Freie Universität Berlin

Web Services

© Klaus Schild, 2006

Block Web Services

Freie Universität Berlin

heutige Vorlesung

- Was sind Web Services?
- Was ist SOAP?
- Was ist WSDL?
- Anwendungen von Web Services
- RPC vs. Messaging

14.6.

- SOAP im Detail

28.6.

- WSDL im Detail

5.7.

- Web Services in der Praxis

© Klaus Schild, 2006

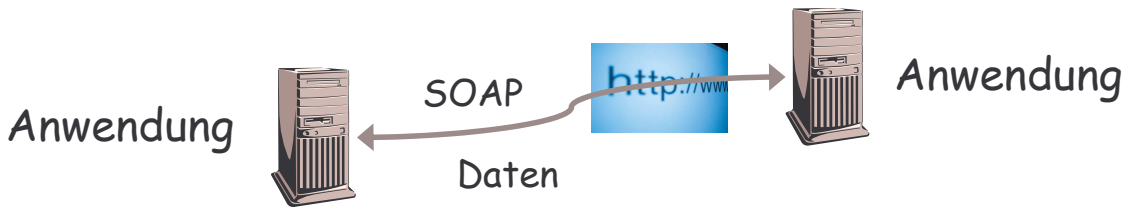
Was sind Web Services?

© Klaus Schild, 2006

traditionelle Web-Anwendung

➔ Mensch-Maschine-Kommunikation

Web Service



➔ Maschine-Maschine-Kommunikation

© Klaus Schild, 2006

Beispiel: Google™ ohne Browser

Freie Universität Berlin

»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/

© Klaus Schild, 2006

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

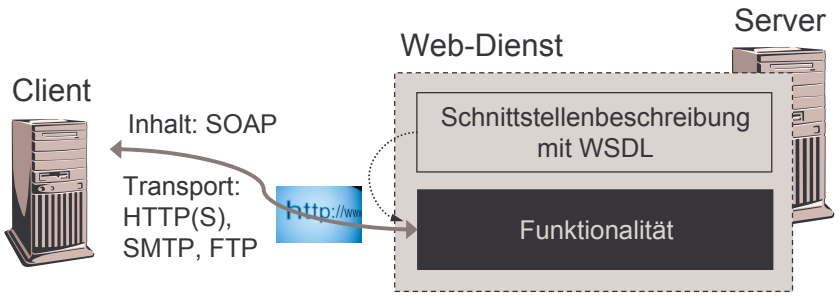
search-for: "XSLT 2.0 Recommendation"

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

- automatisch neue Trends im WWW zu identifizieren

© Klaus Schild, 2006

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.

© Klaus Schild, 2006

Eigenschaften von Web Services

- 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)

© Klaus Schild, 2006

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

© Klaus Schild, 2006

Freie Universität Berlin

Was ist SOAP?

© Klaus Schild, 2006

Nachrichtenformat SOAP

Freie Universität Berlin

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

© Klaus Schild, 2006

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">3289754870548097</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: SOAP-Anfrage kann auch als URL kodiert werden (REST)

© Klaus Schild, 2006

SOAP-Anfrage als URL kodiert

Freie Universität Berlin

GET /search/beta2/doGoogleSearch?q=Beginning+XML HTTP/1.1
Host: api.google.com
Accept: application/soap+xml

- ruft doGoogleSearch(q="Beginning XML") auf
- URL muss gesamte SOAP-Nachricht kodieren!
- Antwort: SOAP
- Amazon bietet solche REST-Schnittstelle an, Google jedoch nicht

