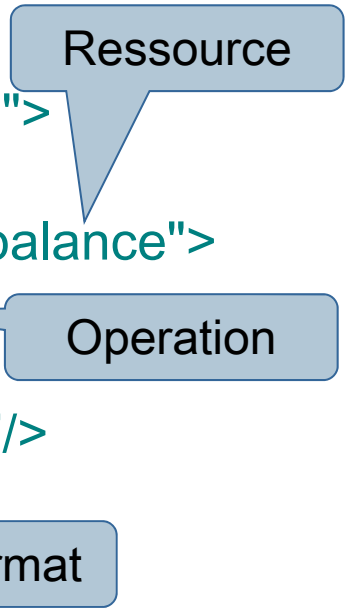
Kontext

Rückgabewert

REST Schnittstellen

- WADL – **Web Application Description Language**
- GET http://www.spb.de/application.wadl

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://research.sun.com/wadl/2006/10">
   <resources base="http://www.spb.de/">
      <resource path="/clients/{cid}/accounts/{accountNo}/balance">
         <method name="POST" id="orderTransfer">
            <response>
               <representation mediaType="application/json"/>
            </response>
         </method>
      </resource>
   </resources>
</application>
```



Diskussion

- **SOAP Web Services** sind komplex, aber Frameworks reduzieren den Aufwand deutlich.
- Fortgeschrittene Standardisierung:
  WS-Security, WS-Conversation, WS-*
  Problem hier: (zu) große Anzahl WS-* Standards
- Anbieter wie Google suchen neue (effizientere) Wege

- **REST Web Services** weniger komplex, leichter zu lernen, benötigen keine aufwendigen Frameworks
- WADL Spezifikation in frühem Stadium
- Noch keine Standards z.B. für Security, Conversation etc.
- Aber durch Einfachheit hohe Beliebtheit in modernen Web-Anwendungen

Acknowledgements

Prof. Dr. Gustavo Alonso, ETH Zürich

- http://www.inf.ethz.ch/personal/alonso/WebServicesBook

- G. Alonso, F. Casati, H. Kuno, V. Machiraju (eds.): Web Services – Concepts, Architectures and Applications. Springer 2004

Vielen Dank für Ihre Aufmerksamkeit.

s-lab – Software Quality Lab
Universität Paderborn
Warburger Str. 100
33098 Paderborn
Tel.: (05251) 60 5390 / 5391

http://s-lab.upb.de
info@s-lab.upb.de