Hochschule Darmstadt
Fachbereich Informatik
Masterstudiengang

Daten- und Systemintegration

SOA und Web Services - v1.0

Prof. Dr. Frank Bühler
Prof. Dr. Günter Turetschek

Agenda

5.1 Definitionen, Ziele, Charakterisierung
5.2 Geschäftsarchitektur
5.3 IT-Anwendungslandschaft
5.4 SOA-Grundmodell und SOA-Architektur
5.5 Services und Servicemanagement
5.6 Ein neues Vorgehensmodell?

5.1 Definitionen, Ziele, Charakterisierung

Definitionen

(1) Unter einer SOA versteht man eine Systemarchitektur,die die Funktionalität verschiedener,zumeist inkompatibler Applikationen als wiederverwendbare und offen zugreifbare **Dienste** repräsentiert und dadurch eine plattform- und sprachunabhängige Nutzung und Wiederverwendung ermöglicht.

Dostal, W. u.a.: Service-orientierte Architekturen mit Web Services, Spektrum Verlag, Heidelberg 2005

(2) SOA ist ein Architekturmuster, das den Aufbau einer Anwendungs-landschaft aus einzelnen fachlichen Anwendungsbausteinen (-komponenten) beschreibt, die jeweils eine klar umrissene fachliche Aufgabe wahrnehmen. Die Anwendungsbausteine sind lose miteinander gekoppelt, indem sie einander ihre Funktionalitäten in Form von Services (als abstrakte fachliche Sicht auf den Anwendungsbaustein) anbieten.

Richter, Jan-Peter u.a.: Serviceorientierte Architektur in: Informatik Spektrum 2005

Ziele einer SOA:

- Flexibilisierung von Geschäftsprozessen
- Barrierefreiheit bei der Nutzung von Diensten
- Wiederverwendung von Anwendungsfunktionalität

damit wird erreicht die

- Senkung von IT-Kosten und Erschließung von Geschäftspotentialen

Fortschritt durch SOA?

Konsequente Verbindung von fachlicher Geschäftsebene mit der IT. SOA ist in erster Linie ein **fachlich** determiniertes Architekturkonzept

Ausgangspunkt einer SOA:

Eine am Geschäft eines Unternehmens orientierte, **fachliche** Beschreibung der Strukturen, Abläufe,…des **Systems Unternehmen.**

5.2 Geschäftsarchitektur

(Hess, A., Humm,B. und M.Voß: Regeln für serviceorientierte Architekturen hoher Qualität in: Informatik Spektrum 2006

Die Architektur besteht aus:

Geschäftsstrategie: z.B. Mit welchen Produkten und Dienstleistungen operiert ein Unternehmen langfristig auf welchen Märkten?

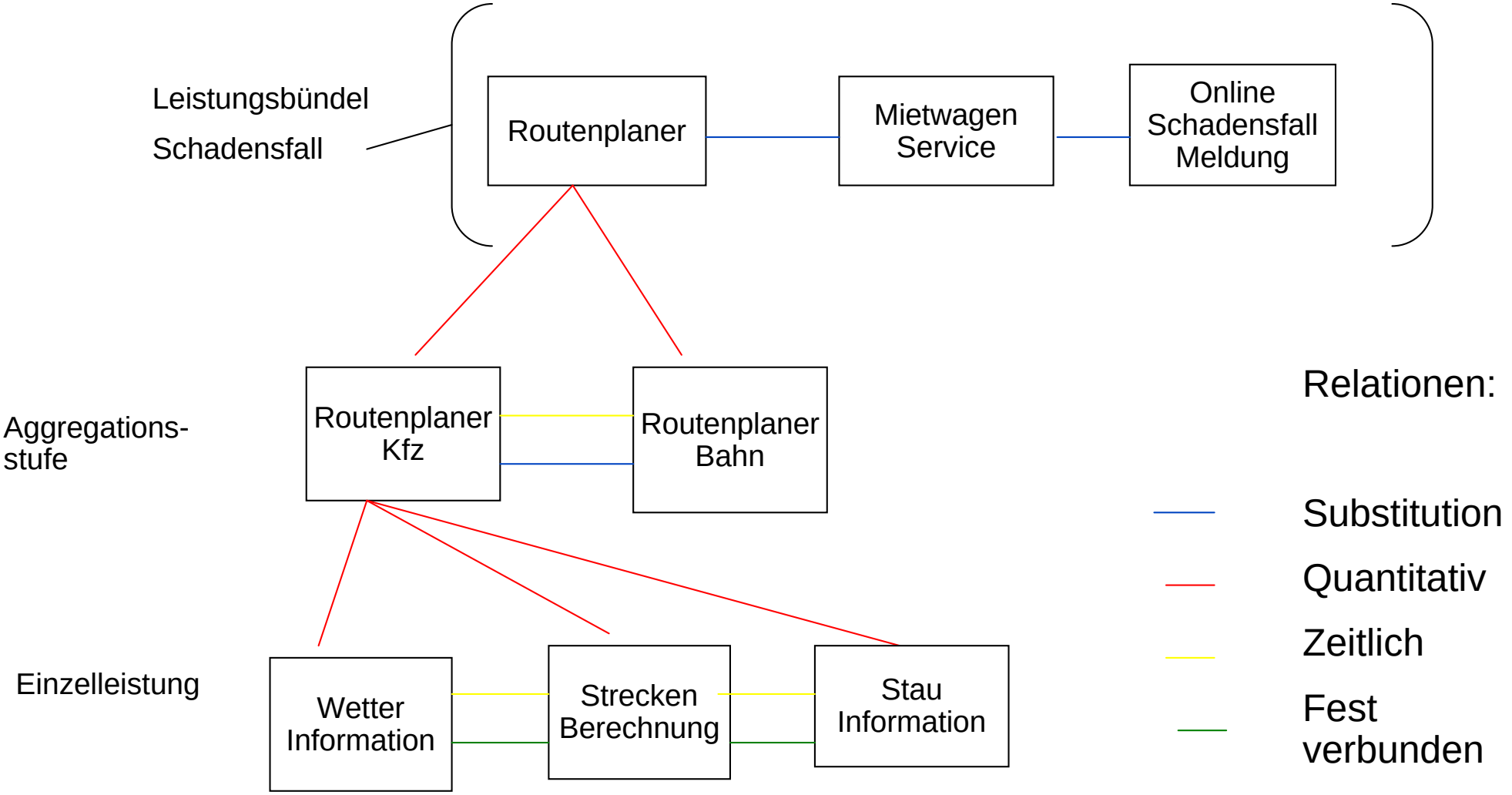
Geschäftsdomänen: z.B. Unternehmenssegmente wie Kundengruppen, Lieferanten, Produkttypen, Vertriebswege

Geschäftsprozessen

Geschäftsfunktionen: Arbeitsschritte (Aktivitäten in Geschäftsprozessen)

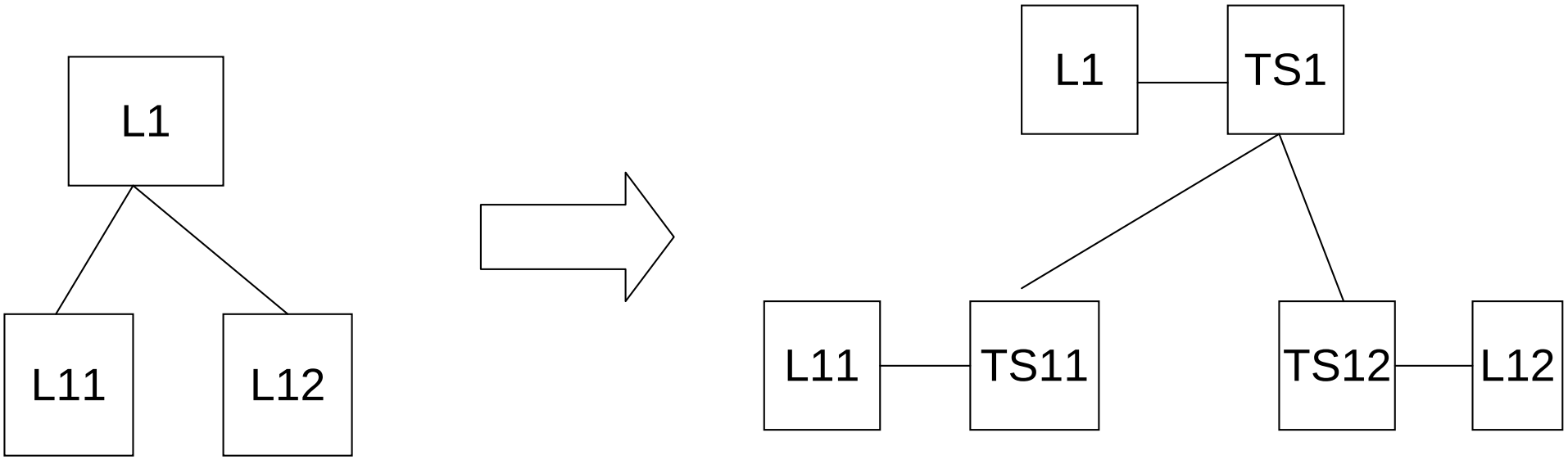
Geschäftsobjekten: z.B. Lieferschein, Bestellung

Beispiel Leistungsbündelung in der Kfz-Versicherung (1)



Beispiel Leistungsbündelung in der Kfz-Versicherung (2)

Servicemodellierung: Abbildung Leistungsbündelstruktur auf Architektur von Anwendungskomponenten



5.3 IT-Anwendungslandschaft

Architektur einer Anwendungslandschaft

(Hess, A. u.a.,…)

Begriffe

Anwendungsdomäne: Menge von Anwendungskomponenten, die gemäß einer Geschäftsdomäne gemeinsam betrachtet werden

Anwendungskomponente: Entwurfs- und Implementierungseinheit, die Schnittstellen importiert und exportiert und Implementierungsdetails verbirgt. Im Folgenden ist die Rede von Komponenten im Großen

Anwendungsservice, kurz **Service**: Schnittstelle einer Anwendungskomponente, definiert fachliche Funktionalität

Serviceoperationen: bilden die Funktionalität von Services ab.

Die Anwendungsarchitektur folgt der Geschäftsarchitektur

Geschäftsarchitektur

Anwendungsarchitektur

Servicekategorien

Anwendungsdomänen

1 2 … n

Geschäftsdomänen →

Interaktions Komponenten

Geschäftsprozess →

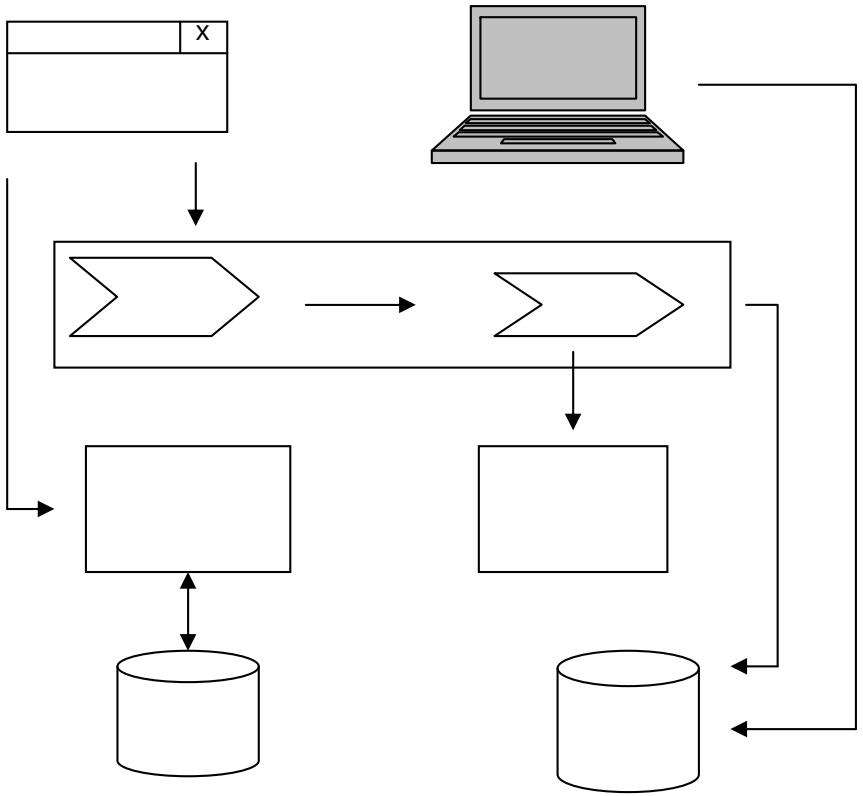
Prozess Komponenten

Geschäftsfunktion →

Funktions Komponenten

Geschäftsobjekt →

Bestands Komponenten



Bestandskomponenten:

Ihre Services bieten Datenpflegeoperationen, bieten lesende Sichten auf die Geschäftsobjekte aber auch elementare fachliche Logik

Funktionskomponenten:

Ihre Services bilden Geschäftsfunktionen ab, nutzen Services von Bestandskomponenten, die Integration von Funktionsservices erfolgt über Prozesskomponenten

Prozesskomponenten:

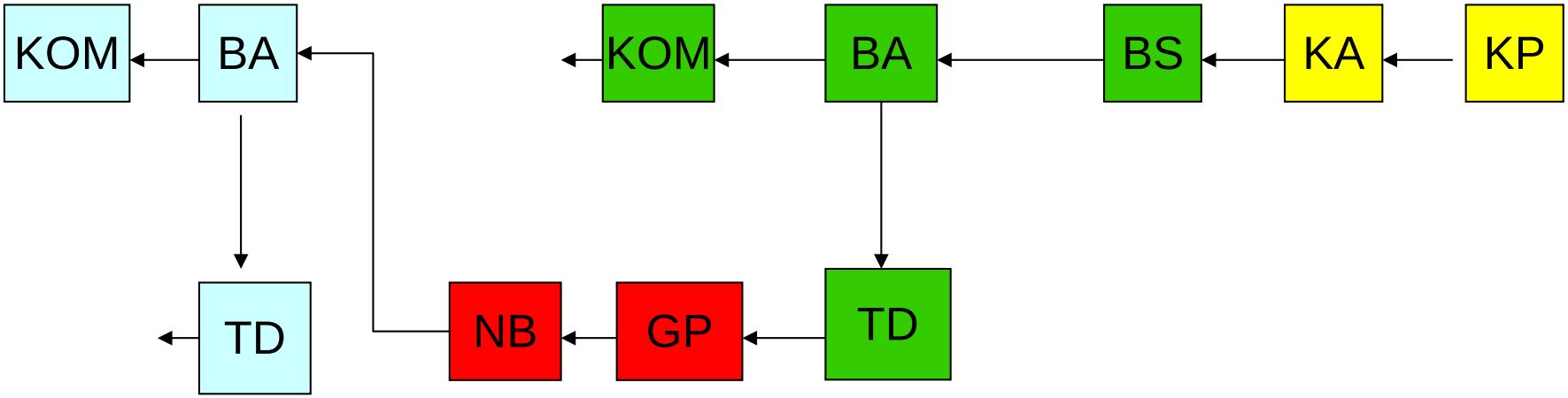
Ihre Services unterstützen Geschäftsprozesse, dies umfasst automatisierte Abläufe und Abläufe mit Benutzerinteraktion. Sie nutzen die Services von Funktions- und Bestandskomponenten

Interaktionskomponenten:

Ihre Services sind die Benutzungsschnittstellen selbst, sie bieten Anwendern den Zugang zu den Services einer Anwendungslandschaft. Ideal ist die Grenze zwischen Anwendungen nicht mehr sichtbar (Einheitliches Layout, Single Sign-on).