© Klaus Schild, 2006

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

© Klaus Schild, 2006

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

© Klaus Schild, 2006

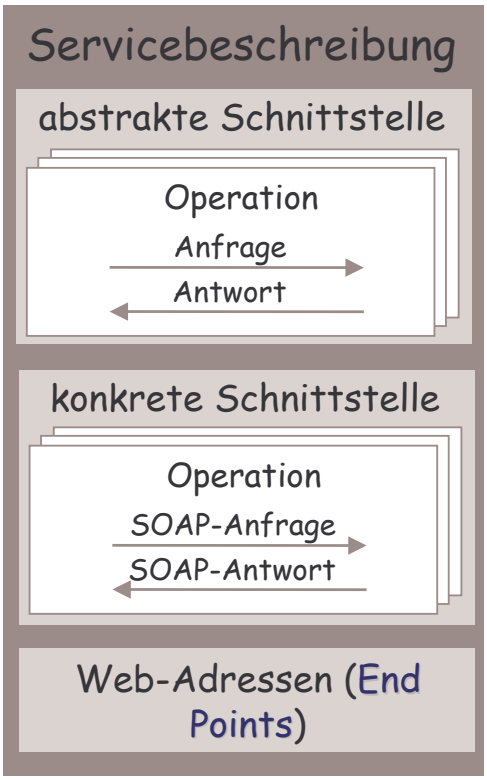
Freie Universität Berlin

Was ist WSDL?

© Klaus Schild, 2006

Web Services Description Language

Freie Universität Berlin



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

abstrakte Schnittstelle (port type)

- Schnittstelle unabhängig von Nachrichtenformaten
- Beschreibung mit XML-Schema

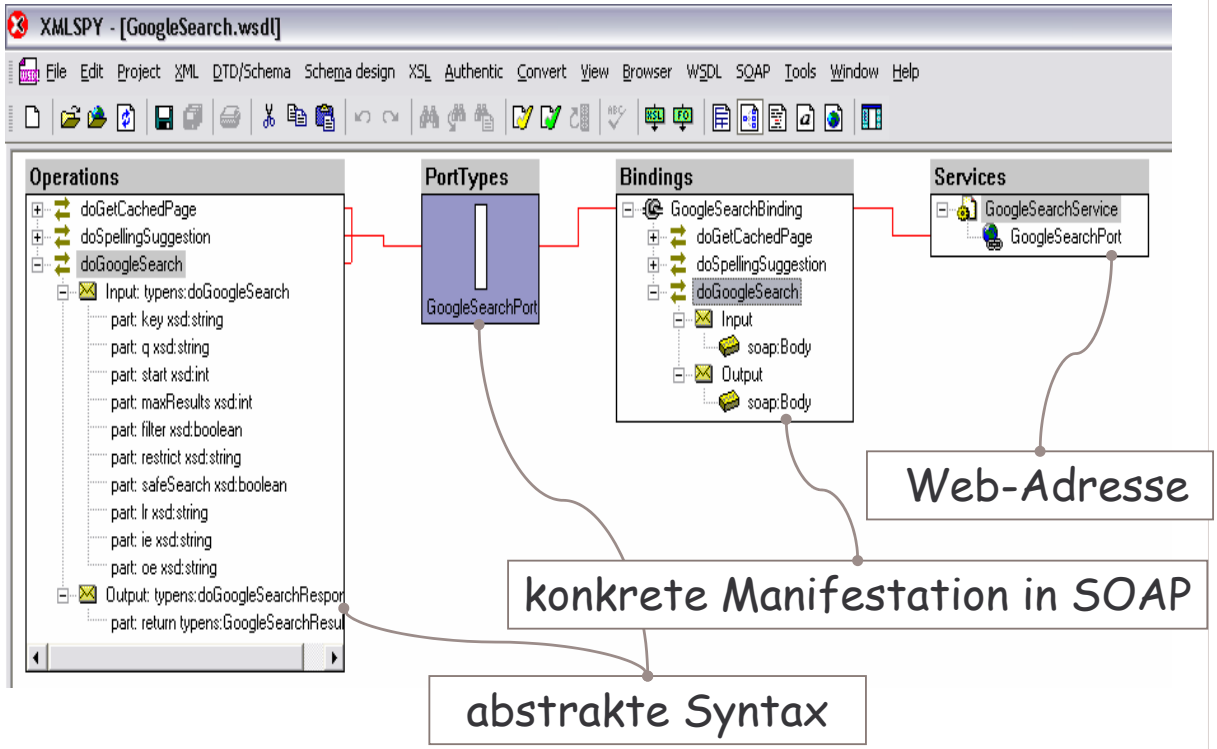
konkrete Schnittstelle (binding)

- Abbildung der abstrakten Schnittstelle auf unterstützte Nachrichtenformate

© Klaus Schild, 2006

WSDL-Beschreibung von Google

Freie Universität Berlin



© Klaus Schild, 2006

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.

