Vorlesung
Software aus Komponenten

3. Komponenten-Modelle

Prof. Dr. Hans-Gert Gräbe / F. Schumacher
Wintersemester 2005/06

Über RPC-Mechanismus hinaus gehende Fragen, die ein objektorientiertes Kommunikationskonzept beantworten muss

1. Wie werden Schnittstellen spezifiziert?
2. Wie werden Objektreferenzen behandelt, wenn der lokale Bereich verlassen wird?
3. Wie werden Dienste aufgefunden und bereitgestellt?
4. Wie wird die Evolution von Komponenten gehandhabt? (Versionsmanagement)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

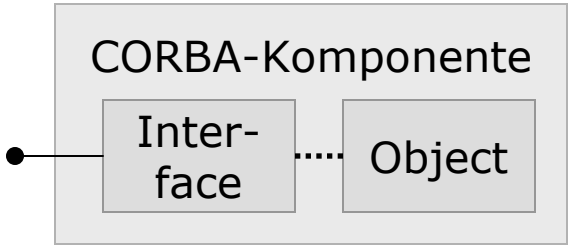
Schnittstellenspezifikation und Objekte

- **Definition:** Interface ist ein abstrakter Datentyp
  - Sammlung von Operationsbezeichnern mit ihren Signaturen
  - Signatur = Typ und Aufrufmodi der Parameter

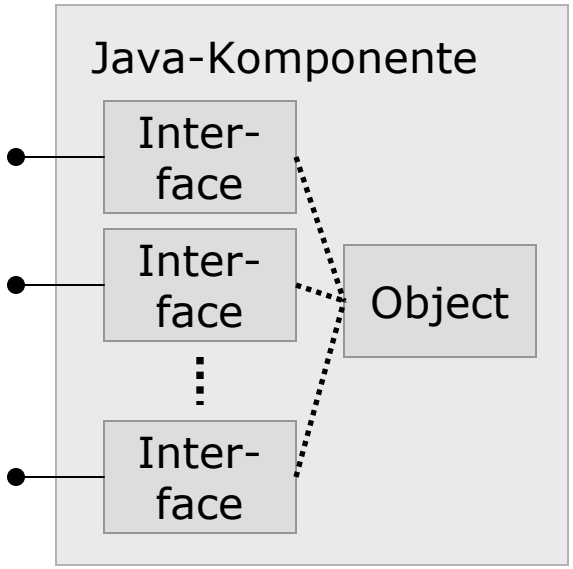
- **Schnittstellen-Beschreibung** durch IDL
  - mehrere Standards koexistieren (insb. OMG IDL und COM IDL)
  - Java und CLR: Keine IDL, sondern sprachspezifisches Meta-Datenformat, das auf jede der IDL abgebildet werden kann
    - dazu sind entsprechende Abbildungen zu spezifizieren
      - *Java to IDL language mapping specification* (Version 1.3 vom Sept. 2003, siehe http://www.omg.org)
    - **Umgekehrt:** Java-Werkzeug **idlj** erzeugt aus einer (OMG) IDL-Beschreibung (u.a.) ein Java *interface*.

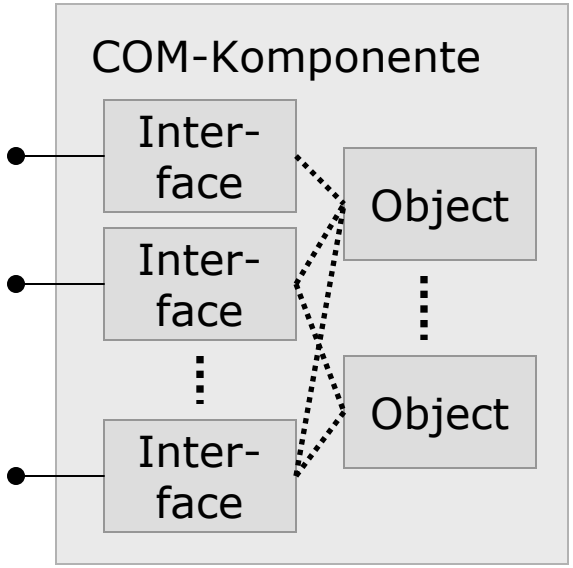
Bindung von Schnittstellen an Objekte

- Drei wesentlich verschiedene Ansätze:
  - 1 O ⇔ 1 S (CORBA 2, SOM)
    - Objekt speichert Programmzustand und Implementierung **seiner** Schnittstelle
    - Schnittstelle kann durch Mehrfachvererbung entstanden sein



- 1 O ⇔ * S (Java, CLR)
- * O ⇔ * S (COM)
  - Problem der Bindung von Schnittstellen an Objekte
  - Identität des Objekts muss geklärt werden