Beispiel Integrationsszenario Automobilfertigung

Zulieferer Hersteller Vertriebsnetz



KP Kundenprofil erstellen
KOM Kommissionierung
NB Nachbestellen

KA Kundenangebot erstellen
TD Teiledisposition
BS Bestellen

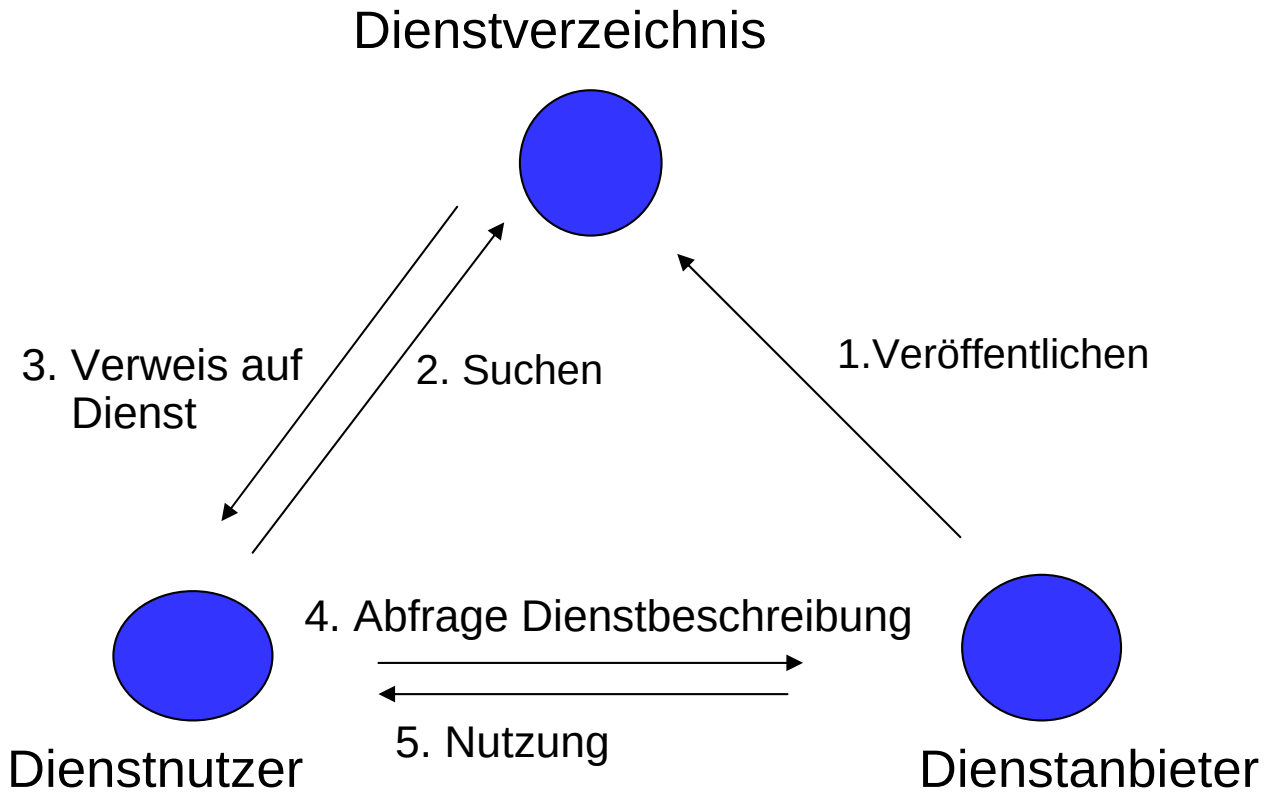
BA Bestellannahme
GP Globale Verfügbarkeit prüfen

ERP I ■ ERP II ■ SCM ■ CRM ■ : **Welches System wird welchen Aktivitäten zugeordnet?**

5.4 SOA Grundmodell und Standardarchitektur

Rollen:

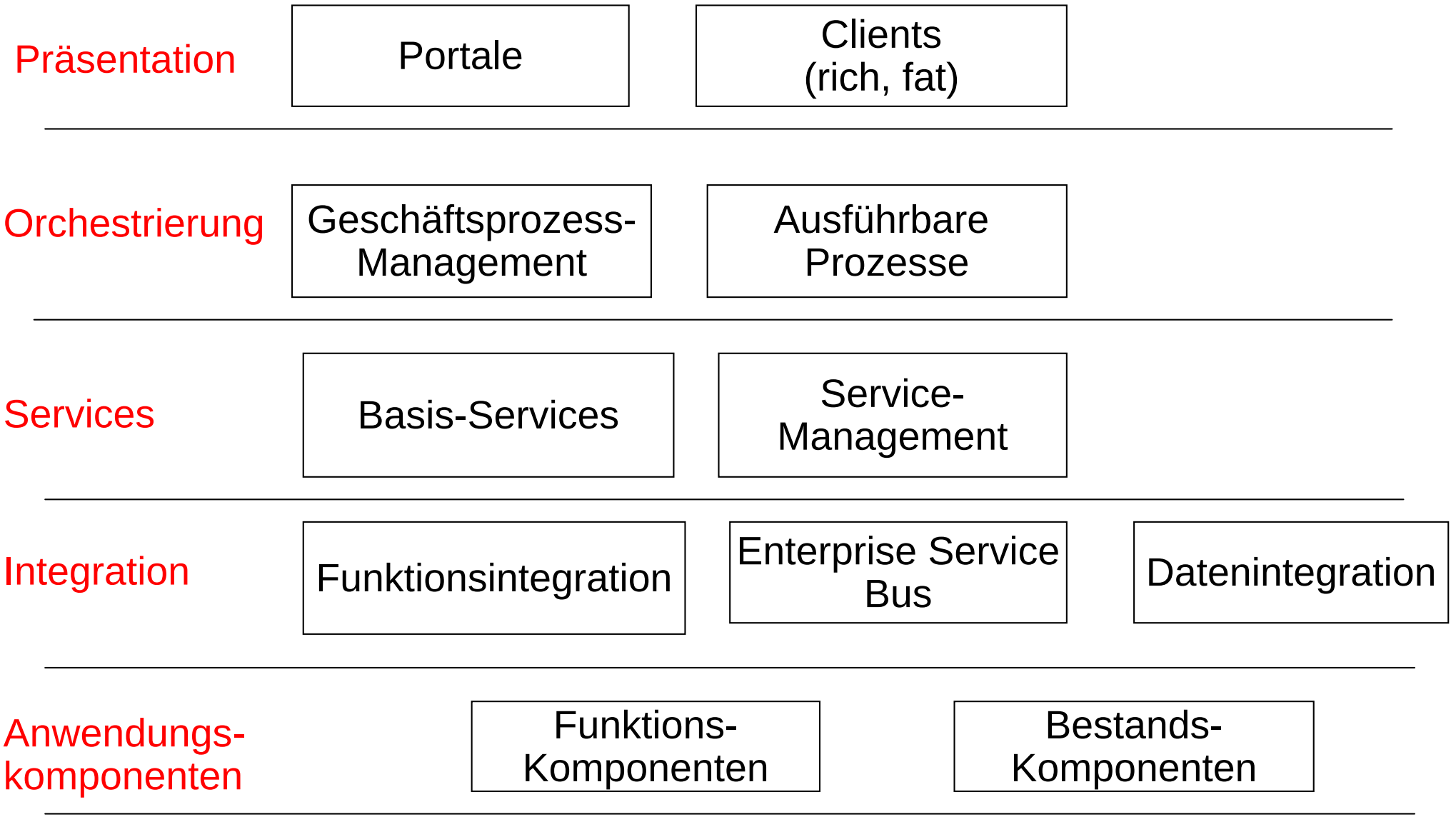
- Anbieter
- Nutzer
- Vermittler



Merkmale einer SOA

1. Lose Kopplung der Komponenten (Dynamische Bindung, Asynchroner Kommunikationsstil, Unabhängige Versionierung Client u. Server, „ Entfernung“ von Komponenten (fachlich,technisch)
2. Verwendung eines Verzeichnisdienstes
3. Verwendung von offenen Standards
4. Automatisierte Kommunikation zwischen Systemen, Benutzer interagiert nur punktuell
5. Leichte Wiederverwendbarkeit von Diensten
6. Ortstransparenz der Dienste
7. Flexibilität beim Austausch von Diensten
8. Plattformunabhängigkeit
9. Sprachunabhängigkeit

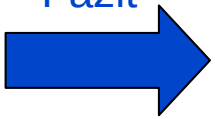
Standardarchitektur



Enterprise Service Bus (ESB)

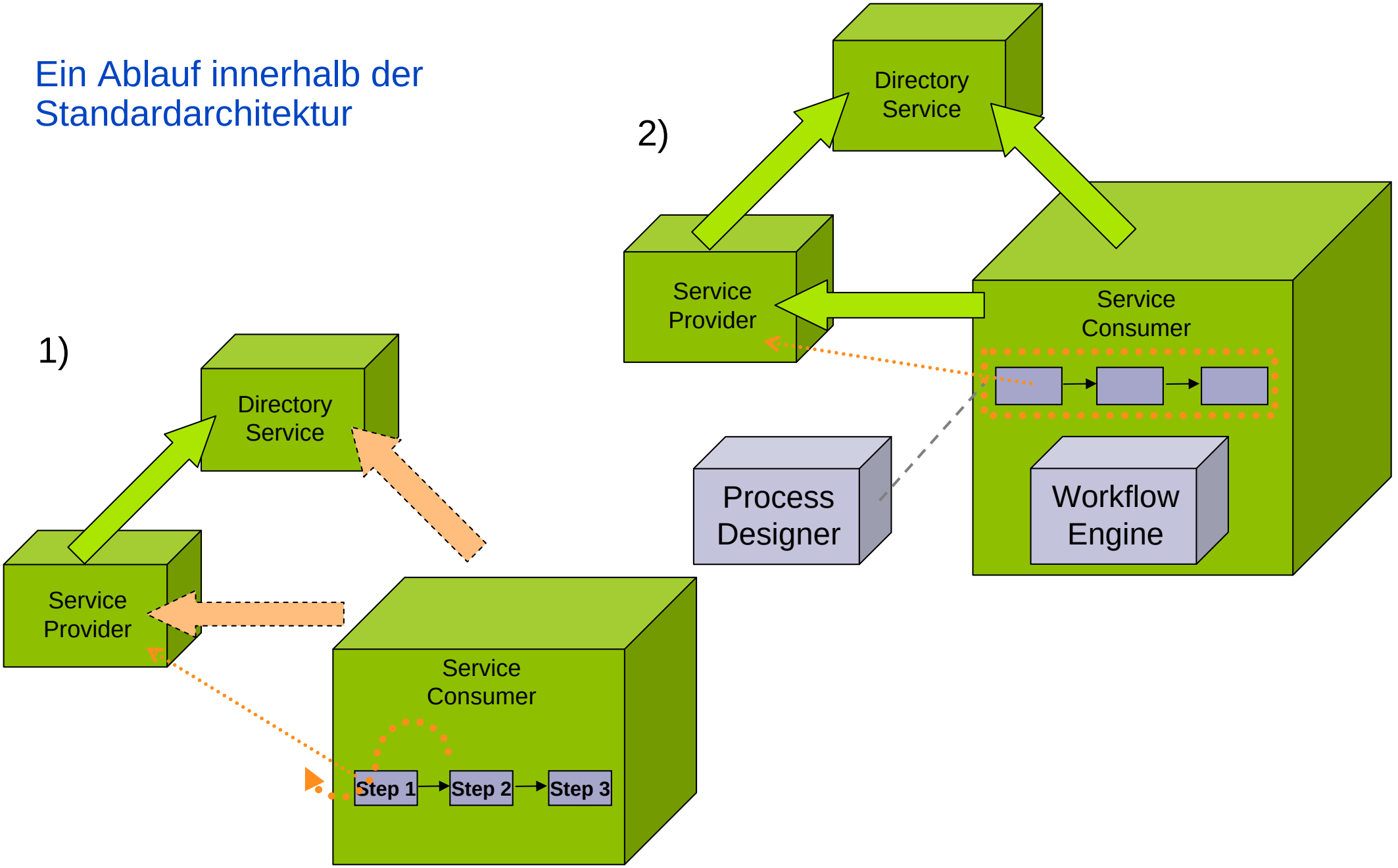
- Integrationsinfrastruktur innerhalb einer SOA
- Wirkt als Broker zwischen Service-Nachfragern und Service-Anbietern
- Entkoppelt die Service-Nachfrage vom Service-Angebot, verhindert Punkt-zu-Punkt Verbindungen
- Bietet intelligente Routing-Mechanismen
- Transformiert Datenformate
- Bietet die Vermittlung zwischen Service-Nachfrage und Service-Angebot unabhängig von den genutzten Protokollen
- Aus der Sicht der Dienste einfach und transparent