© Klaus Schild, 2006

Freie Universität Berlin

Anwendungen von Web Services

© Klaus Schild, 2006

Anwendungen von Web Services

Freie Universität Berlin

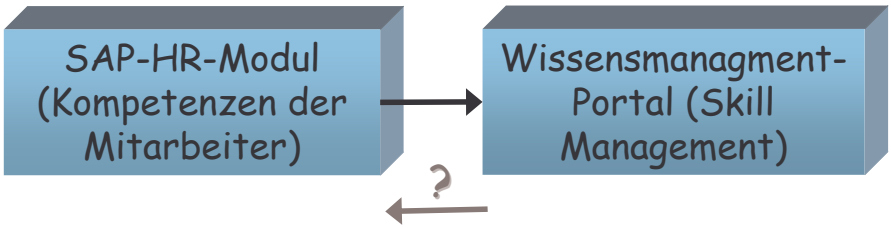
1. Enterprise Application Integration
2. diensteorientierte Architektur
3. Microsofts Indigo

© Klaus Schild, 2006

1. Enterprise Application Integration

Freie Universität Berlin

- technologisch: inkompatible IT-Systeme miteinander verbinden
- inkompatibel kann bedeuten:
  - unterschiedliche Betriebssysteme
  - unterschiedliche Programmiersprachen
  - unterschiedliche Kommunikationsprotokolle
- organisatorisch: Geschäftsprozesse optimieren
- Beispiel:



© Klaus Schild, 2006

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

© Klaus Schild, 2006

Bedeutung der Systemintegration

Freie Universität Berlin

Systemintegration bindet

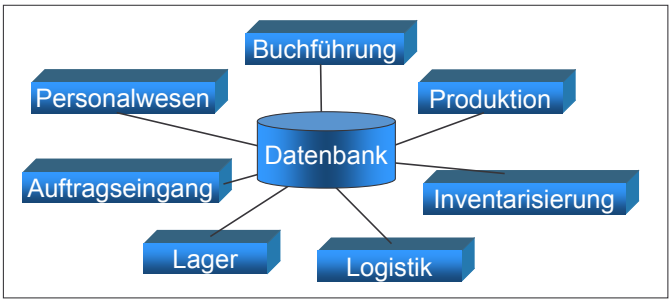
- 35% der IT-Personal-Ressourcen eines Unternehmens (Forrester Research, 2002)
- 65% der Arbeitszeit eines Programmierers (Gartner)

© Klaus Schild, 2006

Enterprise Resource Planning

Freie Universität Berlin

E-Commerce ?



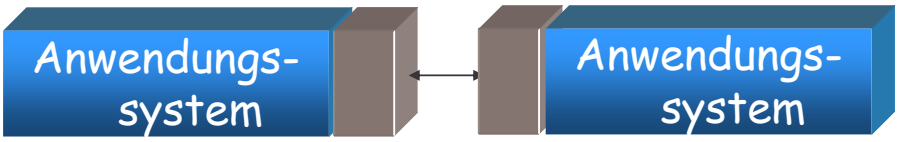
- Beispiel: SAP/R3
- großer Fortschritt für Datenintegration
- sehr teuer:
  > 53.000 $ (TCO) pro Nutzer für ersten 2 Jahre (Meta Group, 2002)
- viele Datensilos durch ein großes Datensilo ersetzt:
- schwierige Integration externer Systeme

