

# Ausblick

# Block Web Services

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

## Web Service Komposition

### Web Services in der Praxis

- SOA in der Praxis
- Anforderungen des E-Business
- Erweiterungen von SOAP/WSDL

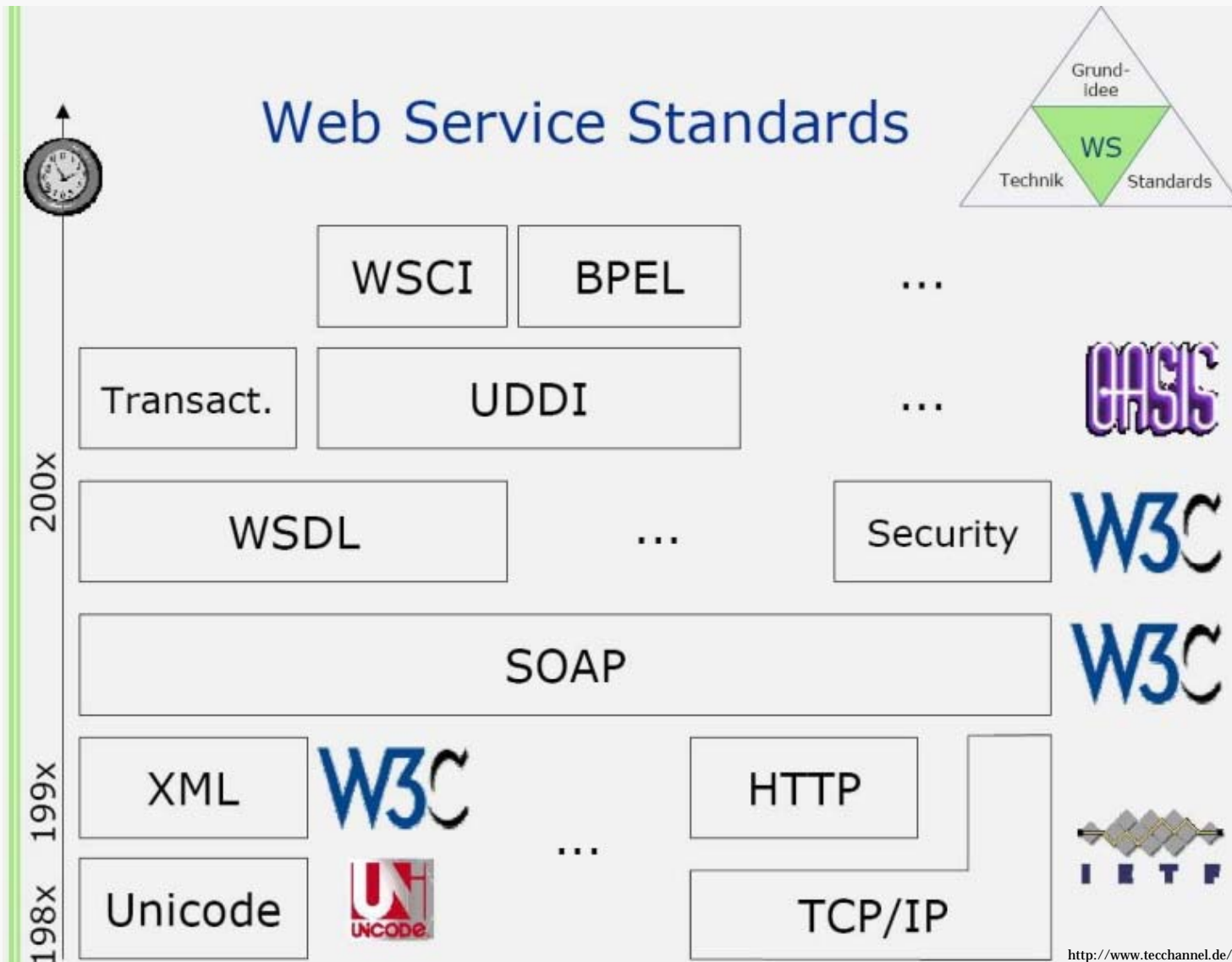
### Semantic Web

- Grundidee & Technologien
- Vergleich mit XML

### Semantic Web Services

- Grundidee

# Standards



# Web Service Komposition

**Web Service Komposition** – Vereinigung bestehender Web Services und anderer Komponenten, um neue Prozesse zu erzeugen

- Bereitstellung von Mehrwertdiensten durch Zusammenschluss mehrerer Web Services
- Serviceorientierte Entwicklung von Systemen
- Ausnutzung wachsender Anzahl bestehender Online-Dienste
- plattformunabhängige Kommunikation zwischen Web Services
- basiert NICHT auf der physikalischen Integration von Komponenten

- **proaktive** vs. **reaktive** Komposition
- **Konversation (conversation) & Konversationsunterstützung**
- **Orchestration** vs. **Choreography**  
(Orchestration/Choreography)

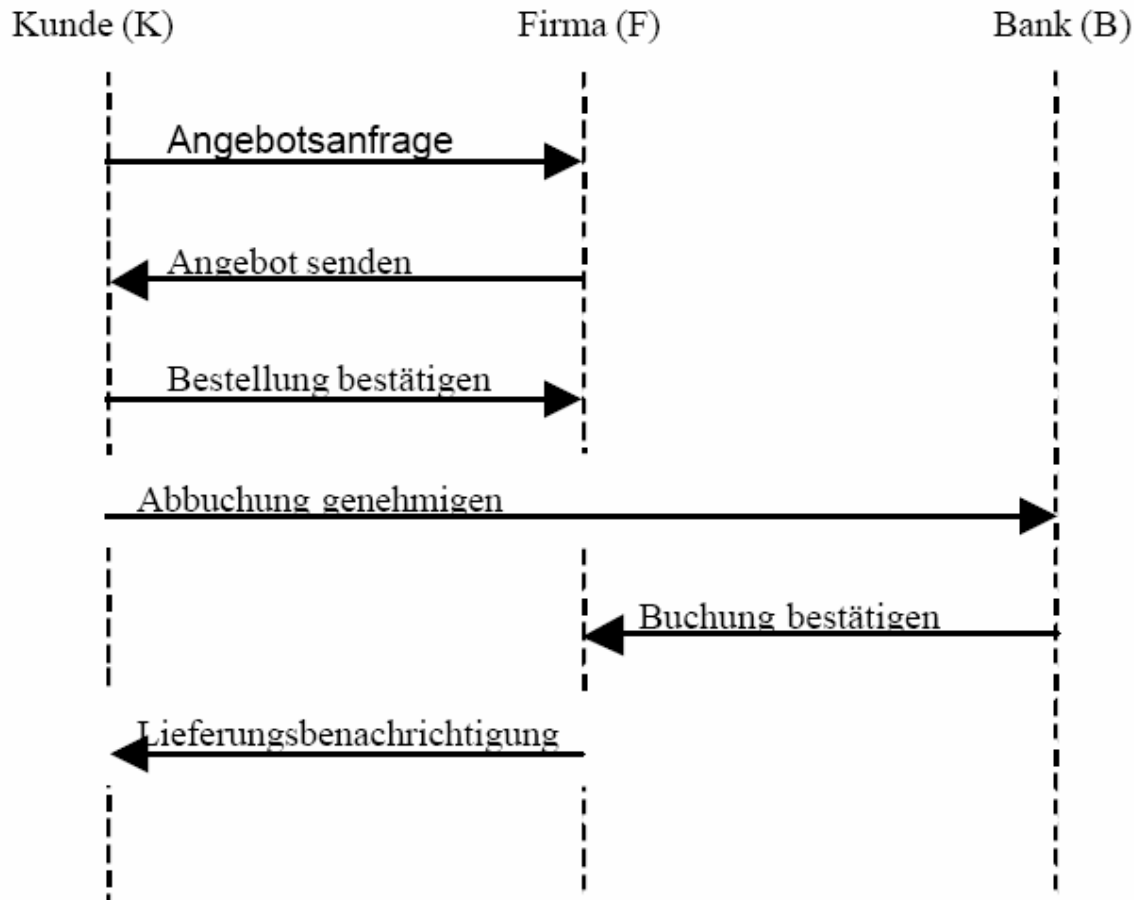
## proaktive Komposition

- alle verwendeten Services sind zur Entwicklungszeit bekannt
- geeignet, wenn die Anwendung
  - oft benutzt wird
  - stabil sein muss
  - lange Laufzeiten benötigt werden
  - die Geschwindigkeit eine Rolle spielt

## reaktive Komposition

- dynamisches Komponieren des neuen Dienstes ( z.B. zu dem Zeitpunkt zu dem er nachgefragt wird)
- erlaubt während der Laufzeit Optimierungen vorzunehmen
- geeignet, wenn der Dienst
  - nur selten nachgefragt wird
  - die einzelnen Komponenten nicht im Vorfeld bekannt sind

# Nachrichtenabfolge (Konversation)



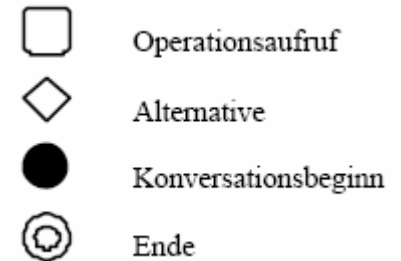
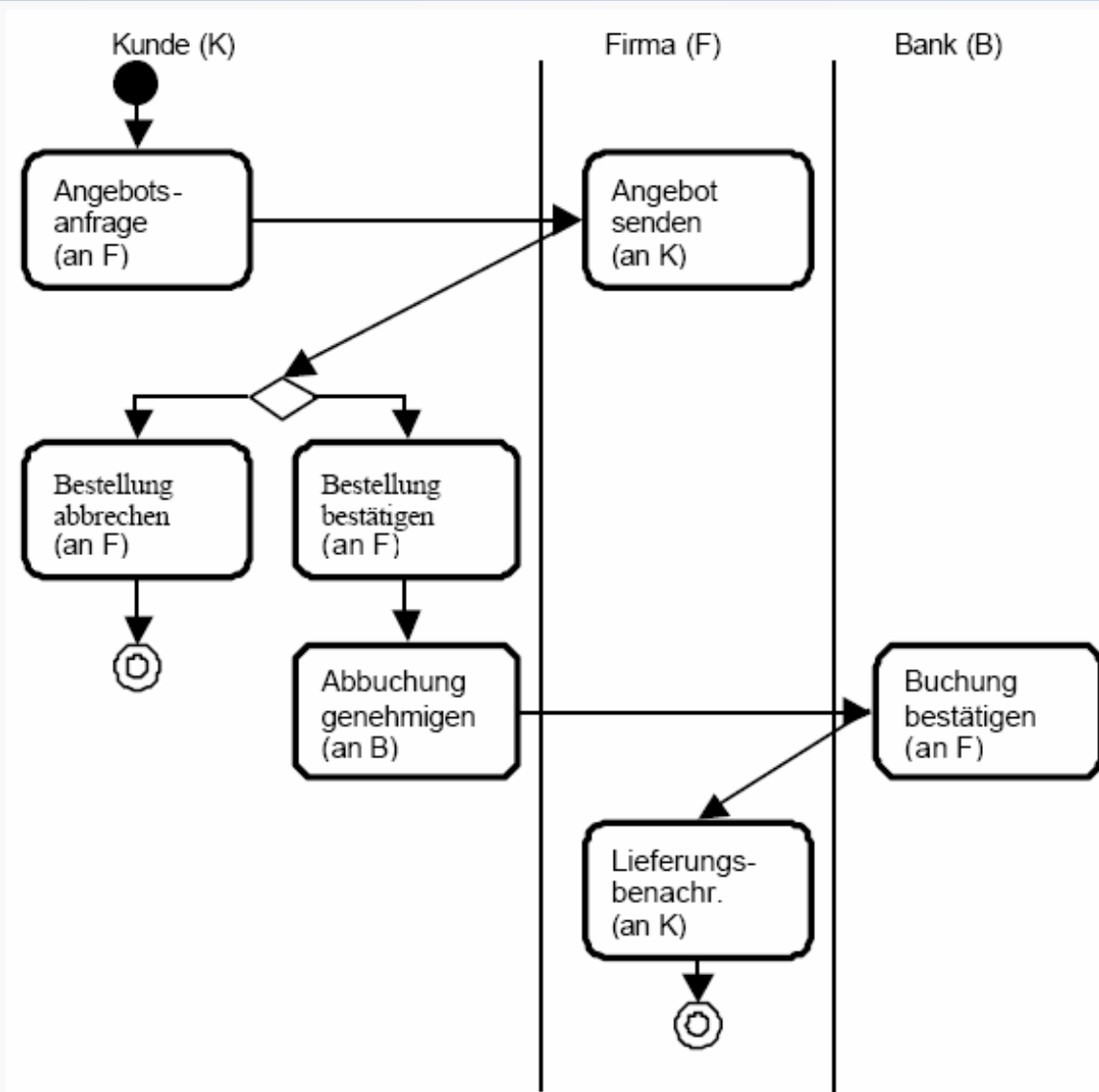
- **Abfolge von Operationen** zwischen Web Services
- **konkrete Kommunikation** von zwei oder mehreren Web Services
- **Kontext** erforderlich