Fazit

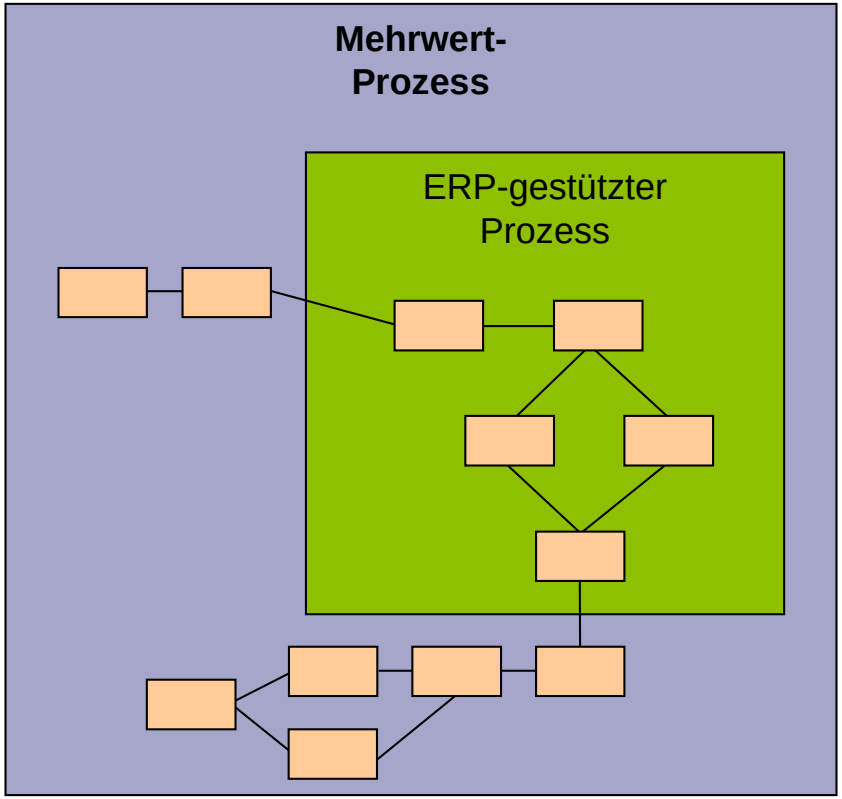


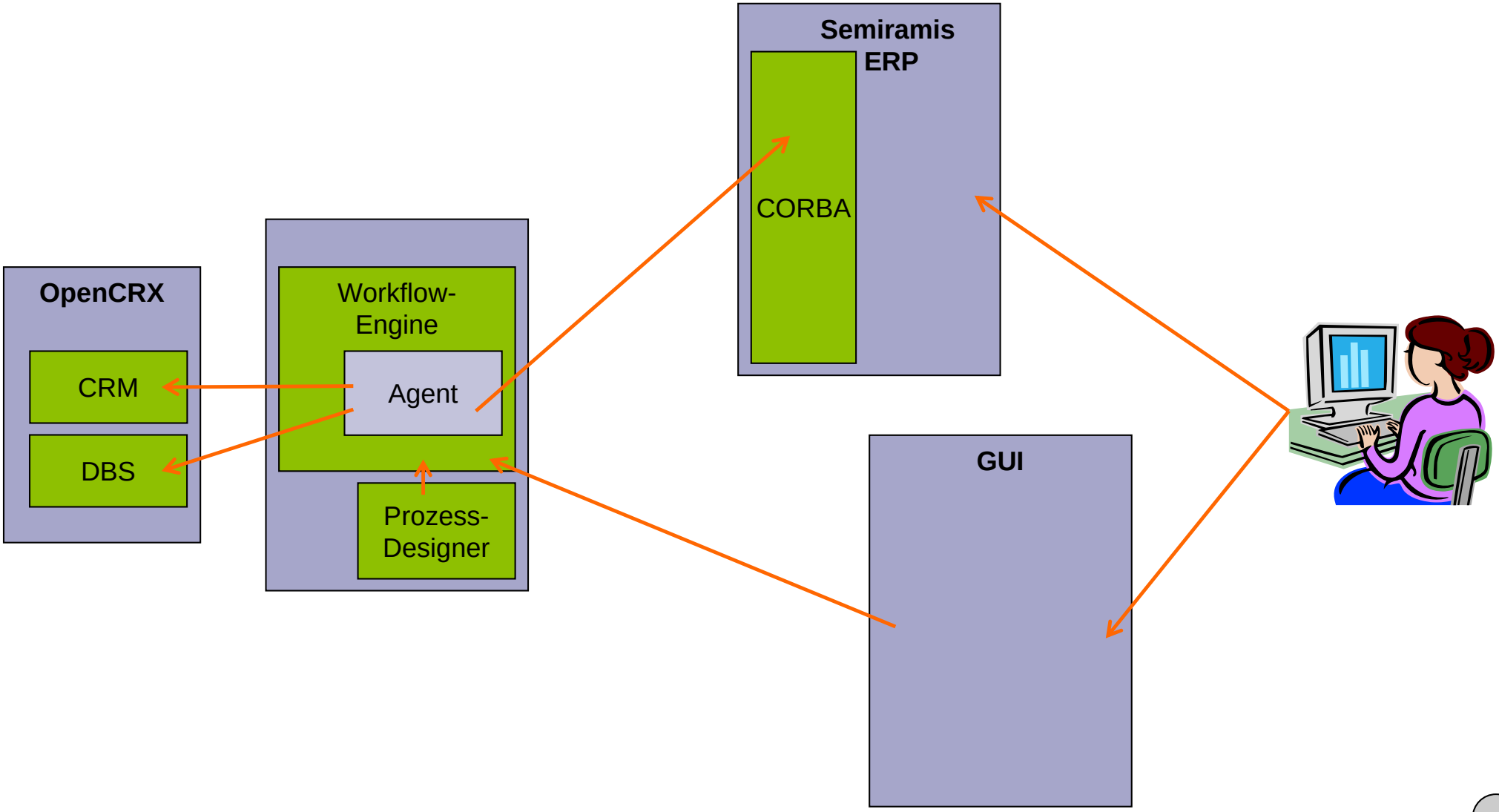
Der ESB stellt eine Erweiterung des Integrationsserver-Konzepts in Richtung einer Service-orientierten Architektur dar.

Ein Ablauf innerhalb der Standardarchitektur



Beispiel: Integration eines ERP/CRM-Systems auf Basis einer SOA




Semiramis
ERP
CORBA
OpenCRX
Workflow-
Engine
CRM
Agent
DBS
Prozess-
Designer
GUI

5.5 Services und Service Management

Im Zentrum stehen die Dienste:

Dienstbeschreibung:
- **Funktionale** Beschreibung: Schnittstelle (Signatur)
- **Nichtfunktionale** Beschreibung: Durch Service Level Agreements wie Antwortzeiten, Verfügbarkeit

Dienstanbieter:
- Stellt die Plattform und Dienste zur Verfügung

Darüber hinaus werden angeboten:
- Sicherheit der Plattform (Authentifzierung und Autorisierung)
- Verfügbarkeit des Dienstes und Wartung

Dienstverzeichnis:

- In der Regel registriert sich der Anbieter aktiv selbst, Klassifizierung des Dienstes für den entsprechenden Eintrag im Verzeichnis.

Dienstnutzer:

- Agiert wie ein Client in einer C/S-Architektur, jedoch erfolgt die Auswahl des Dienstes ohne explizite Codierung.
- Kommunikation zwischen Anbieter und Nutzer erfolgt über ein Protokoll, das beiden bekannt sein muss.

Aktionen zur Nutzung des Dienstes:

- Veröffentlichung (vorher muss der Dienst „deployed“ sein)
- Suchen auf Basis von Taxonomien (-> gelbe Seiten)
- Abfrage der Schnittstellenbeschreibung
- Bindung an den Dienst

Service-Management-Aufgaben

Entsprechend den Aufgaben des IT-Servicemanagements

Speziell:

- Organisatorische Verantwortlichkeit von Service-Providern und Service-Consumern

- Lifecycle Management

- Planung des Roll outs

- Festlegung Service Level Agreements (SLA)

- Abrechnungsmodelle für Dienste

Managed Services und Prozess-Outsourcing

Vorläufer: ASP – Application Service Providing

Der Provider bietet in seinem Rechenzentrum dem Kunden auf Mietbasis den Betrieb von Anwendungen (software as a service), so dass dieser damit seinen Geschäftsprozess abdecken kann.

Im Gegensatz zum Application hosting werden die Services nicht exklusiv von einem Kunden sondern von *n Kunden* genutzt.

Vision:

Ganze Geschäftsprozesse oder Prozessteile werden als orchestrierte Service-pakete (composite services) angeboten, die benötigten Applikationen werden „on the fly“ vom Kunden eingebunden.

5.6. Ein neues Vorgehensmodell?

Phasen:

Analyse: wie bei konventioneller SW – Entwicklung

Ziel: Erstellung einer Anforderungsdefinition
(benutzerorientiert, nicht technisch)

Ergänzt um: Prüfung des Projekts auf Eignung für
SOA, Sichtung verfügbarer Dienste.

Entwurf

- Identifikation von Services aus dem (SOA-basierten) UML–Entwurf
- Prüfung auf Wiederverwendung von Services (Recycling aus früheren Projekten, Einsicht in die Dokumentationen möglich?)
- Bestimmung der Granularität der Dienste, d.h. des Funktionsumfangs der Dienste (Welche Größe ist optimal: "fine-grained" vs. "coarse-grained")
- Definition der Schnittstellen nach internationalen Standards (WSDL)
- Entwurf der Integration/Komposition von Diensten zur Workflow-Unterstützung

Implementierung

- Teamorientiert mit „ Serviceverantwortlichkeit“,
  Einsatz eines SOAP-Frameworks
- Komposition der Services

Betrieb

- Wartung und Pflege (Vertragsgestaltung ?)
- Anpassung bei neuen Prozessanforderungen durch Austausch von Services
- Verfügbarkeit von Services
- Sicherheit

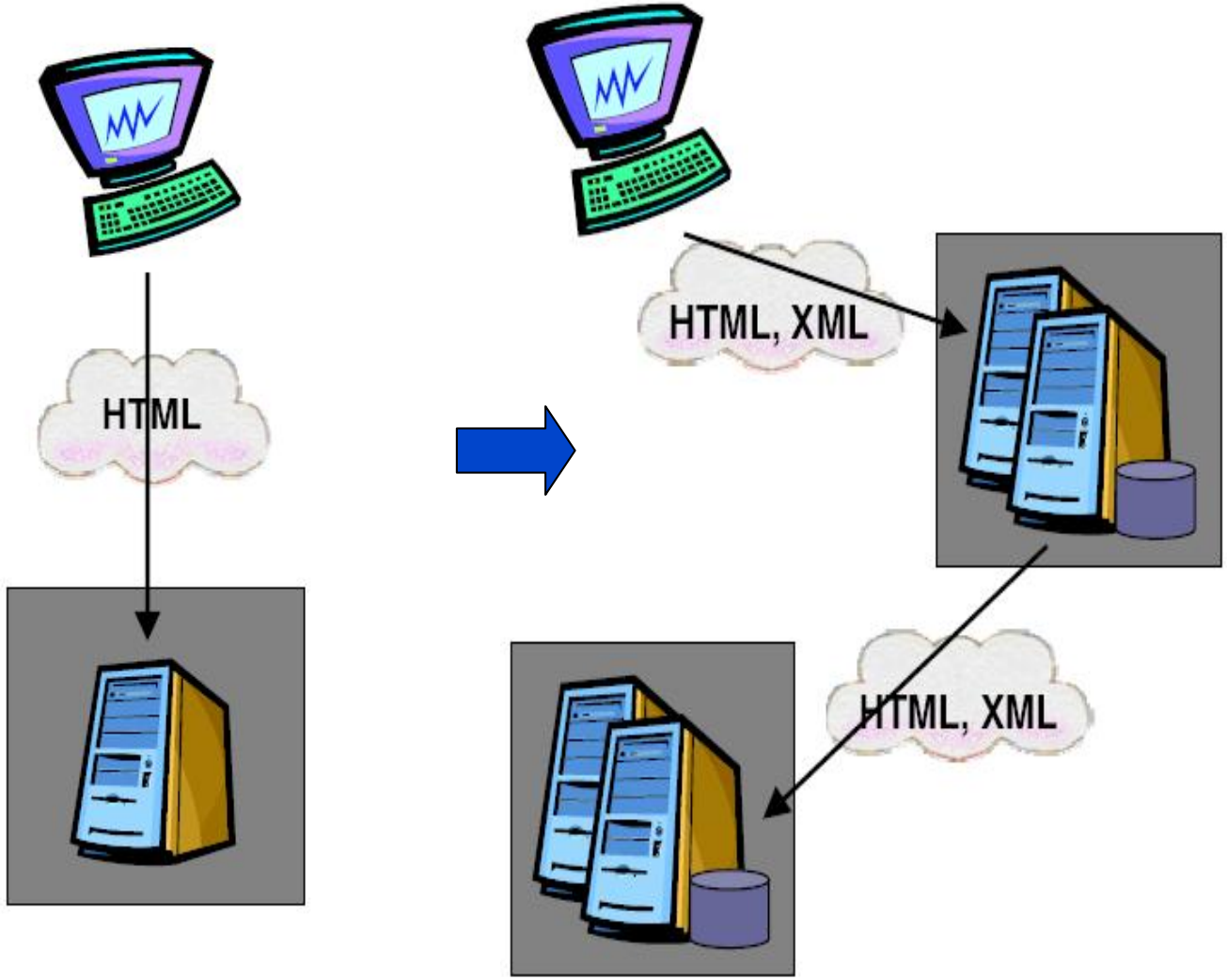
Vorlesung „ Daten- und Systemintegration“