Schnittstellen und Vererbung

- mehrere Implementierungen derselben Schnittstelle möglich
- Implementierung kann mehr Funktionalität bereit stellen als durch die Schnittstelle definiert
- CORBA: Traditioneller Objektansatz
  - Jede Komponente (=Objekt) hat nur ein Interface
  - Mehrfachvererbung möglich
  - Erwartete Schnittstelle darf Subtyp der bereitgestellten sein
    - Zusatzeigenschaften können dynamisch herausgefunden werden
- COM: Komponente hat mehrere Schnittstellen in einer Liste
  - Schnittstellen bedienen mehrere Objekte
  - Interface unveränderbar
    - einmal veröffentlicht -- weder erweiter- noch änderbar
    - aber Schnittstellenliste kann dynamisch erweitert werden
  - einfache Schnittstellenvererbung möglich

Schnittstellen und Vererbung

- Java: Objekte können mehrere Schnittstellen implementieren, aber stärker am Vererbungskonzept orientiert
  - Default-Implementierungen von Interfaces durch abstrakte Klassen
  - Klasse kann mehrere Interfaces implementieren, aber nur von einer abstrakten Klasse erben
- Problem der Namenskollision, wenn Methoden aus unterschiedlichen Schnittstellen denselben Namen haben
  - Java: Überladen und Überschreiben
    - qualifizierte Namensgebung ist möglich
  - COM und CLR: unterschiedliche Schnittstellen sind unterschiedliche Namensräume

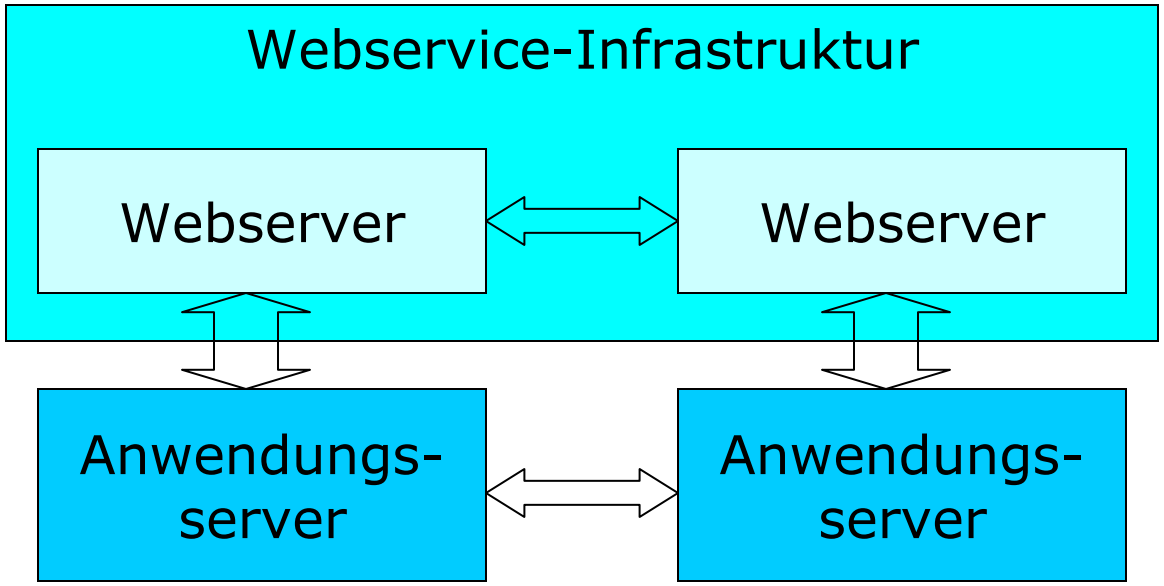
Namensgebung und Auffinden von Diensten

- Dienste werden über ihren Namen identifiziert
  - OMG: UUID als Standard der Open Software Foundation (DCE)
    - genügend lange Zeichenkombinationen
  - COM (Microsoft) verwendet modifizierte Version: Global Unique IDentifier (GUID)
    - Namensgebung für Interfaces (IID), Gruppen von Interfaces (categories = CATID) und Klassen (CLSID)
    - CLR: Identität durch private / public-key auf Komponentenebene
  - Java: Eindeutigkeit über zusammengesetzte Pfadnamen (Anlehnung an URL)

- Über den Namen muss wenigstens folgende Funktionalität zur Laufzeit abrufbar sein:
  - Typtest der Schnittstellen
  - Introspektion der Schnittstellen
  - dynamisches Erzeugen neuer Objekte