Quelle: M. Reichert, D. Stoll; *Komposition, Choreographie und Orchestrierung von Web Services – Ein Überblick*, EMISA Forum, Band 24, Heft 2, 2004, S. 21-32

# Komposition: Choreography

- beschreibt aus öffentlicher Sicht die **zulässige Abfolge von Nachrichten** zwischen einem oder mehreren Partnern
- keine Angaben über die interne Prozesslogik der Teilnehmer
- kein zentraler Koordinator
- nicht als Prozess ausführbar
- jeder Web Service "weiss", mit wem er interagieren soll
- **Konversations-** oder **Koordinationsprotokoll** (*coordination protocol*) – Menge zulässiger Nachrichtenabfolgen

# Choreography – Beispiel

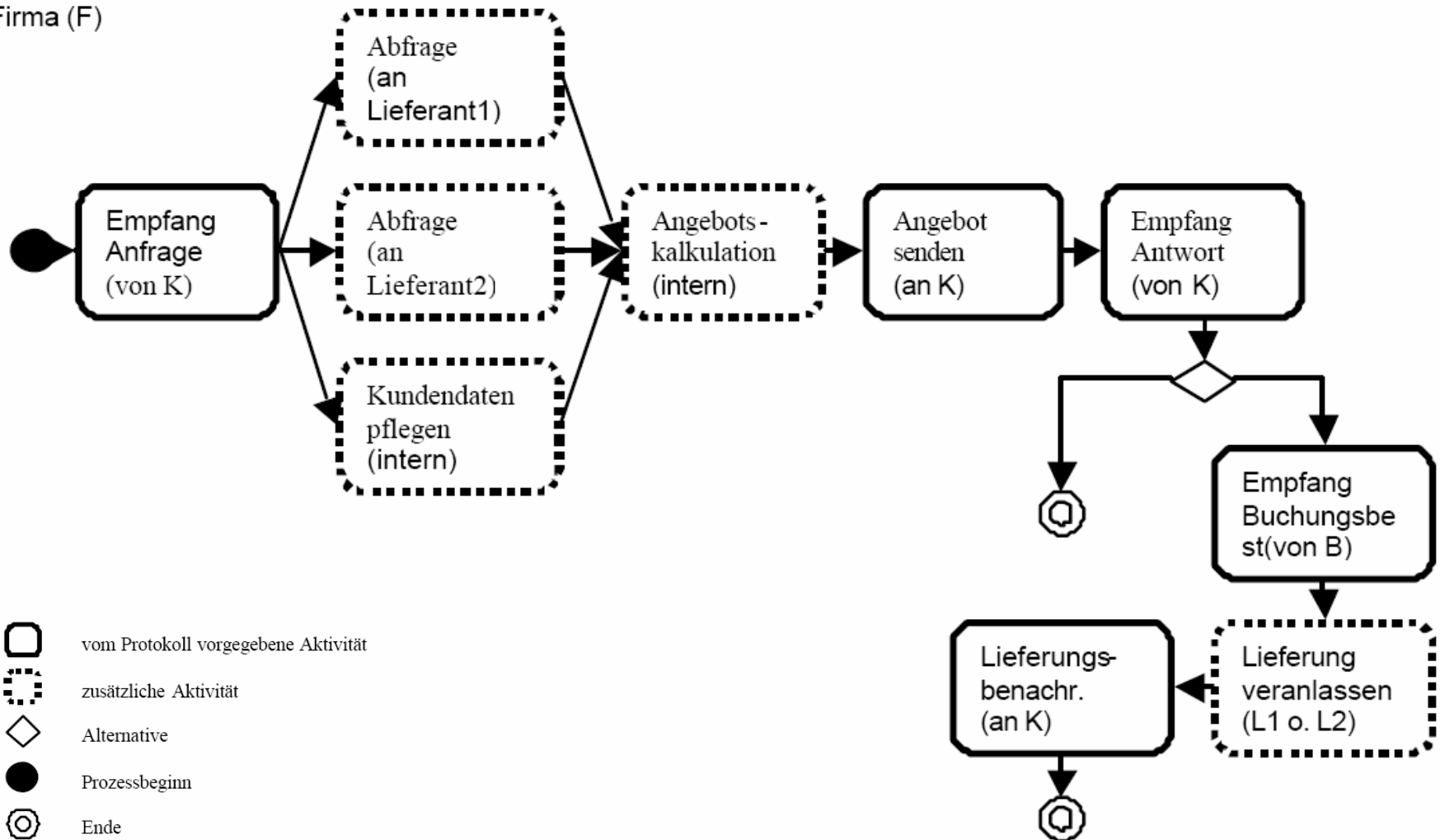


# Komposition: Orchestrierung

- Kopplung einzelner Web Services, um einen komplexen Geschäftsprozess aufzubauen
- beschreibt die Geschäftsprozesslogik aus der Sicht eines Teilnehmers
- ein zentraler Prozess koordiniert die Operationen der Web Services, die in den Prozess eingebunden sind

# Orchestrierung – Beispiel

Firma (F)

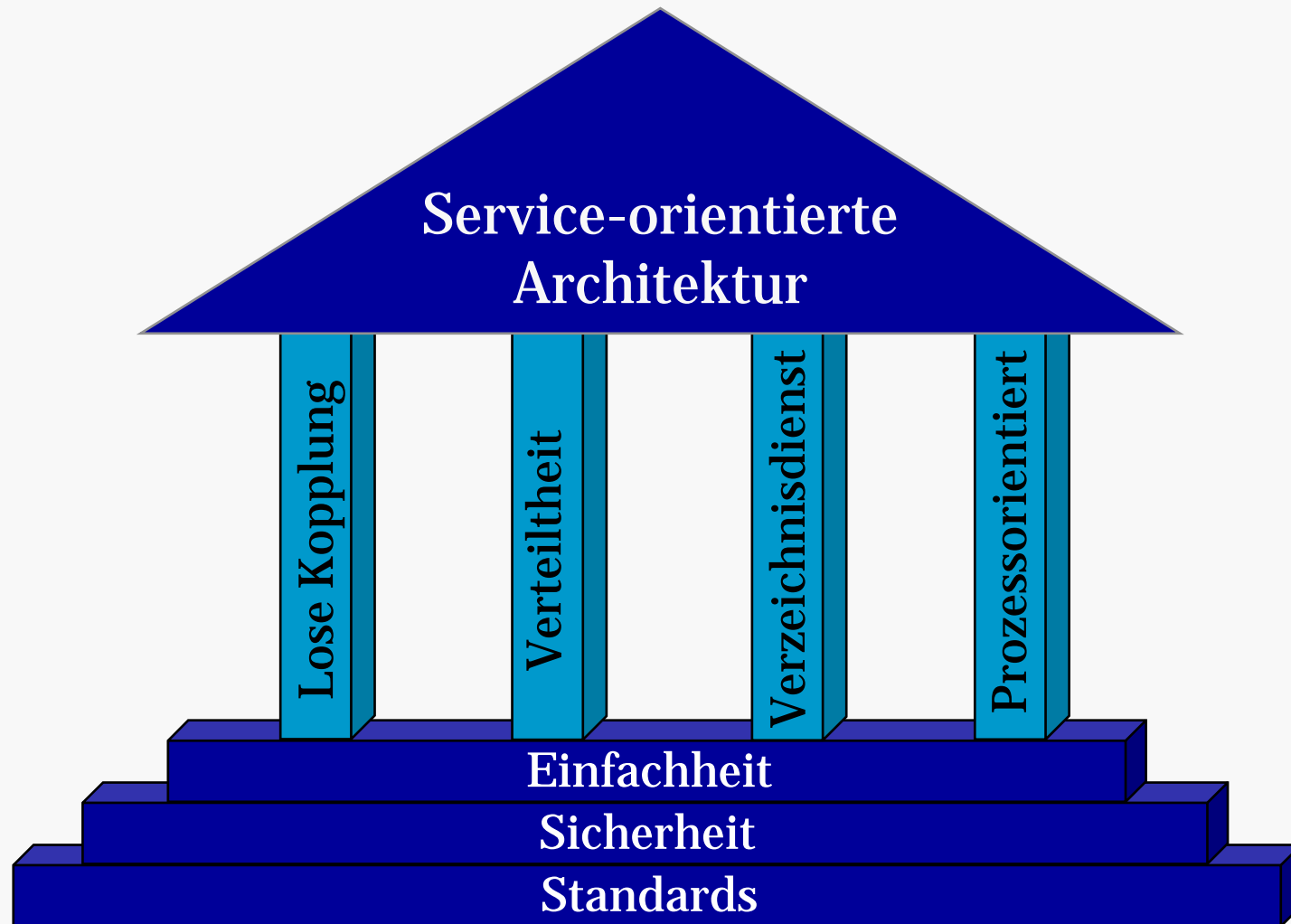


# SOA in der Praxis

- 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 ist...

- SOA ist ein Architekturmodell
- SOA ist kein Standard
- SOA ist keine Spezifikation
- SOA ist kein Produkt
- SOA Idee: Nutzung bestehender Softwarekomponenten die miteinander interagieren

- Unternehmensweite IT Strategie nur fachbereichsspezifisch
- starke Koppelung zwischen verschiedenen Anwendungen
- komplexe IT Infrastruktur (zahllose Systeme mit nur Teilnutzen)
- begrenzte Wiederverwendbarkeit der Systeme (fehlende Schnittstellen, Trennung von Schnittstellen- und Businesslogik)
- hohe Wartungskosten
- langsame und schwierige Anpassung an neue
- Marktanforderungen und Betriebsstrategien.

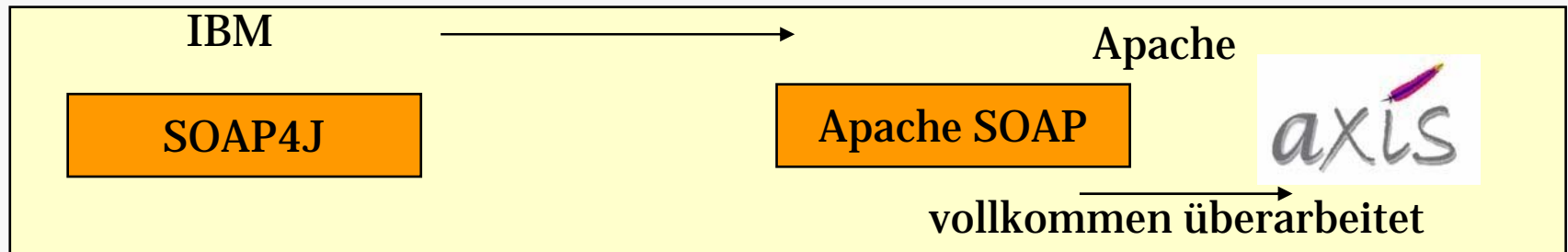
# Ein Web Service aufrufen



## WSDL → Stubs

- Aus WSDL-Beschreibung automatisch Schnittstelle zur Anwendung (Stubs) generieren
- Stubs abstrahieren von SOAP und vom konkreten Übertragungsprotokoll.
- Web Service erscheint als lokale Bibliothek.

- Apache **EX**tensible **I**nteraction **S**ystem
- SOAP-Engine (auch für reines SOAP nutzbar)
- Entwicklung:



- AXIS v. 1.4 – April 2006
- Open Source unter ASFL (ähnlich der LGPL)
- als Plug-in für Tomcat

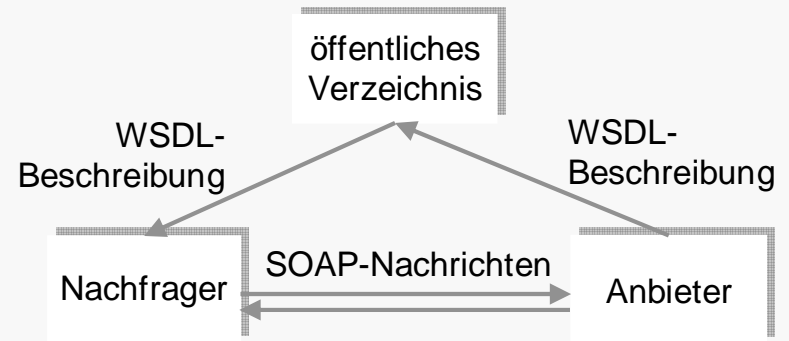
- Semi-automatisches Typemapping zwischen Java und XSD
- Flexible Architektur, gut anpassbar und erweiterbar
- unterstützt SOAP 1.1, SOAP 1.2 , WSDL 1.1
- WSDL-Unterstützung
  - Automatische Generierung für WS
  - Generierung von Stubs zur Benutzung des WS durch Klient

## Beispiel

- personalisierte Informations-Webseite
- Benutzer möchte Echtzeitkurse von bestimmter Aktie einer bestimmten Börse
- Informations-Webseite sucht in öffentlichen Verzeichnis passenden Web Service
- Informations-Webseite bindet den Web Service automatisch ein



⇒ dynamische Einbindung zur Laufzeit



## 4 Schritte notwendig:

1. Anwendung sucht in Verzeichnis passenden Web Service
2. Suchergebnis: WSDL-Beschreibung
3. WSDL-Beschreibung → Stubs
4. Anwendung ruft Stubs auf

⇒ Dienst muss automatisch gefunden und aufgerufen werden.

