Freie Universität Berlin

Web Services

Block XML

Freie Universität Berlin

Vorlesungen – 6 Termine	Übung – 5 Termine
XML-Grundlagen einsch. Namenräume	XML-Syntax, Namensräume
DTD & XML-Schema	DTD
XML-Schema im Detail	XML Schema
SAX & DOM Parser	
XSLT	XPath, XSLT
XML und Datenbanken	XML & Datenabanken

Block Web Services

Freie Universität Berlin

Vorlesungs-termin	Vorlesung 4 + 1 + 1 Termine	Übung – 2 Termine	Übungs-termin
06.06. (heute)	Web Services, RPCs vs. Messaging		
20.06.	SOAP im Detail	SOAP	25./26.06
27.06.	WSDL im Detail	WSDL	02./03.07
04.07.	Web Services in der Praxis & Ausblick		
11.07.	Rückblick + (14:00-16:00) Sprechstunde vor der Klausur, Fabeckstr. 15		
18.07.	Klausur		

Block Wiederholung + Klausur

Freie Universität Berlin

Vorlesungs-termin	Vorlesung 4 + 1 + 1 Termine	Übung – 2 Termine	Übungs-termin
06.06. (heute)	Web Services, RPCs vs. Messaging		
20.06.	SOAP im Detail	SOAP	25./26.06
27.06.	WSDL im Detail	WSDL	02./03.07
04.07.	Web Services in der Praxis & Ausblick		
11.07.	Rückblick + (14:00-16:00) Sprechstunde vor der Klausur, Fabeckstr. 15		
18.07.	Klausur		

Block Web Services

Freie Universität Berlin

heutige Vorlesung

- Was sind Web Services?
- Web Services – Basistechnologien:
  - SOAP
  - WSDL
  - UDDI
- Enterprise Application Integration
- Dienstorientierte Architektur (SOA)
- RPC vs. Messaging

Was sind Web Services?

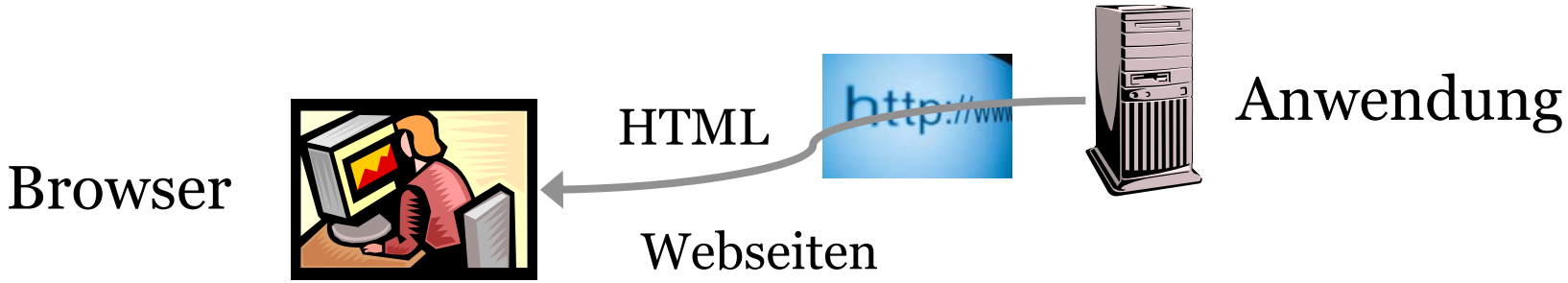
Entwicklung

Freie Universität Berlin

- Email: Mensch-Mensch-Kommunikation
  - Versendung von uninterpretiertem Text
- WWW: Mensch-Computer-Kommunikation
  - Mensch – aktive: konsumiert & gibt an, was er als nächstes lesen möchte.
  - der Computer – passiv
- Web Service: Computer-Computer-Kommunikation
  - Computer erbringt für einen anderen eine Dienstleistung

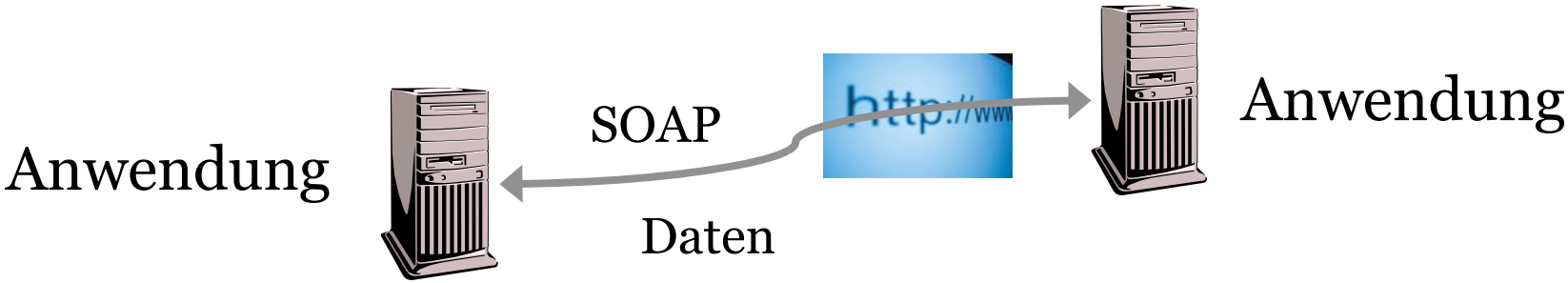
Was sind Web Services?

traditionelle Web-Anwendung



➔ Mensch-Maschine-Kommunikation

Web Service



➔ Maschine-Maschine-Kommunikation

Beispiel: Google ohne Browser

»With Google Web APIs, your computer can do the searching for you.«

Google als Web Service

- Suche
- Rechtschreibkorrektur
- Zugriff auf Web-Cache

Suche als Web Service

- Suchanfrage als SOAP-Nachricht
- Suchergebnis als SOAP-Nachricht

➔ http://www.google.com/apis/

Google - Suche ohne Browser

Freie Universität Berlin

- Google-Suche kann aus Anwendungsprogramm heraus aufgerufen werden

mögliche Anwendungen

- in periodischen Abständen zu bestimmten Thema nach neuen Webseiten zu suchen:

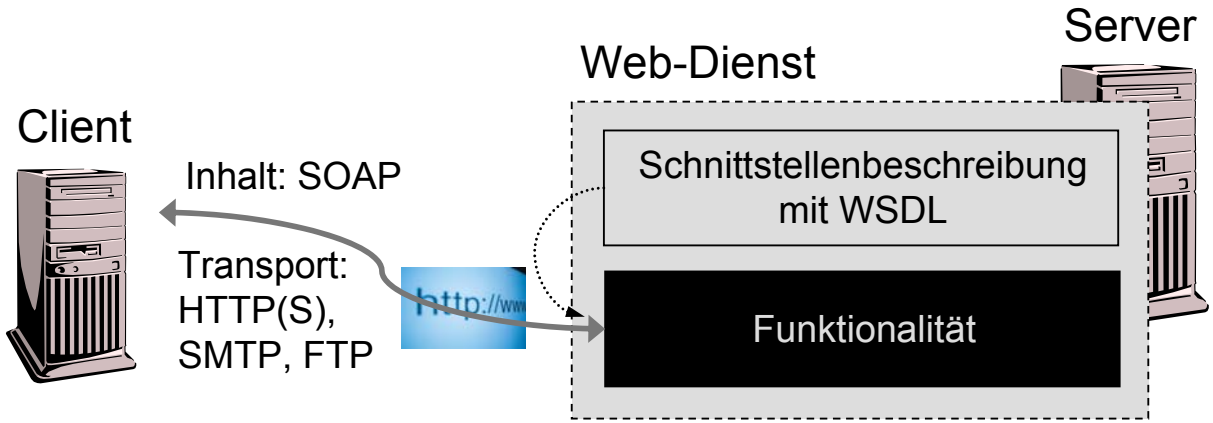
Web Alert

search-for: "XSLT 2.0 Recommendation"

notify new web page: mymail@inf.fu-berlin.de

- automatisch neue Trends im WWW zu identifizieren

Definition



Ein Web Service ist eine Softwareanwendung, die

1. mit einer URI eindeutig identifizierbar ist,
2. über eine WSDL-Schnittstellenbeschreibung verfügt,
3. nur über die in ihrer WSDL beschriebenen Methoden zugreifbar ist und
4. über gängige Internet-Protokolle unter Benutzung von XML-basierten Nachrichtenformaten wie z.B. SOAP zugreifbar ist.

Eigenschaften von Web Services

Freie Universität Berlin

- implementieren häufig keine neuen Systeme, sondern sind Fassade für bestehende Systeme
- abstrahieren von Programmiersprache und Plattform mit der die Anwendung realisiert ist:
  Virtualisierung von Software

- zwei Erscheinungsformen:
  - RPCs (synchron)
  - Messaging (asynchron)

Alter Wein in neuen Schläuchen?