Webservices als Komponententechnologie

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

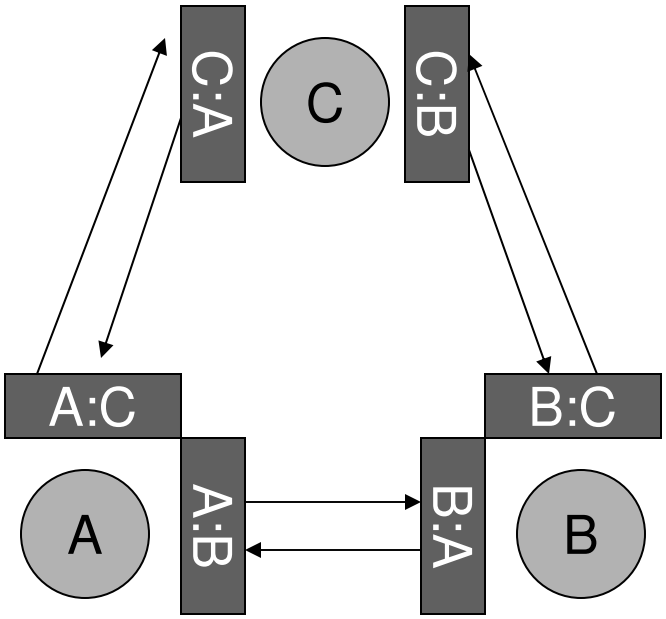


Webservices

klassische Komponenten-technologie

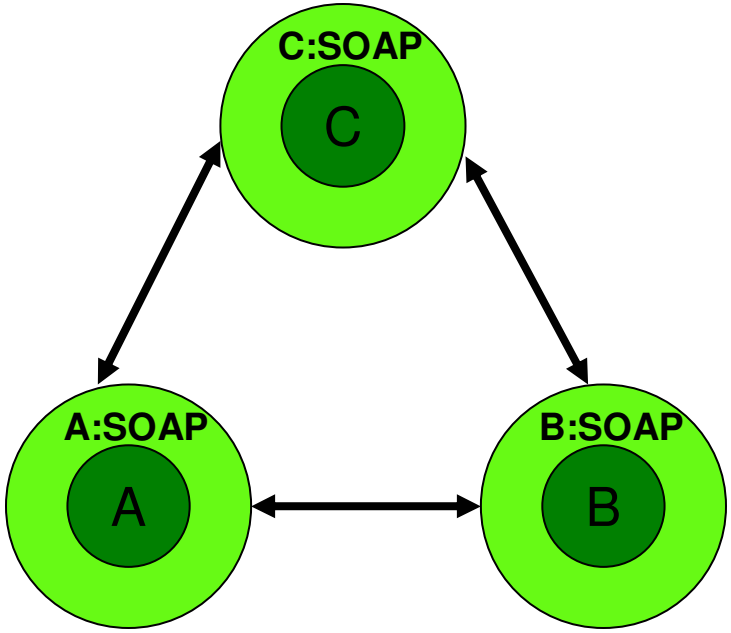
Webservices als EAI-Ansatz

Interface-Strukturen ohne Webservices



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

Interface-Strukturen mit Webservices



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

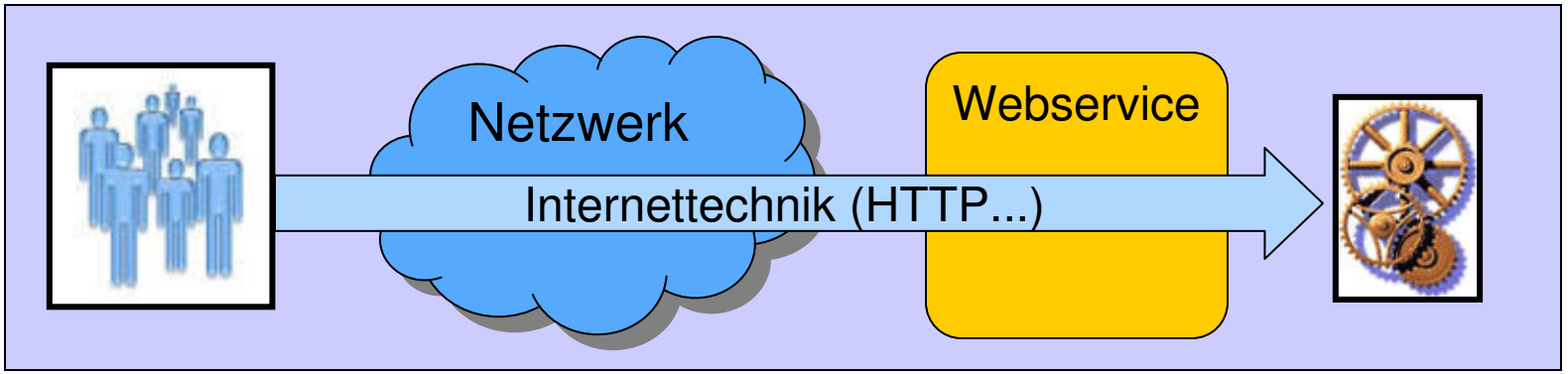
Einsatzziele für Webservices

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

Definition „Webservice"