Kapitel 5 SOA und Web-Services, Teil 2

5.7 WebServices

5.8 Session Beans als WebServices

5.7 WebServices
Die Idee



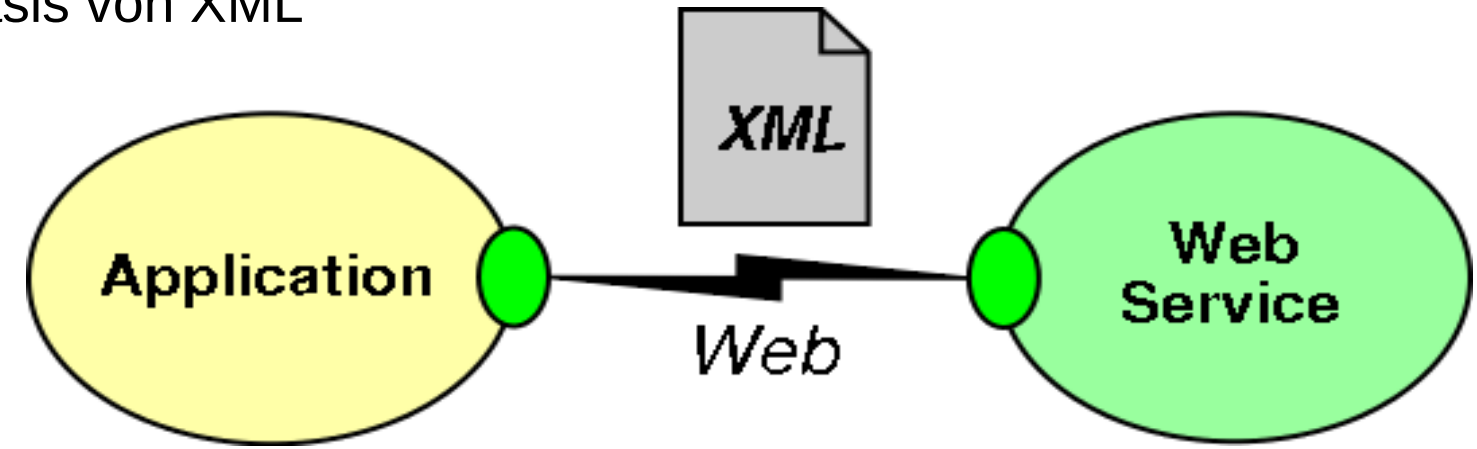
5.7 WebServices
Die Idee

Web Services bestehen aus

XML-Protokoll für **Interprozess-Kommunikation**
SOAP („Simple Object Access Protocol“)

XML – basierte **Beschreibungssprache** für Web Services
WSDL (Web Service Description Language) („IDL“ für Web Services)

Verzeichnis, in dem Web Services mit ihren WSDL Daten eingetragen werden: Universal Description, Discovery and Integration (UDDI), auf Basis von XML



5.7 WebServices
Definition

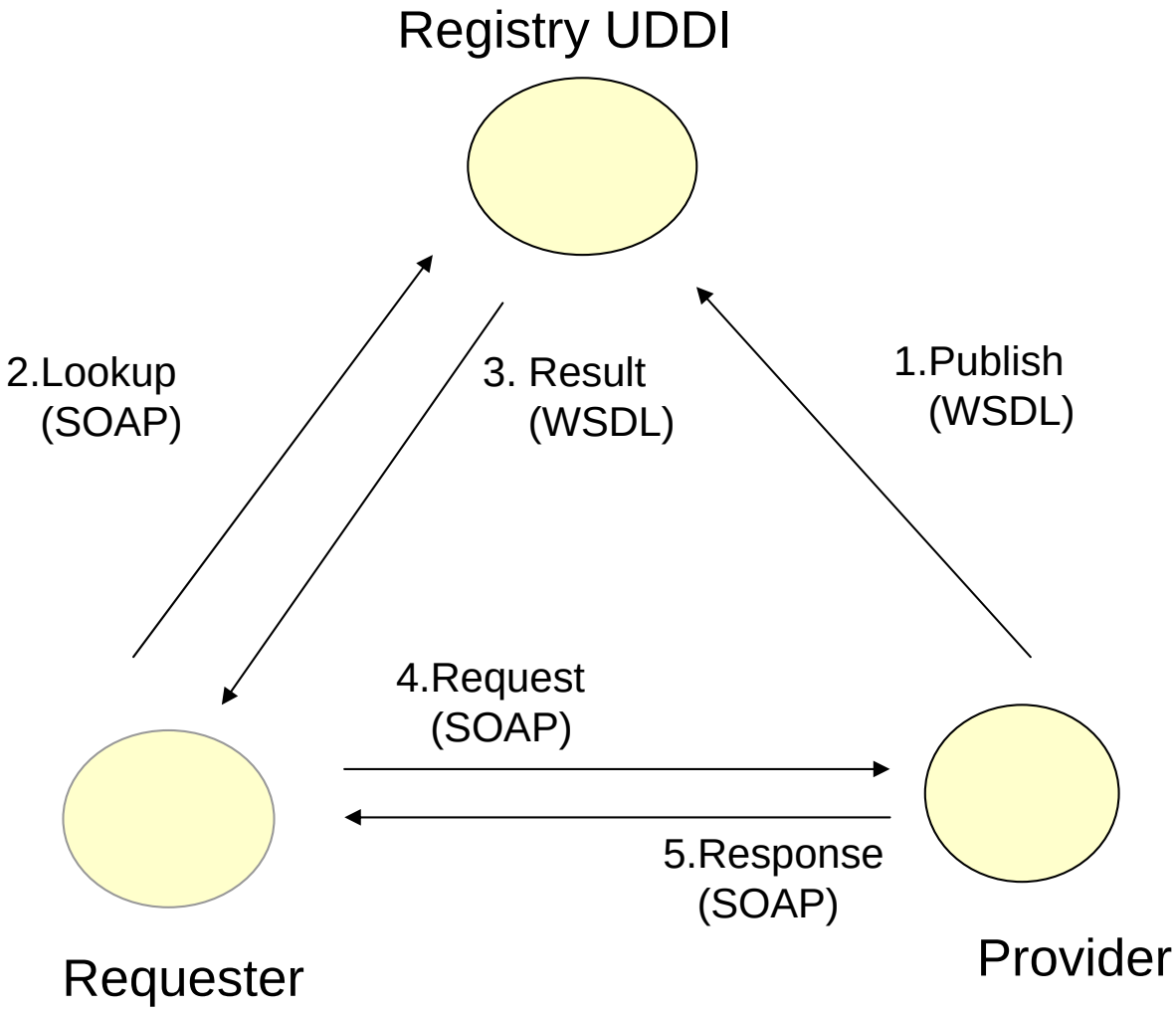
„A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.“

W3C 2003

„ A Web Service is a piece of software that is easily accessible over the Web. A Web Service is built on technology which are independent of the operating system, programming languages and the component model.“

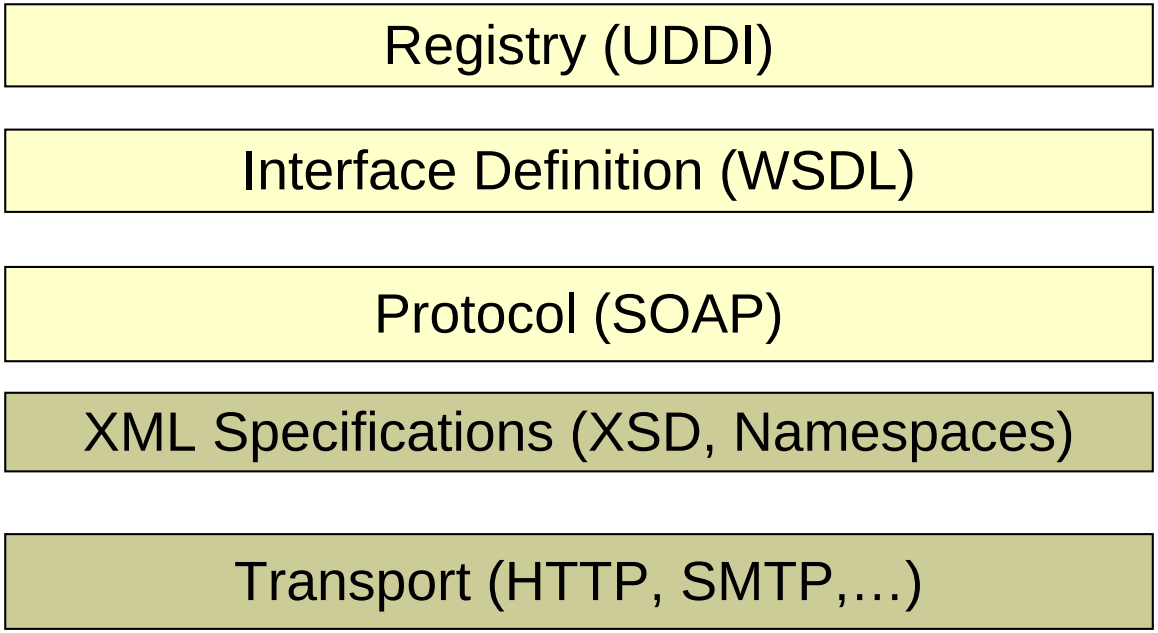
5.7 WebServices

Die WebService-Architektur als eine mögliche Implementierung einer SOA



5.7 WebServices

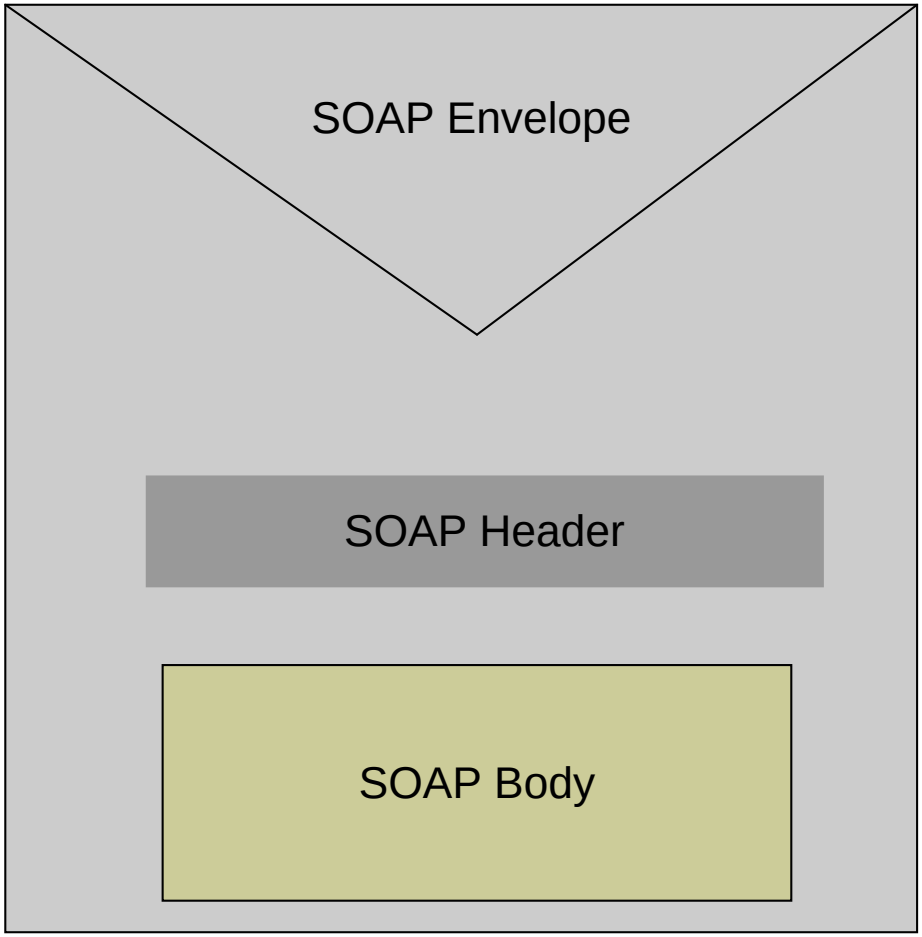
Die WebService-Architektur als eine mögliche Implementierung einer SOA



Der WebService-Stack (vereinfacht)

5.7 WebServices

Simple Object Access Protocol (SOAP)



Aufbau einer SOAP-Nachricht

Simple Object Access Protocol (SOAP)

Eine SOAP-Nachricht besteht aus drei Teilen:

- Einem Umschlag (Envelope) als Wurzelement des XML–Dokuments, der beschreibt, was in einer Nachricht enthalten ist und wie diese zu verarbeiten ist.

- Optional einem Header

- Einem Body, der die eigentliche Nutzinformation enthält (Methodenaufrufe mit Rückgabewerten (RPC style) oder beliebige Dokumenteninhalte (document style)).