© Klaus Schild, 2006

Systemintegration mit Web Services

Freie Universität Berlin

- Anwendungssysteme durch standardisierte Schnittstelle erweitern

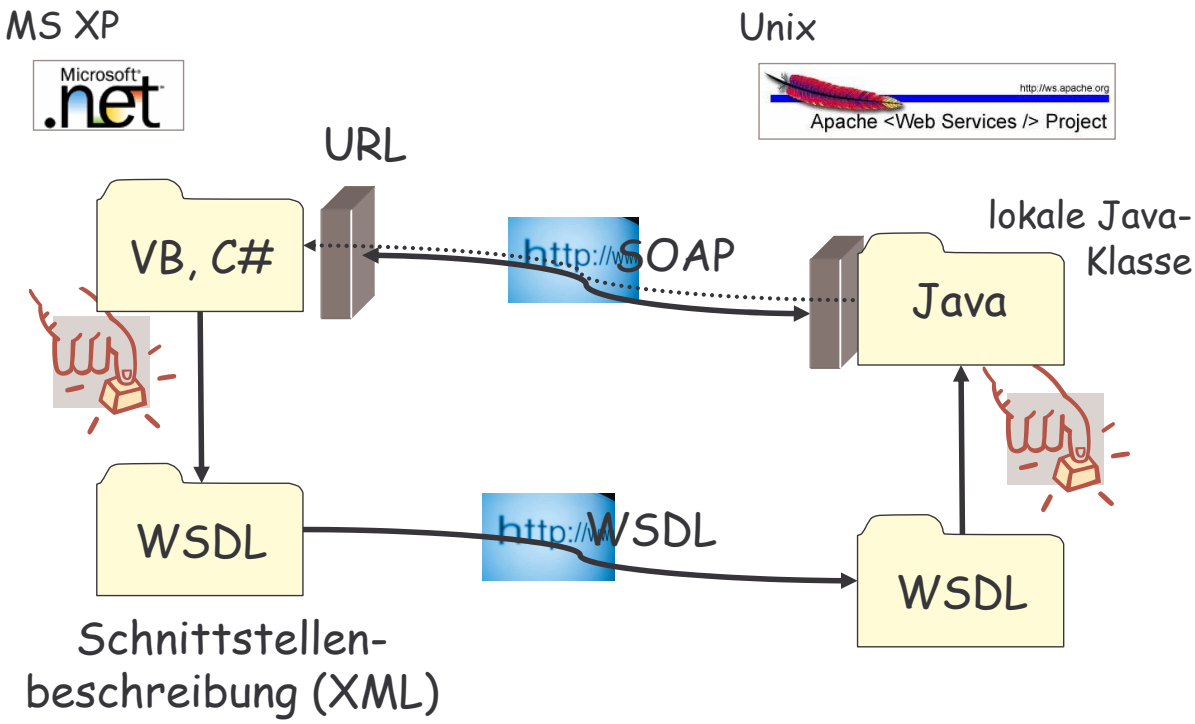


- Schnittstelle muss allgemein akzeptiert sein
- bei Web Services ist dies der Fall
  - ✓ SOAP
  - ✓ WSDL
  - ✓ Internet-Protokolle
- bei n Systemen:
  statt n^2 Schnittstellen nur n Erweiterungen!

© Klaus Schild, 2006

Systemintegration per Knopfdruck!