# Schritt 1: Dienst finden

- gegeben: bekannter/standardisierter Web Service
- gesucht: Anbieter des Web Services, einschl. WSDL-Beschreibung
- Um Anbieter automatisch zur Laufzeit zu finden, muss eine der folg. Bedingungen erfüllt sein:
  - a) Web Service **kostenlos** und **unkritisch**, daher kein Vertrag nötig  
Beispiel: kostenlose, verzögerte Aktienkurse als unverbindliche Information
  - b) **Vertrag besteht bereits**  
Beispiel: SAP/R3-Web-Service auf unterschiedlichen Servern

# Schritt 4: Aufruf der Stubs

- gegeben: WSDL-Beschreibung
- Um Web Service automatisch mit Stubs aufzurufen, muss er **bekannt/standardisiert** sein:
  - WSDL beschreibt zwar Syntax der Schnittstelle, nicht aber Bedeutung der Prozedur/Parameter
  - Bedeutung muss außerhalb von WSDL festgelegt sein:  
Web Service muss also bekannt/standardisiert sein

## Float Aktienkurs(Integer WKN, String Boersenplatz)

- Stub kann automatisch generiert werden

**aber:**

- Was bedeutet *WKN*?
- Was bedeutet *Boersenplatz*?

Wie wird z.B. der Xetra-Handel in Frankfurt kodiert?

- Was bedeutet *Aktienkurs*?

Echtzeit oder verzögert? Wie lange verzögert?

- Was bedeutet Ergebnistyp *Float*?

Welche *Währung*?

## Float Aktienkurs(Integer WKN, String Boersenplatz)

- Stub kann automatisch generiert werden

aber:

- Was bedeutet *WKN*?
- Was bedeutet *Boersenplatz*?  
Wie wird z.B. der Xetra in Frankfurt kodiert?
- Was bedeutet *Aktienkurs*?  
Echtzeit oder verzögert? Wie lange verzögert?
- Was bedeutet *Ergebnistyp Float*?  
Welche *Währung*?

**muss außerhalb von WSDL festgelegt werden!**

## 1. neuer Dienst zur Laufzeit

- vollautomatisches Einbinden prinzipiell mit WSDL möglich, wird aber *Ausnahme* bleiben
- *Grund*: nur für standardisierte, kostenlose und unkritische Dienste möglich

## 2. neuer Dienst zur Entwicklungszeit

- aus WSDL können automatisch Stubs generiert werden
- Integration in Anwendungslogik realisiert Entwickler

## 3. Ersatz für bestimmten Dienst zur Laufzeit

- aus WSDL können automatisch Stubs generiert werden
- Integration in Anwendungslogik bereits realisiert

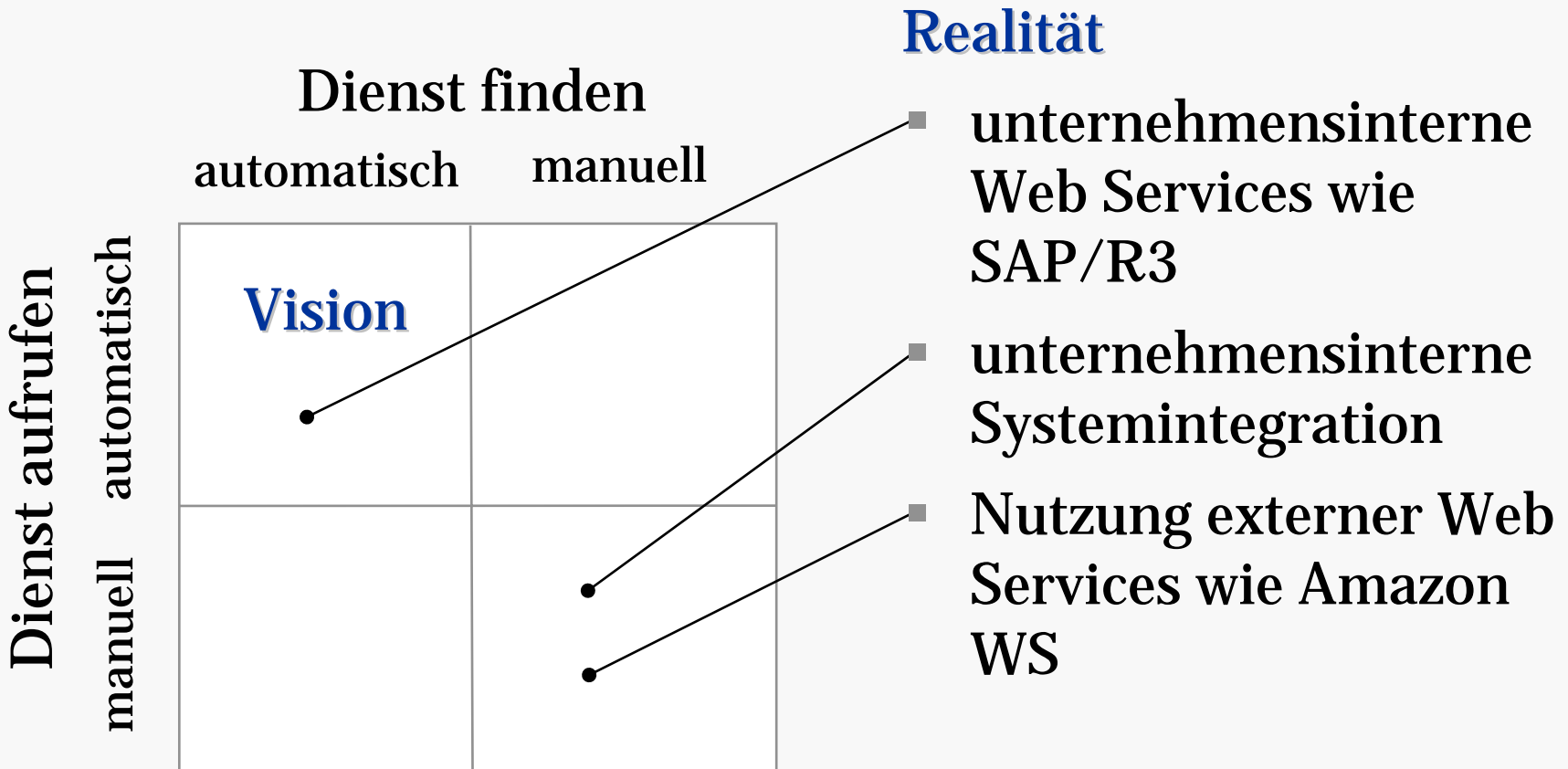
## automatischer Aufruf zur Laufzeit Vision

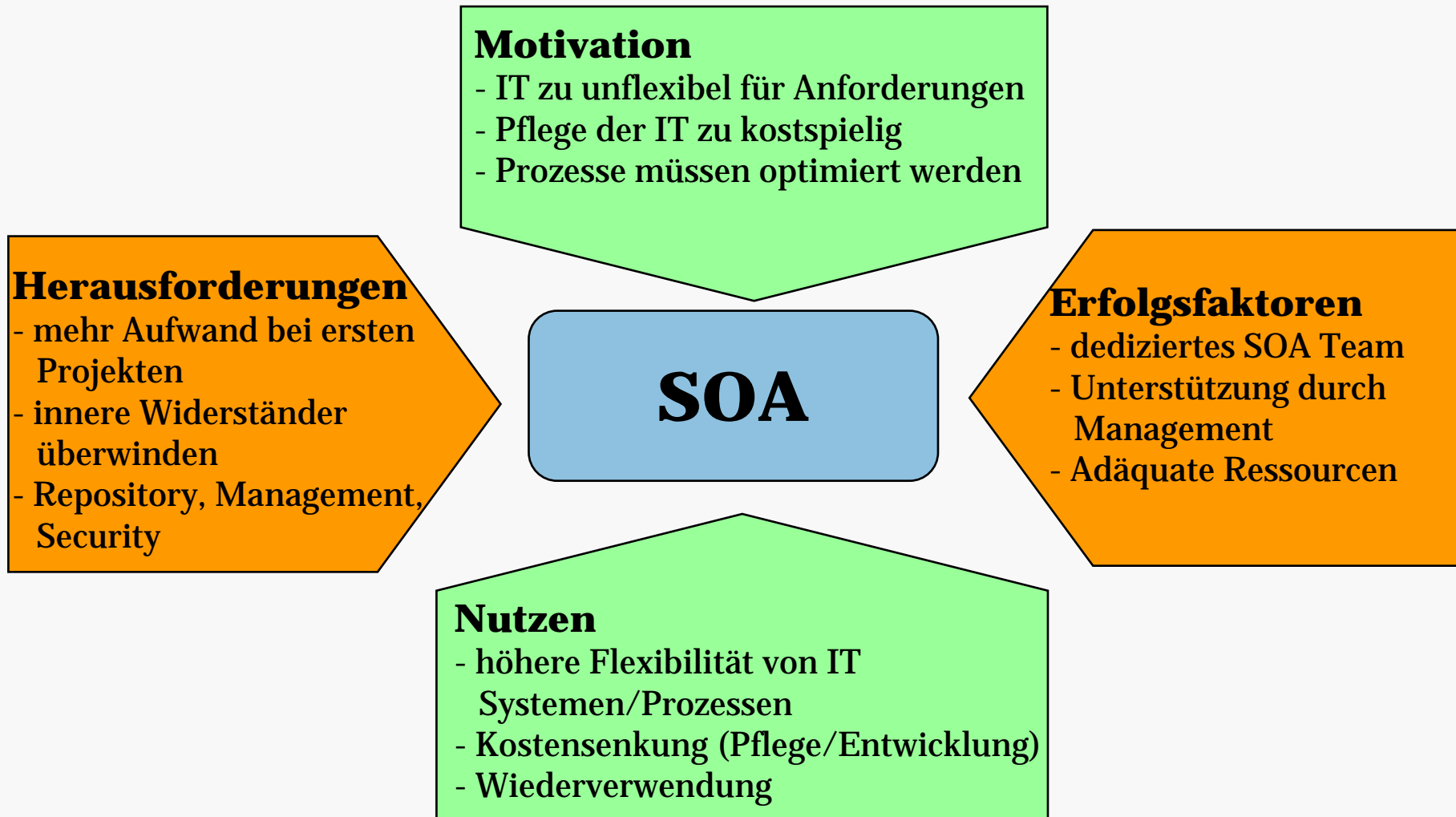
- Schritt 3: Übersetzung WSDL → Stubs erfolgt automatisch
- Schritt 4: Integration von Stubs in Anwendungslogik ***erfolgt automatisch***

## manueller Aufruf zur Entwicklungszeit Praxis

- Schritt 3: Übersetzung WSDL → Stubs erfolgt automatisch
- Schritt 4: Integration von Stubs in Anwendungslogik ***programmiert Entwickler***

