

# Einführung in WebServices

Grundlagen und Praxis von WebServices

Seminarleiterin: Dipl.-Ing. Mahbouba Gharbi

# Zielsetzung und Voraussetzungen

---

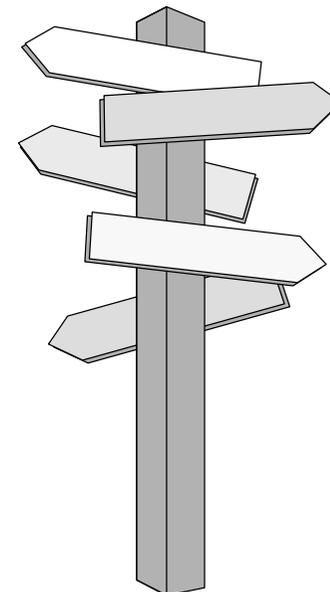


- Zielsetzung
  - Nutzen von WebServices kennenlernen
  - Grundbegriffe der WebServices Architektur verstehen
  - Die Kommunikation für die Web Service Nutzung praktisch gesehen haben
- Voraussetzungen
  - Grundlagen Software-Architekturen (Client/Server, Middleware)
  - XML Basiswissen
  - Java Basiswissen (um das Beispiel zu verstehen)

# Übersicht

---

- Grundlagen WebServices
- Beschreiben von WebServices mit WSDL
- Kommunikation zu WebServices mit SOAP
- Suchen und Finden von WebServices mit UDDI
- WebServices in der Praxis



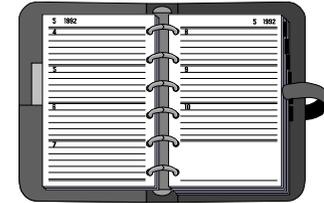
# Grundlagen WebServices

Was sind WebServices?

Welche Probleme sollen WebServices  
lösen?

Welche Vorteile versprechen  
WebServices?

# Themen



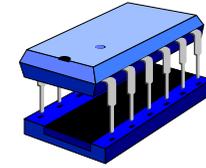
- ✘ Einführung WebServices
  - Definition
  - Diskussion WebServices: Ziele, Vorteile, Probleme
- ✘ Exkurs: Grundlagen Architektur
- ✘ Struktur von WebServices
  - Rollenmodell: Provider, Requestor, Broker
  - Architektur: WSDL, SOAP und UDDI

# Anforderungen an Software-Systeme

---

- Alles ist Internet!
  - e-Commerce als strategischer Erfolgsfaktor
- Hohe Verfügbarkeit
- Hoher Durchsatz, Skalierbarkeit, Datenschutz
- Kurze Time-To-Market
  - Software-Systeme müssen „agiler“ und besser wartbar werden
- Integrierbarkeit!

# Software-Komponenten



- Austauschbare Software-Bausteine
  - Definierte Schnittstellen für die Bereitstellung der Verantwortlichkeit
- Wünsche
  - Effizientere Software-Erstellung durch Wiederverwendung eigener und eingekaufter Komponenten
  - Industrielle Software-Produktion statt Manufaktur



# Was sind WeBservices?

---

**Komponente**, die ihre Funktionalität über eine **veröffentlichte Schnittstelle** anbietet und über ein **offenes Protokoll im Internet** zugreifbar ist

- Verwendet ausschließlich die Internetstandards XML und HTTP
- Bausteine:
  - WSDL (Web Service Definition Language)
  - UDDI (Universal Service Description, Discovery, and Integration)
  - SOAP (Simple Object Access Protocol)

# Anwendungsbeispiele

---

-  **amazon.de**
  - Ermitteln strukturierter Daten (Produktname, Hersteller, Preis etc.) über Produkte von Amazon, die auf Parametern wie Schlagwortsuchbegriffen und Browse-Tree-Nodes basieren.
-  **Google**
  - Integration der Google-Suchmaschine in eigene Anwendungen.