Freie Universität Berlin



© Klaus Schild, 2006

2. Diensteorientierte Architektur

Freie Universität Berlin

- 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

© Klaus Schild, 2006

Beispiel Amazon

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

© Klaus Schild, 2006

Vorteile aus Sicht von Amazon

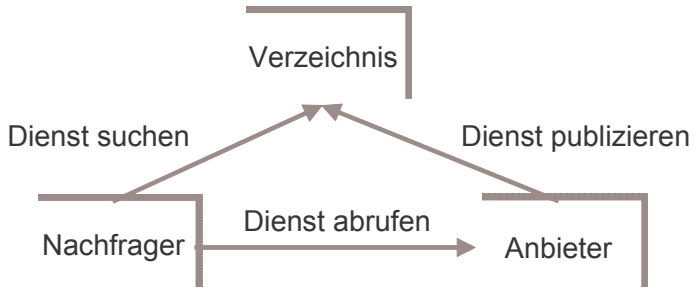
Freie Universität Berlin

- 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

© Klaus Schild, 2006

Dienstorientierte Architektur

Freie Universität Berlin

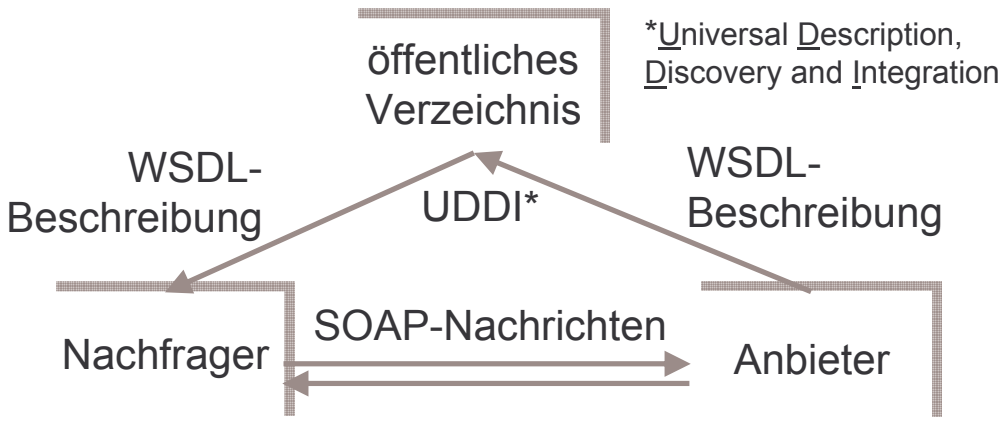


- publizieren (publish): Beschreibung eines Dienstes in einem Verzeichnis (registry) veröffentlichen.
- suchen (find): Beschreibung eines Dienstes suchen, entweder dynamisch oder zur Entwicklungszeit
- abrufen (bind): Beschreibung des Dienstes verwenden, um Dienst abzurufen, entweder dynamisch oder zur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

© Klaus Schild, 2006

Standards

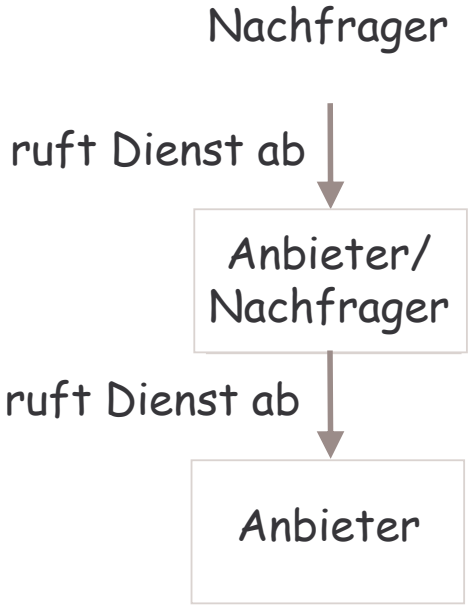
Freie Universität Berlin



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

© Klaus Schild, 2006

Nachfrager vs. Anbieter von Diensten

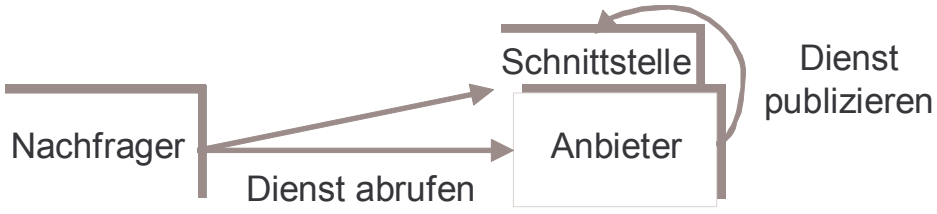


- Dienst-Anbieter (service provider) bietet Dienste an.
- Dienst-Nachfrager (service requestor) nutzt Dienste anderer Anbieter.
- Anbieter von Diensten kann gleichzeitig Dienste anderer nutzen (und Nachfrager) sein.

⇨ Diese Begriffe sind relativ.