# Web Services: Vision und Realität

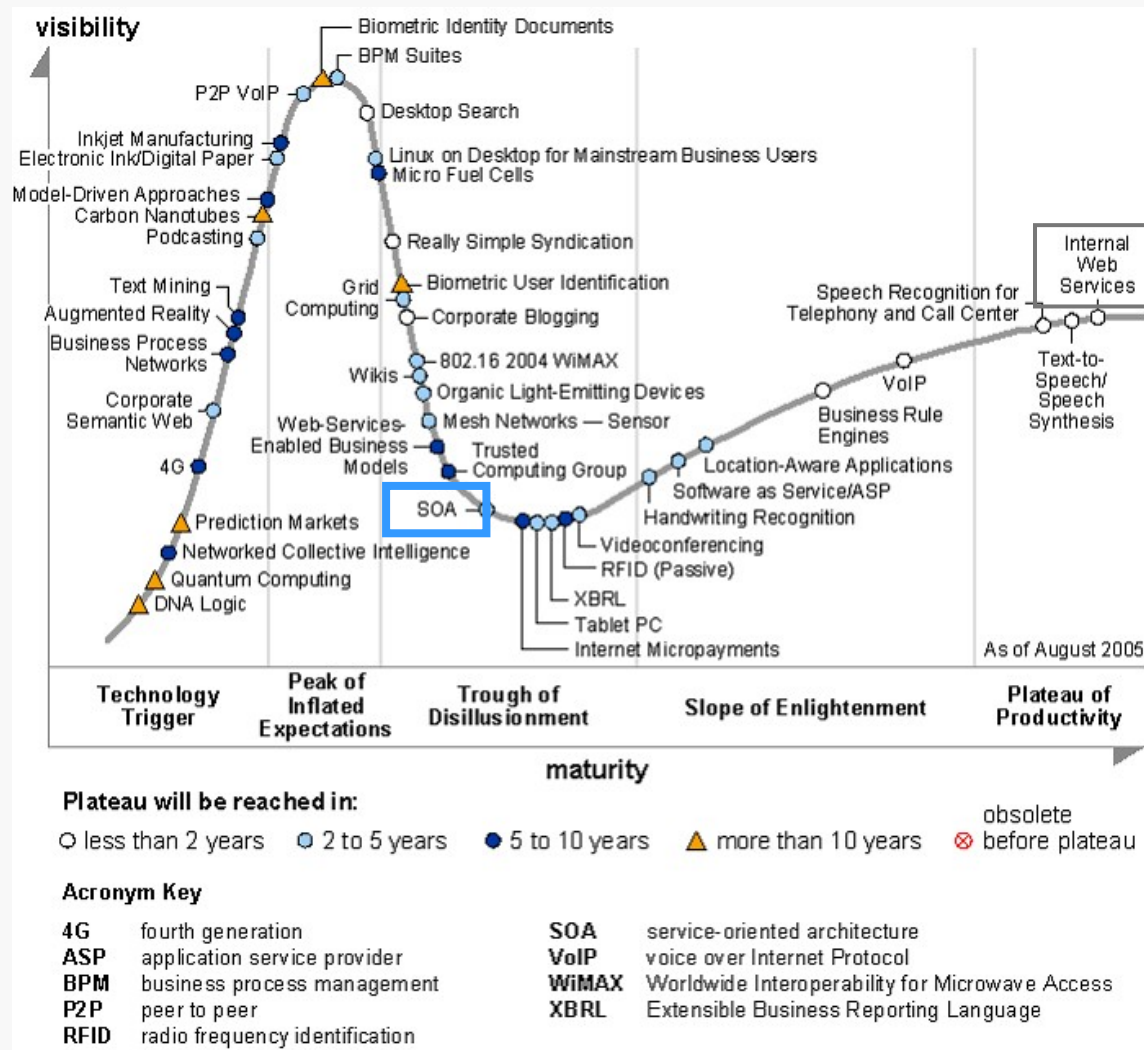




- **Konsolidierung:** Gibt es mehrere Anwendungen, die im Kern das gleiche tun?
- **Umsatzsteigerung/Produktdiversifikation:** Existieren Anwendungen/Daten, die als Standard Service auch für Partner/Kunden einen Nutzen darstellen?
- **Substitution:** Nicht immer sind zusätzliche, unverbundene Kern-Systeme erforderlich, um einen konzentrierten Umgang mit Informationen entlang eines Prozesses zu gewährleisten.

- **Produktwahl ohne Kompromisse** → üblicherweise ist keine der wichtigsten Plattformen für jeden Einsatzzweck die beste Wahl
- **Weiche „Migration“** → lässt ohne Auswirkung auf die zugreifenden Komponenten genügend Spielraum, um die Implementierung der angebotenen Funktionalität zu ersetzen

# Web Services: Entwicklungsstand



# Meinungen zu SOA

Durch die Kopplung von BPM und SOA lässt sich ein Unternehmen realisieren, welches am Markt durch zeitnahe Reaktion auf Veränderungen am Markt reagieren kann und wettbewerbsfähig bleibt.

Quelle: Gartner Group Studie aus CIO Ausgabe 17.06.2004

Gegen Ende des Jahres 2006 wird SOA zur wichtigsten Technologie auf dem IT-Markt herangewachsen sein.“

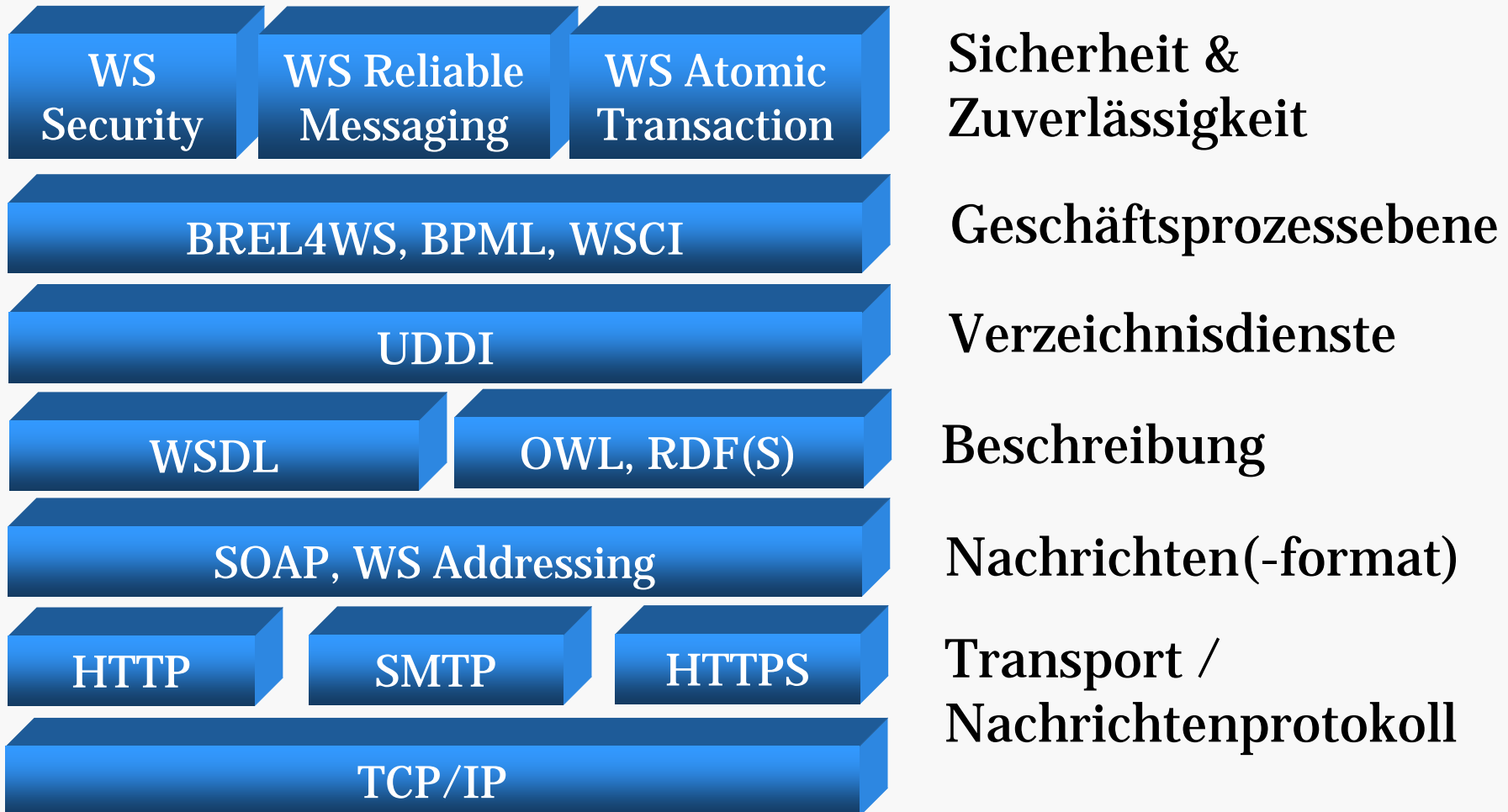
Quelle: InfoWorld Research 2006

2008 werden mehr als 60% der Betriebe **SOA** als „**Das führende Prinzip**“ im Unternehmen anwenden.

Quelle: Gartner Group Statement 2006

# Erweiterungen von SOAP/WSDL

# Erweiterungen von SOAP/WSDL



## Zusatzinformationen im SOAP-Header:

- **digitale Signatur**
  - Identifikation des Absenders
  - Unversehrtheit der Nachricht
- **Anforderung einer Empfangsbestätigung**
  - Nachricht mindestens einmal zugestellt
- **eindeutige Nachrichtenreferenz**
  - Erkennung von Duplikaten
- **Verweis auf vorherige Nachrichten, z.B. Bestellanfrage**
  - Berücksichtigung des Workflows

```
<S:Envelope ... >
  <S:Header>
    <wsa:ReplyTo>
      <wsa:Address>http://business456.com/User12</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>http://fabrikam123.com/Traffic</wsa:To>
    <wsa:Action>http://fabrikam123.com/Traffic/Status</wsa:Action>
  </S:Header>
  <wssec:Security>
    <wssec:BinarySecurityToken
      ValueType="wssec:X509v3"
      EncodingType="wssec:Base64Binary">
      dWJzY3JpYmVyLVBlc...eFw0wMTEwMTAwMD
    </wssec:BinarySecurityToken>
  </wssec:Security>
  <wsrm:Sequence>
    <wsu:Identifier>http://fabrikam123.com/seq1234</wsu:Identifier>
    <wsrm:MessageNumber>10</wsrm:MessageNumber>
  </wsrm:Sequence>
  </S:Header>
  <S:Body>
    <app:TrafficStatus
      xmlns:app="http://highwaymon.org/payloads">
      <road>520W</road> <speed>3MPH</speed>
    </app:TrafficStatus>
  </S:Body>
</S:Envelope>
```

WS-Addressing

WS-Security

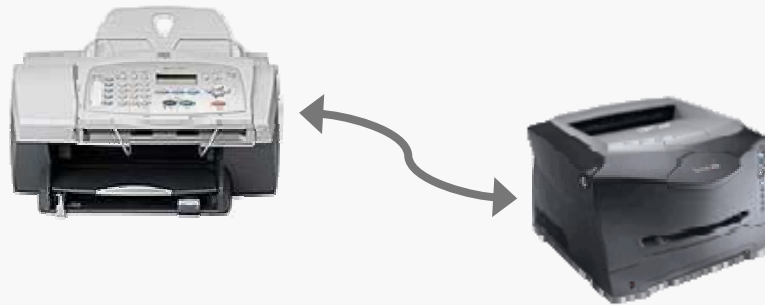
WS-Reliable Messaging

Bild: Ferguson et al. 2003

# So einfach ist es dann doch nicht...

## Problem der **Interoperabilität**:

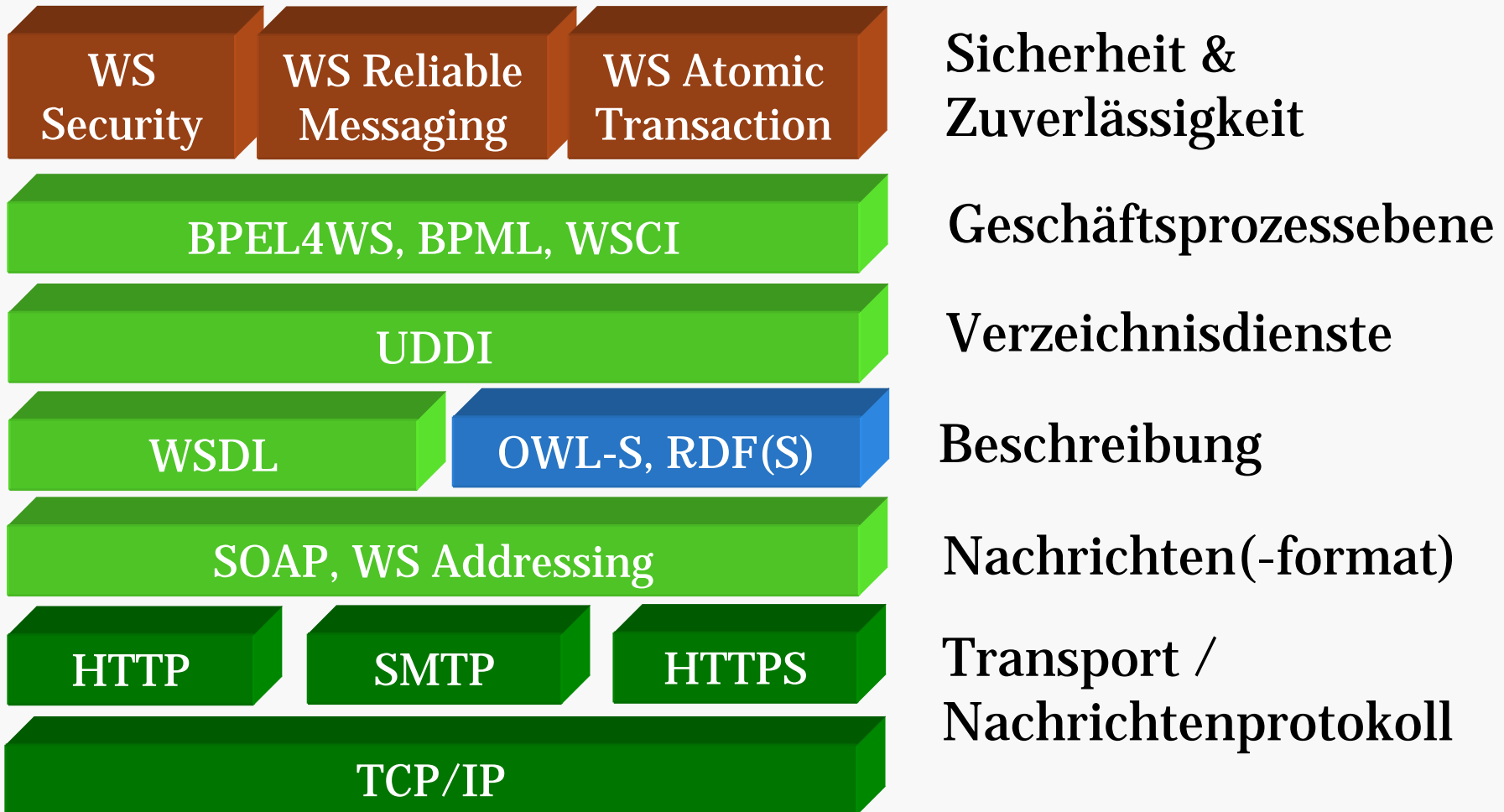
- WSDL beschreibt zwar Syntax der Schnittstelle, einschl. der Zusatzinformationen im SOAP-Header.
- Man kann nicht davon ausgehen, dass jede Web-Service-Umgebung diese Erweiterungen auch verarbeiten kann.
- Grund: **Erweiterungen noch nicht etabliert**