# SOA: Service Oriented Architecture

---

- Software Architektur, bei der die grundlegende Infrastruktur/Funktionalität in Form von Diensten organisiert ist
  - Geschäftsprozesse werden dynamisch aus Diensten erstellt
  - Definierte Schnittstellen der Dienste
- Noch nicht standardisiert, existiert bisher nur als Idee
- WebServices als konkrete Implementierung einer SOA

# Vorteile WebServices

---

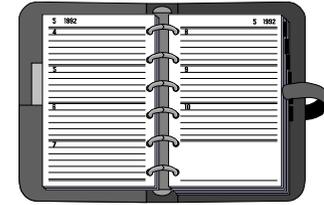
- Alle Vorteile komponentenbasierter Architekturen wie CORBA, J2EE oder .NET
  - Unabhängigkeit von Plattformen, Systemen, Sprachen
  - Flexibilität, evolutionäres Vorgehen der Entwicklung
  - Skalierbarkeit, Wiederverwendbarkeit, Integration, ...
- + Hohe Akzeptanz und Verbreitung
  - basiert auf im Internet etablierten Standards (HTTP, XML)
  - einfach (leichtgewichtige Kommunikation)

# Nachteile/Probleme von WebServices

---

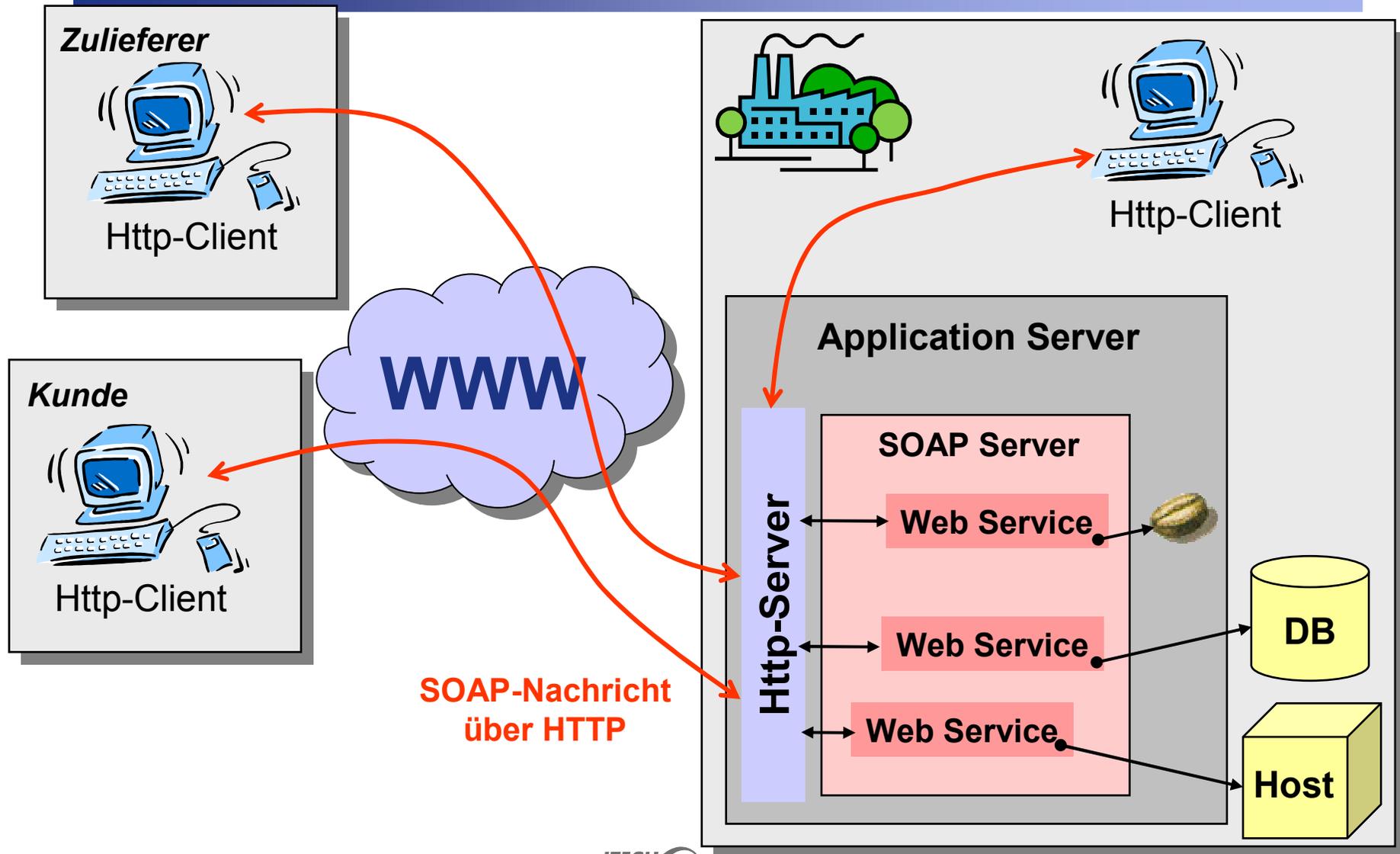
- Offene Punkte/Fehlende Funktionalität
  - Sicherheit von Diensten
  - Transaktionskontrolle
  - Prozess-Steuerung
- Nur wenige Dienste bisher verfügbar