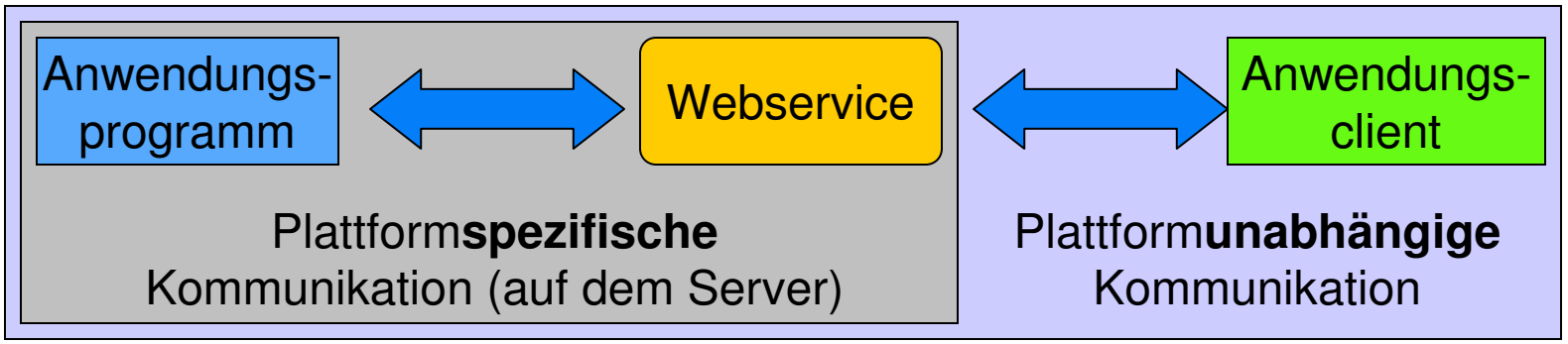
Ein Web-Service ist eine über ein Netzwerk zugängliche nachrichten-basiert gesteuerte Schnittstelle zu Anwendungsfunktionen:

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



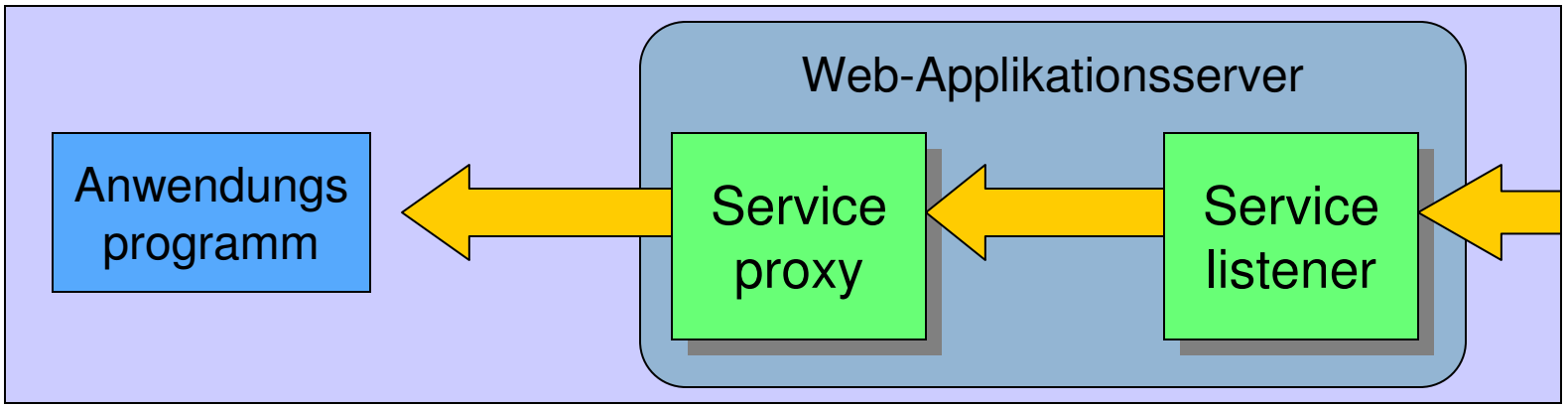
Definition „Webservice"