Fax uneingeschränkt interoperabel:

- Faxgerät kann durch ein anderes ersetzt werden, ohne mit potentiellen Sendern Protokolle auszuhandeln.
- Z.B. muss nicht geklärt werden, wie Empfang bestätigt wird.
- Grund: etablierter internationaler Fax-Standard

# Erweiterungen von SOAP/WSDL



# Erweiterungen: Fazit

- noch keine etablierten Standards
- + einzelne Erweiterungen unabhängig voneinander
- + meist gemeinsame Vorschläge von Microsoft und IBM

**Keine prinzipiellen Hürden, jedoch noch ein langer Standardisierungsweg zu gehen!**

# Semantic Web

# Semantic Web

“The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.”

Berners-Lee, Hendler, and Lassila, 2001.



Foto: W3C



Foto: Homepage



Foto: Homepage

Das Ziel des Semantic Web ist es  
WWW-übertragende Daten  
durch Menschen mit  
Bedeutungsinformationen (Semantik)  
anzureichen für  
die **Verarbeitung durch Maschinen**  
und **Nutzung durch Menschen.**

# Bildersuche: „Apache“



- Maschinen fehlt dieser Kontext aus Begriffen und Zusammenhängen
- Kontext muss Maschinen zusätzlich bereitgestellt werden

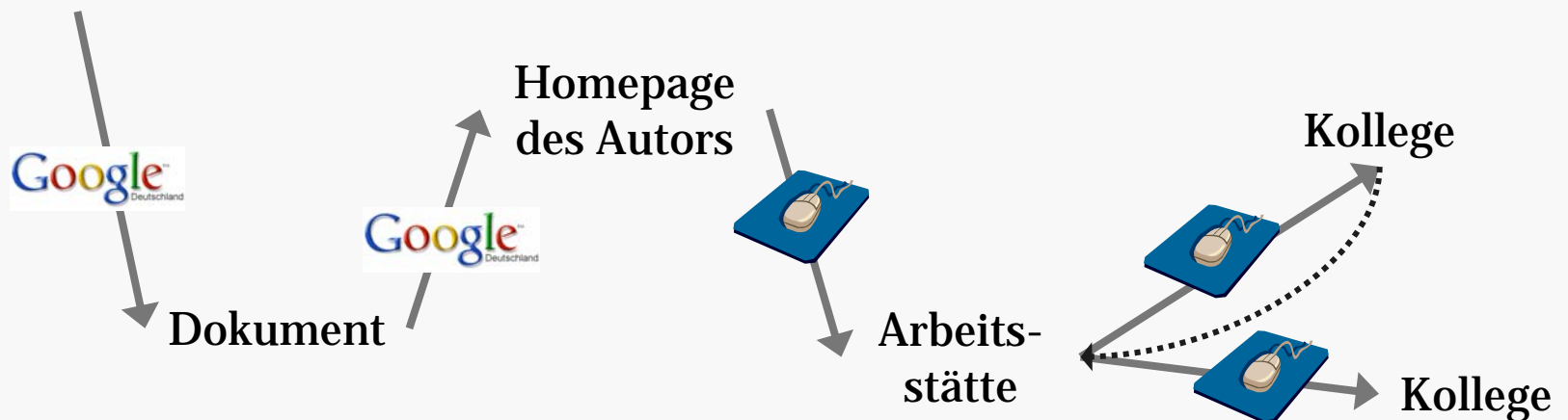
- Damit Metadaten nutzbar sind
  - muss der Informationsanbieter sich so ausdrücken, dass Informationsnutzer ihn verstehen
  - muss der Informationsnachfrager so fragen, dass er etwas finden kann
- Gemeinsame Benutzung von Konzepten
- Gemeinsame Sprache
  
- **Ontologie** zur Definition einer gemeinsamen Sprache
  - Es gibt Konzepte, die wir mit „Bank“ und „Sparkasse“ benennen
  - Es gibt ein Konzept, das wir „Geldinstitut“ nennen und das die Konzepte „Bank“ und „Sparkasse“ umfasst

# Vision von Berners-Lee

- Webinhalte und ihre Vernetzung werden für Maschinen verständlich.
- Auch komplexe Anfragen können ans Web gestellt werden.



- Beispiel: Mit welchen Kollegen arbeitet der Autor eines bestimmten Dokumentes zur Zeit zusammen?



# Knowledge Technologies

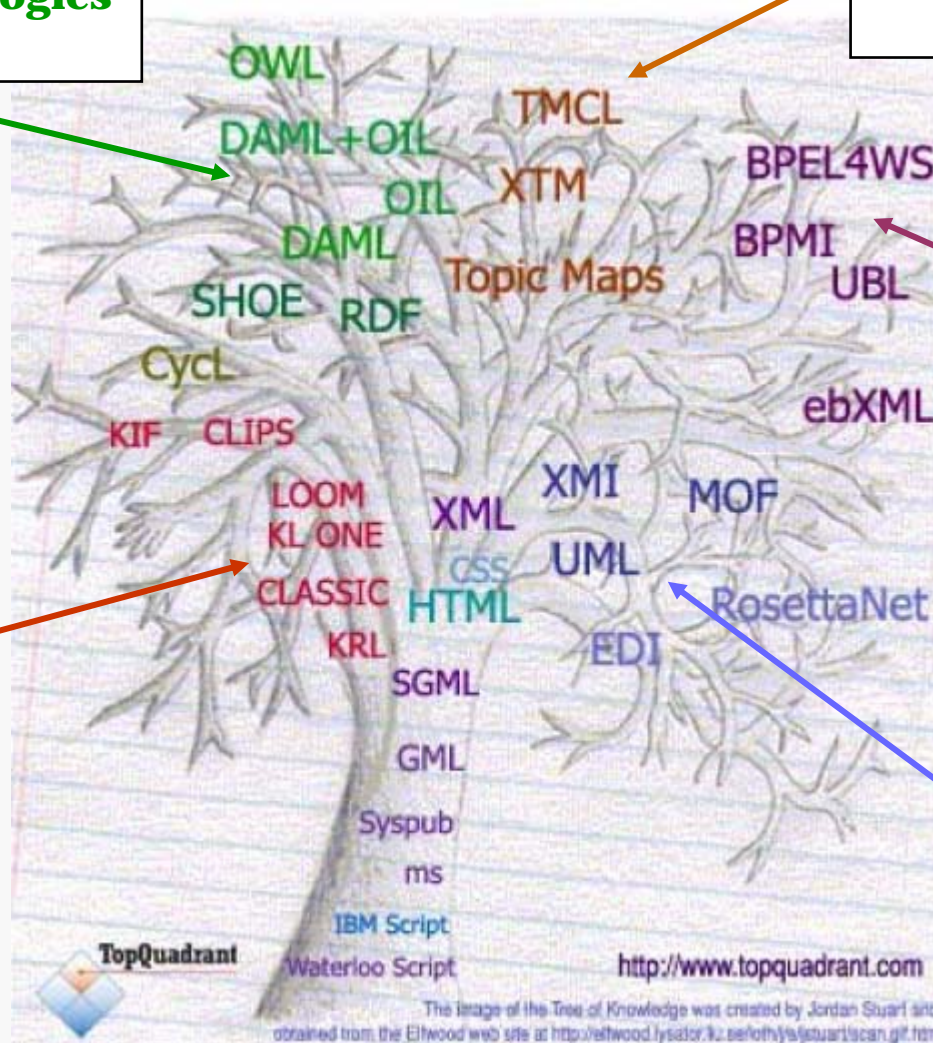
**Semantic Technologies Languages**

**Content Management Languages**

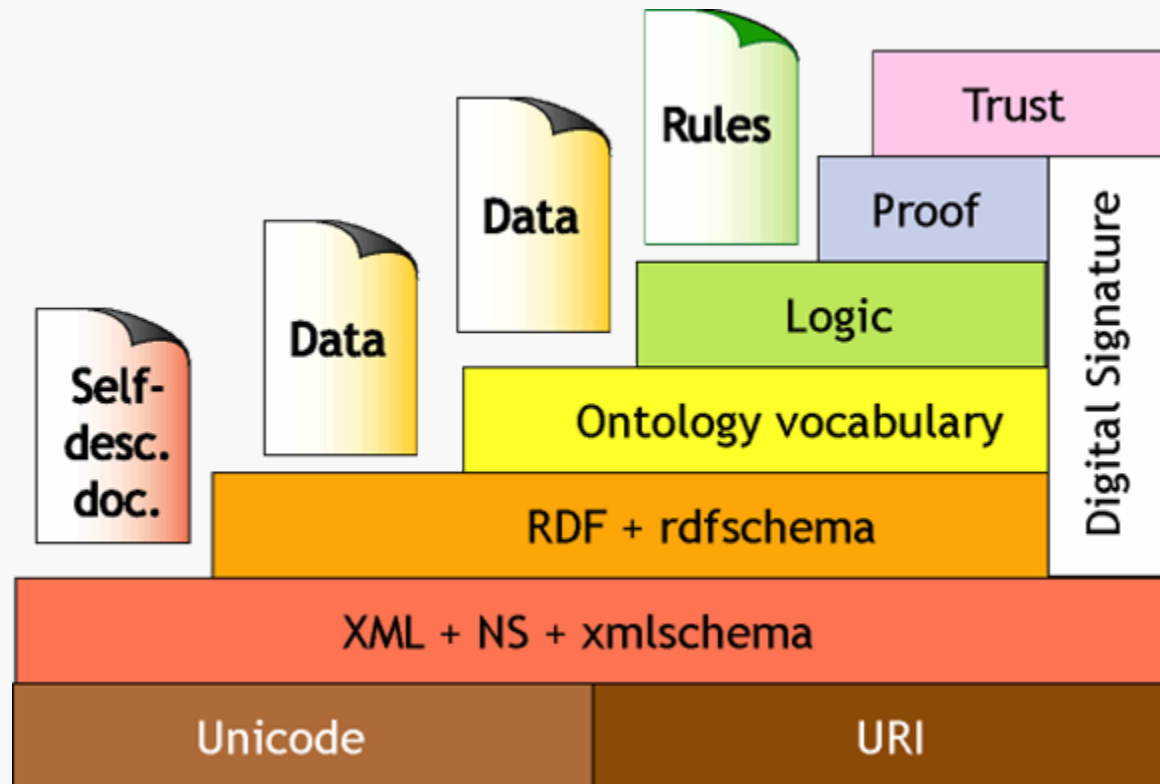
**Process Knowledge Languages**

**AI Knowledge Representation**

**Software Modeling Languages**



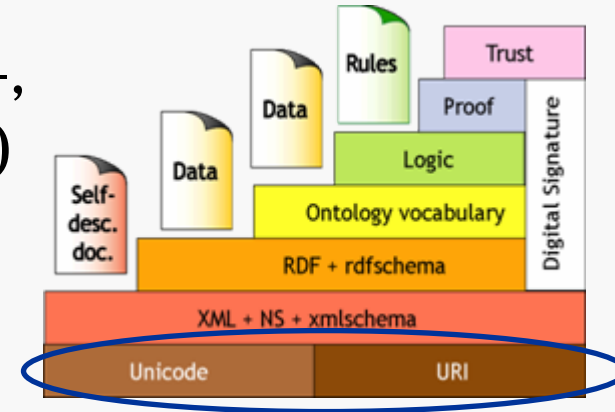
# Semantic Web Stack (W3C)



auch „Semantic Web Layer Cake“ oder „Semantic Web Tower“ genannt

## Unicode

- jedes Zeichen eigene Nummer (system-, programm- und spracheunabhängig)
- Unicode-Codierung – Zeichensätze für fast jede natürliche Sprache