5.7 WebServices

Simple Object Access Protocol (SOAP)

Die SOAP Spezifikation des W3C liegt seit 2003 in der Version 1.2 vor:

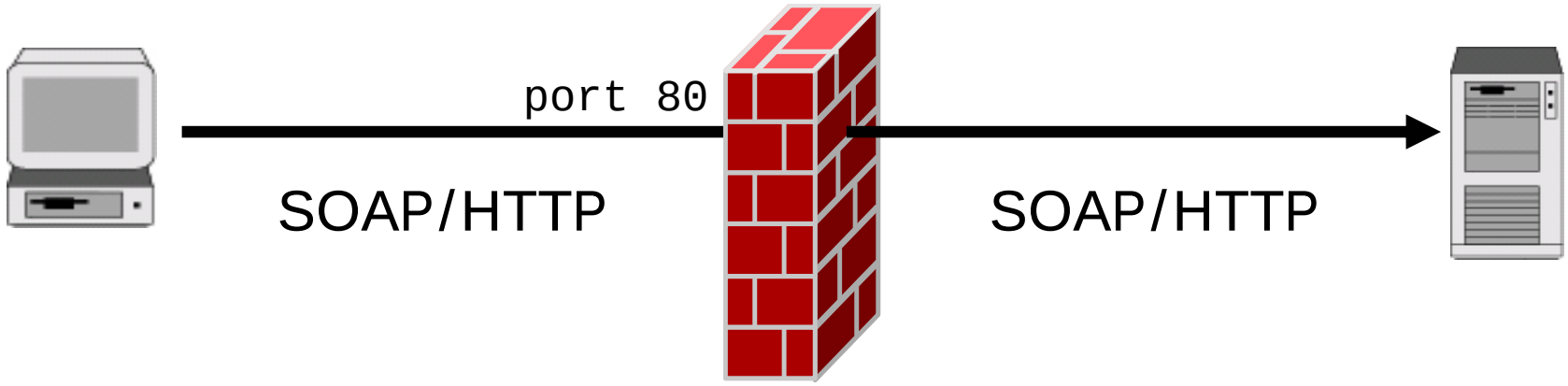
- SOAP dient als Kommunikationsprotokoll zum Austausch von Nachrichten in verteilten Umgebungen. Es basiert auf XML.

- SOAP benutzt als Transportprotokolle HTTP, SMTP (Simple Mail Transfer Protocol) oder andere Protokolle. So wird bei Verwendung von HTTP ein SOAP-Request in einem **HTTP-Request** verpackt und in der gleichen Verbindung in einem **HTTP-Response** der SOAP-Response geschickt

- Durch Verwendung des Transportprotokolls HTTP werden Firewall Probleme überwunden, da diese HTTP-Pakete in der Regel durchlassen.

Risiko Web Service

HTTP wird von Firewalls nicht gefiltert!



5.7 WebServices

Simple Object Access Protocol (SOAP)

Der SOAP-Envelope (Beispiel mit einem Header und einem Body)

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV=http://www.w3.org/2003/2005/soap-envelope>

    <SOAP-ENV:Header>
    ...
    </SOAP-ENV:Header>

    <SOAP-ENV:Body>
     ...
     SOAP-ENV:encodingStyle= "http://www.w3.org/2003/2005/soap-encoding"

    </SOAP-ENV:Body>
      ...

</SOAP-ENV:Envelope>
```

Optionale Metainformationen
(z. B. Authentifizierung, Autorisierung)

Nutzinformationen

5.7 WebServices

Simple Object Access Protocol (SOAP)

Exkurs: Namensräume und XML Schema

Ein Namensraum (namespace) ist eine Sammlung von eindeutigen Namen für Elementtypen und Attribute, identifiziert durch einen **URI**.

XML-Dokument

```
<lit:Buch xmlns:lit='www.melibo.de/schema'>
        <lit:Titel>Melibo und Winnie</lit:Titel>
        <lit:Autor>AZB</lit:Autor>
        <lit:Autor>DW</lit:Autor>
        <lit:Verlag>
            <lit:Name>Melibo Verlag</lit:Name>
            <lit:Adresse>
                <lit:Ort>Darmstadt</lit:Ort>
            </lit:Adresse>
        </lit:Verlag>
</lit:Buch>
```

xmlns
= XML Namespace

Definition XML-Schema

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="www.melibo.de/schema">
        <xs:element name="Adresse">
            <xs:complexType>
                <xs :sequence>
                    <xs:element name="Ort" Type="string"/>
                </xs :sequence>
            </xs:complexType>
        <xs:element/>
        <xs:element name="Verlag">
            <xs:complexType>
                <xs :sequence>
                    <xs:element name="Name" Type="string"/>
                    <xs:element ref="Adresse"/>
                </xs :sequence>
            </xs:complexType>
        <xs:element/>
        <xs:element name="Buch">
            <xs:complexType>
                <xs :sequence>
                    <xs:element name="Titel" Type="string"/>
                    <xs:element name="Autor" Type="string"
                        maxOccurs="5" minOccurs="1"/>
                    <xs:element ref="Verlag"/>
                </xs :sequence>
            </xs:complexType>
        <xs:element/>
</schema>
```

Der SOAP-Header (Beispiel)

Metainformationen (Sicherheit, Transaktionen,…)

```
<env:Header>
        <t:Transaction xmlns:t="URI"
                        env:mustUnderstand="1">TRID57738
        </t:Transaction>
</env:Header>
```

Der SOAP-Body (Beispiel)

Anwendungsdaten (Methodenaufrufe/ Parameter / Rückgabewerte)

```
<env:Body>
            <m:ErmittleAutor xmlns:m="www.melibo.de/schema">
                        <Titel>Melibo und Winnie</Titel>
            </m:ErmittleAutor>
</env:Body>
```

Web Service Description Language (WSDL)

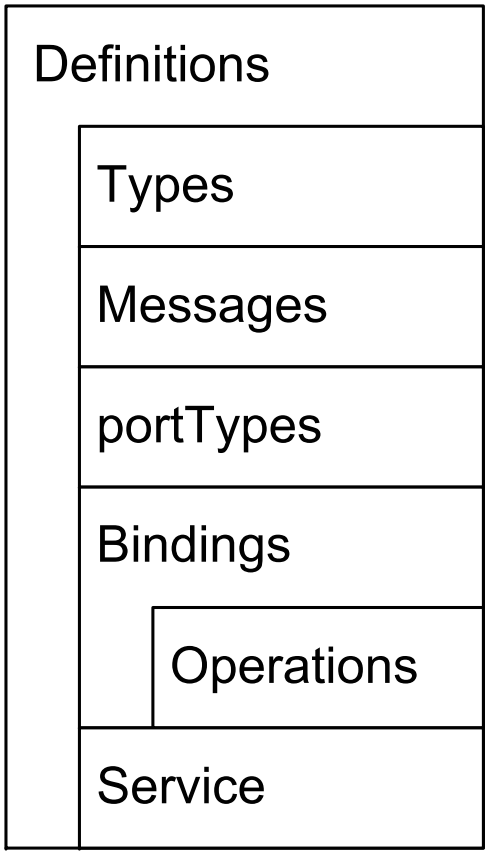
- Beschreibungsprache für WebService-Schnittstellen
- Standardisiert von W3C

Web Services Schnittstellen werden in WSDL Dateien beschrieben.
WSDL Dateien sind spezielle XML Dateien.

5.7 WebServices

Web Service Description Language (WSDL)

Aufbau einer WSDL-Datei



Das Wurzel–Element <definition> der Datei enthält die Beschreibungselemente, um einen Web Service zu beschreiben:

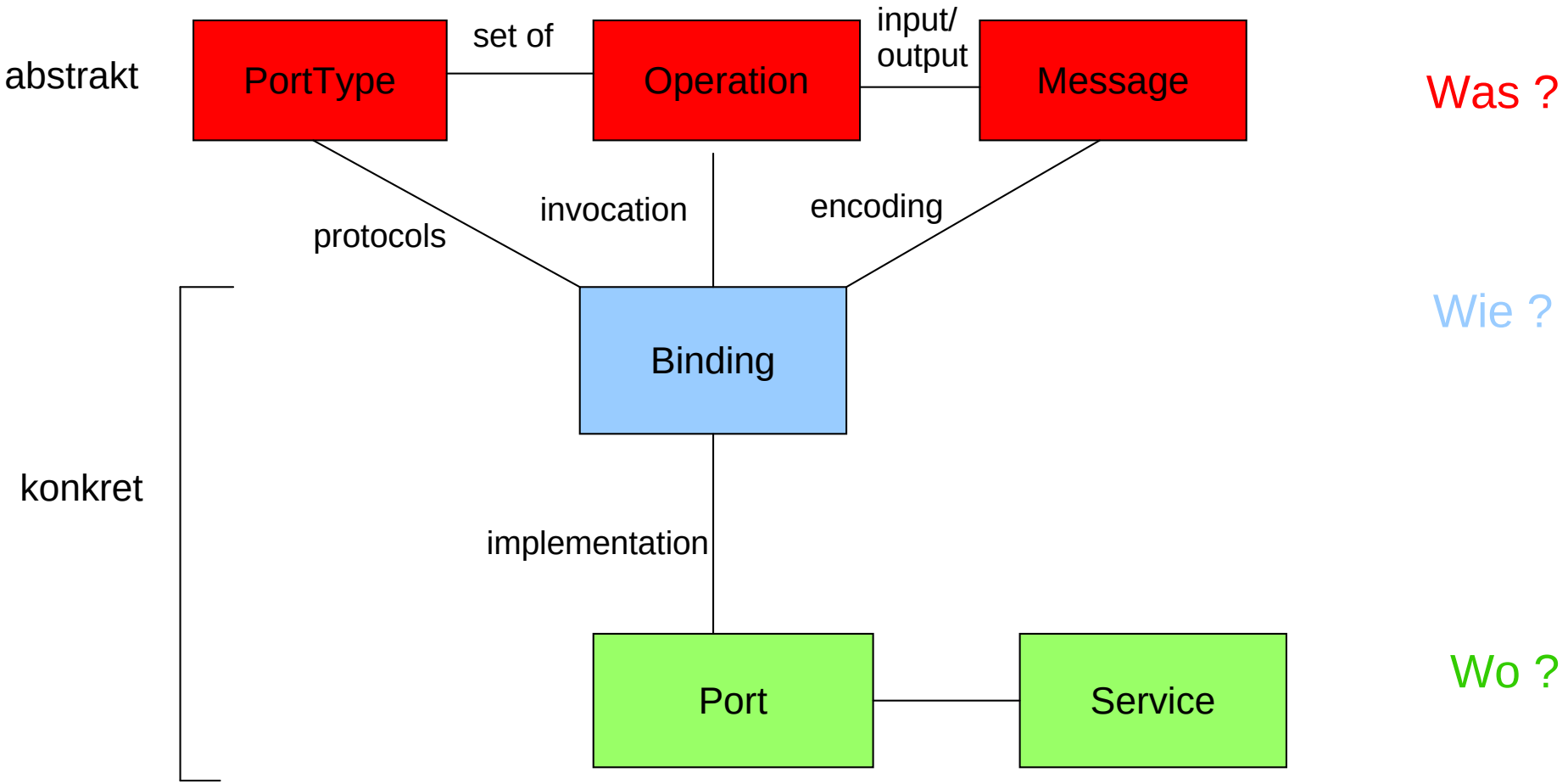
<types>	definiert die Datentypen, z.B. XML-Schema-Typen
<message>	definiert die verwendeten Parameter einer Nachricht
<portType>	definiert eine Menge von Operationen (die Schnittstelle)
<binding>	definiert das Nachrichtenformat und Protokoll
<operation>	definiert einen Methodenaufruf
<service>	definiert die konkrete Adresse des Web Service

WSDL-Datei (Beispiele für Elemente)

- **Types:** Integer, String, Float, Array,…
- **Messages:** ErmittleAutorInput, ErmittleAutorOutput
- **Operation:** ErmittleAutor, bestehend aus Input-Message und Output-Message
- **Port Type:** Verlags-Service mit allen Operationen, die der Service zur Verfügung stellt
- **Binding:** Bindung an das SOAP–Protokoll
- **Service:** URL des Web Services

5.7 WebServices

Web Service Description Language (WSDL)



Web Service Description Language (WSDL)

- WSDL beschreibt Web-Dienste als **Satz von Endpunkten**

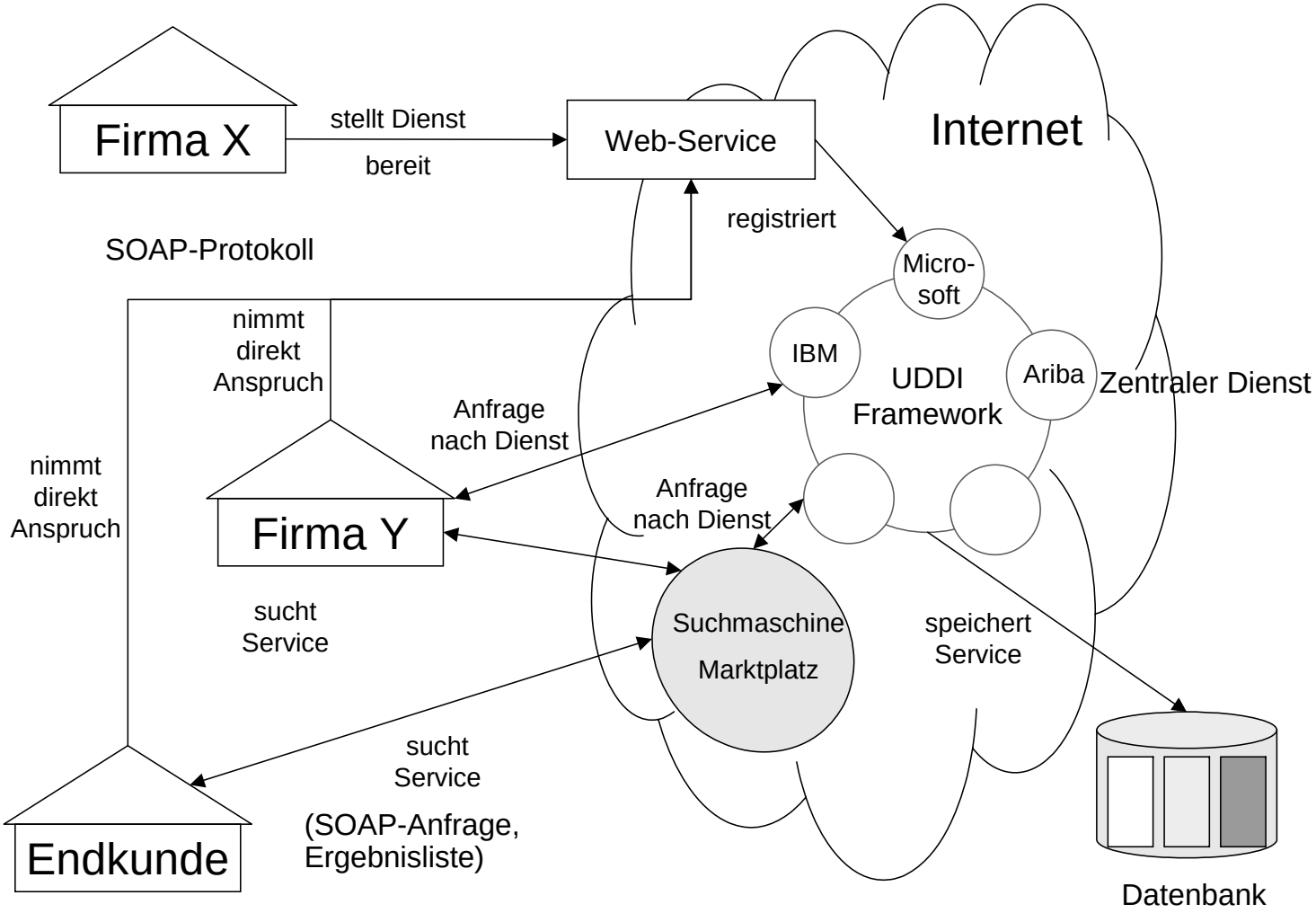
- Die Operationen und Nachrichten werden abstrakt beschrieben und später als konkretes Protokoll und Nachrichtenformat zusammengebunden, um einen entsprechenden Endpunkt darzustellen. Abhängige oder zusammengehörige Endpunkte können dann zu Composite Services kombiniert werden.