Freie Universität Berlin

- Web Services keine revolutionär neue Technologie
- große Ähnlichkeiten mit CORBA

Neu ist jedoch:

- alle bedeutenden IT-Unternehmen auf Standards geeinigt: XML, SOAP und WSDL
- CORBA hingegen nie von Microsoft unterstützt

Außerdem neu:

- statt proprietäre Protokolle (wie IIOP und DCOM) gängige Internet-Protokolle
- nicht nur RPCs, sondern auch Messaging

Protokolle

Freie Universität Berlin

Layer	Protokoll/Standards
Messaging	HTML,...
Transport	HTTP, FTP, SMTP
Network	TCP/IP, UDP

Die Protokolle und Standards aller Layer nur in Auswahl
Quelle: http://www.aki-dpg.de/Dokumente/Bad_Honnef_2003/webservicestutorial.pdf

XML-gekapselte Protokolle

Freie Universität Berlin

Layer	Protokoll/Standards
Content	Content Information
Messaging	SOAP, XML-RPC
Transport	HTTP, FTP, SMPT
Network	TCP/IP, UDP

Die Protokolle und Standards aller Layer nur in Auswahl
Quelle: http://www.aki-dpg.de/Dokumente/Bad_Honnef_2003/webservicestutorial.pdf

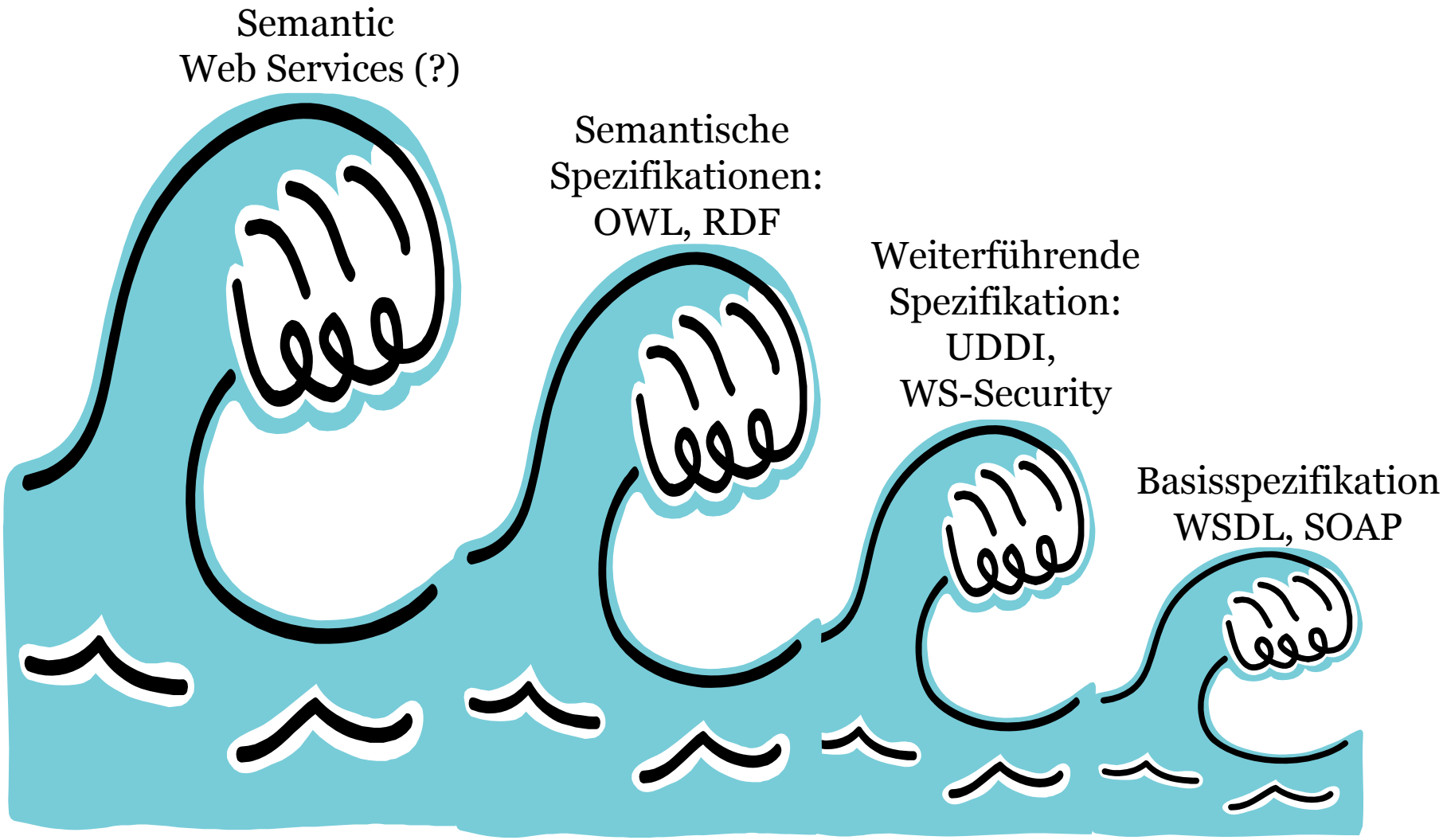
Web Service Technology Stack

Freie Universität Berlin

Layer	Protokoll/Standards
Discovery	UDDI, DISCO, WSIL
Description	WSDL, RDF
Messaging	SOAP, XML-RPC
Transport	HTTP, FTP, SMPT
Network	TCP/IP, UDP

Die Protokolle und Standards aller Layer nur in Auswahl
Quelle: http://www.aki-dpg.de/Dokumente/Bad_Honnef_2003/webservicestutorial.pdf

Web Services Wellen



Web Service Basiskomponenten:

SOAP

Austausch einer SOAP-Nachricht

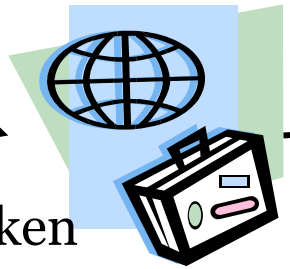
Freie Universität Berlin

Sender

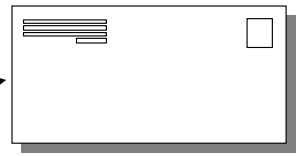
Empfänger

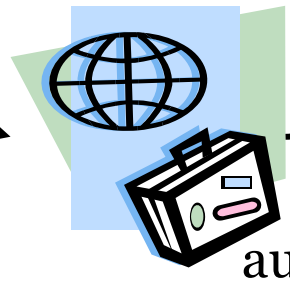
Information

verpacken
(serialisieren)



Nachricht

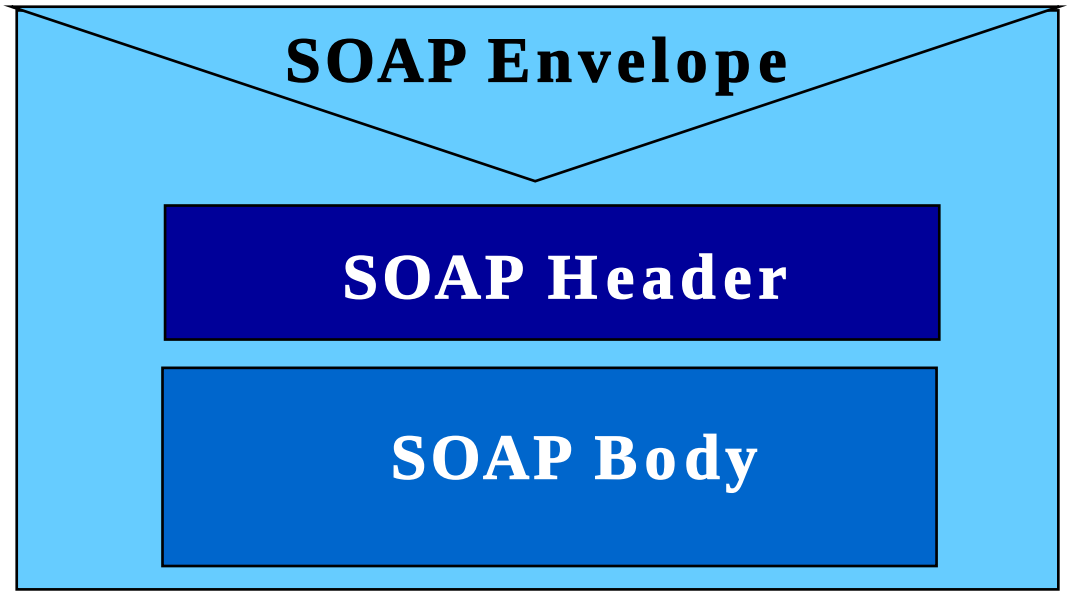




Information

auspacken
(deserialisieren)

Aufbau einer SOAP-Nachricht




```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap/envelope/">
    <!- SOAP Header -->
    <!- SOAP Body -->
</env:Envelope>
```

SOAP Version 1.2

SOAP Version 1.1:
http://schemas.xmlsoap.org/soap/envelope/

Nachrichtenformat SOAP

HTML

```
<html>
  <head>
    Zusatzinformationen
  </head>
  <body>
    Inhalt: Webseite
  </body>
</html>
```

SOAP

```
<Envelope>
  <Header>
    Zusatzinformationen
  </Header>
  <Body>
    Inhalt: XML-Daten
  </Body>
</Envelope>
```

- XML-basierter W3C-Standard
  - SOAP 1.1: W3C Note von 2000
  - SOAP 1.2: W3C Recommendation von 2003
- seit SOAP 1.2: SOAP ≠ Simple Object Access

Eine SOAP-Anfrage an Google

Freie Universität Berlin

```
<?xml version='1.0' encoding='UTF-8'?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
                    xmlns:xsd="..."  xmlns:xsi="…">
  <env:Body>
    <doGoogleSearch xmlns="urn:GoogleSearch">
      <key xsi:type="xsd:string">328975487054809 7</key>
      <q xsi:type="xsd:string">Eine Anfrage</q>
      <start xsi:type="xsd:int">0</start>
      <maxResults xsi:type="xsd:int">10</maxResults>
        …
    </doGoogleSearch>
  </env:Body>
</env:Envelope>
```

- doGoogleSearch(key, q, start, maxResults,...)
- hier kein Header
- Beachte: Web Service-Aufruf kann als URL kodiert werden (REST)

Und die Antwort von Google

Freie Universität Berlin

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="..."  xmlns:xsi="...">
    <env:Body>
      <ns1:doGoogleSearchResponse xmlns:ns1="urn:GoogleSearch" ...>
          <return xsi:type="ns1:GoogleSearchResult">
                ...
          </return>
      </ns1:doGoogleSearchResponse>
  </env:Body>
</env:Envelope>
```

- Antwort: doGoogleSearchResponse(return(...))
- Datentyp ns1:GoogleSearchResult in WSDL-Beschreibung definiert

REpresentational State Transfer

Freie Universität Berlin

- kein Protokoll sondern ein Architekturstil
- Verwaltet beliebige Menge von Ressourcen
- RESTful → eine Anwendung konform zum REST-Architekturstil

- Amazon – der populärster Anbieter einer REST-Anwendung
- Google bietet solche REST-Schnittstelle NICHT an

REST bei Amazon

Freie Universität Berlin

http://ecs.amazonaws.com/onca/xml?Service=AWSECommerceService&
AWSAccessKeyId=[ID]&Operation=ItemSearch&SearchIndex=Books&
Title=Harry%20Potter

Ausschnitt der Antwort des Amazon Servers

