



## Web Services in der Praxis & Ausblick

Malgorzata Mochol  
Freie Universität Berlin  
Institut für Informatik  
Netzbasierte Informationssysteme  
[mochol@inf.fu-berlin.de](mailto:mochol@inf.fu-berlin.de)

<b>Vorlesungs- -termine</b>	<b>Vorlesung (4 + 1 + 1)</b>	<b>Übung – 2 Termine</b>	<b>Übungs- termine</b>
10.06.	Web Services, RPCs vs. Messaging		
17.06.	SOAP im Detail	SOAP	24./25.06
24.06.	WSDL im Detail		
01.07. (heute)	Web Services in der Praxis & Ausblick	WSDL	01./02.07
08.07.	Rückblick		
15.07.	Klausur		

## **Web Service Komposition**

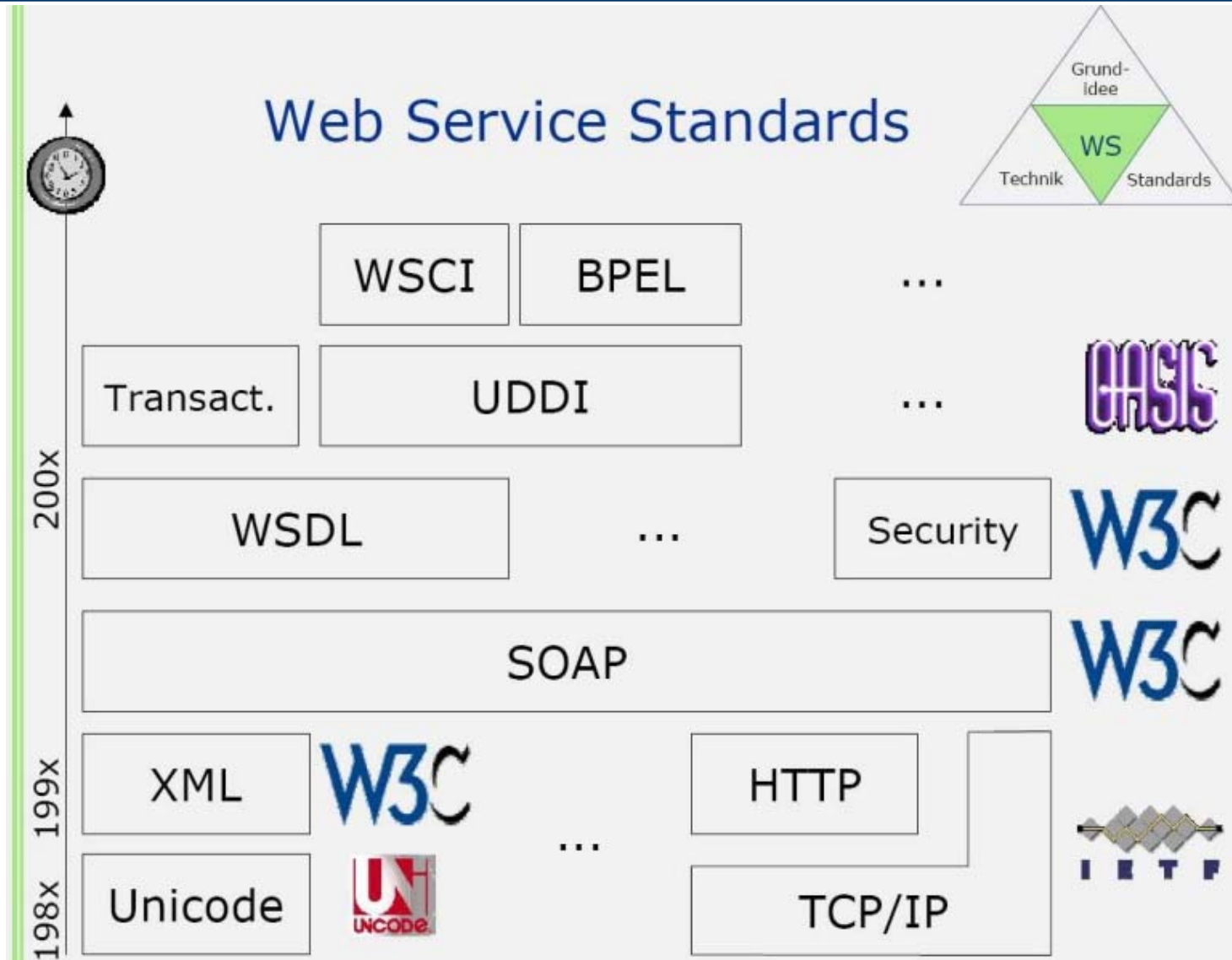
- SOA in der Praxis
- SOA-Sünden

## **Semantic Web**

- Grundidee & Technologien

## **Semantic Web Services**

- Grundidee





## 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

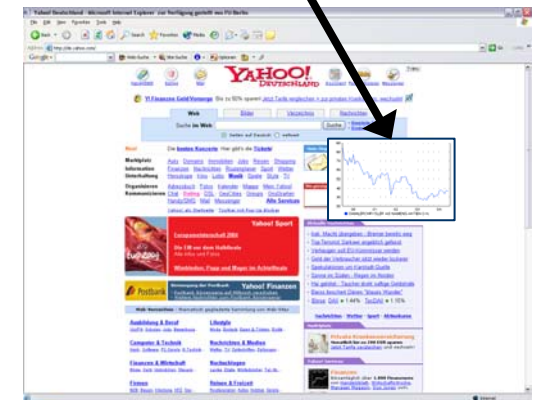
- SOA ist ein **Architekturmodell**
- SOA ist kein Standard
- SOA ist keine Spezifikation
- SOA ist kein Produkt
- SOA Idee: Nutzung bestehender Softwarekomponenten die miteinander interagieren

## Heutige Situation (meistens)

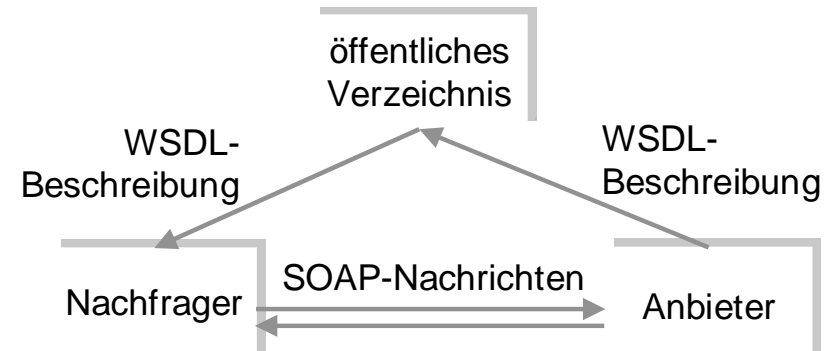
- Unternehmensweite IT Strategie nur fachbereichsspezifisch
- starke Koppelung zwischen verschiedenen Anwendungen
- komplexe IT Infrastruktur
  - → zahllose Systeme mit nur Teilnutzen
- begrenzte Wiederverwendbarkeit der Systeme
  - → fehlende Schnittstellen, Trennung von Schnittstellen- und Businesslogik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

## 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 im Verzeichnis passenden Web Service
2. Suchergebnis: WSDL-Beschreibung
3. WSDL-Beschreibung → Stubs
4. Anwendung ruft Stubs auf

⇒ Dienst muss automatisch gefunden und aufgerufen werden.

- 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

- 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 *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