- Durch die abstrakte Beschreibung können Teile öfter wiederverwendet werden. Unter einem Teil kann z.B. ein Endpunkt verstanden werden, der in Verbindung mit anderen Endpunkten einen Service definiert. Dies verbietet aber nicht, den beschriebenen Endpunkt dazu zu verwenden, einen oder mehrere andere Services mit zu beschreiben. Wenn entsprechende Definitionen vorliegen, ist eine Service Beschreibung mit einem **Puzzle-Spiel** zu vergleichen. Es werden die passenden, schon vorliegenden Teile ausgesucht und zu einem Gesamtpaket (Composite Service) zusammengesetzt.

5.7 WebServices

Universal Description, Discovery and Integration (UDDI)

Funktionsweise von UDDI am Beispiel eines beliebigen Web-Services:



UDDI-Einbindung

Ablauf:

1. Eine Firma X stellt einen Web-Service bereit (z.B. Börsenkursinformationen).
2. Sie registriert diesen im UDDI-Framework, wo die entsprechenden Eigenschaften (Firmeninformationen, Schnittstelle usw.) in einer Datenbank abgespeichert werden.
3. Eine Firma Y oder ein Endkunde möchten nun einen Web-Dienst (z. B. eine Kursinformation über Siemens) erfragen. Sie kennen aber keinen Anbieter, der ihnen diese Information bereitstellt.
4. Beide (Firma Y bzw. Endkunde) versuchen eine Suche in einer Suchmaschine.
5. Diese Suchmaschine ermittelt im UDDI-Service (über SOAP-Anfragen) eine Liste von Unternehmen inkl. Firma X, die der Anfrage entsprechen. Diese Liste wird an die Firma Y und den Endkunden als Ergebnis versendet.
6. Beide können jetzt direkt mit dem Web-Service der Firma X kommunizieren und seinen Dienst in Anspruch nehmen.
7. Die Firma Y kann auch direkt mit dem UDDI-Framework kommunizieren, um z.B. die Suchanfrage in der Anwendung automatisch durchführen zu lassen.

5.7 WebServices

Universal Description, Discovery and Integration (UDDI)

- Der "Universal Description, Discovery and Integration"-Service ist ein plattformunabhängiger, offener Dienst, der es erlaubt, verschiedene Web-Services im Internet zu finden (Discovery) und zu beschreiben (Description).

- Er liefert darüber hinaus Informationen, wie die gefunden Services in die eigene Anwendung integriert (Integration) werden können.

- Man kann ihn als eine Art "**Branchenbuch**" für Web-Dienste bezeichnen, der ständig weiterentwickelt wird und primär „Business to Business“ - Geschäfte erleichtern soll.

- IBM®, Microsoft® und Ariba® sind die Begründer des UDDI-Services und haben seit September 2000 eine erste Testplattform im Internet zur Verfügung gestellt.

5.7 WebServices

Universal Description, Discovery and Integration (UDDI)

- UDDI ist ein verteilter Internet-Service mit mehreren Knotenpunkten, die auf täglicher Basis ihre Daten untereinander abgleichen, damit alle den gleichen Stand aufweisen. Dieser Service kann über Programm-API's oder im Web verfügbare Suchmaschinen angesprochen werden.

- Nach einer Suche z.B. des Firmennamens oder einer Servicekategorie werden alle gefundenen Anbieter mit zusätzlichen Informationen aufgelistet und bei Bedarf Schnittstellendetails zu ihren Web-Diensten geliefert.

- Diese Schnittstellendetails können dazu benutzt werden, den Dienst entsprechend zu starten. Damit wird verhindert, dass eine Firma nicht mit jedem Kunden eine eigene Online - Beziehung unterhält. Der Aufwand, jeweils eine eigene Schnittstelle mit ihren Standards und Protokollen zu pflegen, wäre umständlich und teuer.

5.7 WebServices

Universal Description, Discovery and Integration (UDDI)

- Der UDDI-Service selbst und seine Schnittstelle ist in XML formuliert. Die Anfragen werden über das SOAP-Protokoll gesendet.
- Die Informationen werden in drei Kategorien unterteilt:

  + Weiße Seiten (Adresse und Kontaktinformation)
  + Gelbe Seiten (Branchenzuordnungen)
  + Grüne Seiten (technische Informationen über die Web-Services, die vom Unternehmen bereitgestellt werden)

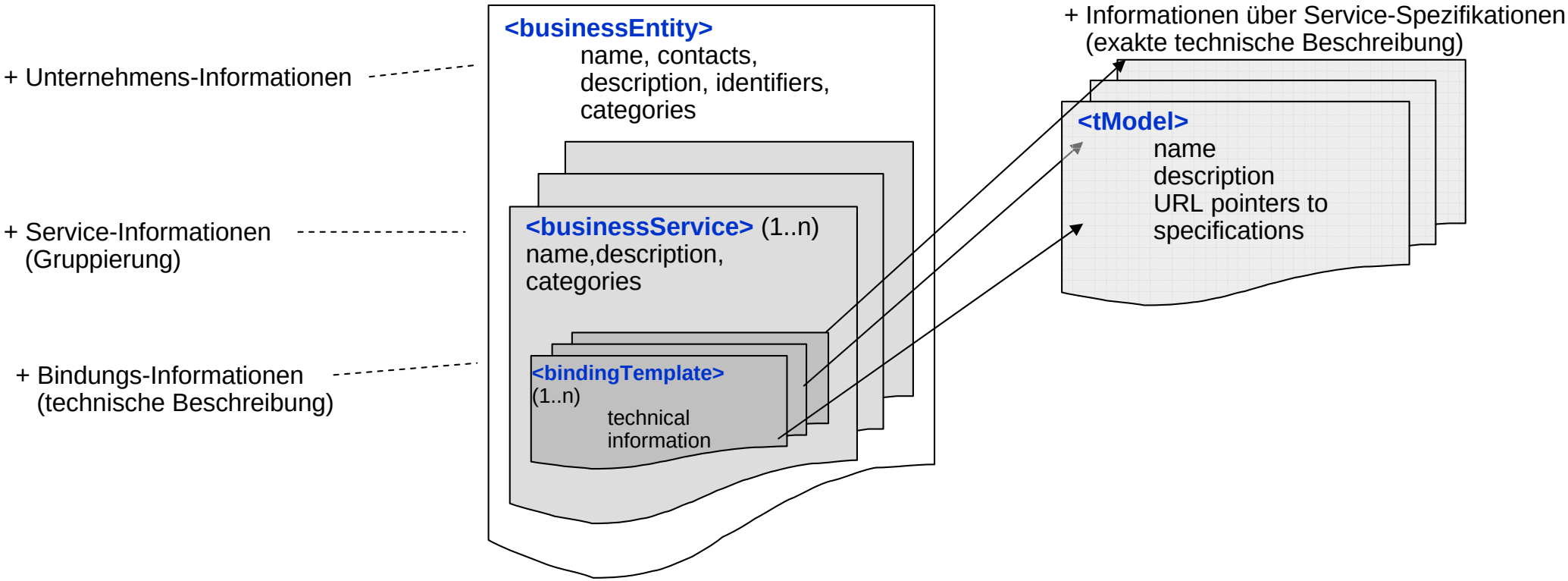
Beispiel: Suche nach Unternehmensdaten von Microsoft

```
<env:Body>
        <find_business generic = “1.0”>
                <name>Microsoft</name>
        </find_business>
</env:Body>
```

5.7 WebServices

Universal Description, Discovery and Integration (UDDI)

Aufbau dieser Informationshierachie als XML-Dokument:



UDDI-Registrierungseintrag eines Unternehmens

5.7 WebServices

SOA-Merkmale erfüllt ?

Merkmal	Web Services	CORBA	J2EE
1 Lose Kopplung der Dienste	+	0	0
2 Dynamische Bindung der Dienste	+	- statisch	- statisch
3 Verwendung eines Verzeichnisdienstes	+	0 Naming S.	0 Naming S.
4 Verwendung von offenen Standards	+	+	+
5 Automatisierte Kommunikation zwischen Systemen, Benutzer interagiert nur punktuell	+	+	+
6 Leichte Wiederverwendbarkeit von Diensten	+	0	0
7 Ortstransparenz der Dienste	+	+	+
8 Flexibilität beim Austausch von Diensten	+	-	-
9 Plattformunabhängigkeit	+	+	+
10 Sprachunabhängigkeit	+	+	- nur Java

Vorlesung „ Daten- und Systemintegration“

Kapitel 4 SOA und Web-Services, Teil 2

5.7 WebServices

5.8 Session Beans als WebServices

5.8 Session Beans als WebServices

Java EE 5 und EJB 3

Wichtige WebService-APIs für EJB-Anwendungen sind:
- JAX-RPC (Java API for XML-based RPC)
- JAX-WS (Java API für XML Web Services)

Die WS-I (Web Services Integration Organization) sorgt mit dem WS-I Basic Profile dafür, dass die WebServices interoperabel sind. In der WS-I sind so wichtige Firmen wie z. B. Microsoft, IBM, BEA, Sun Microsystemens und Oracle vertreten.

5.8 Session Beans als WebServices

JAX-RPC (Java API for XML-based RPC)

- Das WS Endpoint Interface basiert auf der WSDL-PortType Spezifikation und kann mittels JAX-RPX Compiler aus der WSDL-Datei generiert werden.
- JAX-RPC definiert das Mapping zwischen XML-Schema-Typen und Java-Datentypen (z. B. xsd:int <-> int, xsd:string <-> java.lang.String)

```
<message name="msgRequest">
        <part name="name" type="xsd:string"/>
 </message>

        <message name="msgResponse">
        <part name="return" type="xsd:int"/>
 </message>
        <portType name="Processor">
        <operation name="msg"/>
          <input   message="tns:msgRequest"/>
          <output message="tns:msgResponse"/>
        </operation>
</portType>
```

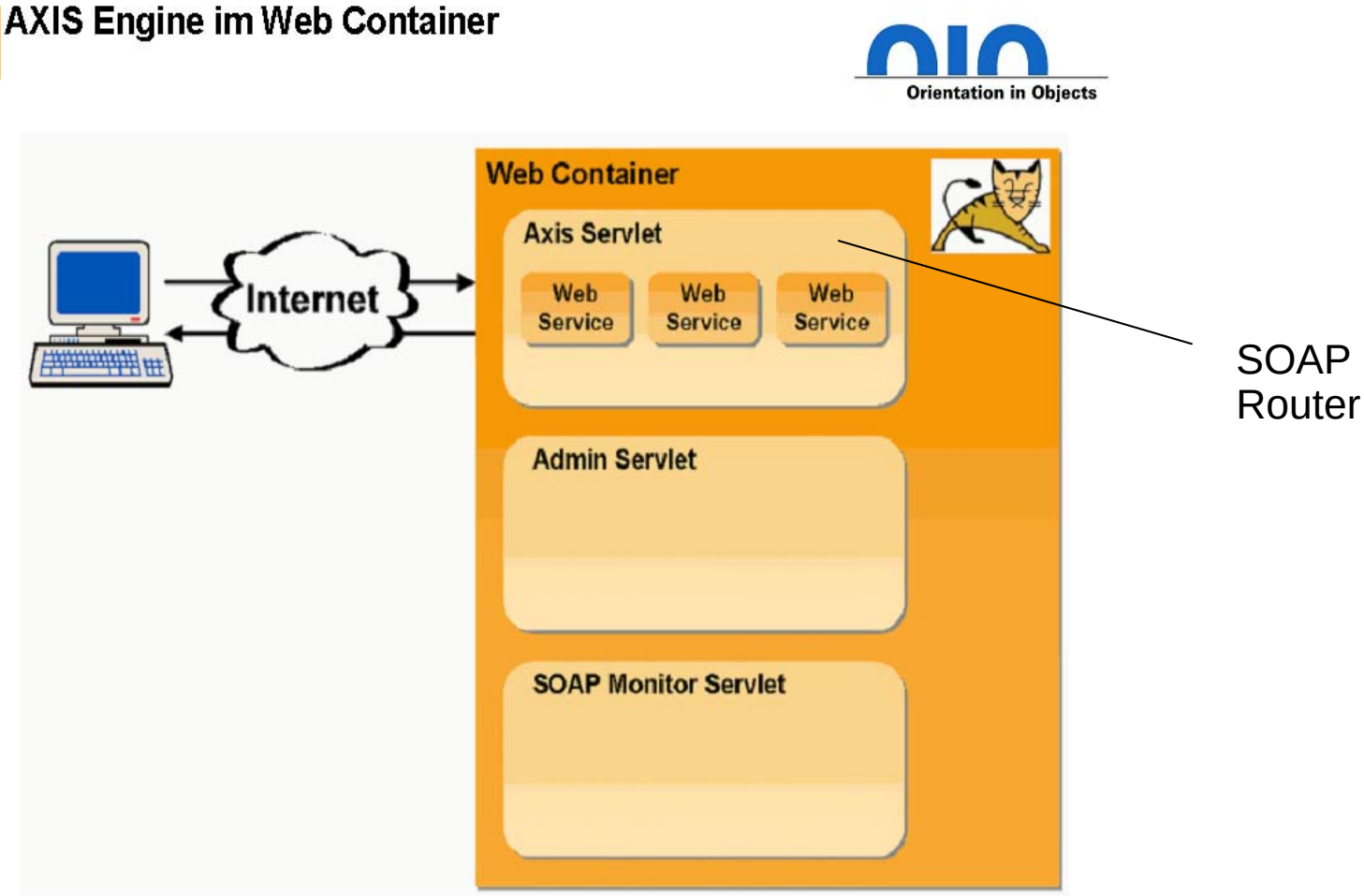
WSDL-Datei

```
public interface Processor extends java.rmi.Remote {
   public int msg(String name)
   throws java.rmi.RemoteException;
}
```