```
– <ItemSearchResponse>
  …
  – <Items>
    – <Request>
      …
      – <ItemSearchRequest>
          <SearchIndex>Books</SearchIndex>
          <Title>Harry Potter</Title>
        </ItemSearchRequest>
      </Request>
      <TotalResults>1076</TotalResults>
      <TotalPages>108</TotalPages>
    – <Item>
        <ASIN>0545010225</ASIN>
      – <DetailPageURL>
          http://www.amazon.com/gp/redirect.html%3FASIN=0545010225%26tag=ws%26l
        </DetailPageURL>
      – <ItemAttributes>
          <Author>J. K. Rowling</Author>
          <Creator Role="Illustrator">Mary GrandPré</Creator>
          <Manufacturer>Arthur A. Levine Books</Manufacturer>
          <ProductGroup>Book</ProductGroup>
          <Title>Harry Potter and the Deathly Hallows (Book 7)</Title>
        </ItemAttributes>
      </Item>
      …
    </Items>
</ItemSearchResponse>
```

Übertragung von SOAP-Nachrichten

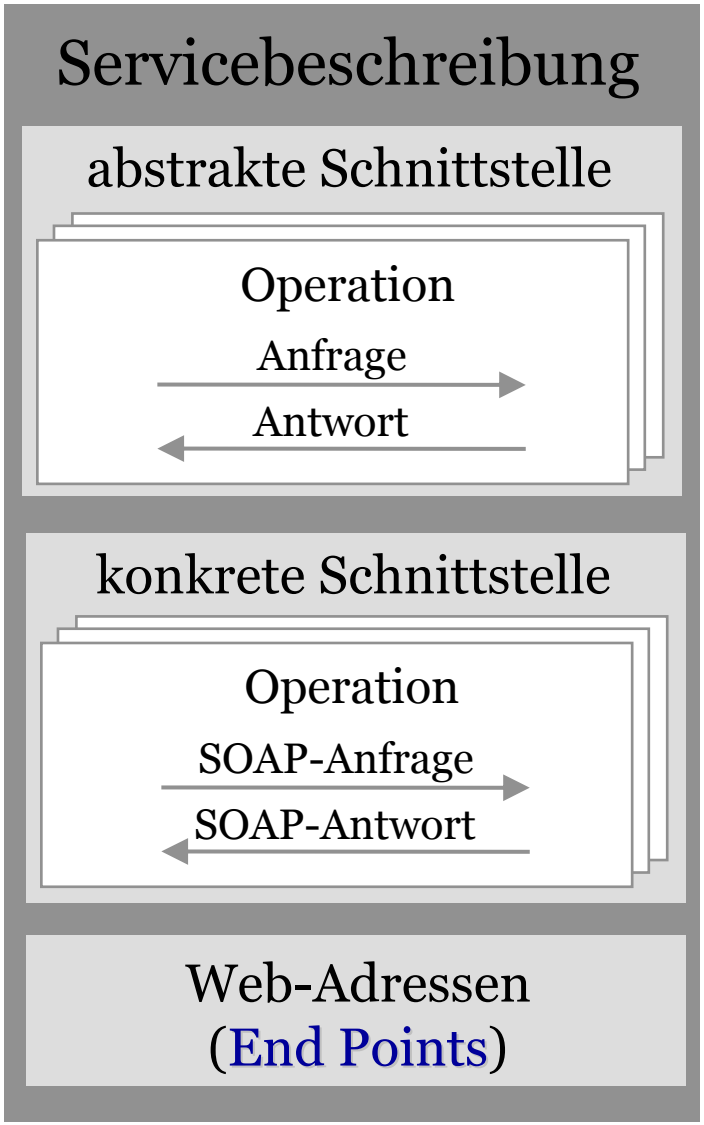
Freie Universität Berlin

- heute meist über HTTP oder HTTPS
- Request-Response-Verhalten von HTTP unterstützt entfernte Prozeduraufrufe (RPCs).
- mit HTTP auch RPCs über eine Firewall hinweg
- Übertragung aber auch z.B. mit SMTP (Messaging) möglich

Web Service Basiskomponenten:

WSDL

Web Services Description Language



- beschreibt Interface (IDL)
- XML-basierter Standard
- W3C Note von 2001

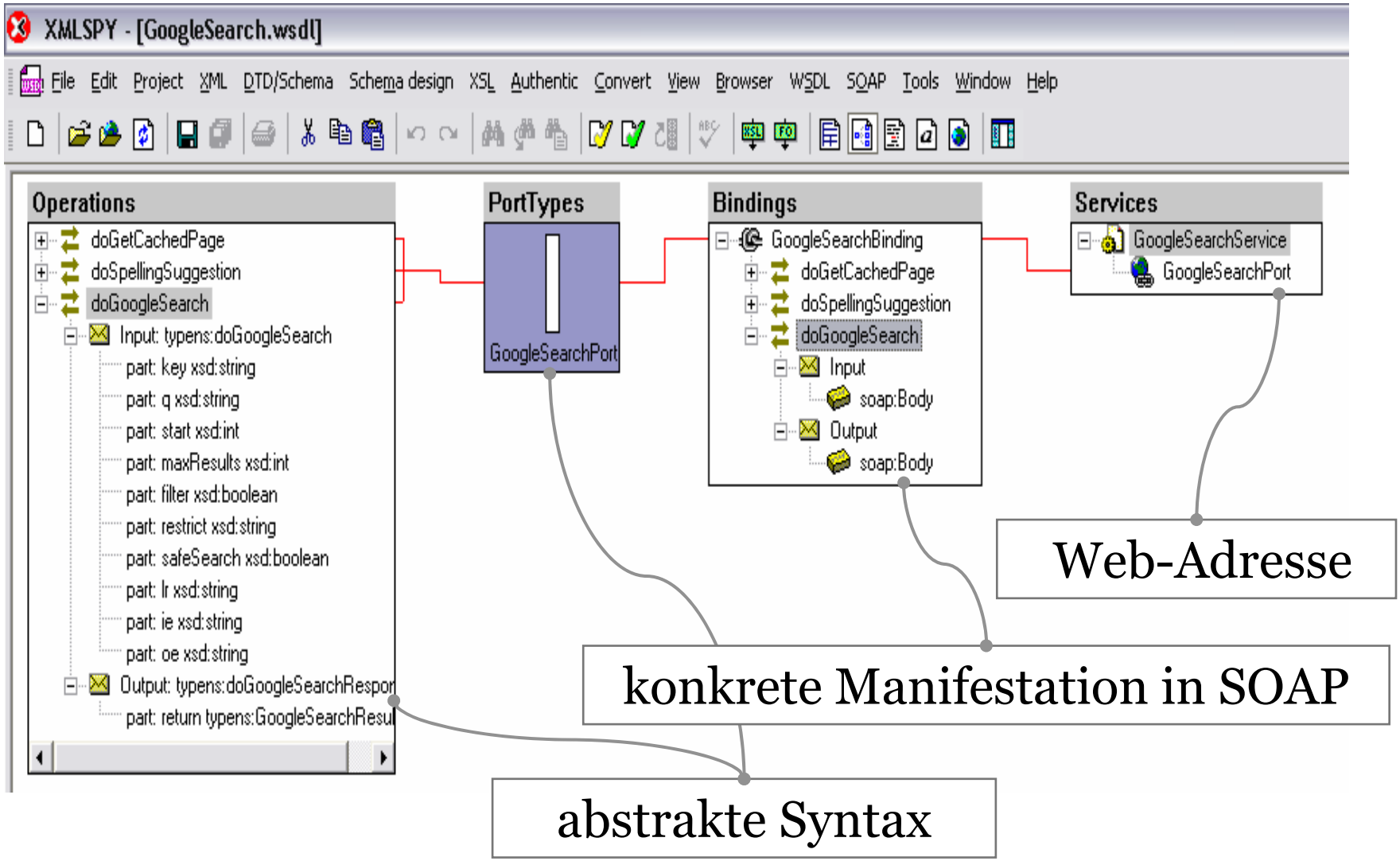
abstrakte Schnittstelle (port type)

- Schnittstelle unabhängig von Nachrichtenformaten
- Beschreibung mit XML-Schema

konkrete Schnittstelle (binding)

- Abbildung der abstrakten Schnittstelle auf unterstützte Nachrichtenformate

WSDL-Beschreibung von Google



Eigenschaften von WSDL

Freie Universität Berlin

- beschreibt Schnittstelle(n) eines Web Services und wo dieser abgerufen werden kann
- baut auf XML-Schema auf

⇨ Syntax einer Schnittstelle kann bis ins kleinste Detail festgelegt werden.

- Grundlegende Interaktionsmuster (wie Anfrage-Antwort) können beschrieben werden.
- Semantische Eigenschaften können nicht beschrieben werden:
  - Funktionalität
  - Verfügbarkeit
  - etc.

Web Service Basiskomponenten:

UDDI

UDDI

Freie Universität Berlin

- Universal (Service) Description, Discovery, and Integration