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



Aufbau von Webservices

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

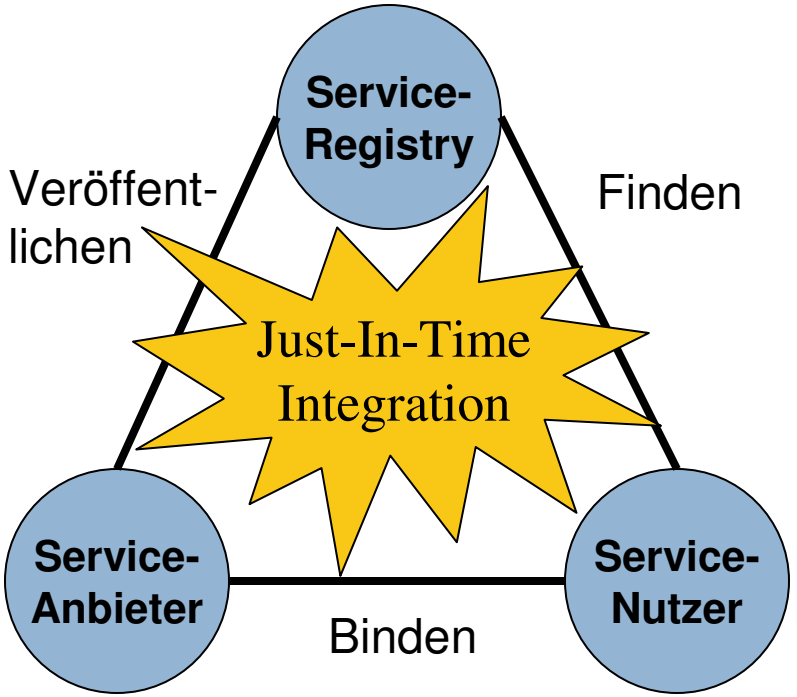


Aufbau von Webservices (2)

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

Webservice-Architektur

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



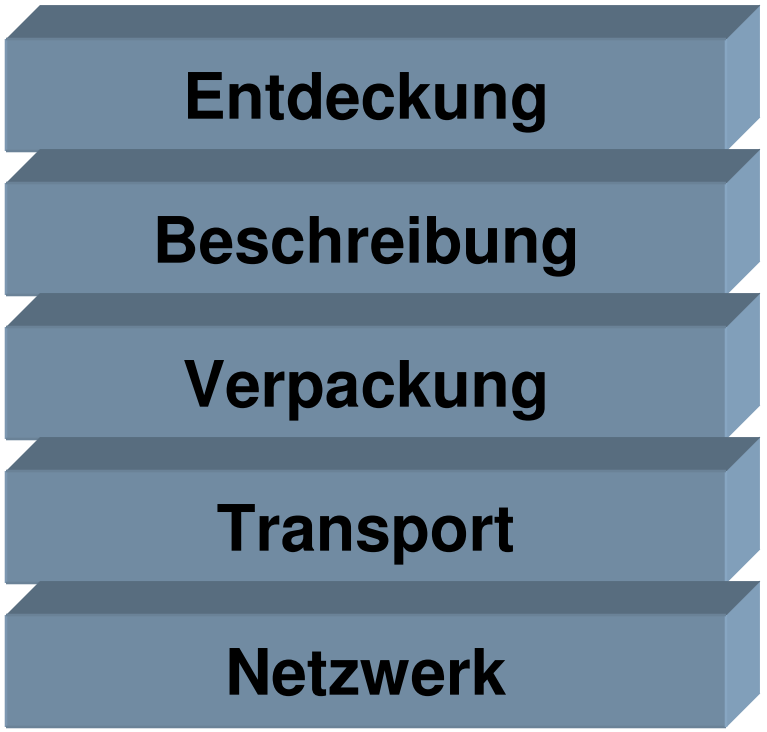
Finden:
Der Service-Nutzer (Mensch, Programm) sucht in der Service-Registry nach einem Service

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

Schichtenmodell der Webservices

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

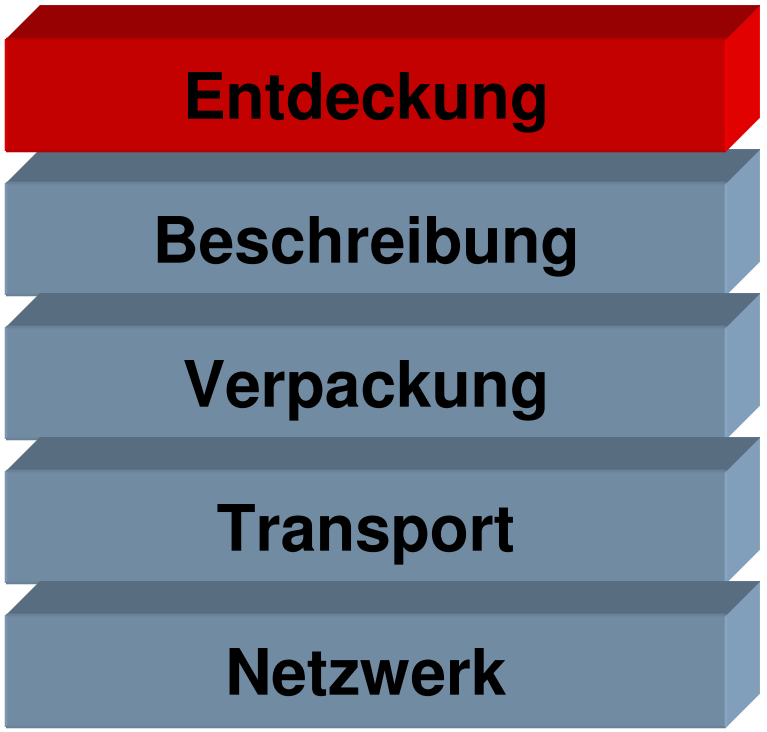
  ermöglichen die **Unabhängigkeit** von der jeweiligen Plattform (wie ISO/OSI-Schichtenmodell)
- Die Schichten decken unabhängige Sachverhalte ab, so dass der Entwickler wählen kann, welche Schichten er implementiert
- Schichten dienen der Modularisierung