JAX-RPC endpoint interface

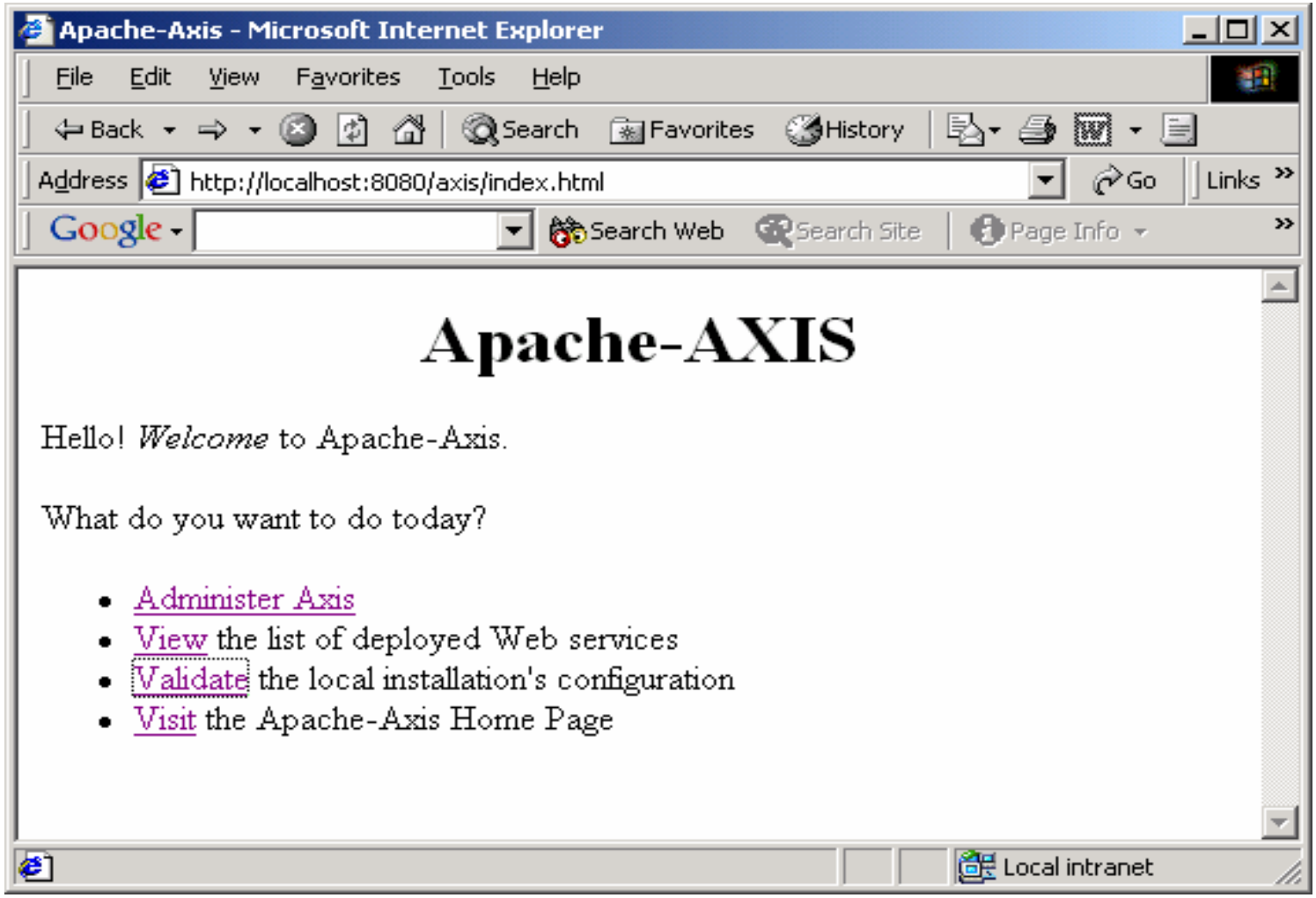
5.8 Session Beans als WebServices

Axis (Apache Extensible Interaction System) als JAX-RPC –konforme SOAP-Engine



5.8 Session Beans als WebServices
Implementierung

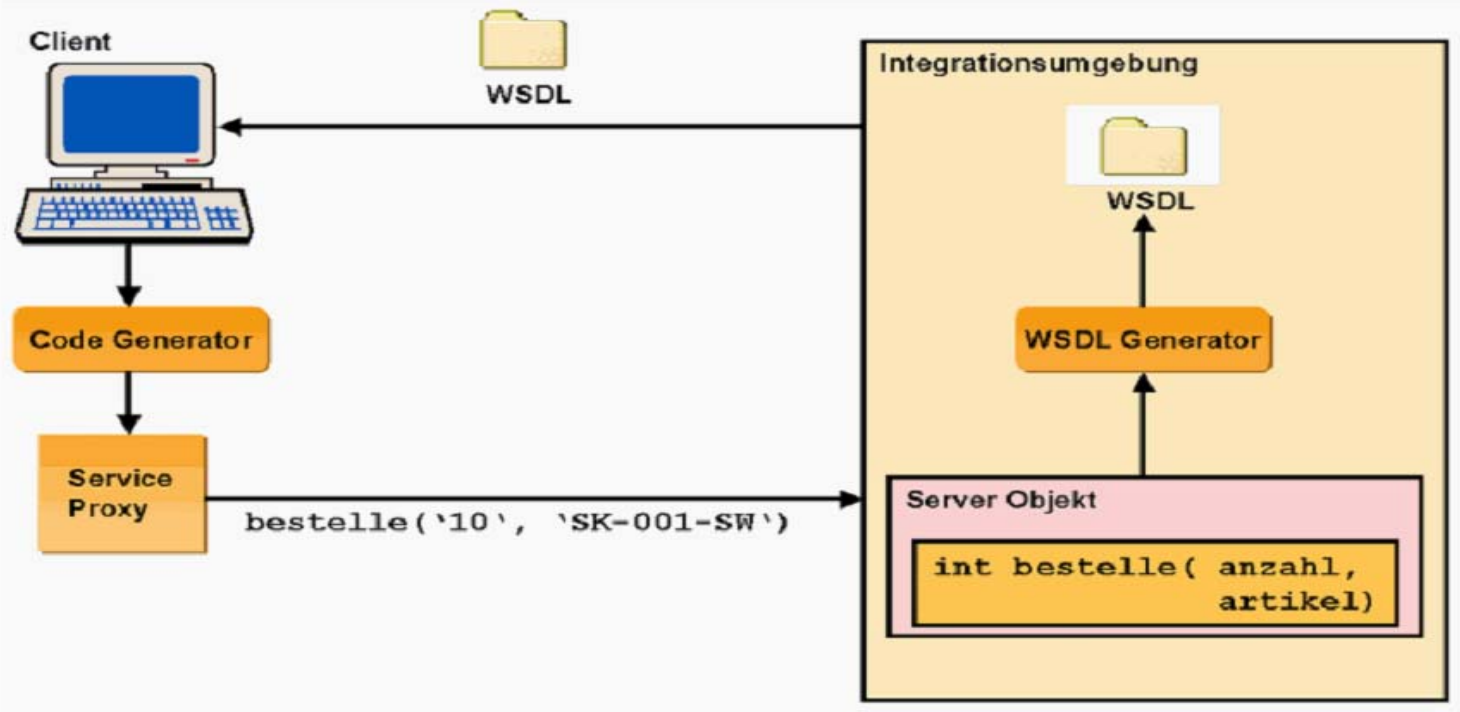
AXIS Administration (z. B. Deployment von WebServices testen)



5.8 Session Beans als WebServices

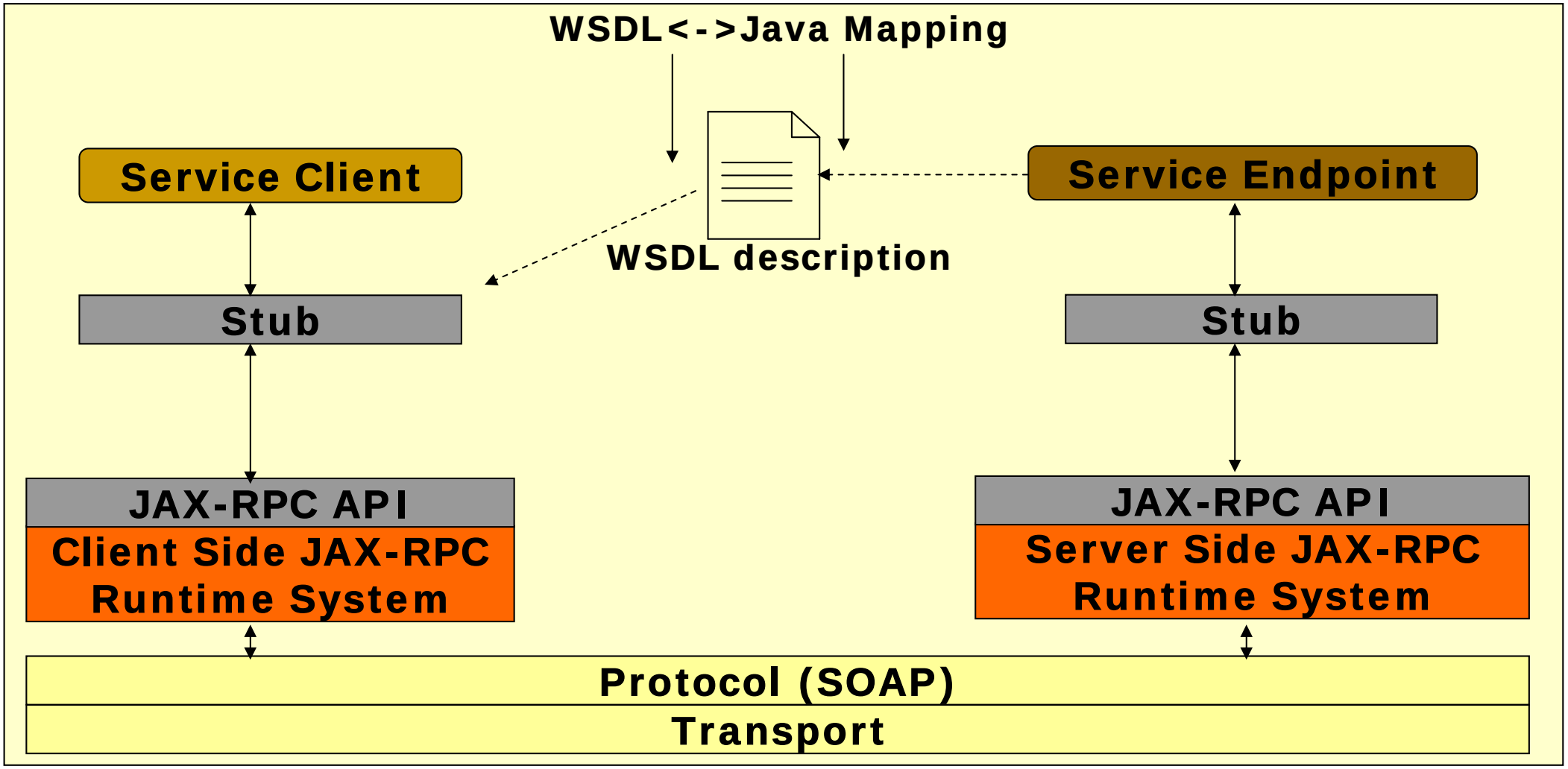
Generierung von Proxys aus WSDL

Orientation in Objects



5.8 Session Beans als WebServices

Implementierung mit Axis



Installation von Apache AXIS

Voraussetzungen**:**
- J2SE SDK >= 1.4
- Servlet Container: z.B. Tomcat

Installation:
- AXIS herunterladen von http://ws.apache.org/axis
- Archiv auspacken und deployen
  Kopieren von *webapps/axis* in das *webapps* Verzeichnis von Tomcat
- Tomcat starten: *bin/startup*