- entwickelt seit Herbst 2000 von Ariba, Microsoft& IBM
- beschreibt einen Verzeichnisdienst für Web Services
- ein SOAP basierter Dienst

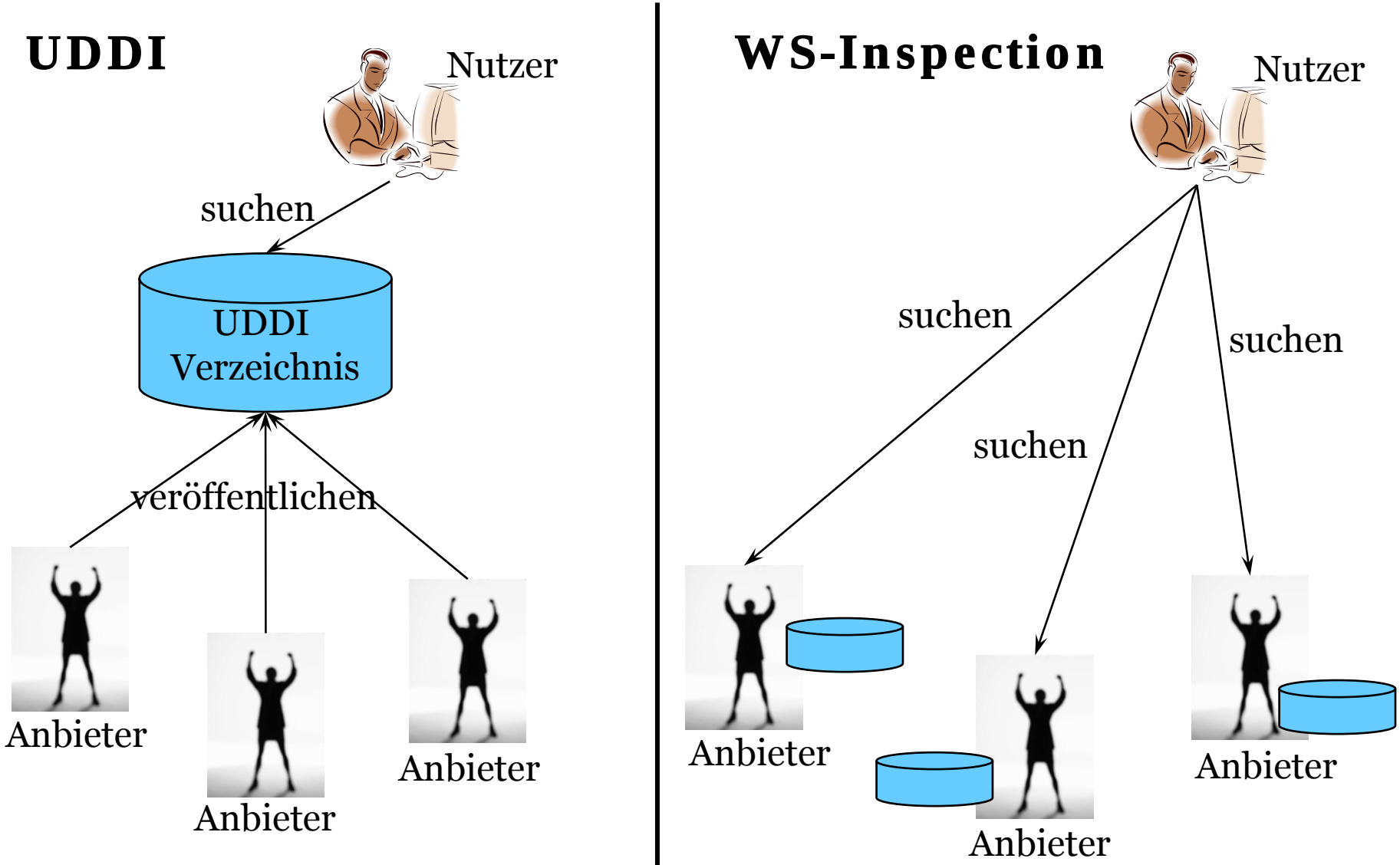
UDDI Komponenten

Freie Universität Berlin

- UDDI-XML-Schema
  - Business-Entity
  - Business-Service
  - Binding-Template
  - Technisch Modelle (TModel)

- UDDI-API
  - Anwendungsschnittstelle in Form von Web Services

UDDI vs. WS-Inspection



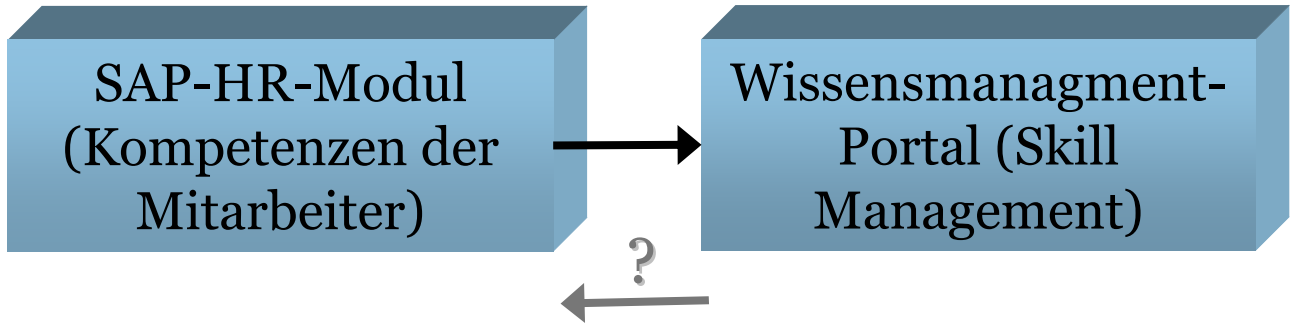
Freie Universität Berlin

Enterprise Application Integration

Enterprise Application Integration

- technologisch: inkompatible IT-Systeme miteinander verbinden
  - inkompatibel kann bedeuten:
    - unterschiedliche Betriebssysteme
    - unterschiedliche Programmiersprachen
    - unterschiedliche Kommunikationsprotokolle

- organisatorisch: Geschäftsprozesse optimieren
- Beispiel:



Zwang zur Systemintegration

Freie Universität Berlin

unternehmensintern

- Beispiel Mercedes-Benz-Werk
  - mehr als 200 EDV-Systeme
  - sehr gut vernetzt (LAN)
  - Systeme also prinzipiell integrierbar
- Neue Systeme müssen Alt-Systeme integrieren.

unternehmensübergreifend

- E-Business setzt Zusammenarbeit von heterogenen Systemen voraus:
  - Unternehmen ⇔ Unternehmen
  - Unternehmen ⇔ Portal

Bedeutung der Systemintegration

Freie Universität Berlin

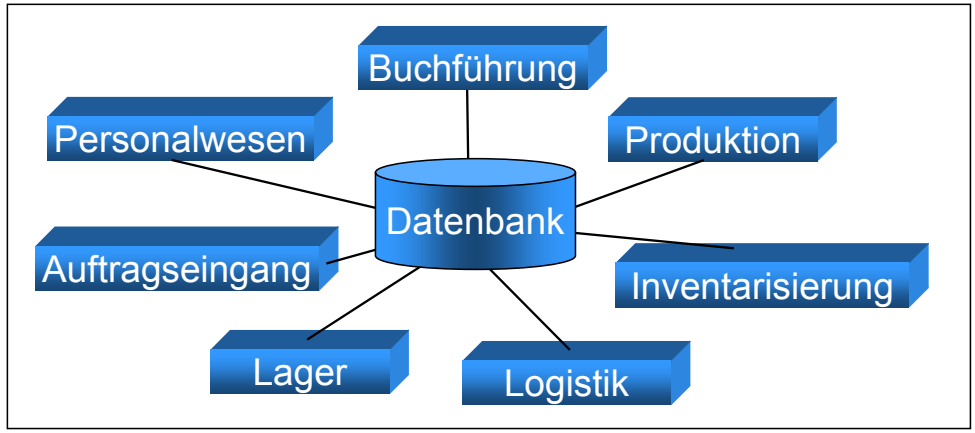
Systemintegration bindet

- 35% der IT-Personal-Ressourcen eines Unternehmens (Forrester Research, 2002)

- 65% der Arbeitszeit eines Programmierers (Gartner)

Enterprise Resource Planning

?

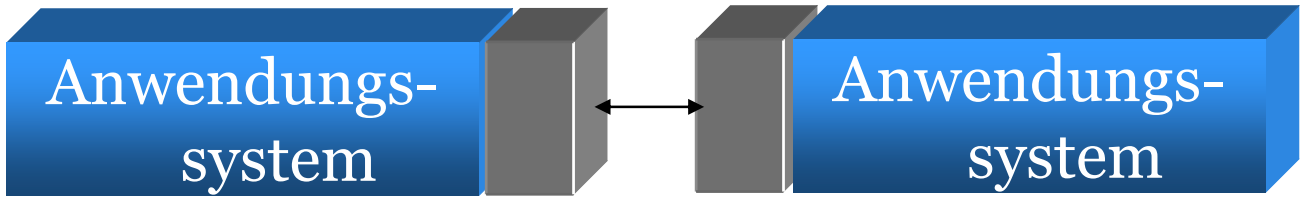


- Beispiel: SAP/R3
- großer Fortschritt für Datenintegration
- sehr teuer:

  > 53.000 $ (TCO) pro Nutzer für ersten 2 Jahre (Meta Group, 2002)
- viele Datensilos durch ein großes Datensilo ersetzt:
- schwierige Integration externer Systeme

Systemintegration mit Web Services

- Anwendungssysteme durch standardisierte Schnittstelle erweitern



  - Schnittstelle muss allgemein akzeptiert sein
  - bei Web Services ist dies der Fall
    - ✓ SOAP
    - ✓ WSDL
    - ✓ Internet-Protokolle
  - bei n Systemen:
    statt n^2 Schnittstellen nur n Erweiterungen!