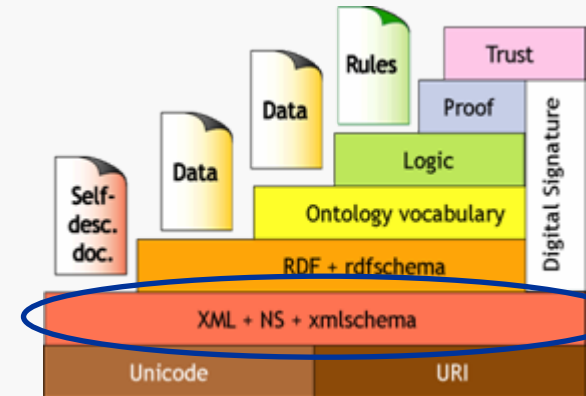


## URI – Uniform Resource Identifier

- eindeutige Identifikation einer Quelle/Ressource → jedes beliebige Objekt verfügt über einen URI
- Mechanismus um Daten verteilt repräsentieren zu können
- URLs – Untergruppe von URIs
- Syntax vom W3C standardisiert

## XML + Namensräume + XML-Schema

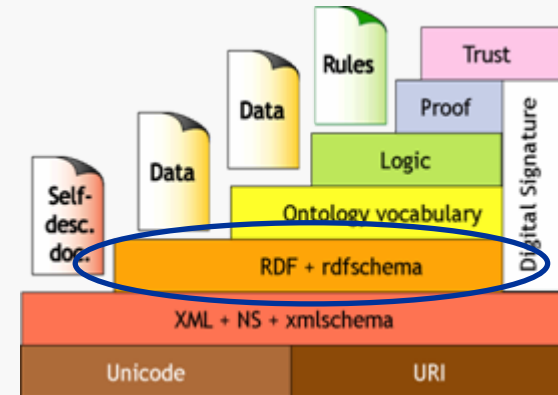
- hierarchisch strukturierte, medienneutrale **Daten**
- Vokabular kann mit **XML-Schema** definiert werden
- Bedeutung des Vokabulars kann mit **Namensräumen** festgelegt werden
- XML-Daten können mit **XLink** verlinkt werden: Links können Namen, aber keinen Namensraum haben



⇒ maschinenverarbeitbare verlinkte Daten, Links jedoch nicht maschinenverarbeitbar

## RDF + Namensräume + RDF-Schema

- Web als Menge vernetzter Ressourcen
- Vokabular für Beziehungen kann mit **RDF-Schema** definiert werden
- Bedeutung des Vokabulars wird mit **Namensräumen** festgelegt
- **RDF Modell** bietet eine syntaxunabhängige Darstellung



⇒ maschinenverarbeitbares  
Netzwerk von Beziehungen

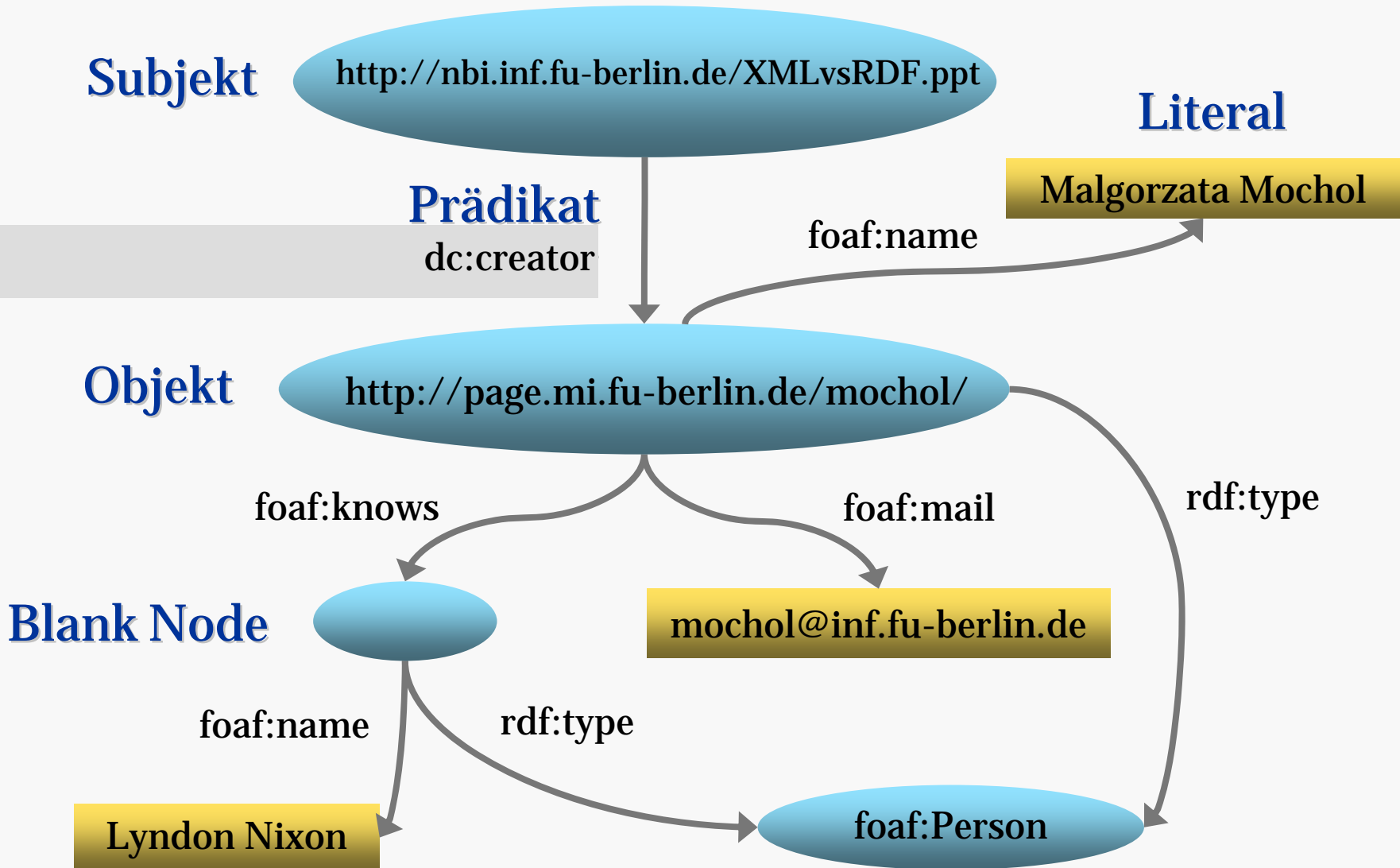
- RDF/XML Syntax Specification – W3C Recommendation seit Feb. 2004
- verschiedene Versionen:
  - Tripel: kompakt, lesbar
  - RDF/XML: für maschinelle Verarbeitung
- Tripel setzen bel. Web-Ressourcen URI-s und URI-o miteinander in Beziehung:

<URI-s, URI-p, URI-o>

URI-s steht zu URI-o in der Beziehung URI-p

- RDF Statement – die kleinste Informationseinheit, die ein Fakt darstellt
- Beispiel:  
***This presentation** was **created** by **Malgorzata Mochol***
  - Subject (Ressource): ***This presentation***
  - Predicate (Property): ***created***
  - Object (Wert): ***Malgorzata Mochol***
- RDF benutzt URIs :
  - Subject: ***<http://nbi.inf.fu-berlin.de/SemWeb.ppt>***
  - Predicate: ***<http://purl.org/dc/elements/1.1/creator>***
  - Object: ***<http://page.mi.fu-berlin.de/mochol/>***

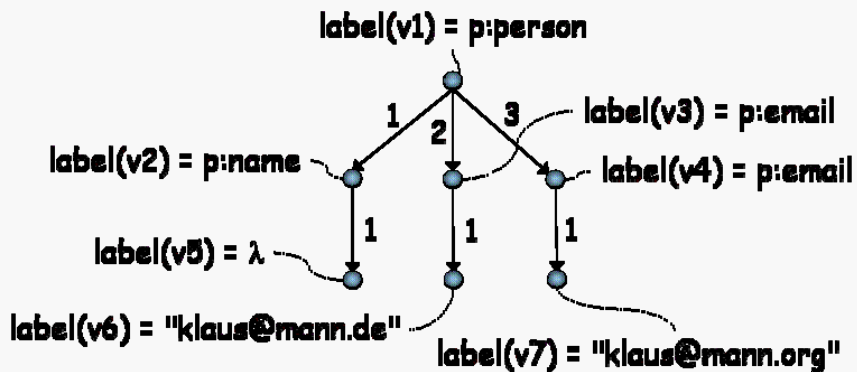
# RDF Model



# Vergleich der Datenmodelle

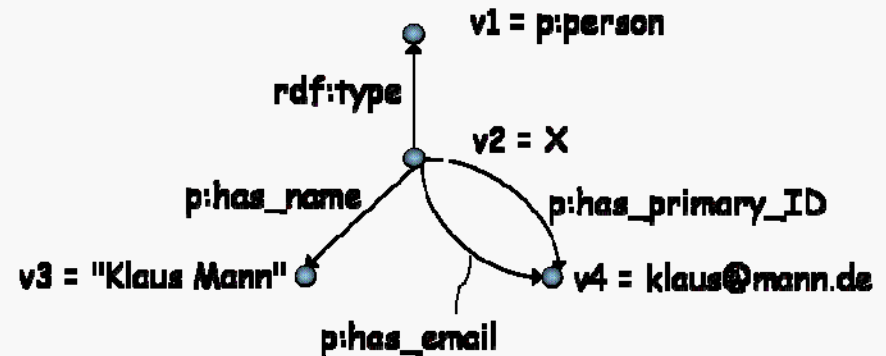
## XML

```
<p:person>
  <p:name/>
  <p:email> klaus@mann.de</p:email>
  <p:email> klaus@mann.org</p:email>
</p:person>
```



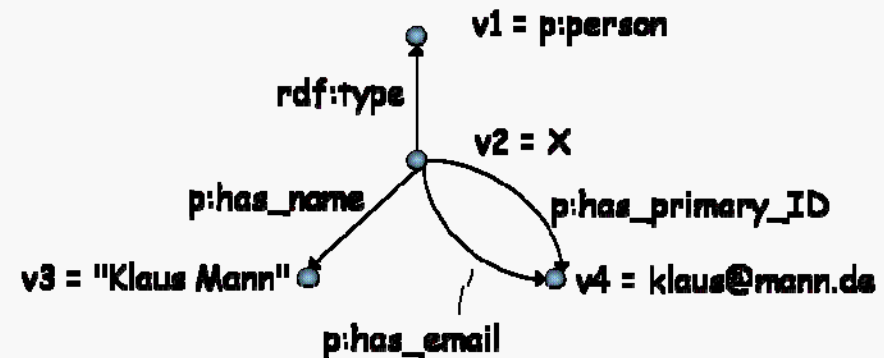
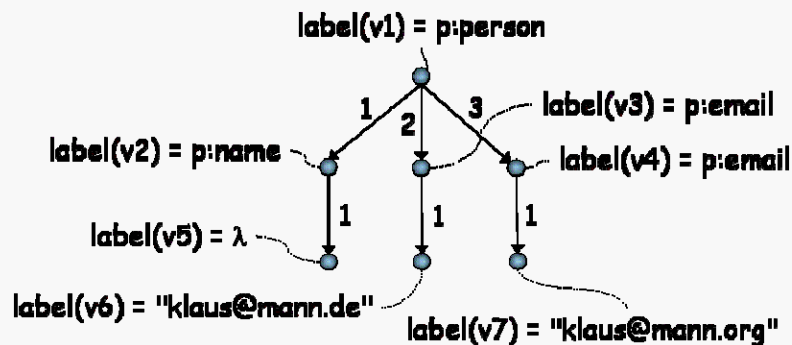
## RDF

```
X rdf:type p:person
X p:has_name "Klaus Mann"
X p:has_email klaus@mann.de
X p:has_primary_ID klaus@mann.de
```