# Themen



- × Einführung WebServices
  - Definition
  - Diskussion WebServices: Ziele, Vorteile, Probleme
- × Exkurs: Grundlagen Architektur
- × Struktur von WebServices
  - Rollenmodell: Provider, Requestor, Broker
  - Architektur: WSDL, SOAP und UDDI

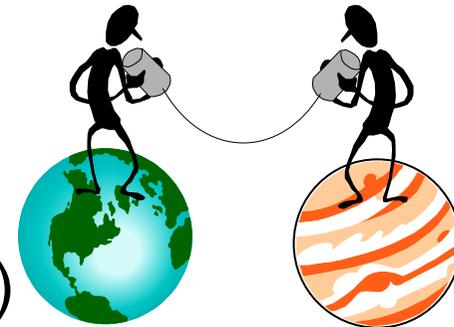
# Architektur-Szenario



# HyperText Transfer Protocol: HTTP

---

- Übertragungsprotokoll
  - Zustandslos
  - TCP/IP-basiert (Anwendungsschicht)
- Grundlage des WWW neben URLs und HTML
  - Protokoll, um entfernte Dateien (HTML-Seiten, Bilder, MP3, PDF-Dateien usw.) auf den eigenen Rechner zu übertragen
- Kommunikationsmethoden:
  - GET, POST, HEAD, PUT, DELETE
- Versionen: 1.0, 1.1



# Extensible Markup Language: XML

---



- Datenbeschreibungssprache
  - Macht Daten portierbar
  - Metasprache
  - Submenge von SGML
  - Nicht nur maschinenlesbar, sondern auch für Menschen
- Grundidee
  - Trennung von Daten und ihrer Darstellung
- Formatbeschreibung
  - Dokumenttypdefinition (DTD)
  - XML Schema