Freie Universität Berlin

Dienstorientierte Architektur

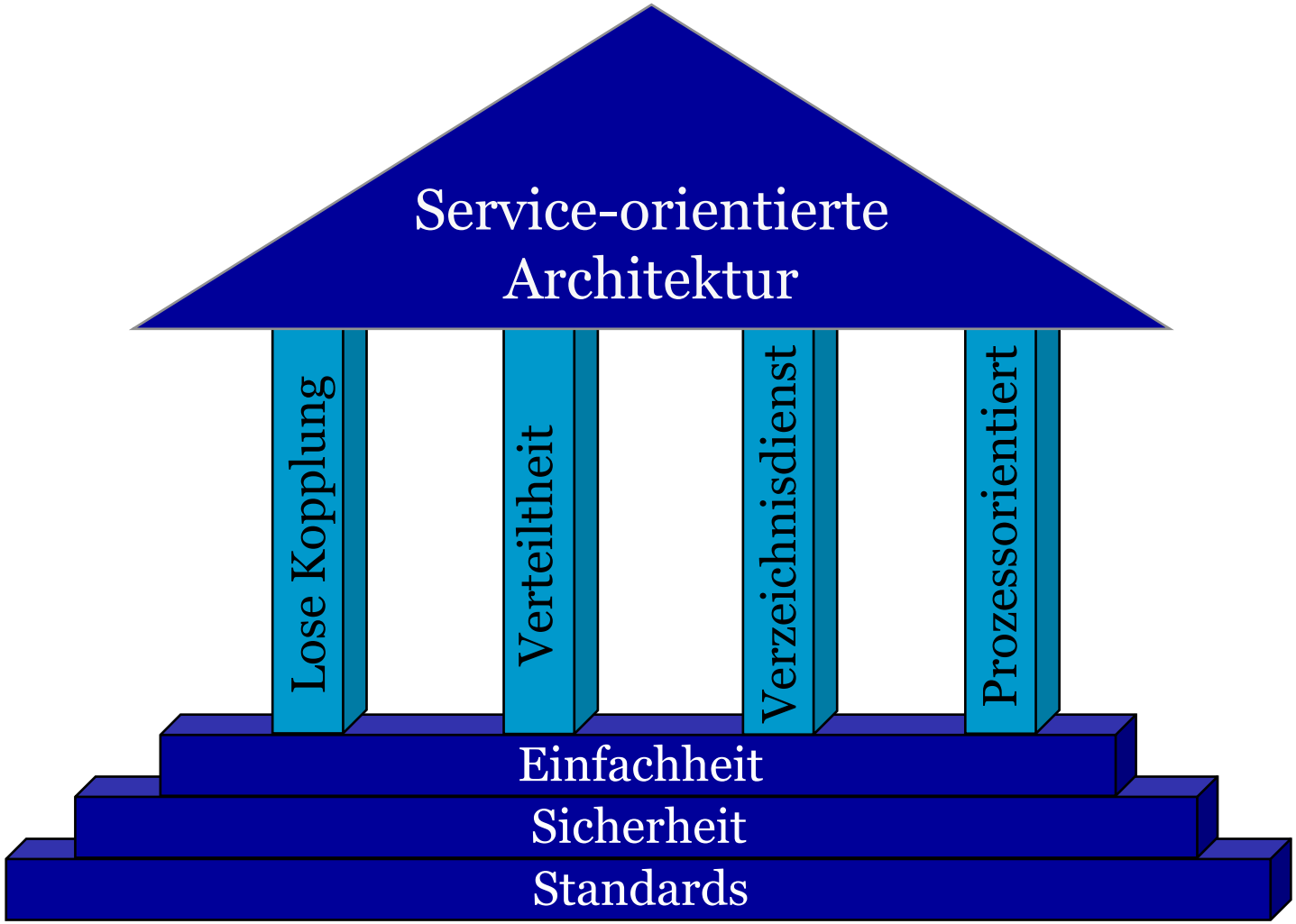
Dienstorientierte Architektur

- engl. service-oriented architecture, kurz SOA
- statt Anwendungen isoliert zu entwickeln, nur um sie später zu integrieren:
- neue Anwendungen von Anfang an auf existierenden Web Services aufbauen
- neue Anwendung wiederum als Web Service anbieten

*„... eine Systemarchitektur, die vielfältige, verschiedene und eventuell inkompatible Methoden oder Applikationen als wiederverwendbare und offen zugreifbare Dienste repräsentiert und dadurch eine plattform- und sprachunabhängige Nutzung und Wiederverwendung ermöglicht.“**

*Quelle: „ *Service-orientierte Architekturen mit Web Services: Konzepte – Standards – Praxis*“, W. Dostal, M. Jeckle, I. Melzer, B. Zengler; Spektrum Akademischer Verlag, 2005

SOA Tempel



Beispiel

Freie Universität Berlin

- konsequente Realisierung einer diensteorientierten Architektur:

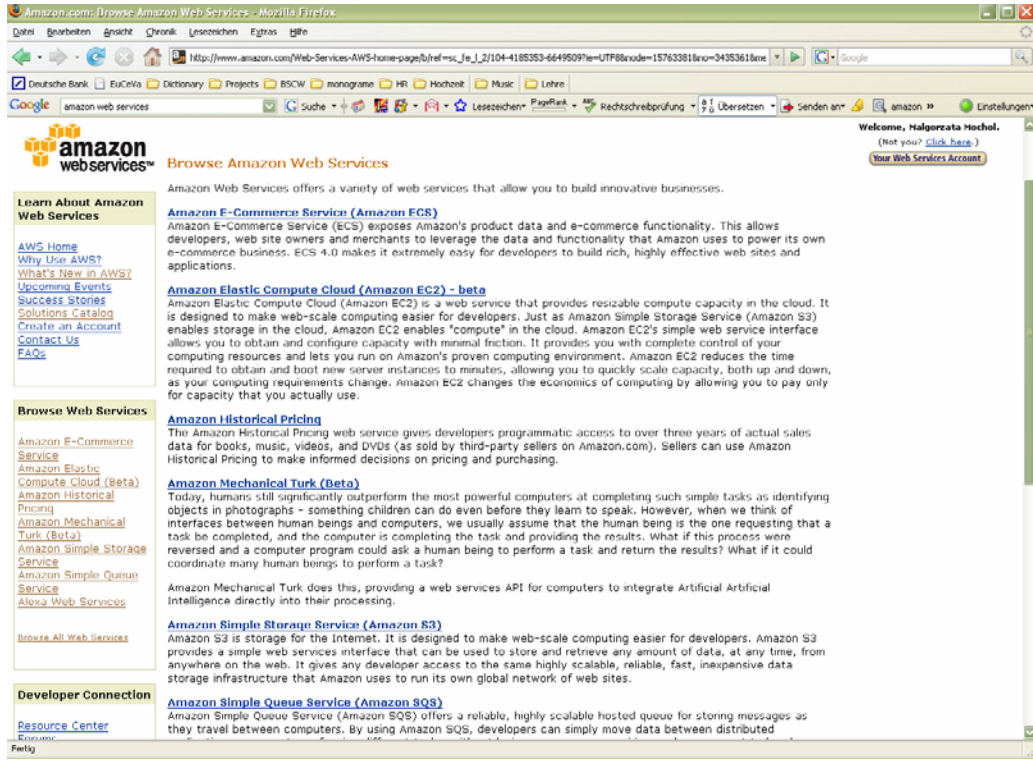
> If you hit the Amazon.com gateway page, the application calls more than 100 services to collect data and construct the page for you (Werner Vogels, CTO von Amazon)

Beispiele für (interne) Web Services:

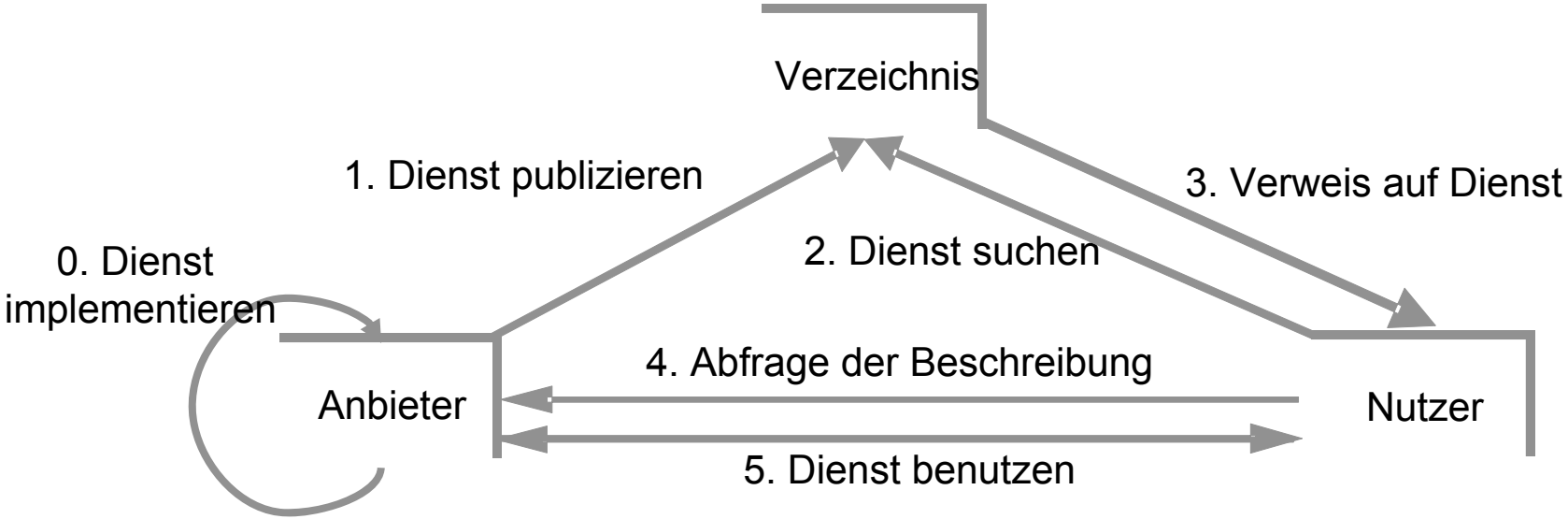
- Webseite generieren
- Kaufempfehlungen generieren
- „Käufer, die diesen Artikel gekauft haben, haben auch folgende Artikel gekauft…“
- Angebote anderer Anbieter

Vorteile aus Sicht von Amazon

- komplexe Anwendung aus relativ einfachen Web Servicen aufbauen
- aus komplexer Anwendung wiederum einfachen Web Service machen
- verteilte, skalierbare, robuste Architektur
- bestimmte Gruppe für Entwicklung und Betrieb eines Web Services verantwortlich

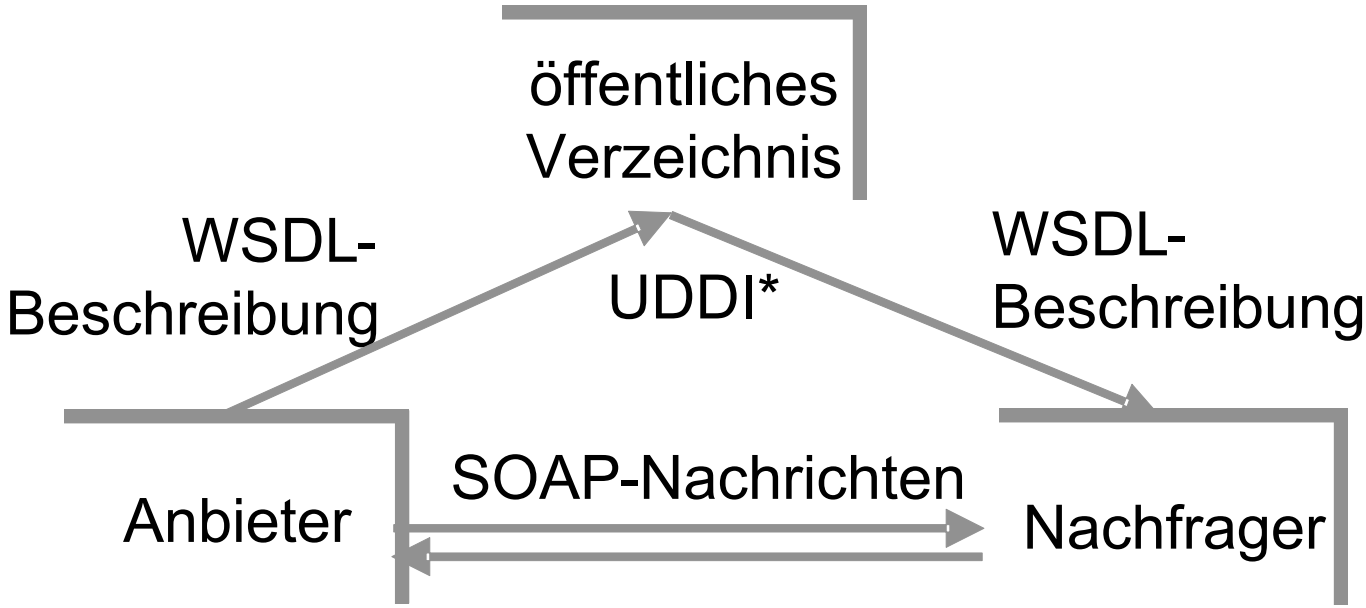


Dienstorientierte Architektur



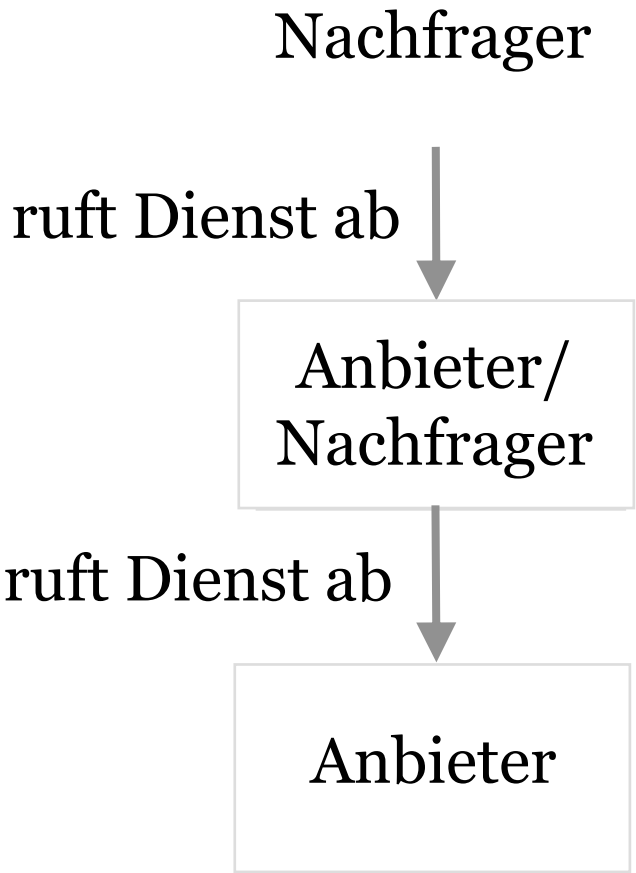
- publizieren (publish): Beschreibung eines Dienstes in einem Verzeichnis (registry) veröffentlichen.
- suchen (find): Beschreibung eines Dienstes suchen, entweder dynamisch oder zur Entwicklungszeit
- abrufen (bind): Beschreibung des Dienstes verwenden, um Dienst abzurufen, entweder dynamisch oder zur Entwicklungszeit

Standards



- SOAP und WSDL allgemein akzeptiert