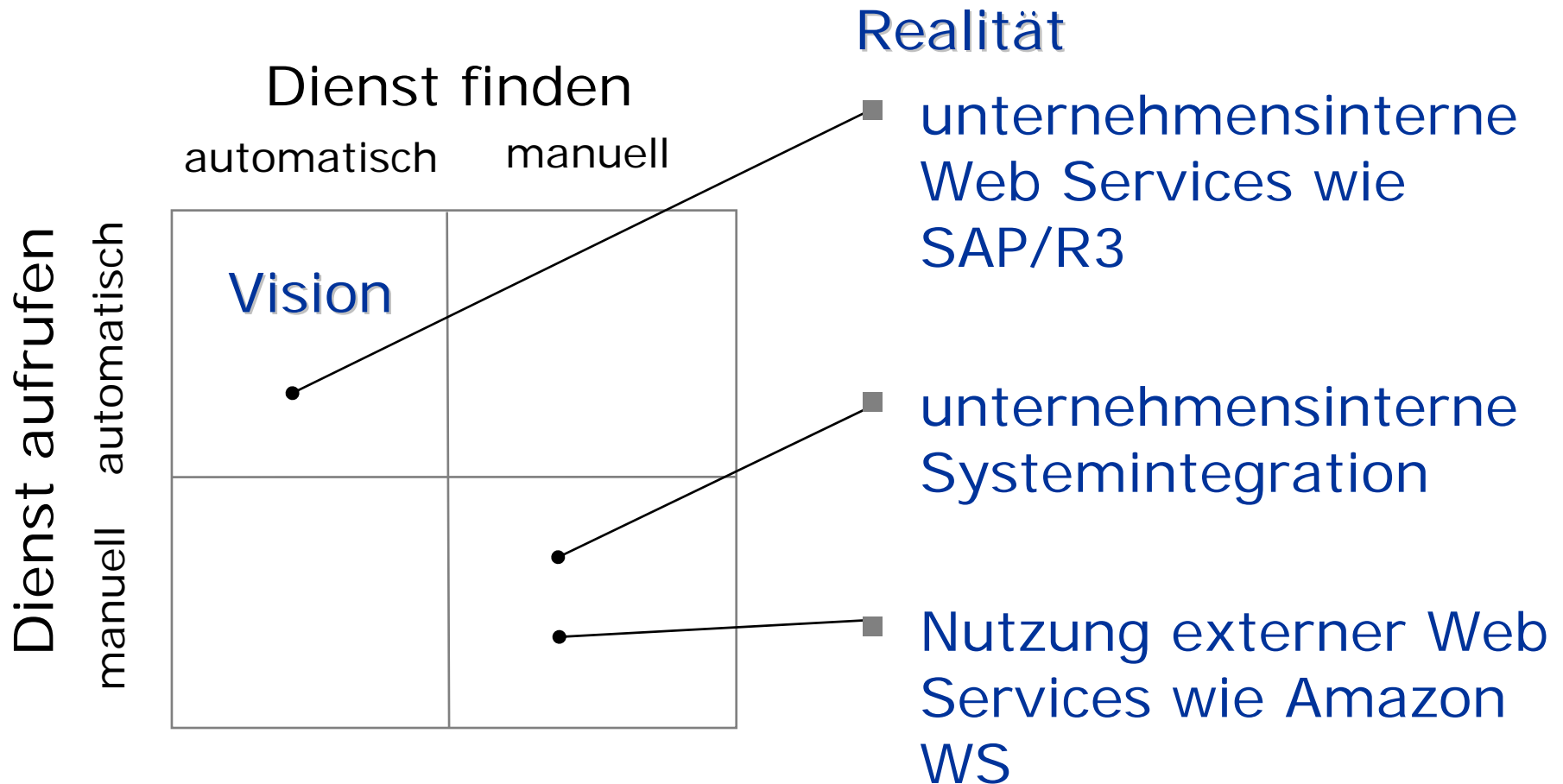
Vision

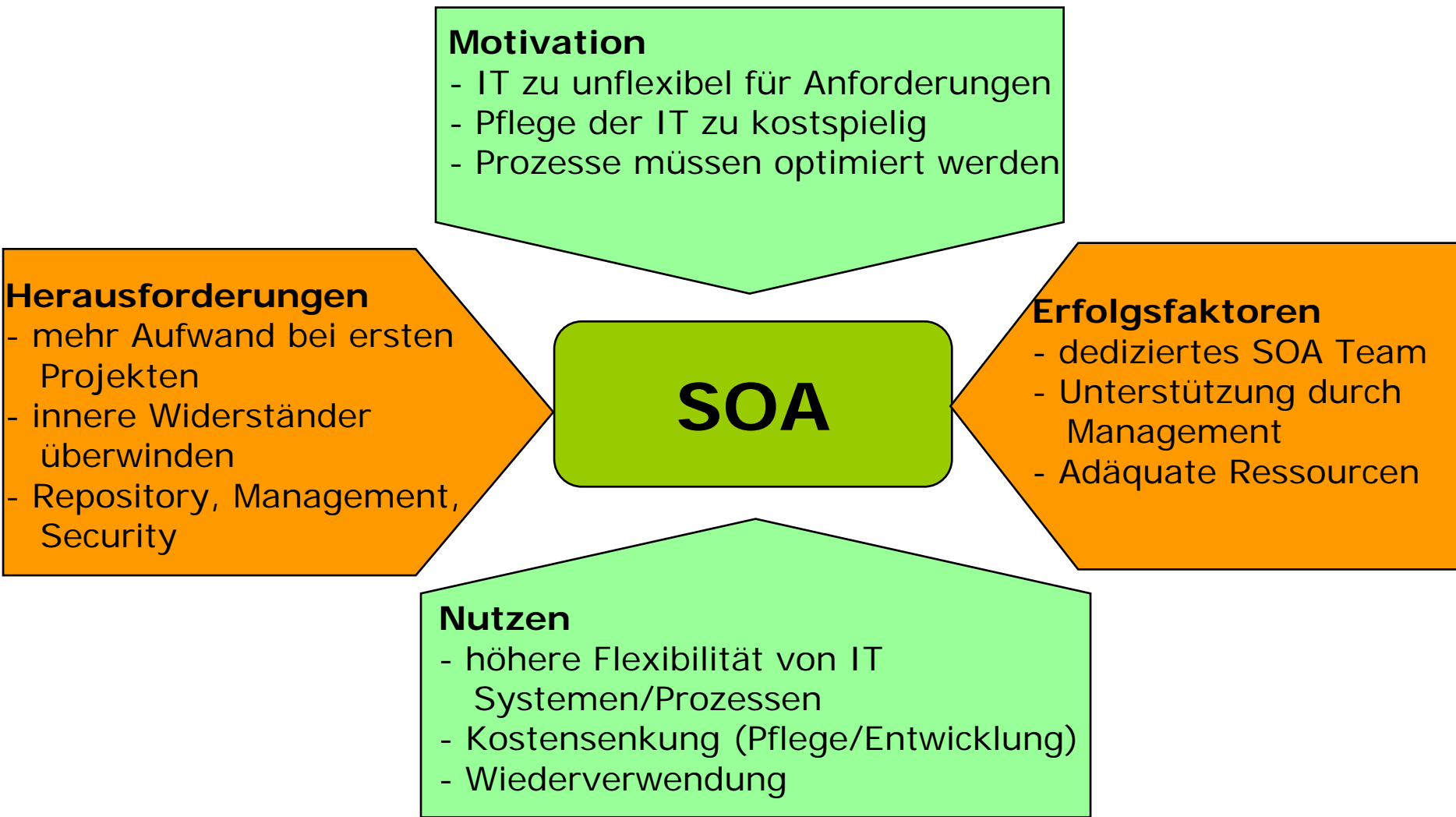
- Schritt 3: Übersetzung WSDL → Stubs erfolgt automatisch
- Schritt 4: Integration von Stubs in Anwendungslogik *erfolgt automatisch*