# Applikations-Server: Standards

---

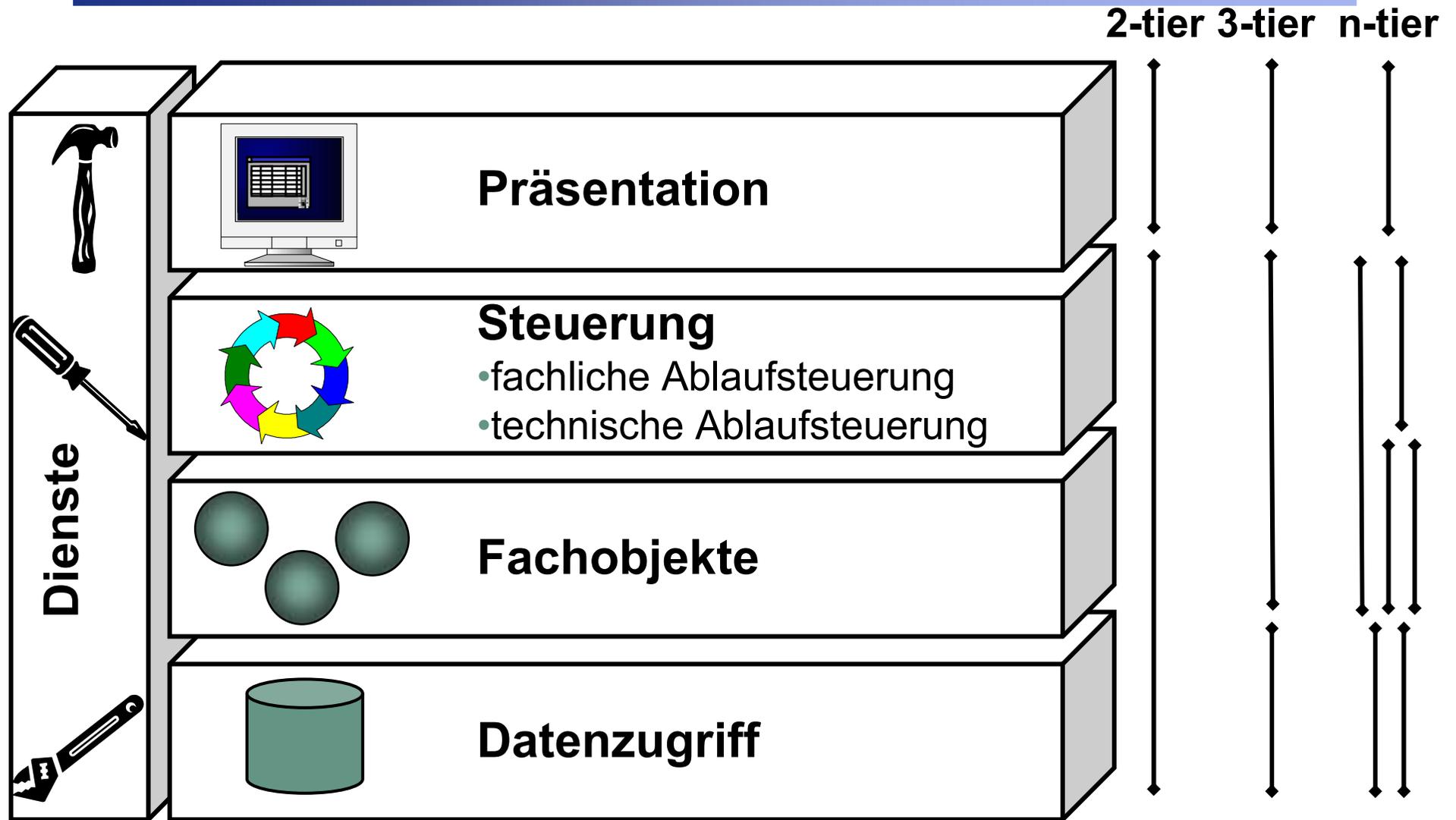
- CORBA  CORBA
  - Offener Industriestandard, viele Sprachen und Plattformen
- Sun Java 2 Enterprise Edition (J2EE) 
  - Standardisierung durch Sun Microsystems kontrolliert, viele Plattformen, nur Java („anderssprachige“ Clients inzwischen möglich)
- Microsoft COM+/.NET 
  - Industriestandard von Microsoft, nur Windows-Plattformen, mehrsprachig, sehr gute Tool-Unterstützung

# Enterprise Application Integration (EAI)

---

- Enterprise Application Integration (EAI) ist ein Markt für vorgefertigte Adapter, welche Protokolle und/oder Daten konvertieren
- Beispiele:
  - Unix zur IBM SNA-Welt
  - Externer Zugriff auf SAP-Module
  - Datenkonvertierung zwischen verschiedenen Anwendungen
- XML als eine der Kern-Technologien

# Schichtenmodell



# Schichtenmodell

---

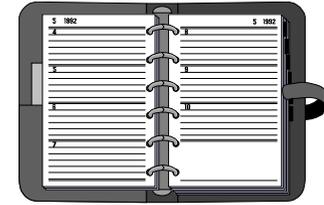
- Aufteilung der Anwendung in mehrere logische Ebenen
  - Präsentation
  - Anwendungslogik
  - Speicherung
- Vorteil
  - „Design for Change“: einzelne Ebenen voneinander unabhängig
  - Innerhalb einer Schicht Freiheit für den Designer
- Nachteil
  - Einzelne Ebenen immer noch sehr komplex => Komponenten
  - Höherer Kommunikationsaufwand