- UDDI: Standard zur Beschreibung von Web-Service-Verzeichnissen
- UDDI umstritten und wenig genutzt

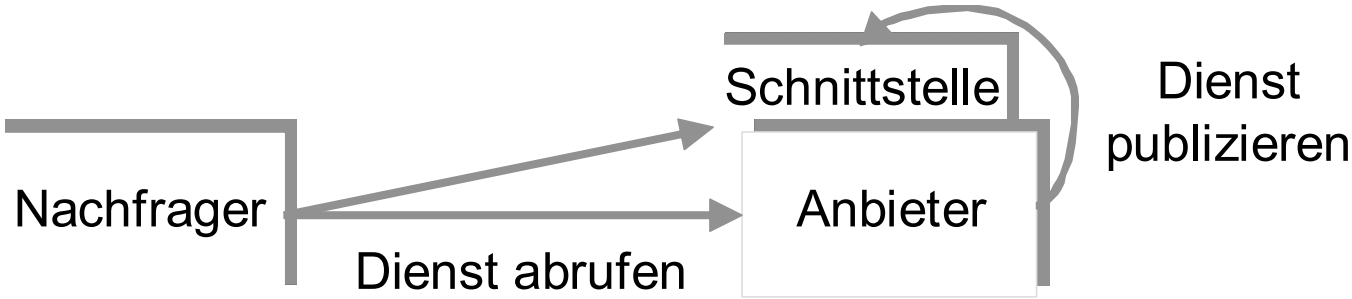
Nachfrager vs. Anbieter von Diensten



- Dienst-Anbieter (service provider) bietet Dienste an.
- Dienst-Nachfrager (service requestor) nutzt Dienste anderer Anbieter.
- Anbieter von Diensten kann gleichzeitig Dienste anderer nutzen (und Nachfrager) sein.

⇨ Diese Begriffe sind relativ.

Öffentliches Verzeichnis



- Öffentliches Verzeichnis nicht zwingend notwendig:
  → Auch ohne öffentliches Verzeichnis kann Dienst gefunden werden.
- → Schnittstelle kann z.B. auf Webseite des Anbieters veröffentlicht werden.
- öffentliche Verzeichnisse heute wenig genutzt

SOA: Vor- und Nachteile

Freie Universität Berlin

ohne dass Nutzer des Web Services es bemerkt, kann

+ neue Version freigeschaltet werden
+ bei Ausfall ein Web Service durch einen anderen Web Service mit gleicher Schnittstelle ersetzt werden
+ Lastverteilung durchgeführt werden

allerdings

- Vertrauen nötig, dass Web-Service-Anbieter WSDL-Beschreibung im Sinne des Nutzers realisiert

RPC vs. Messaging

Wie SOAP-Nachrichten übertragen?

Freie Universität Berlin

über HTTP

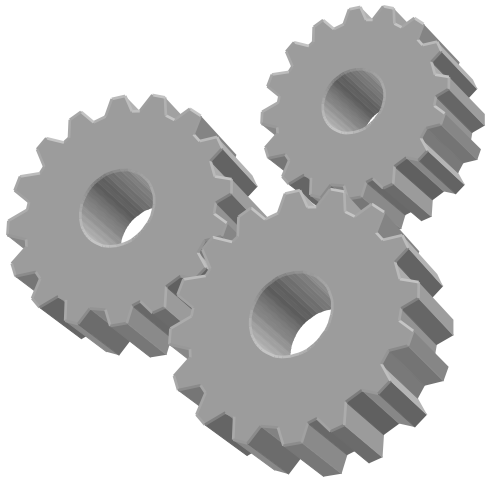
- heute üblich
- Request-Response-Verhalten von HTTP unterstützt RPCs.
- verbindungsorientiert
- Übertragung: Sender und Empfänger müssen präsent sein.
- typischerweise synchron

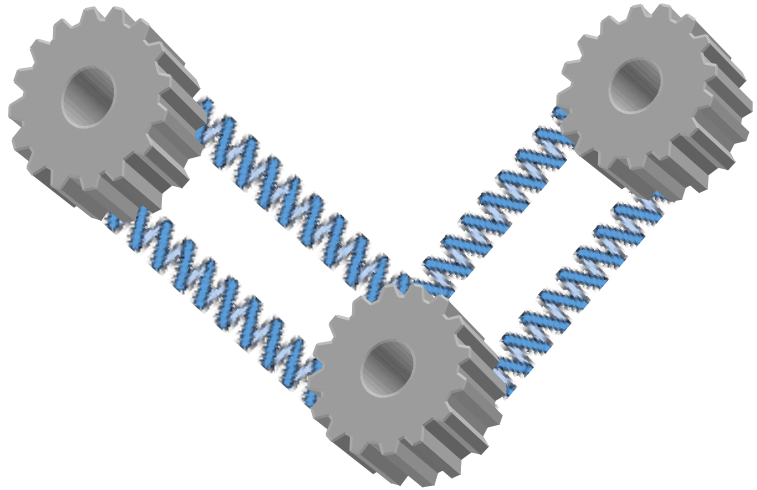
über SMTP

- heute eher selten
- realisiert Messaging
- persistente Kommunikation
- Übertragung: Weder Sender noch Empfänger muss präsent sein.
- typischerweise asynchron
- erlaubt Lastverteilung und Priorisierung

⇨ weitreichende Designentscheidung!

Weitreichende Designentscheidung





enge Kopplung

- verbindungs-orientierte, synchrone Kommunikation
- z.B. SOAP/HTTP

lose Kopplung

- gepufferte, asynchrone Kommunikation
- z.B. SOAP/SMTP
- robuster, aber auch komplexer zu entwerfen

RPC mit SOAP

Freie Universität Berlin

- Nachrichten konform:
  - zu einer vordefinierten Beschreibung des Funktionsaufrufes
  - zu der zu erwartenden Rückantwort

- Arten der Kommunikation:
  - Anfrage (Request)