Schichtenmodell der Webservices (2)

Entdeckungsschicht

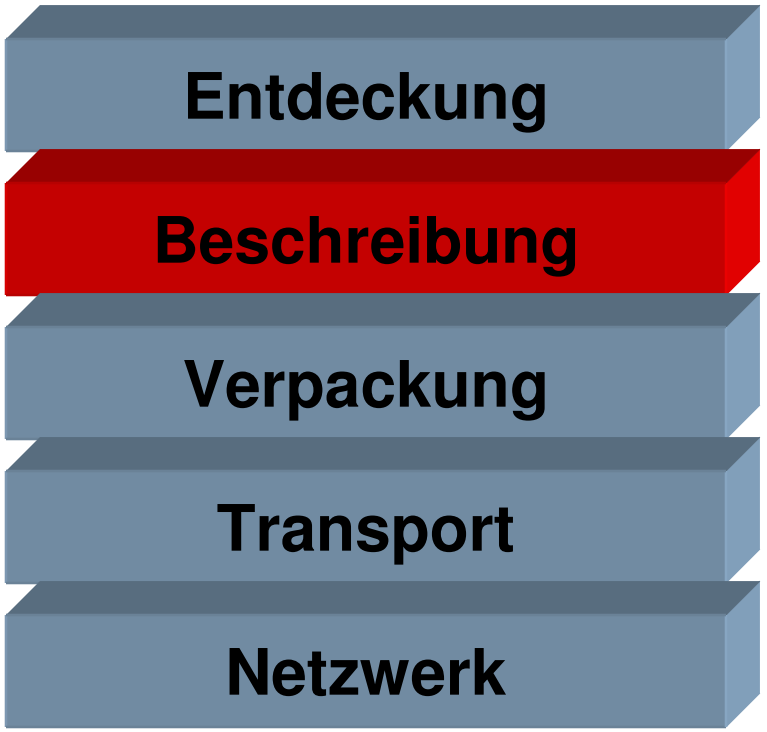
- Mechanismen für Service-Nutzer, um die Beschreibung über den Dienst und den Service-Anbieter zu bekommen
- Ausprägungen:
  - UDDI
  - WS-Inspection



Schichtenmodell der Webservices (3)

Beschreibungsschicht

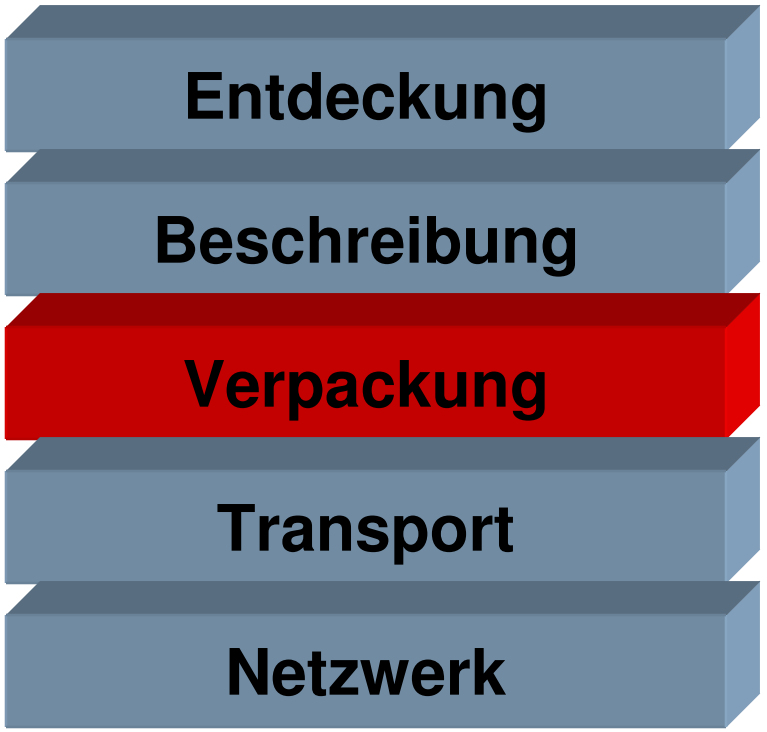
- Bereitstellung von Informationen über die Protokolle für Netzwerk, Transport und Verpackung
- Hilft dem Service-Nutzer, den Webservice zu kontaktieren und zu verwenden
- **Ausprägungen:**
  - WSDL (Web Service Description Language)
  - RDF (Resource Description Framework)
  - DAML (DARPA Agent Markup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



Schichtenmodell der Webservices (4)