## manueller Aufruf zur Entwicklungszeit

Praxis

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



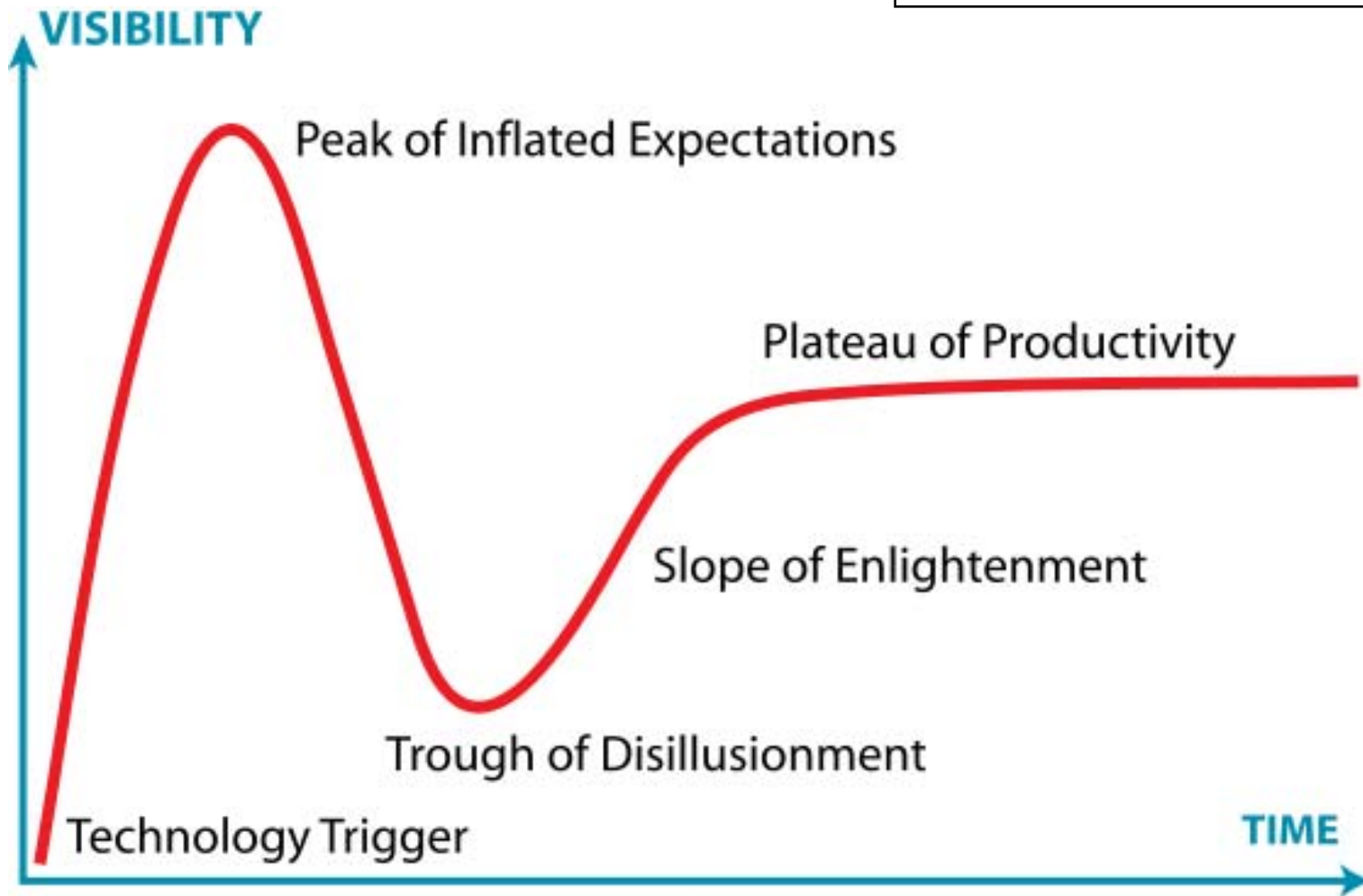


Quelle: SOA Einsatz → Quelle: „SOA in der Praxis – Wie Unternehmen SOA erfolgreich einsetzen“, Berlecon Research GmbH 2006

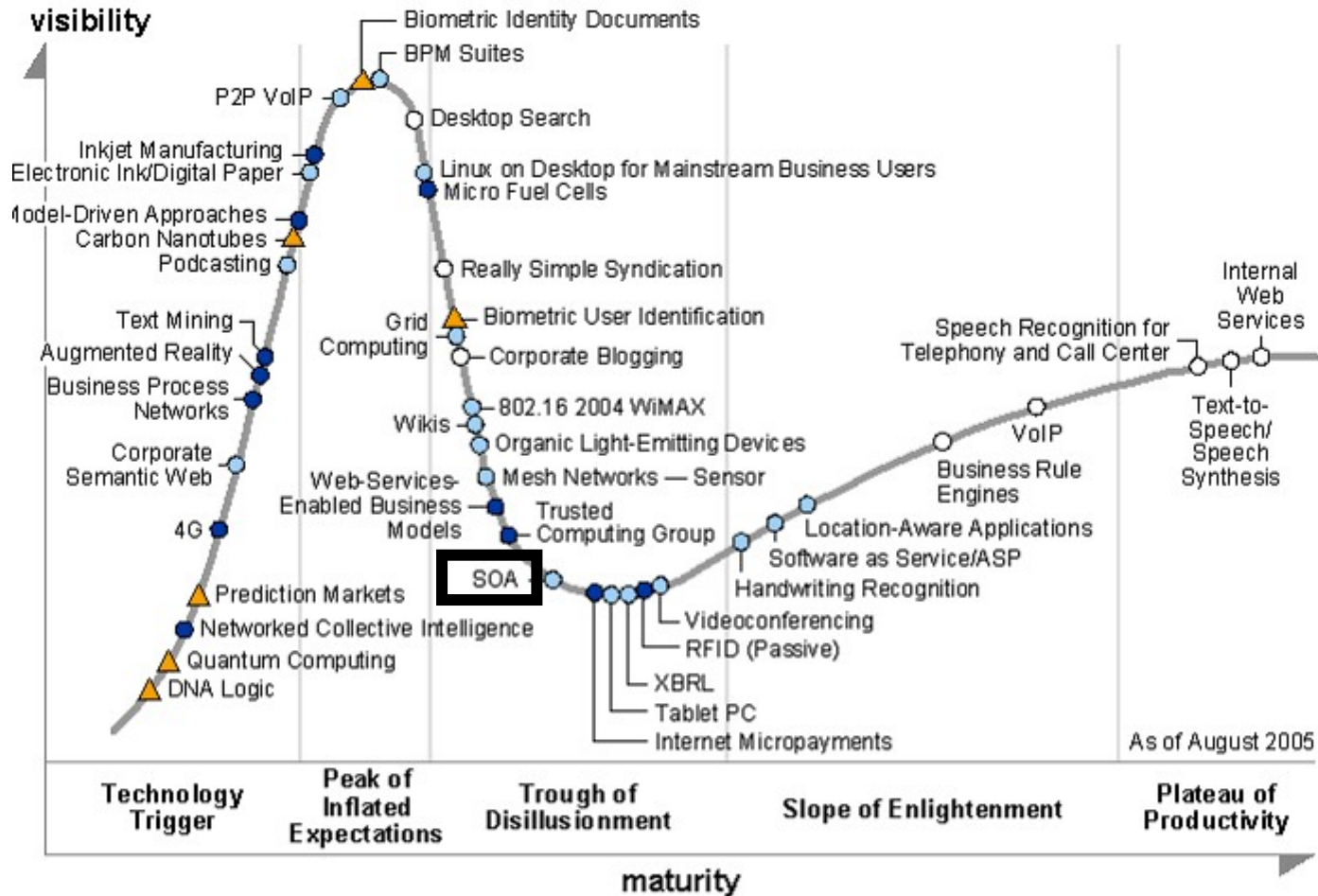
- **Konsolidierung:** Gibt es mehrere Anwendungen, die im Kern das gleiche tun?
- **Umsatzsteigerung/Produktdiversifikation:** Existieren Anwendungen/Daten, die als Standard Service auch für Partner/Kunden einen Nutzen darstellen?
- **Substitution:** Nicht immer sind zusätzliche, unverbundene Kern-Systeme 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