# WebServices Einsatzmöglichkeiten

---

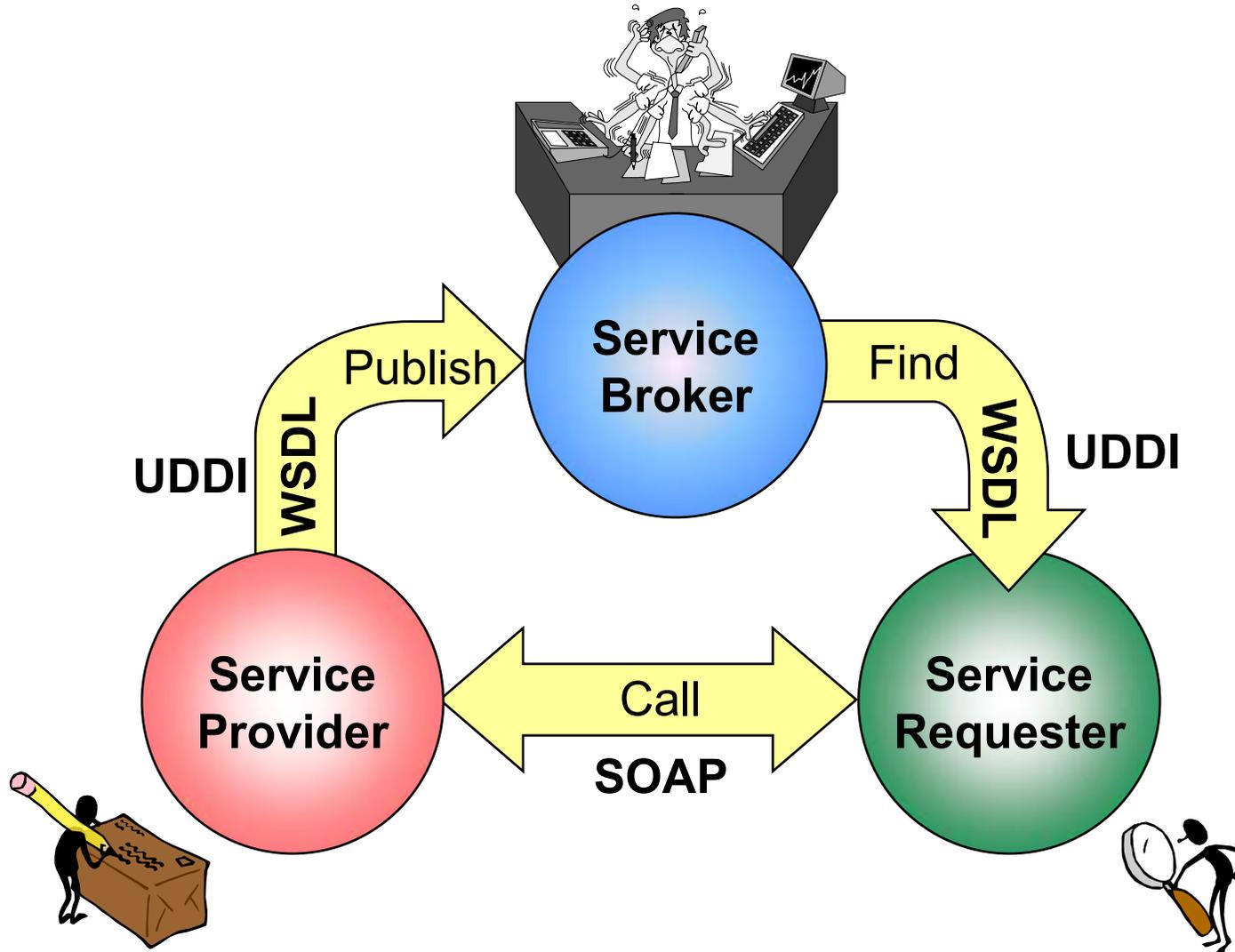
- Vertikal: Zwischen einzelnen Schichten
  - Zugriff auf Präsentationslogik
  - Zugriff auf Geschäftslogik (Legacy, Host)
  - Zugriff auf Daten (DB, Host)
- Horizontal: Zwischen Anwendungen
  - Innerhalb eines Unternehmens
  - Zwischen Unternehmen (B2B, B2C)