# Vergleich der Datenmodelle

	<b>XML</b>	<b>RDF</b>
Datenmodell	<b>hierarchisches</b> Modell	<b>Netzwerkmodell</b>
Was wird dargestellt?	<b>einzelner Baum:</b> <ul style="list-style-type: none"> <li>▪ benannte Knoten</li> <li>▪ unbeschriftete, aber <b>geordnete</b> Kanten</li> </ul>	möglicherweise <b>unendlich</b> viele <b>gerichtete Multi-Graphen:</b> <ul style="list-style-type: none"> <li>▪ benannte Knoten</li> <li>▪ benannte Kanten</li> <li>▪ Knoten = Name</li> </ul>



# Vergleich der Schema-Sprachen

	<b>XML Schema</b>	<b>RDF Schema / OWL</b>
Abstraktions- ebene	~ <b>Datenbankschema</b>	~ <b>ER-Diagramm</b>
Prinzip	nur zulässig, was explizit erlaubt: <b>Closed World Assumption (CWA)</b>	alles zulässig, was Randbedingungen erfüllen kann: <b>Open World Assumption (OWA)</b>
Validierung bzgl. Schema	<b>möglich</b>	<b>nicht möglich</b>
Berechnungs- komplexität	<b>polynomial</b>	RDF Schema: <b>NP-vollständig</b>

CWA:

nicht zulässig

explizit  
erlaubt

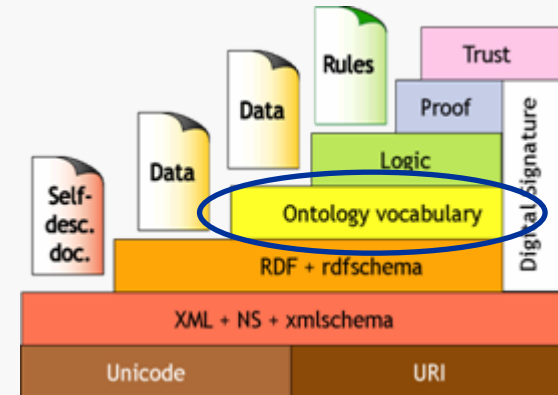
OWA:

zulässig

Randbedingungen  
nicht erfüllbar

## Ontologien

- Vokabulare
- Begriffsbeziehungen (Unterklasse, Untereigenschaft, Wertebereiche, ..., selbstdefinierte)
- Sprachen für Web-Ontologien:
  - DAML+OIL
  - OWL – Web Ontology Language
    - Erweiterte Beschreibungsmöglichkeiten
    - In unterschiedlichen Mächtigkeiten/Komplexitäten (OWL-Lite, OWL-DL, OWL-Full)



# Weitere Technologien des Semantic Web

## Logik

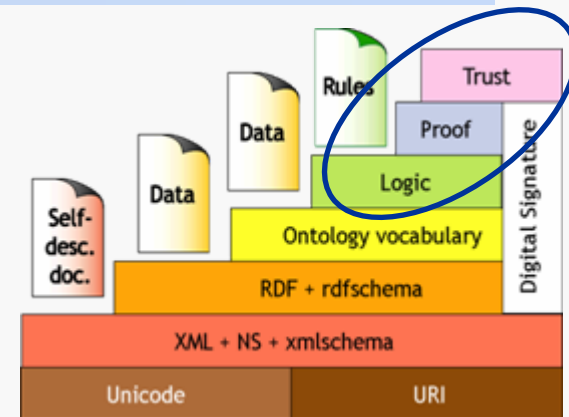
- Semantik auf logischer Basis
- Ableitungsregeln

## Proof

- Konsistenz
- Ableitung (Inferenz)

## Trust

→ Beginn der Entwicklung



## XML

- XML heute omnipräsent, wenn auch nicht immer sichtbar

## RDF

- HTML-Seiten und XML-Dokumente werden erstellt, aber kaum RDF.
- neue XHTML-Version wird RDF integrieren:  
Jedes XHTML-Element kann dann RDF-Meta-Informationen haben.
- Woher kommen diese Meta-Informationen?

## Organisatorisch:

- Stellenanbieter nutzen gemeinsames kontrolliertes Vokabular für die Annotierung von Stellenangeboten
- Stellensuchende nutzen gleiches Vokabular für Stellengesuche/Bewerberprofile

## Technisch:

- **Einfache Annotation → Reichere Annotation → Ersatz von Freitext durch RDF**
- Stelleangebote direkt auf der Web-Seite des Unternehmens
- Semantische Suchmaschinen :
  - sammeln Informationen
  - Vergleich auf Basis von semantischen Informationen (Semantic Matching)

- Mit RDF und Bezug auf gemeinsames Vokabular (z.B. abgeleitet von HR-XML)

```
<html>
```

```
<head>
```

```
<rdf:RDF xmlns:rdf="...#" xmlns:jpp="...#">  
  <jpp:JobPosting  
    rdf:about="http://www.example.org/jp1.html"/>  
</rdf:RDF>
```

```
</head>
```

```
<body>
```

```
  ...Job posting in free text...
```

```
</body>
```

```
</html>
```

- Suchmaschinen können so Stellenangebote identifizieren

# Reichere Annotation

```
<html>
```

```
<head>
```

```
<rdf:RDF xmlns:rdf="...#" xmlns:jpp="...#" xmlns:skills="...#">
  <jpp:JobPositionPosting
    rdf:about="http://www.example.org/jp1.html"/>
  <jpp:requiredCompetence>
    <skills:Java>
      <skills:hasCompetenceLevel rdf:resource="...#expert"/>
    </skills:Java>
  </jpp:requiredCompetence>
</rdf:RDF>
```

```
</head>
```

```
<body>
```

```
... Job posting in free text ...
```

```
</body>
```