© Klaus Schild, 2006

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

© Klaus Schild, 2006

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

© Klaus Schild, 2006

3. Indigo von Microsoft

Freie Universität Berlin

- Kommunikations-Infrastruktur von Windows Vista™ (vormals Longhorn)
- Nachfolger von COM
- basiert auf SOAP und (vereinfachter Version) von WSDL

© Klaus Schild, 2006

RPC vs. Messaging

© Klaus Schild, 2006

Wie SOAP-Nachrichten übertragen?

über HTTP

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

über SMTP

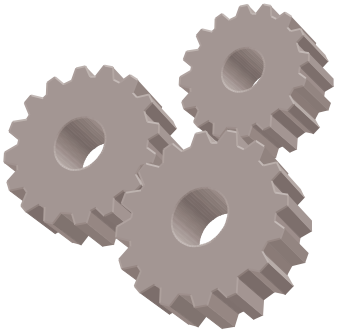
- heute eher selten
- realisiert Messaging
- persistente Kommunikation
- Übertragung: Weder Sender noch Empfänger muss präsent sein.
- typischerweise asynchron
- erlaubt Lastverteilung und Priorisierung

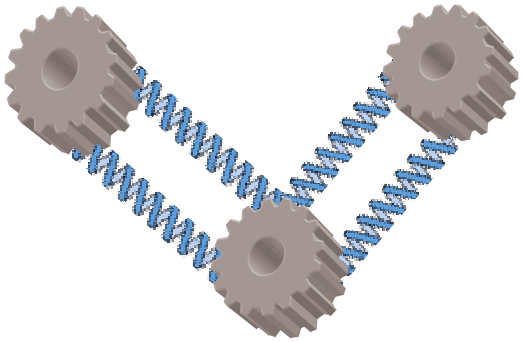
⇨ weitreichende Designentscheidung!

© Klaus Schild, 2006

Weitreichende Designentscheidung

Freie Universität Berlin





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

© Klaus Schild, 2006

Komplexität loser Kopplung

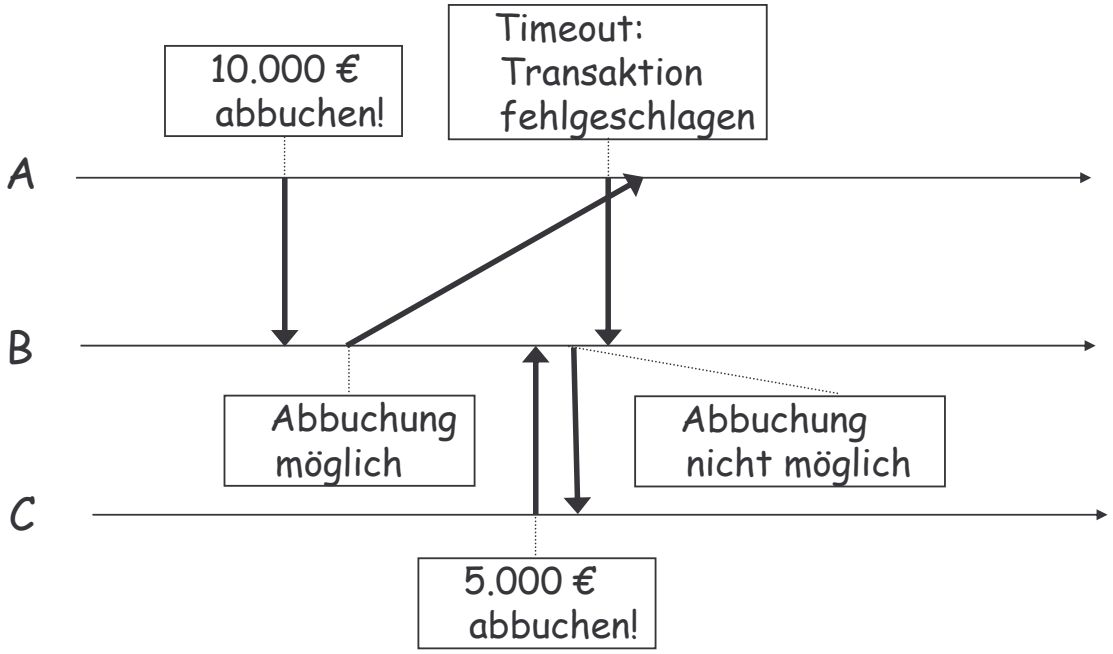
Freie Universität Berlin

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.