# Themen



- ✘ Einführung WebServices
  - Definition
  - Diskussion WebServices: Ziele, Vorteile, Probleme
- ✘ Exkurs: Gr©undlagen Architektur
- ✘ Struktur von WebServices
  - Rollenmodell: Provider, Requestor, Broker
  - Architektur: WSDL, SOAP und UDDI

# Struktur von WebServices



# WebServices Rollen

---

- Service Provider

- Dienstleister, der sein Angebot beim Broker bekannt macht



- Service Requester

- Sucht einen Dienst

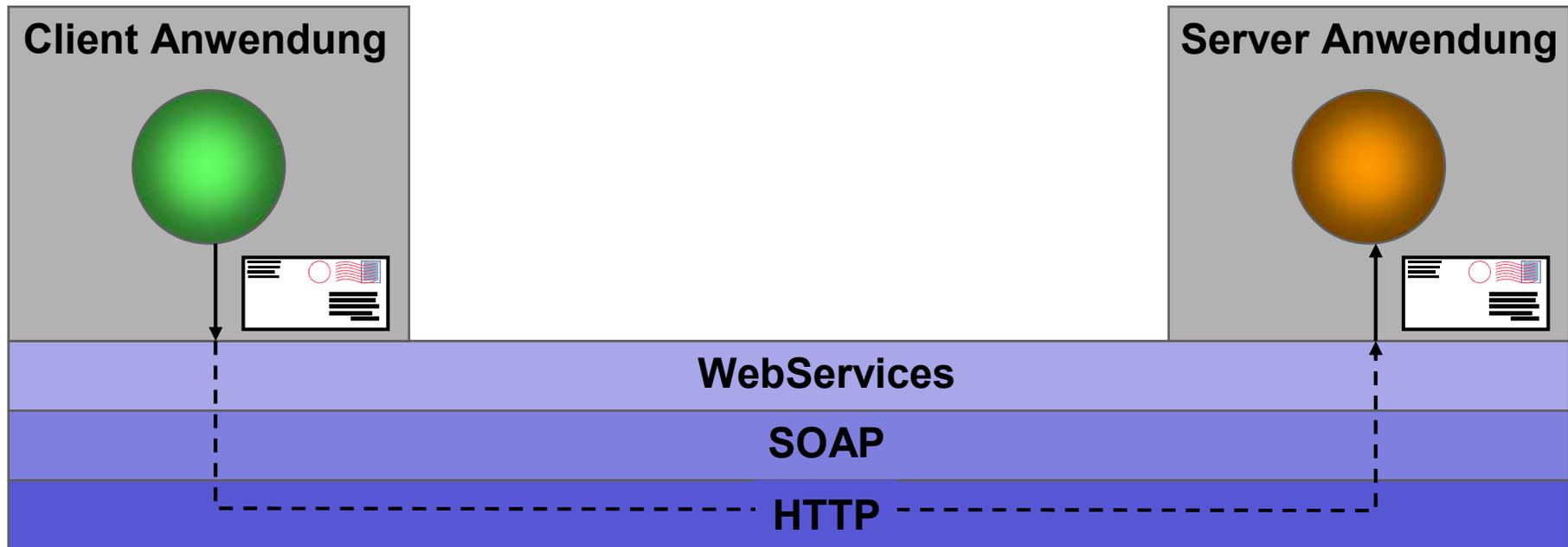


- Service Broker / Service Registry

- Registriert Dienstleistungen
- Führt eine Echtzeitsuche nach Diensten durch und leitet Beschreibungen von Diensten an Suchende weiter