Verpackungsschicht

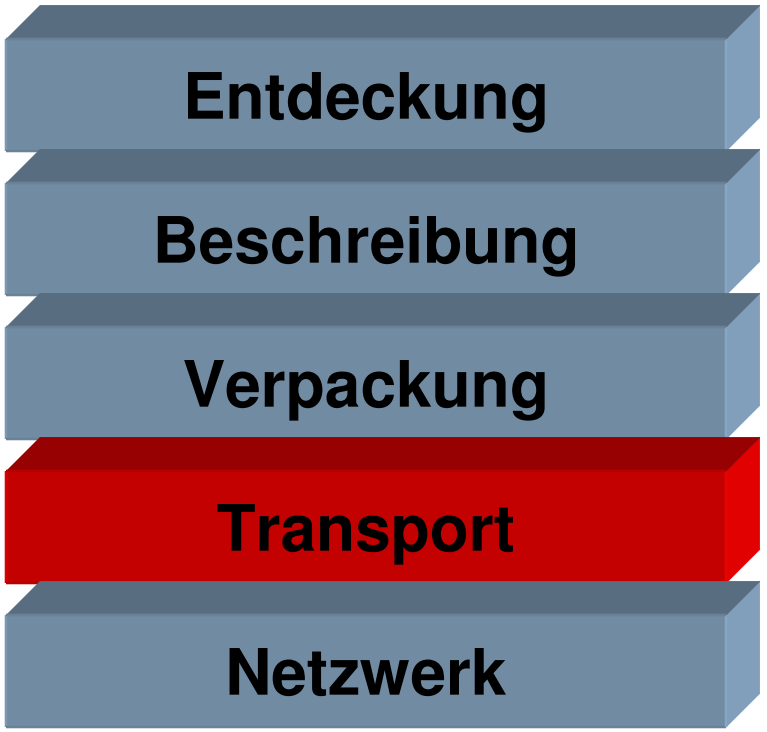
- Verpackt die Anwendungsdaten in XML, damit sie über die Transportschicht übertragen werden können (Serialisierung)
- Ausprägungen:
  - SOAP
    (Simple Object Access Protocol)
  - XML-RPC



Schichtenmodell der Webservices (5)

Transportschicht

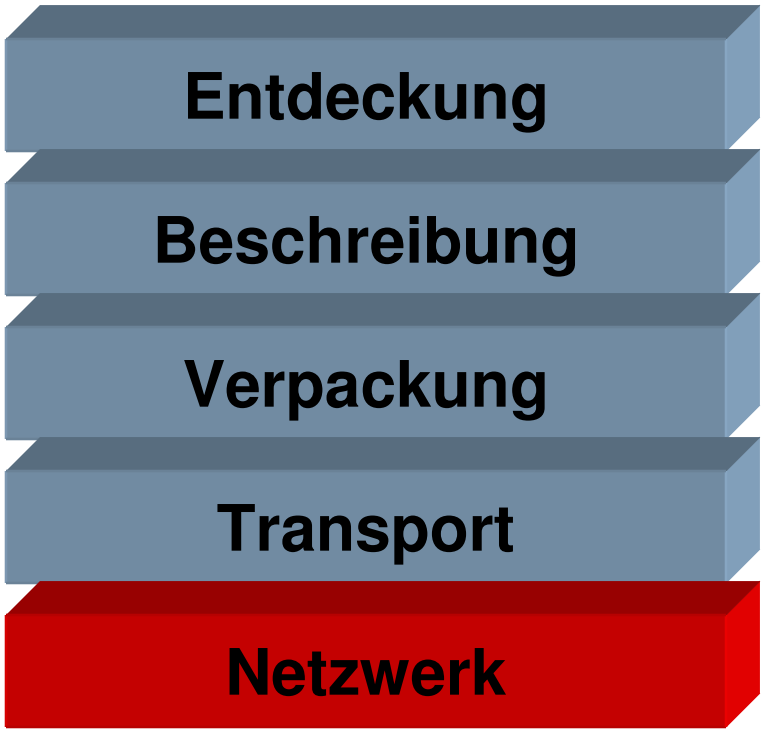
- Ermöglicht auf Basis der Netzwerkschicht die Kommunikation der Anwendungen
- Verwendete Techniken:
  - TCP
  - HTTP
  - SMTP
- Webservices können auf beliebigen Transportprotokollen aufsetzen
- HTTP ist am meisten verbreitet



Schichtenmodell der Webservices (5)