© Klaus Schild, 2006

Beispiel



⇨ Komplexität ist Preis für die gewonnene Robustheit

© Klaus Schild, 2006

Messaging

- 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

© Klaus Schild, 2006

1. Kommunikationsstruktur

Freie Universität Berlin

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

© Klaus Schild, 2006

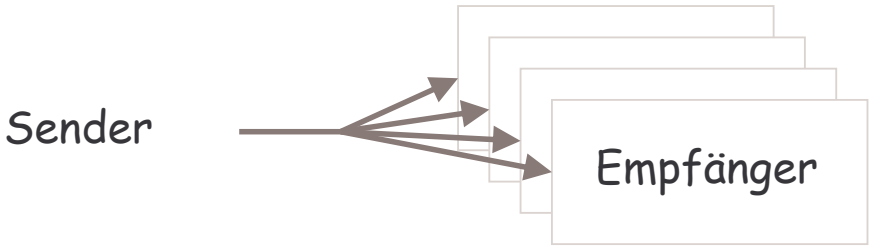
Eins-zu-Eins-Kommunikation

Freie Universität Berlin

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

© Klaus Schild, 2006

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

© Klaus Schild, 2006

2. Interaktionsmuster

Client	→	Server	Einweg (one way, fire and forget)
Client	⇄	Server	Anfrage-Antwort (request-response)
Client	←	Server	Benachrichtigung (notification)
Client	⇄	Server	Benachrichtigung-Antwort (notification-response)

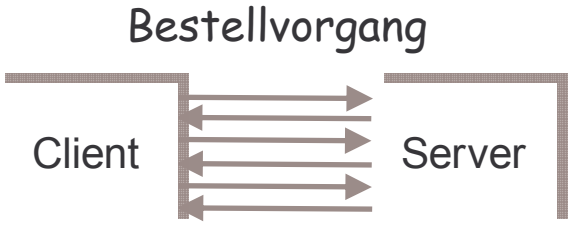
Beachte: „Antwort" bezieht sich auf jeweilige Anwendung, nicht auf das Netzwerk.

© Klaus Schild, 2006

Komplexe Interaktionsmuster

Freie Universität Berlin

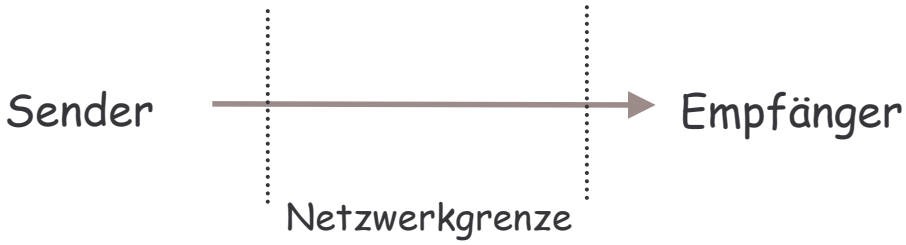
- → Bestellanfrage mit spätestem Liefertermin
- ← Angebot mit zugesichertem Liefertermin
- → Bestellung
- ← Bestätigung des Eingangs der Bestellung
- → Bestätigung der Lieferung
- ← Rechnung