  - Antwort (Response)
  - Fehlerfall (Fault)

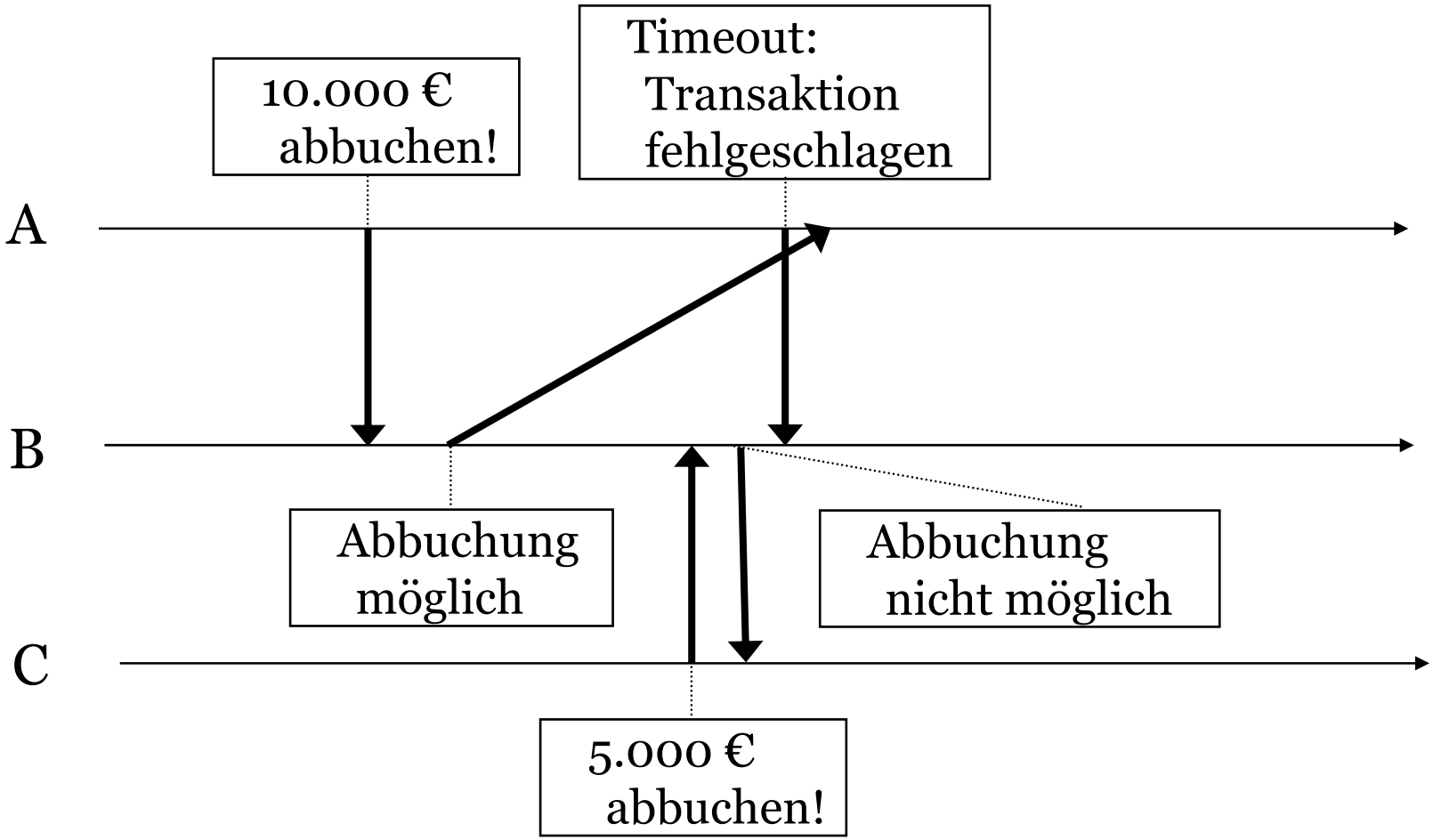
Komplexität loser Kopplung

Freie Universität Berlin

asynchroner Nachrichtenaustausch

- auch nach Timeout Antwort noch möglich
- Was tun mit der Antwort?
- häufig muss Absender informiert werden, dass Antwort nicht verarbeitet werden konnte
- Was tun, wenn diese Warnung nicht rechtzeitig beim Absender ankommt?
- Absender hat vielleicht im falschen Glauben, dass seine Antwort verarbeitet wurde, weitergearbeitet.
- Dieser Vorgang muss dann evtl. rückgängig gemacht werden.

Beispiel



⇨ Komplexität ist Preis für die gewonnene Robustheit

Messaging

Freie Universität Berlin

- Anwendungen interagieren durch Austausch von Nachrichten miteinander:
- Kommunikation in Anwendung sichtbar: senden und empfangen

⇨ Unterschied zu RPCs

- verschiedene Formen des Messaging:

1. Kommunikationsstruktur
2. Interaktionsmuster
3. flüchtig vs. persistent
4. synchron vs. asynchron
5. Quality of Service

1. Kommunikationsstruktur

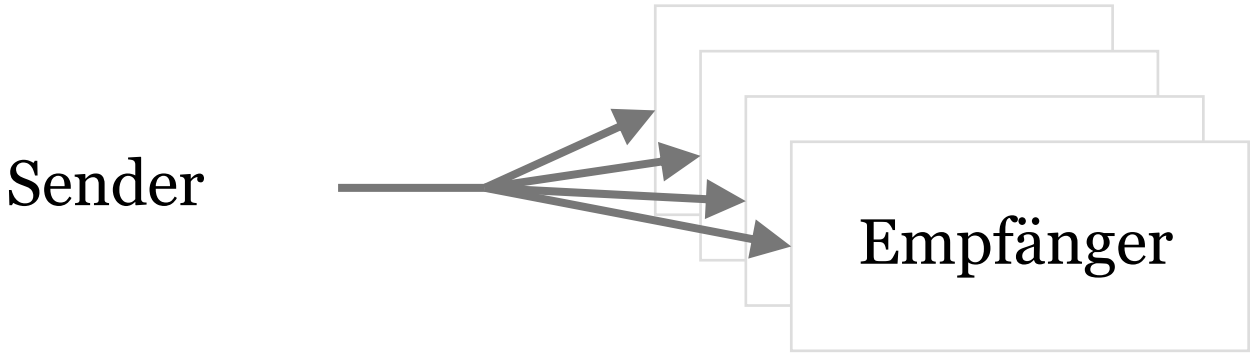
Freie Universität Berlin

- Anzahl und Organisation der Kommunikationspartner
- wichtigste Kommunikationsstrukturen:
  - One-to-One-Kommunikation
  - One-to-Many-Kommunikation

One-to-One-Kommunikation

- Sender sendet Nachricht an bestimmten Empfänger.
- Beispiel: Kunde sendet Bestellung per E-Mail an Firma

One-to-Many-Kommunikation



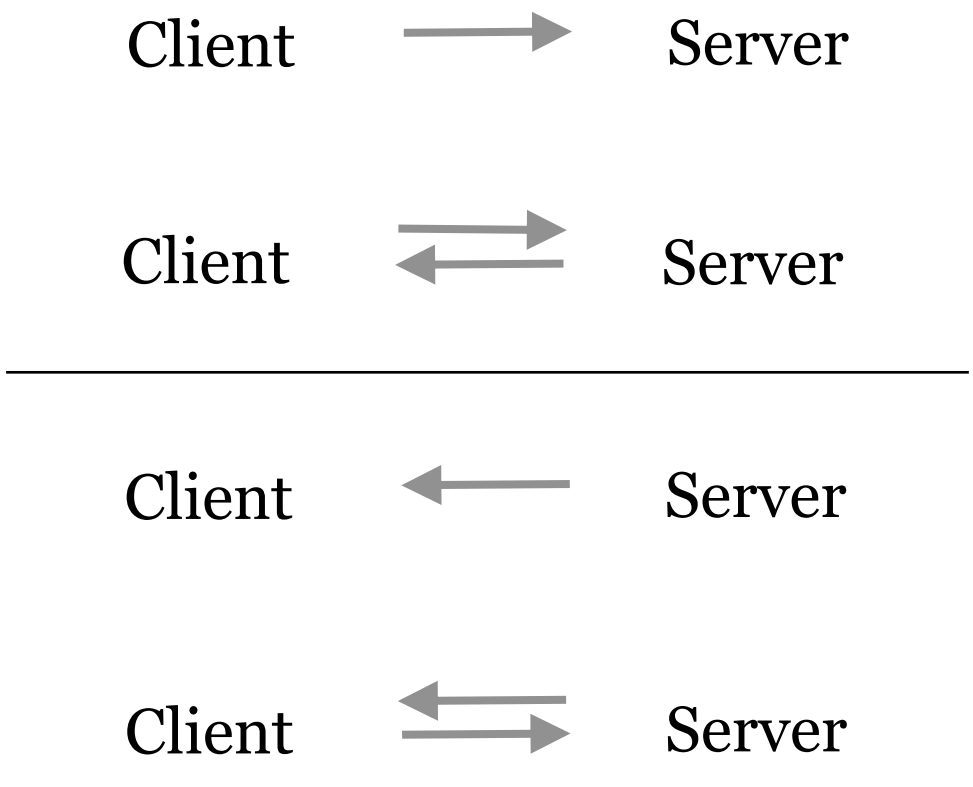
- Sender sendet identische Kopie gleichzeitig an mehrere Empfänger.
- Sender (publisher) veröffentlicht Nachricht zu bestimmten Thema (topic), zu dem sich die Empfänger (subscriber) angemeldet haben.

⇨ auch publish-subscribe oder topic-based messaging genannt

- Beispiel: Mailing-Liste

2. Interaktionsmuster

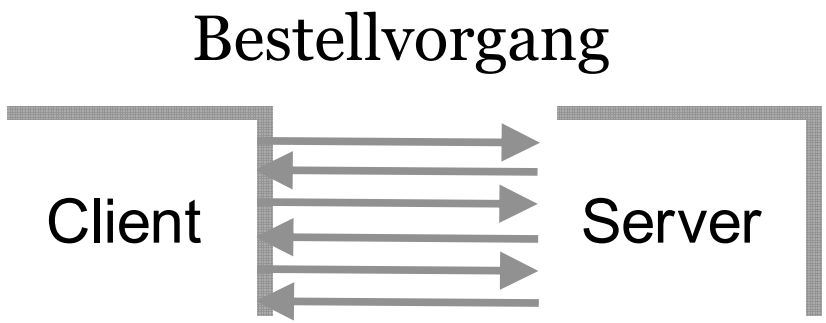
Freie Universität Berlin



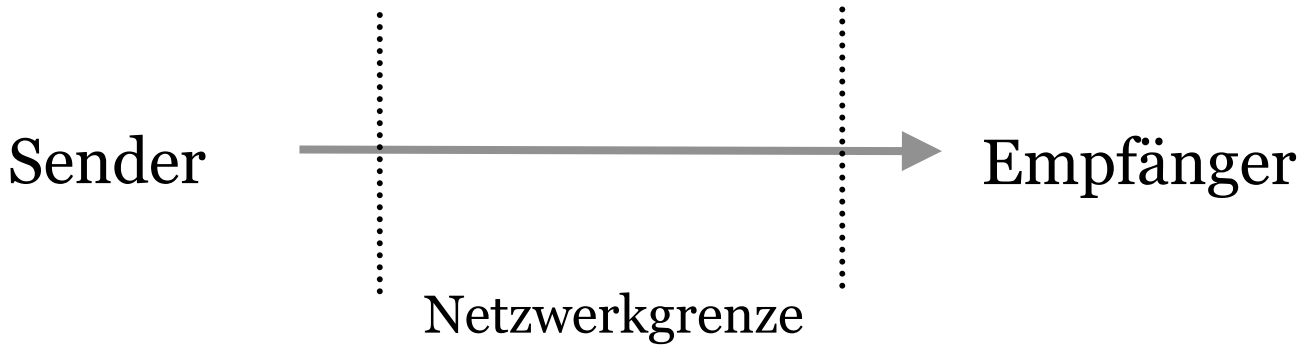
Beachte: „Antwort“ bezieht sich auf jeweilige Anwendung, nicht auf das Netzwerk.

Komplexe Interaktionsmuster

- Bestellanfrage mit spätestem Liefertermin
- Angebot mit zugesichertem Liefertermin
- Bestellung
- Bestätigung des Eingangs der Bestellung
- Bestätigung der Lieferung
- Rechnung



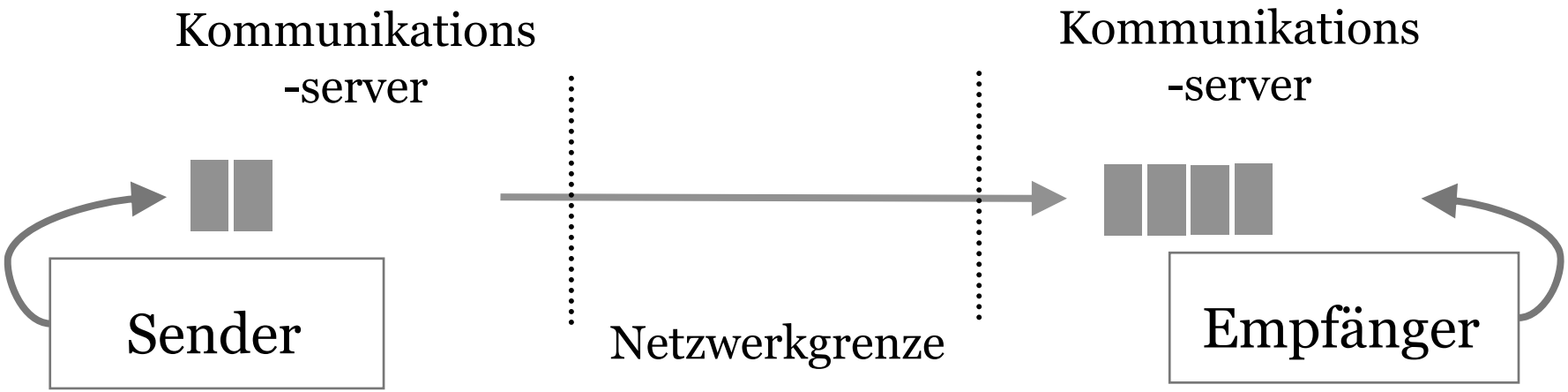