5.8 Session Beans als WebServices
Implementierung

Funktionalität einer POJO (Plain Old Java Object) als Web-Service bereitstellen

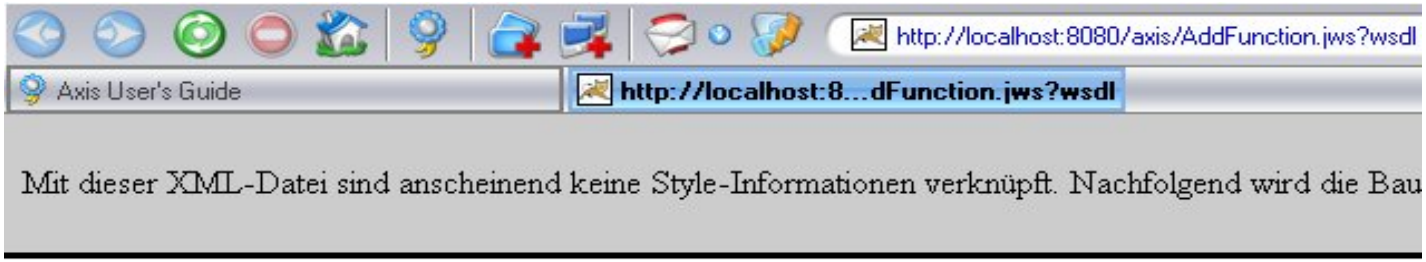
Datei *AddFunction.java*

```
public class AddFunction {

  public int addIntegers(int a, int b){
    return(a+b);
  }

}
```

- Datei umbenennen in *Addfunction.jws*
- Kopieren von *AddFunction.jws* nach *webapps/axis*
- WSDL ansehen unter: http://localhost:8080/axis/AddFunction.jws?wsdl

5.8 Session Beans als WebServices
Implementierung

WSDL-Datei herunterladen (http://localhost:8080/axis/AddFunction.jws?wsdl)




```
- <wsdl:definitions targetNamespace="http://localhost:8080/axis/AddFunction.jws">
  - <!--
      WSDL created by Apache Axis version: 1.3
      Built on Oct 05, 2005 (05:23:37 EDT)
    -->
  - <wsdl:message name="addFunctionRequest">
      <wsdl:part name="a" type="xsd:int"/>
      <wsdl:part name="b" type="xsd:int"/>
    </wsdl:message>
  - <wsdl:message name="addFunctionResponse">
      <wsdl:part name="addFunctionReturn" type="xsd:int"/>
    </wsdl:message>
  - <wsdl:portType name="AddFunction">
    - <wsdl:operation name="addFunction" parameterOrder="a b">
        <wsdl:input message="impl:addFunctionRequest" name="addFunctionRequest"/>
        <wsdl:output message="impl:addFunctionResponse" name="addFunctionResponse"/>
      </wsdl:operation>
    </wsdl:portType>
```

5.8 Session Beans als WebServices
Implementierung

Generierte WSDL Datei: Ausgetauschte Nachrichten im SOAP

Aufruf

```
<wsdl:message name="addFunctionRequest">
   <wsdl:part name="a" type="xsd:int"/>
   <wsdl:part name="b" type="xsd:int"/>
</wsdl:message>
```

Antwort

```
<wsdl:message name="addFunctionResponse">
        <wsdl:part name="addFunctionReturn" type="xsd:int"/>
</wsdl:message>
```

Generierte WSDL Datei: Methoden

- Beschreibung der verfügbaren Methoden
- Beschreibung der Parameter und Rückgabetypen

```
<wsdl:portType name="AddFunction">
   <wsdl:operation name="addFunction" parameterOrder="a b">
         <wsdl:input message="impl:addFunctionRequest"
         name="addFunctionRequest"/>
         <wsdl:output message="impl:addFunctionResponse"
         name="addFunctionResponse"/>
   </wsdl:operation>
</wsdl:portType>
```

Generierte WSDL Datei: Protokolle und Kodierung

- Welches Protokoll wird verwendet?
- Welche Kodierung wird verwendet (wird pro Methode angegeben)?

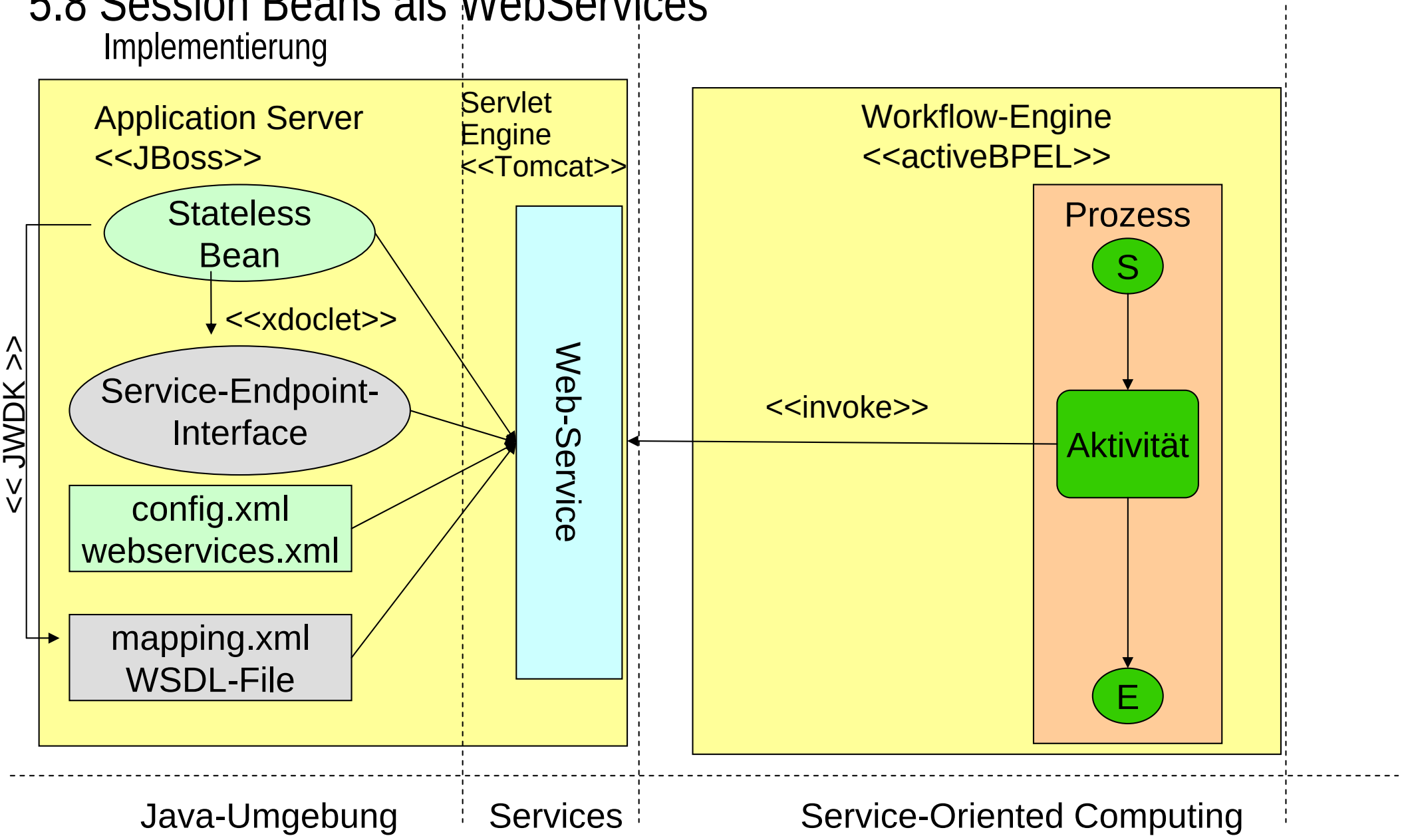
```
<wsdl:binding name="AddFunctionSoapBinding"
                    type="impl:AddFunction">

   <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
                         style="rpc"/>

   <!–- Hier noch weitere Bindings pro Operation -->
…
</binding>
```

Generierte WSDL Datei: URL zum Web-Service

```
<wsdl:service name="AddFunctionService“>
   <wsdl:port binding="impl:AddFunctionSoapBinding“ name="AddFunction">
         <wsdlsoap:address
         location="http://localhost:8080/axis/AddFunction.jws"/>
   </wsdl:port>
</wsdl:service>
```

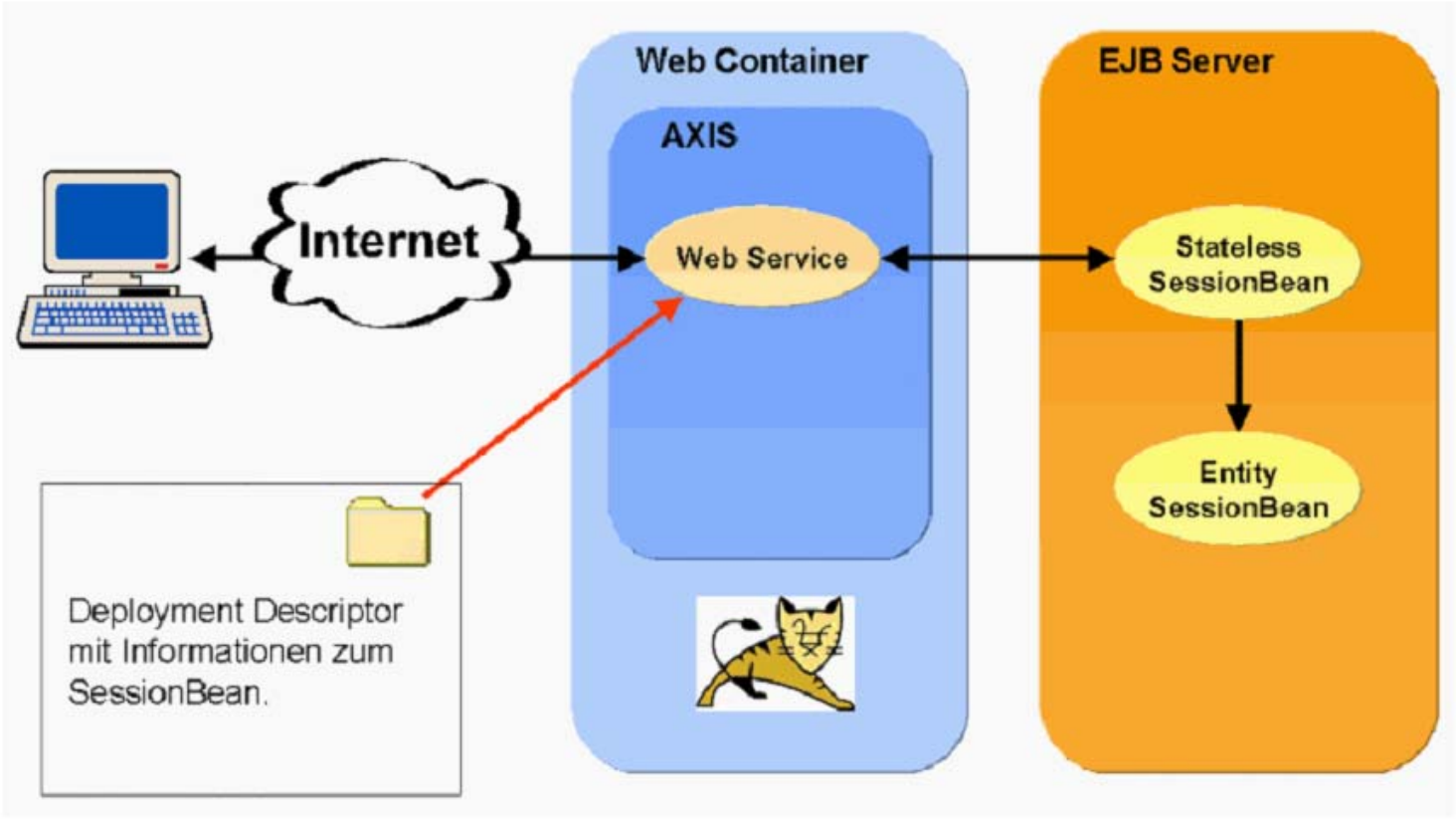
5.8 Session Beans als WebServices

Implementierung



Enterprise JavaBeans als Web Service

Orientation in Objects



5.8 Session Beans als WebServices
Implementierung

```
/**
 * @ejb.bean name="BusinessLogicWS" display name="BusinessLogicWS"
 *  type="Stateless" description="WS Endpoint for BusinessLogic"
 *  jndi-name="ejb/BusinessLogicWSGrpX" view-type="service-endpoint"
 */
public class BusinessLogicWSBean implements SessionBean {
…
    /**
    * @ejb.interface-method view-type = "service-endpoint"
    */
    public int additionWS(int a, int b) throws RemoteException {
            return a + b;
    }
…
}
```

xDoclet generiertes Service-Endpoint-Interface

```
public interface BusinessLogicWS extends java.rmi.Remote {
    public int additionWS( int a,int b )    throws    java.rmi.RemoteException;
}
```

5.8 Session Beans als WebServices
Implementierung

JAX-WS (Java API für XML Web Services)

- JAX-RPC erfordert verschiedene Artefakte (WSDL-Datei, JAX-RPC Mapping-Datei, Endpoint Interface Datei, Deployment-Deskriptor: webservice.xml), um eine Stateless Session Bean als WebService bereit zu stellen.
- Obwohl die meisten Dateien automatisch erzeugt werden, ist die Implementierung mittels JAX-RPC doch recht komplex.
- JAX-WS vereinfacht die Implementierung und Publizierung eines WebServices mittels verschiedener Annotationen.

5.8 Session Beans als WebServices
Implementierung

JAX-WS (Java API für XML Web Services)

```
import javax.ejb.Stateless;
import javax.jws.WebService;
import javax.jws.WebMethod;

@Stateless
@WebService
Public class additionBean
{
    @WebMethod
    public int additionWS(int a, int b) {
        return a + b;
    }
}
```

Beispiel

Durch verschiedene Annotationen können die WS-Operatioen definiert werden, wie z. B.

@WebService

@WebMethod
(alle Methoden einer Klasse oder nur die Ausgezeichneten)

@WebParam
(Mode: IN, OUT, INOUT, …)

@WebResult
(Name des Return-Wertes)

@SOAPBinding
(Style: RPC, Use: Literal | Encoded, ..)