© Klaus Schild, 2006

3. Flüchtig vs. persistent

Freie Universität Berlin

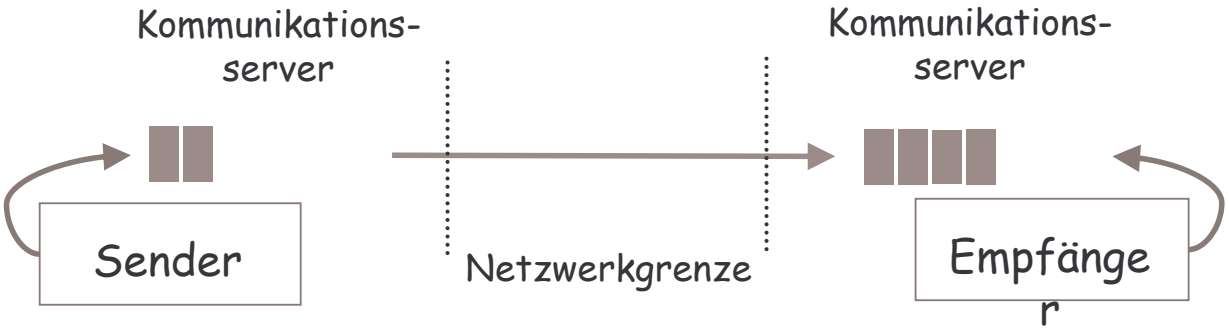


flüchtige Kommunikation

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

© Klaus Schild, 2006

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

© Klaus Schild, 2006

4. Synchron vs. Asynchron

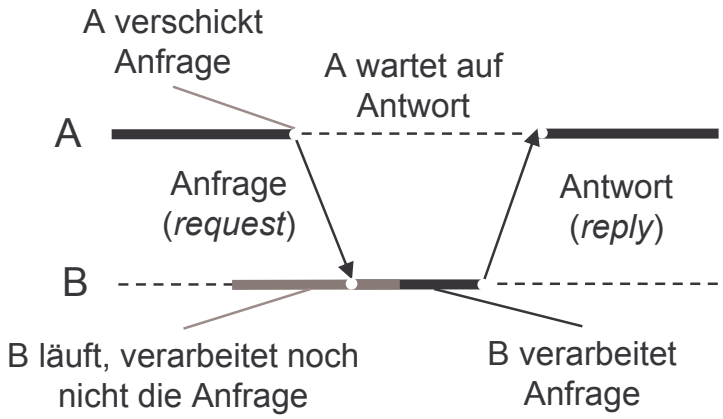
- bei persistenter Kommunikation:

  Messaging normalerweise asynchron
- bei flüchtiger Kommunikation:

  Messaging kann sowohl synchron als auch asynchron sein.

© Klaus Schild, 2006

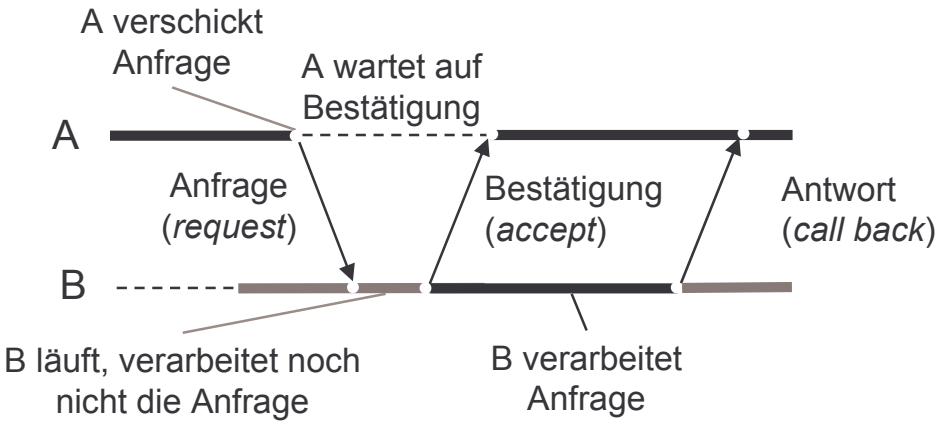
Beispiel 1



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

© Klaus Schild, 2006

Beispiel 2



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

© Klaus Schild, 2006

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

© Klaus Schild, 2006

Block Web Services

Freie Universität Berlin

heutige Vorlesung

- ☑ Was sind Web Services?
- ☑ Was ist SOAP?
- ☑ Was ist WSDL?
- ☑ Anwendungen
- ☑ RPC vs. Messaging

Übung

- diese Woche keine Übungen

nächste Vorlesung

- SOAP im Detail

© Klaus Schild, 2006