# WebServices Layer-Modell



# Web Service Description Language (WSDL)

---

- XML Dokument mit der Beschreibung eines Web Service
  - Was macht der Dienst?
  - Wie kommuniziert man mit dem Dienst?
  - Wo läuft der Dienst?
- W3C Standard
- Entkoppelt Implementierung und Interface

## WSDL

# Elemente

---

- Types
  - Datentypdefinitionen
- Message
  - Typisierte Definition der kommunizierten Daten
- Operation
  - Beschreibung einer vom Service zur Verfügung gestellten Aktion
- Port Type
  - Operationen, die von Endpunkt(en) zur Verfügung gestellt werden
- Binding
  - Konkrete Protokoll-/Datenformat-Spezifikation für einen Port-Typ
- Port
  - Endpunkt definiert als Kombination von Binding + Netz-Adresse
- Service
  - Eine Menge zusammengehöriger Endpunkte

# Universal Service Description, Discovery, and Integration (UDDI)

---

- Verzeichnisdienst für WebServices
  - Orientiert sich an Gelben Seiten und Telefonbuch
  - Übernimmt Ideen der CORBA Services Naming, Property
  - Urheber: OASIS
- Verwaltung von WebServices
  - Identifikation, Kategorie, technische Informationen
  - Mehrere Versionen, Aliase, Zugriffskontrolle
- Ist selbst ein Web Service
- Referenzierung der UDDI-Elemente über UUID

# Exkurs: Universal Unique ID (UUID)

---

- Auch: Global Unique ID (GUID)
- Eindeutige 128-bit ID
- Algorithmus für die Erzeugung
  - In OSF DCE spezifiziert
- Identifizierung von
  - UDDI-Elementen
  - CORBA-Elementen (Schnittstelle, Klasse usw.)
  - ...
- Aufbau
  - Node-ID: 48 Bit
    - 3Byte Herstellerkennung (IEEE-Adresse)
    - 3Byte Kartenkennung der Netzwerkkarte
  - Timestamp: 60 Bit
    - 100ns-Intervall seit dem 15.10.1582 00:00 Uhr
  - Clocksequence: 48 Bit
    - Verhindert Duplikate (z.B. bei Umstellung Sommer- auf Winterzeit)

# Simple Object Access Protocol (SOAP)

---

- Kommunikationsprotokoll zwischen Provider und Requester
- XML basiert
- Unabhängig vom zugrunde liegenden Transportprotokoll
  - HTTP, JMS, Mail-Transportprotokoll, usw.
- W3C Standard
  - aktuelle Version: 1.2 (1.1 noch üblich)
  - Ursprünglich von Microsoft (XML-RPC)