# Exkurs: Gartner Hype Cycle

<http://www.gartner.com/>



# Gartner Hype Cycle 2005 - SOA



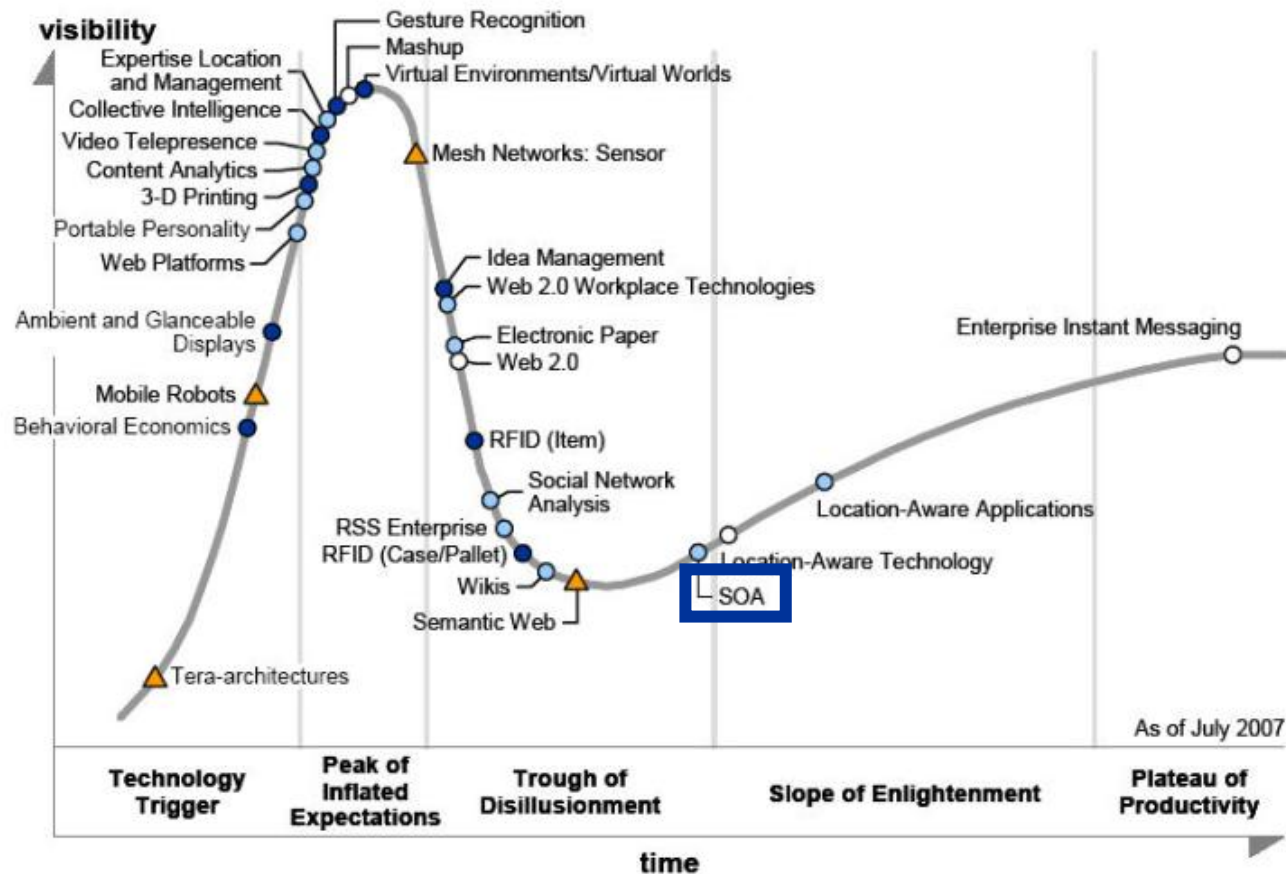
Plateau will be reached in:

- less than 2 years
- 2 to 5 years
- 5 to 10 years
- ▲ more than 10 years
- ⊗ obsolete before plateau

Quelle: Gartner (2005)

# Gartner Hype Cycle 2007 - SOA

Hype Cycle for Emerging Technologies, 2007



As of July 2007

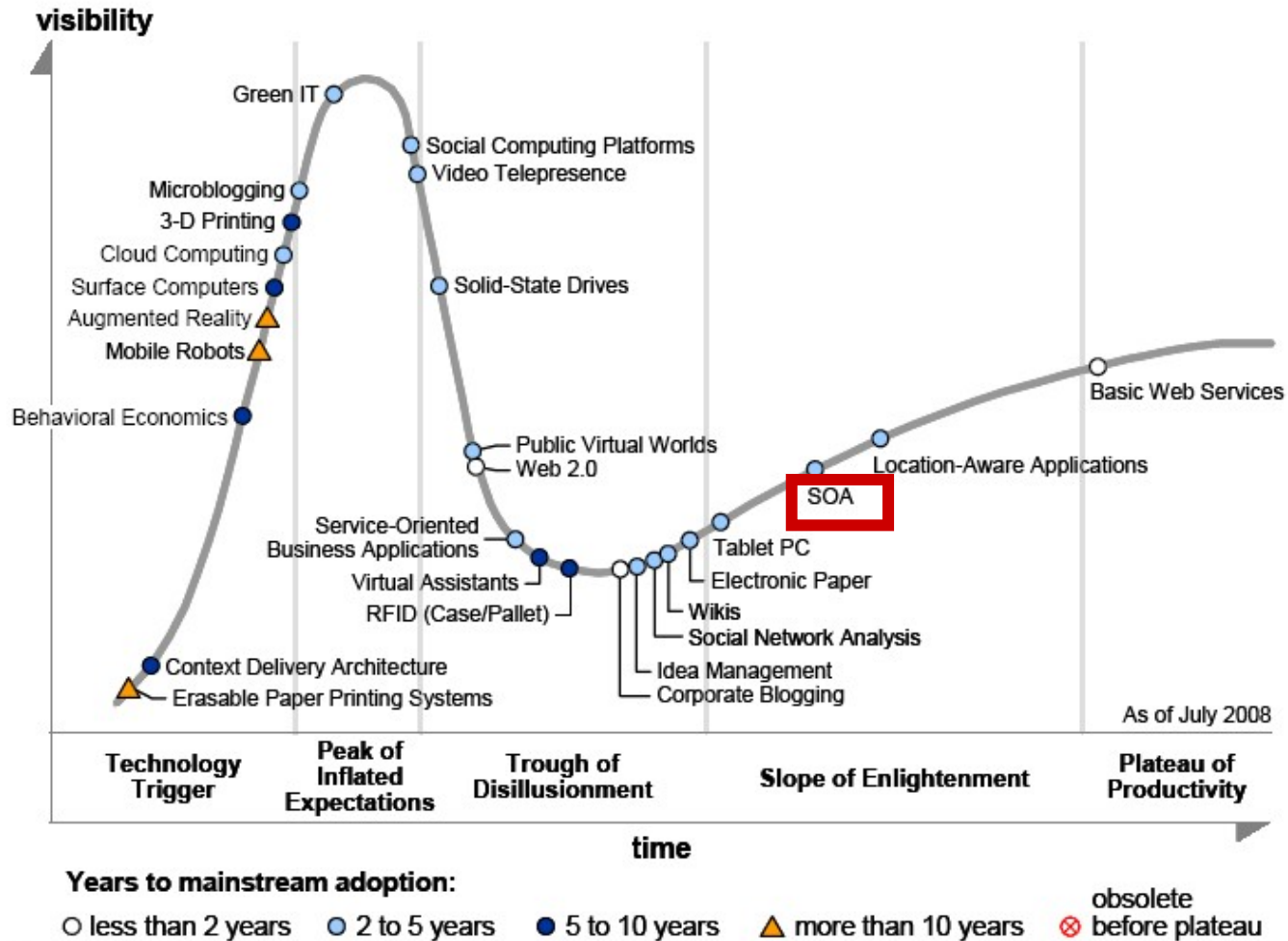
**Years to mainstream adoption:**

- less than 2 years
- 2 to 5 years
- 5 to 10 years
- ▲ more than 10 years
- ⊗ obsolete before plateau