3. Flüchtig vs. persistent



flüchtige Kommunikation

- Sender und Empfänger kommunizieren direkt ohne Puffer miteinander.
- Sender und Empfänger müssen während der gesamten Übertragung präsent sein
- engl. transient
- Beispiele: HTTP, UDP

Flüchtig vs. persistent



persistente Kommunikation

- Nachricht solange gespeichert, bis sie tatsächlich zugestellt wurde.
- Weder Sender noch Empfänger müssen während Übertragung präsent sein.
- Beispiele: E-Mail, MQSeries (IBM), JMS

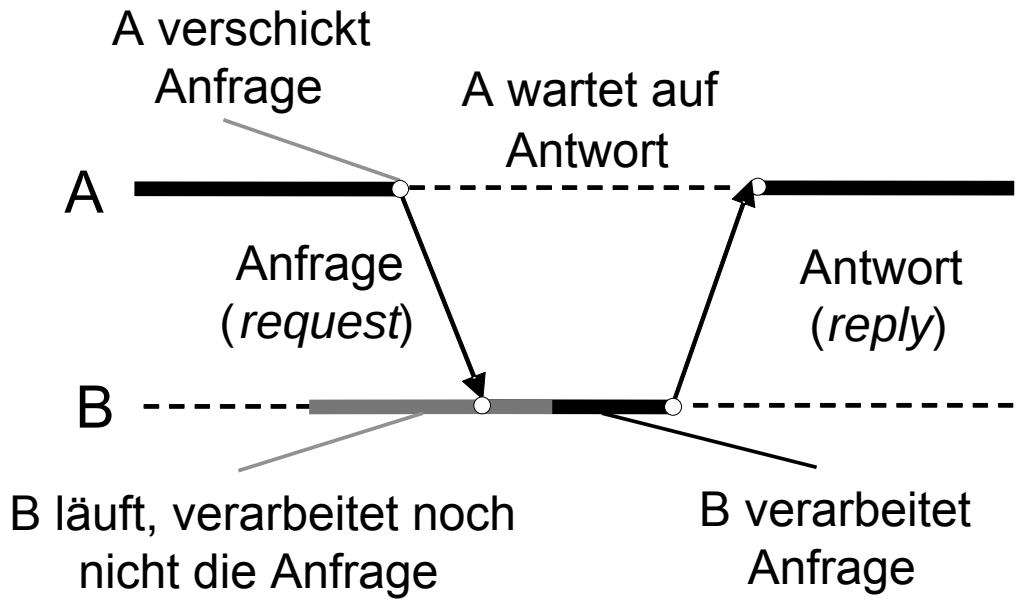
4. Synchron vs. Asynchron

Freie Universität Berlin

- Asynchron
  - Senden und Empfangen zeitlich versetzt
  - ohne Blockieren des Prozesses (ohne Warten auf die Antwort)
- Synchron
  - Senden/Empfangen synchronisieren → warten (blockieren) bis die Kommunikation abgeschlossen ist.

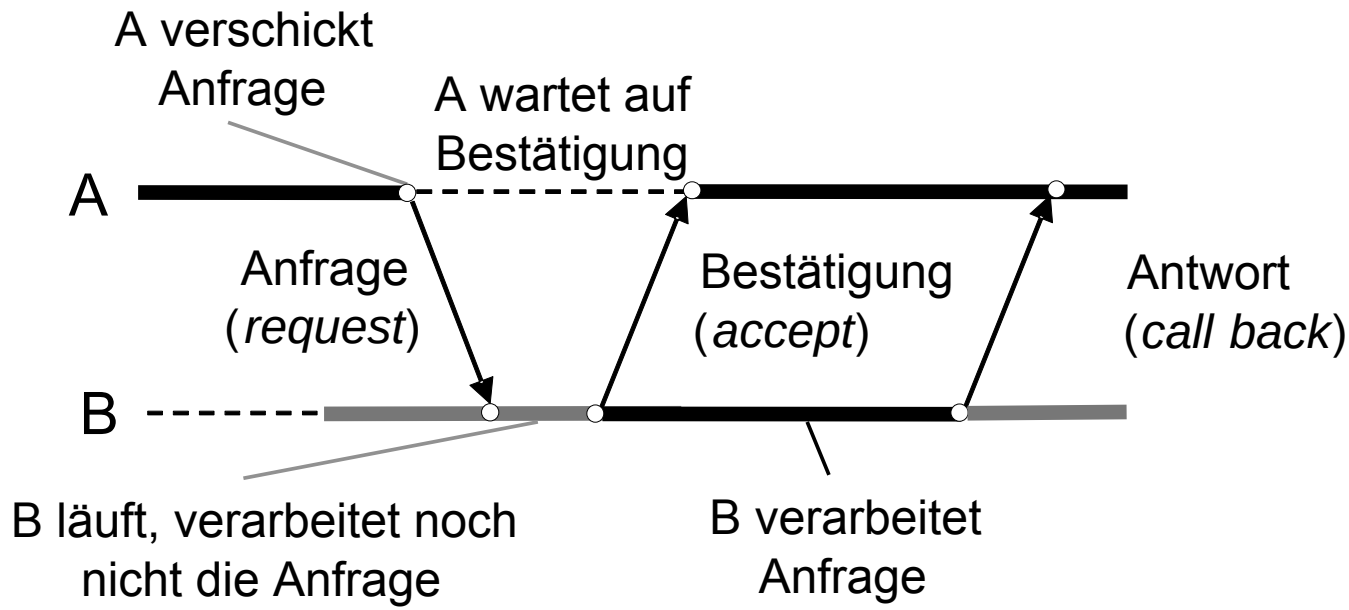
- bei persistenter Kommunikation → Messaging normalerweise asynchron
- bei flüchtiger Kommunikation → Messaging kann sowohl synchron als auch asynchron sein.

Beispiel 1



- flüchtige, synchrone Kommunikation
- auch antwortorientiert (response-based) genannt
- implementiert synchrone RPCs
- jedoch ≠ RPC: Kommunikation für Anwendung sichtbar

Beispiel 2



- flüchtige, asynchrone Kommunikation
- auch bestätigungsorientiert (delivery-based) genannt
- implementiert asynchrone RPCs
- jedoch ≠ RPC: Kommunikation für Anwendung sichtbar

RPC oder Messaging?

Freie Universität Berlin

RPC ⇨eng gekoppelte, starre Systeme

+ einfach, abstrahiert von Kommunikation
- nur Eins-zu-Eins-Kommunikation
- Client und Server müssen präsent sein.
- skaliert weniger gut

Messaging ⇨lose gekoppelte, robuste Systeme

- abstrahiert nicht von Kommunikation
+ erlaubt auch One-to-Many-Kommunikation
+ Weder Sender noch Empfänger müssen präsent sein.
+ erlaubt Priorisierung und Lastverteilung
+ skaliert sehr gut

Block Web Services

Freie Universität Berlin

heutige Vorlesung

- ☑ Was sind Web Services?
- ☑ Basistechnologien (SOAP, WSDL, UDDI)
- ☑ Enterprise Application Integration
- ☑ Dienstorientierte Architektur (SOA)
- ☑ RPC vs. Messaging

nächste Vorlesung (20.06)

- SOAP im Detail

Nächste Woche
→Keine Vorlesung
→ Keine Übung