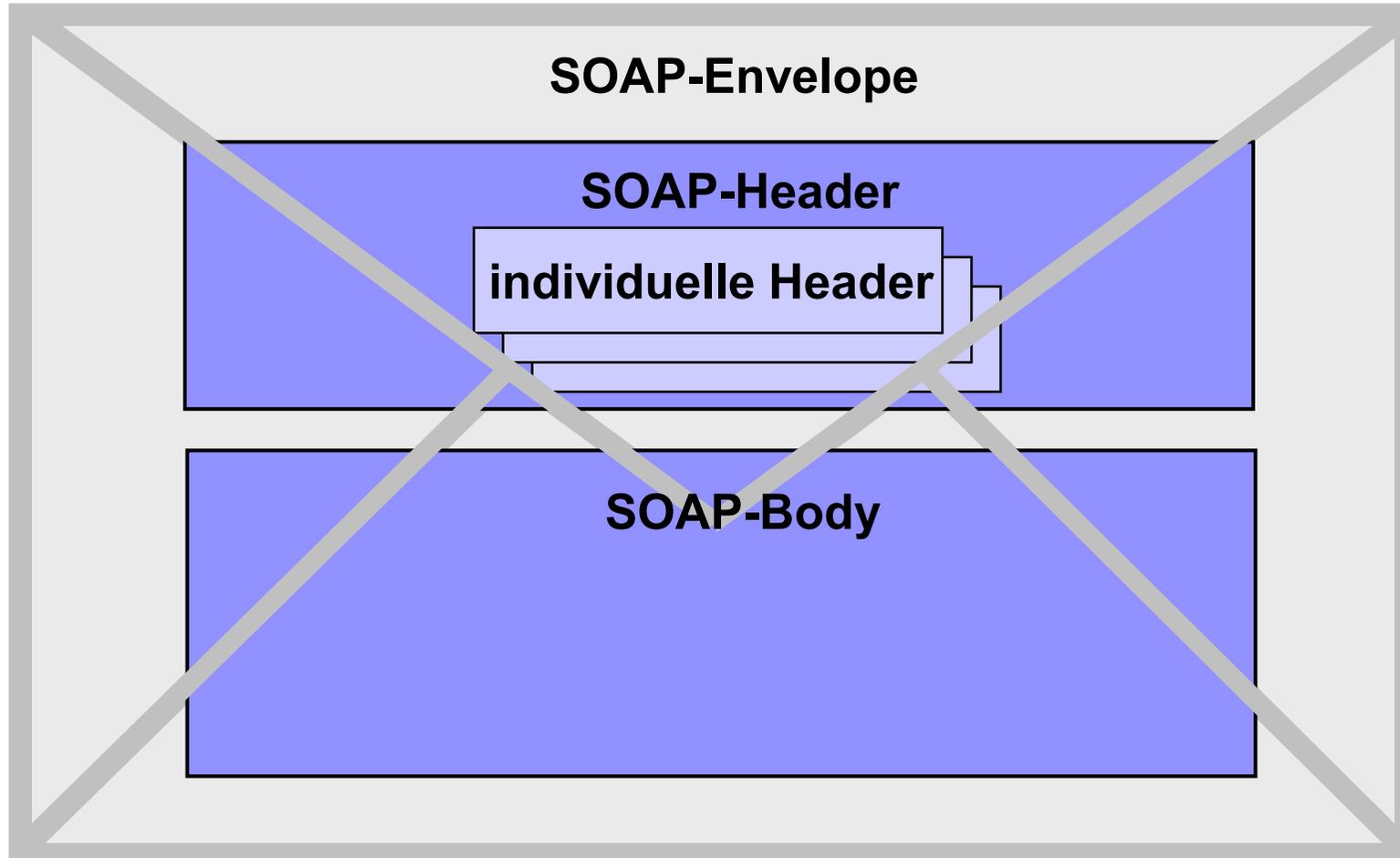
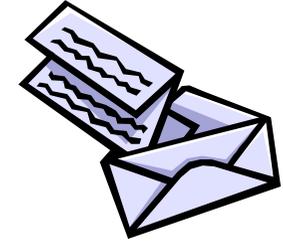
# SOAP

---

- Beschreibt den Inhalt einer Nachricht und wie diese verarbeitet wird
  - SOAP Envelope
  - SOAP Transport Binding Framework
- Leichtgewichtige Kommunikation
  - Details wie Garbage Collection, Aktivierung von Objekten usw. werden nicht berücksichtigt
- Interaktionsmuster
  - Methodenaufruf: Remote Procedure Call (RPC)
  - Dokumentenaustausch: (document/literal)

SOAP

# Aufbau einer SOAP-Nachricht



# Implementierung von WebServices

---

- J2EE
  - JWSDP (Java WebServices Developer Pack)
- .NET
- C++
- Perl

# Links

---



- ⊕ [www.webservices.org](http://www.webservices.org)
- ⊕ [java.sun.com/webservices/](http://java.sun.com/webservices/)
- ⊕ [www.w3.org/2000/xml/Group/](http://www.w3.org/2000/xml/Group/) (SOAP)
- ⊕ [www.uddi.org/](http://www.uddi.org/)
- ⊕ [www.w3.org/TR/wsdl/](http://www.w3.org/TR/wsdl/)
- ⊕ [www.omg.org/news/meetings/workshops/webservices\\_2003.htm](http://www.omg.org/news/meetings/workshops/webservices_2003.htm)
- ⊕ [www.jeckle.de/webServices/](http://www.jeckle.de/webServices/)