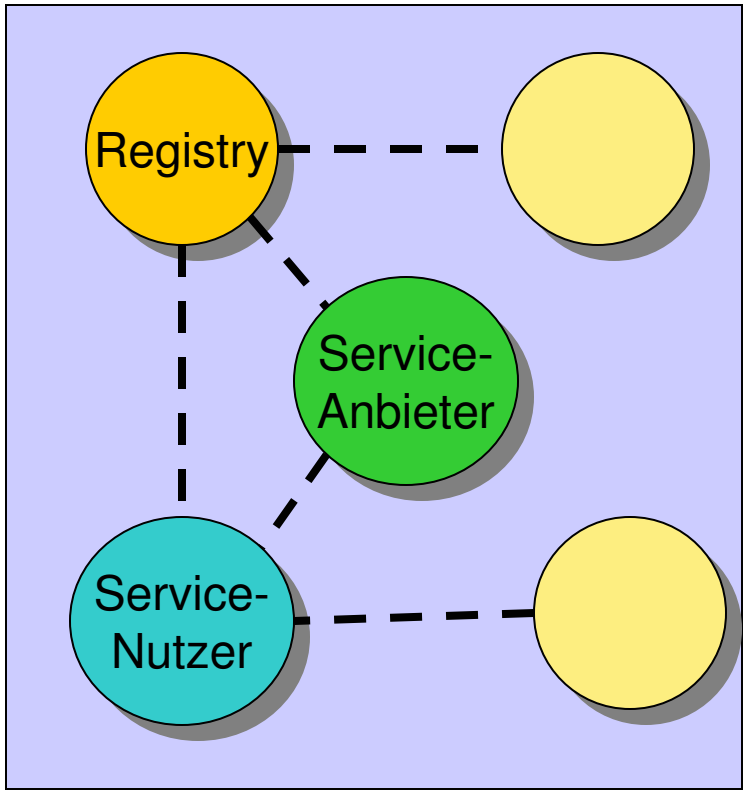
Netzwerkschicht

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

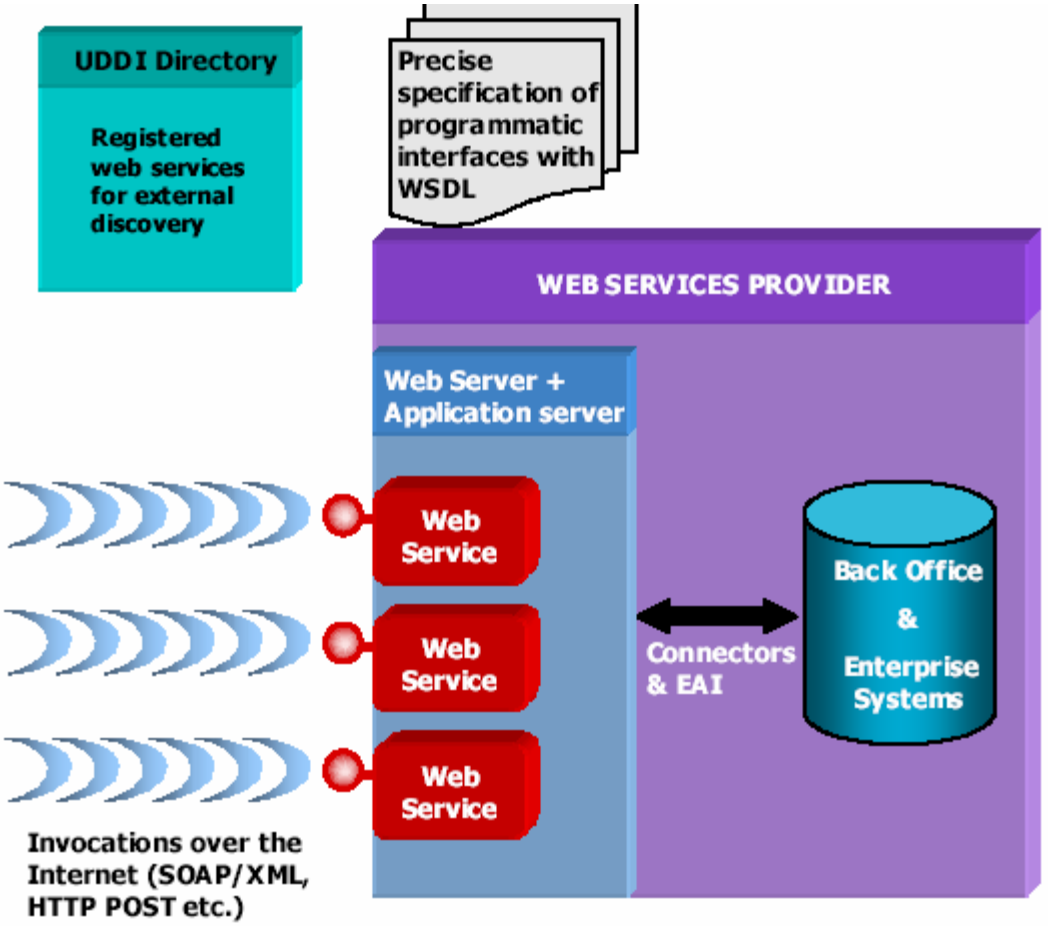


Das Peer-Service-Modell

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



Überblick über Webservice-Komponenten



XML-Protokolle

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