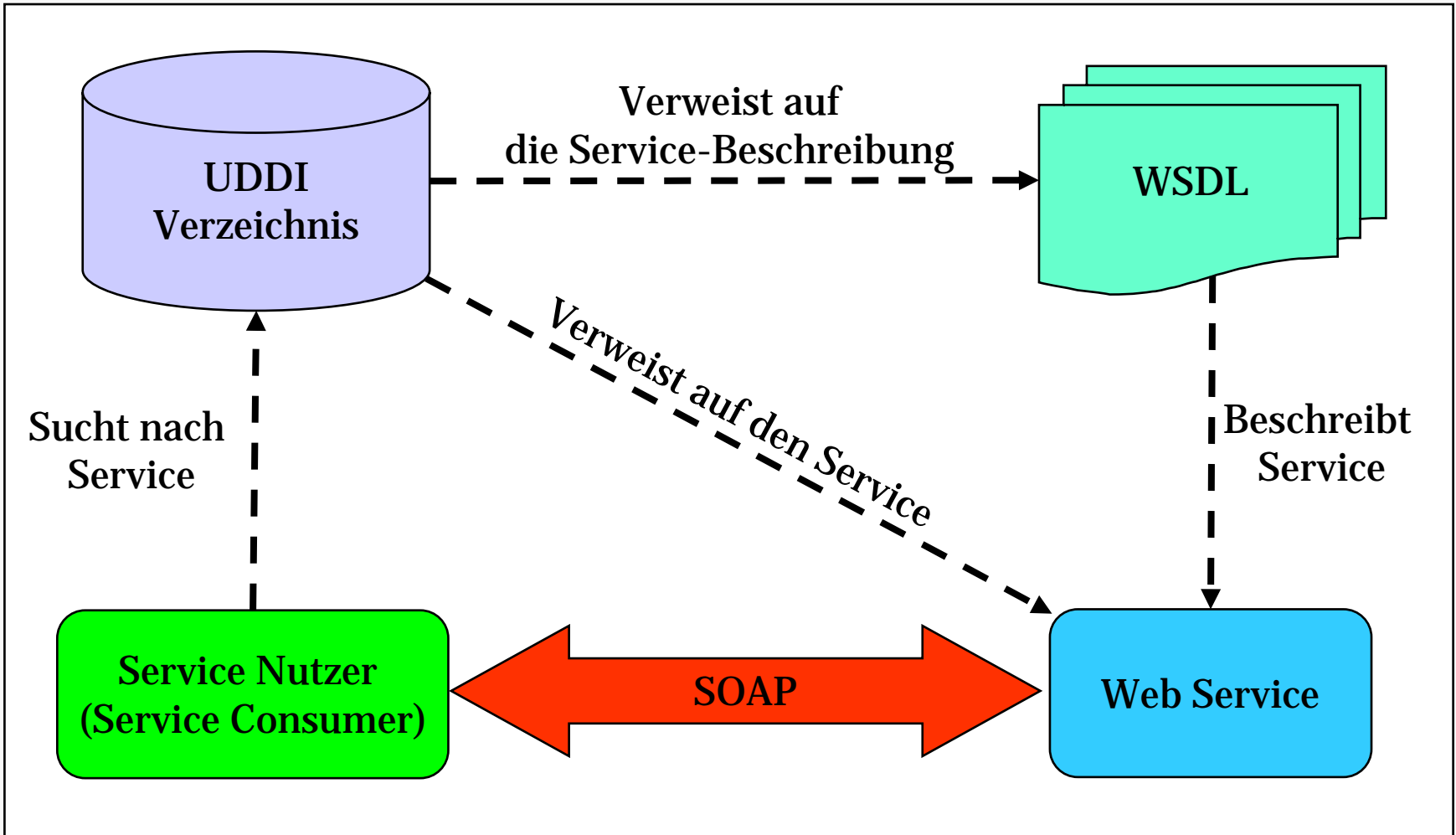
```
</html>
```

# Ersatz von Freitext durch RDF

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:rdf="...#" xmlns:jpp="...#" xmlns:skills="...#">
  <jpp:JobPositionPosting rdf:about="#JobPositionPostingId-inf-44">
    <jpp:hasHiringOrganisation>
      <org:Organisation>
        <org:name>Freie Universität Berlin</org:name>
      </org:Organisation>
    </jpp:hasHiringOrganisation>...
    <jpp:requiredCompetence>
      <skills:Java>
        <skills:hasCompetenceLevel rdf:resource="...#expert"/>
      </skills:Java>
    </jpp:requiredCompetence>...
  </jpp:JobPositionPosting>...
</rdf:RDF>
```

# Semantic Web Services

# Web Services



## Schnittstellen

- Schnittstellen ausgelegt um mit einem und demselben Programm zu kommunizieren

## Standards

- unterschiedliche Standards, da sie von unterschiedlichen Firmen für unterschiedliche Zwecke entwickelt wurden → keine einheitlichen Schnittstellen

## Beschreiben, Auffinden und Nutzen

- Services können sich nicht einem anderen Computerprogramm beschreiben und somit können Computerprogramme Services auch nicht ohne weiteres finden
- ein Service kann einen anderen Service meist nur mit erheblichem implementationstechnischen Aufwand nutzen
- heutige Technologien erlauben keine Automatisierung der Web Services Prozesse

**Semantic Web Services (SWS)**

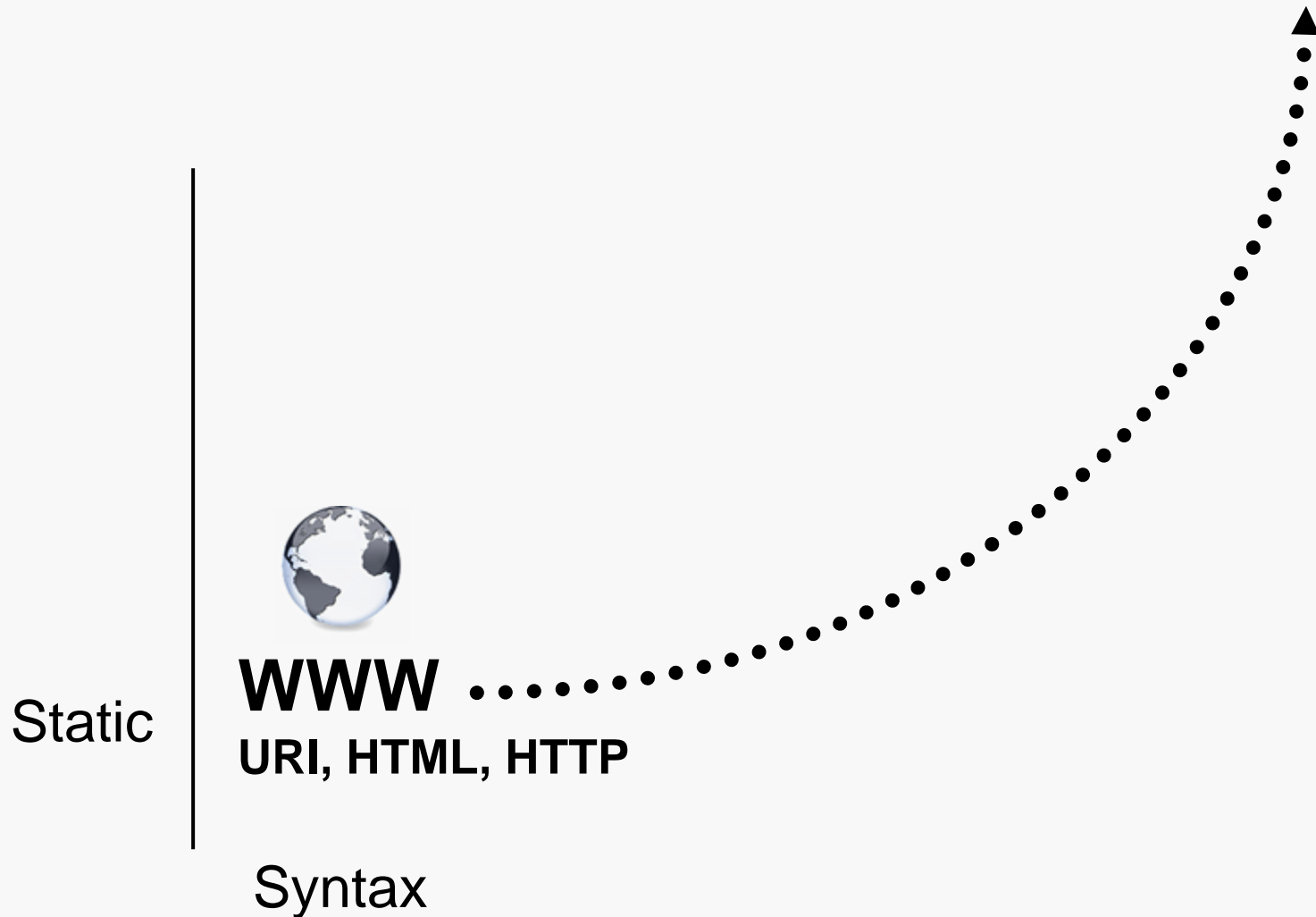
=

**Semantic Web Technology**

+

**Web Service Technology**

# Semantic Web & Web Services – Vision



Statisch



**WWW**  
URI, HTML, HTTP

Syntax

## Probleme mit

- Finden von Informationen (finding)
- Gewinnen von Informationen (extracting)
- Darstellen von Informationen (representing)
- Interpretieren von Informationen (interpreting)
- Pflege von Informationen (maintaining)



**Semantic Web**  
RDF, RDF(S), OWL

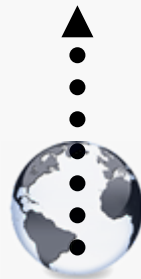
Semantik

# Semantic Web & Web Services – Vision

Dynamisch

## Web Services

UDDI, WSDL, SOAP



Bringing the  
computer back as  
a device for  
computation

Statisch

WWW

URI, HTML, HTTP



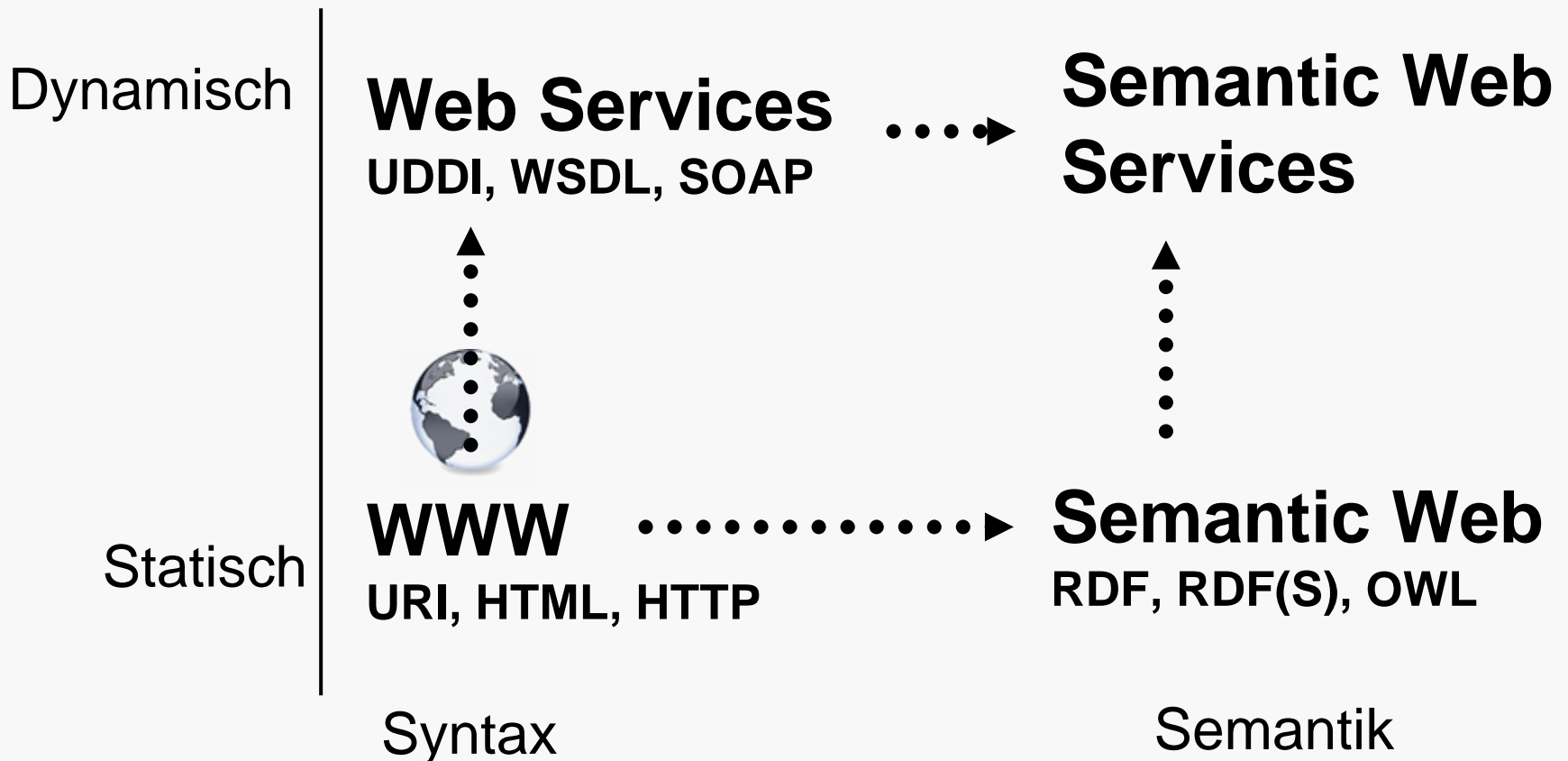
Semantic Web

RDF, RDF(S), OWL

Syntax

Semantik

# Semantic Web & Web Services – Vision



- teilweise oder vollständige Automatisierung des Prozess der Bildung einer SOA durch das Hinzufügen von Semantik
- dynamische Bindung von Nutzer und Anbieter zur Laufzeit
  - Auswahl des besten bzgl. aktuellen Anforderungen Services zur Laufzeit (price vs. delivery time)
  - neue Services können ohne Kode-Änderungen integriert werden
  - Verbesserung der Wiederverwendung
  - Ausfallsicherung im Falle einer Störung
- Aufhebung der Heterogenitätsprobleme zwischen Nutzer und Anbieter auf dem Daten-, Protokoll- und Prozesslevel
- automatischer Service Aufruf

- hoher manueller Aufwand
  - *Lösung: maschinenlesbare Daten um das Finden, Ansprechen und Interpretieren zu automatisieren*
- Web ist dynamisch und unbeständig → Wenn sich ein Service ändert oder nicht verfügbar ist, wird ein Geschäftsprozess unterbrochen
  - *Lösung: (1) dynamisches Auffinden von Services anhand erforderlicher Funktionalität und (2) semantic mediation (Vermittlung) zwischen Erforderlichem und Gefundenem*
- Service Dokumentation in XML und Free Text – kann zur Missverständnissen führen
  - *Lösung: gemeinsames Verständnis von Web Services und ihrer Funktionalität*

# Keine SWS Standards

- nur syntaktische Standards
- keine semantik-basierten Repositories
- keine Standardframeworks um Discovery, Composition and Execution zu erleichtern
- Keine Tools/Plattformen, die semantische Aninreicherung von existierenden Web-Inhalten erlauben

- **Publication** → stellt eine Beschreibung der Einsatzmöglichkeiten eines Services bereit
- **Discovery** → entdeckt passende Web Services
- **Selection** → wählt das am besten geeignete Service zwischen den vorhandene Services
- **Composition** → kombiniert Services
- **Mediation** → löst Daten- und Prozessfehlانpassung (mismatch) zwischen den kombinierten Services auf
- **Execution** → spricht ein Service an

- 4. European Semantic Web Conference 2007 (ESWC2007) <http://www.eswc2007.org/>
- SWS Themen:
  - **Service Beschreibung** → ein Model für ein Service Interface das erlaubt, eine zusätzliche Beschreibung eines Services vom Provider zur Laufzeit abzurufen
  - **Contracting** (auf operational level & semantic level) ermöglicht, das Verhalten von Schnittstellen bzw. Operationen durch Vor- und Nachbedingungen (Kontrakte) zu beschreiben
  - **Discovery** → Integriertes semantic matchmaking für Web Service Discovery

# Prognosen bzgl. SWS

## Technologie-Portfolio zur Tendaussage 4: Das semantische Web ermöglicht den Übergang von Information zu Wissen.

Der Durchmesser der Kreise entspricht der erwarteten wirtschaftlichen Bedeutung der Themen

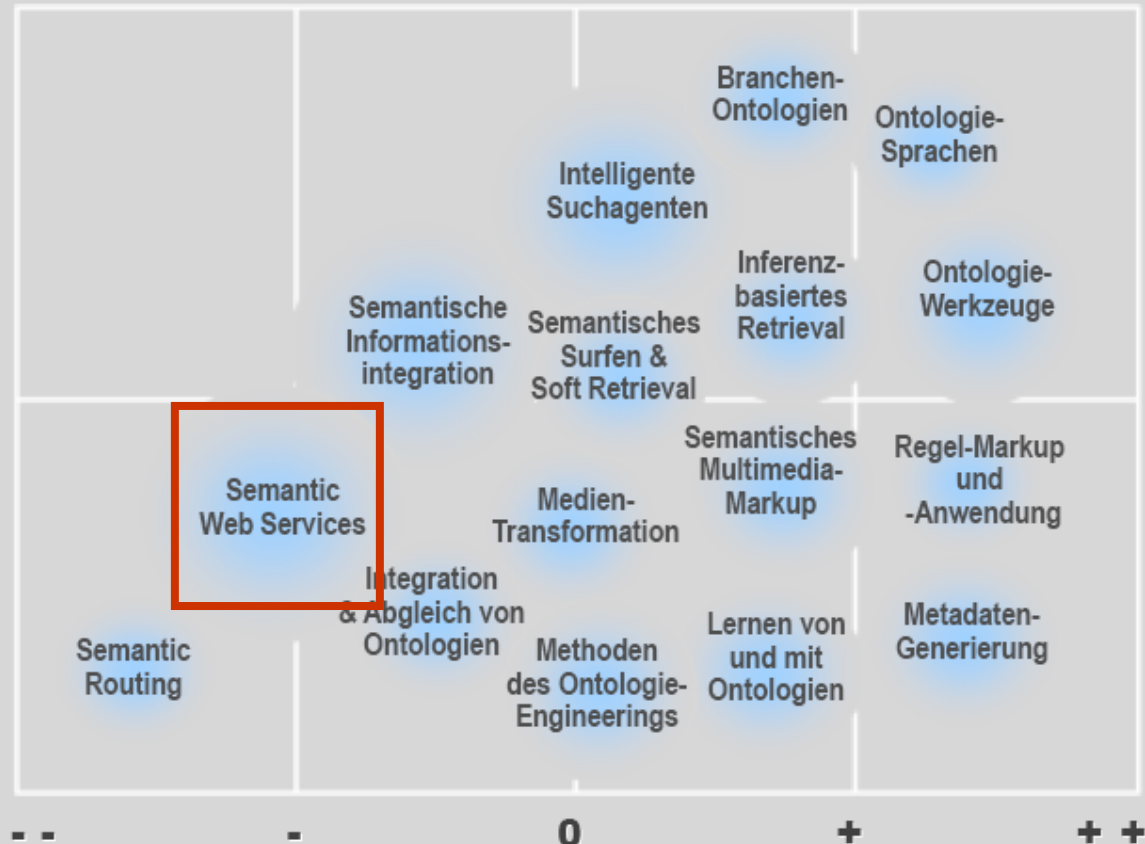


### Anwendungsreife

hoch

mittel

gering



Wettbewerbsstellung von Deutschland

# Wie geht es weiter?

---

## Mi 11.7. 12:15-13:45

- Rückblick (Klausurvorbereitung)

## Mi 11.7. 14:15-16:45

- Sondersprechstunde in der Fabeckstr. 15

## Mi 18.7. 12:00-14:00

- Klausur