Source: Gartner (July 2007)

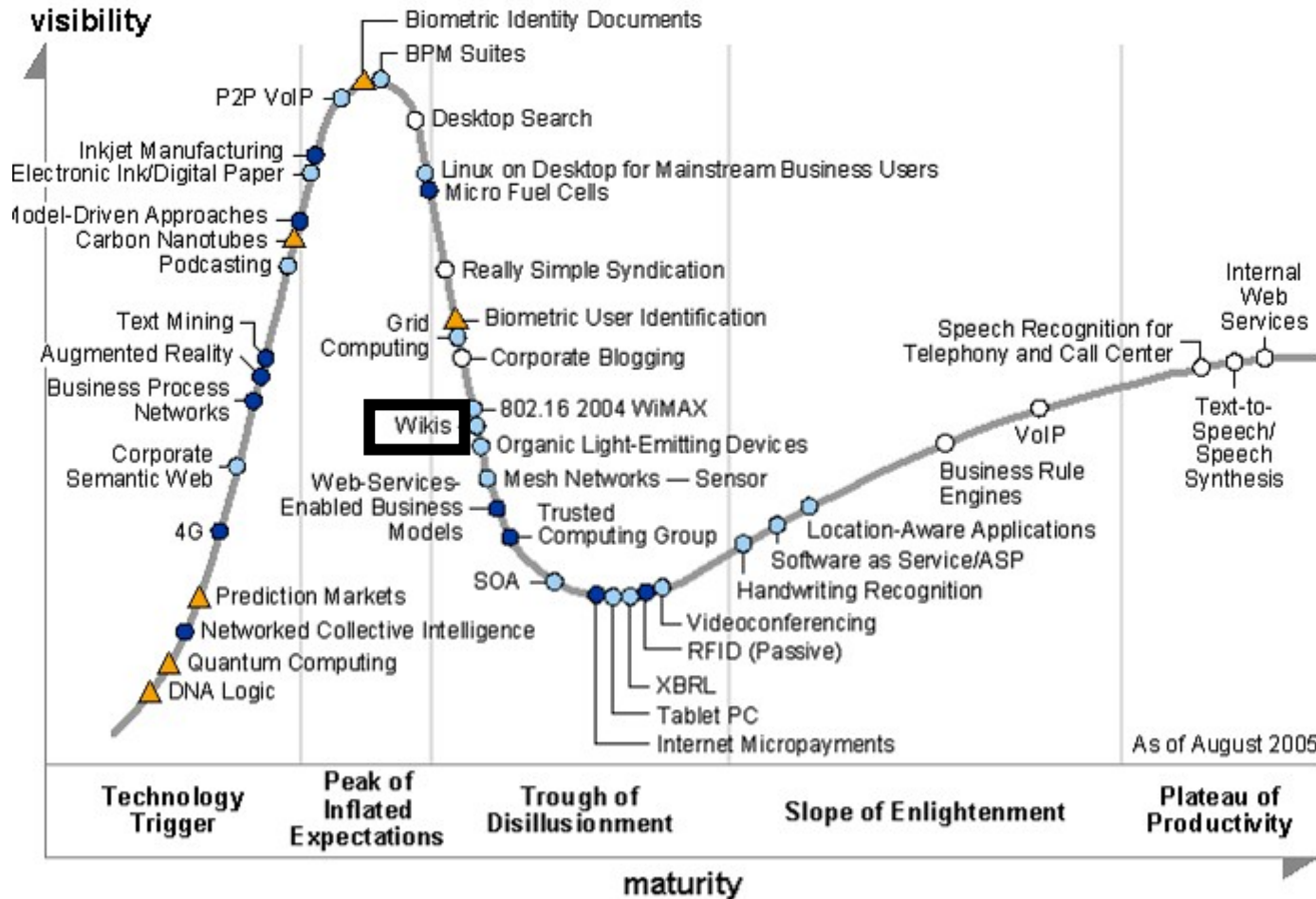
# Gartner Hype Cycle 2008 - SOA

Figure 1. Hype Cycle for Emerging Technologies, 2008



Source: Gartner (July 2008)

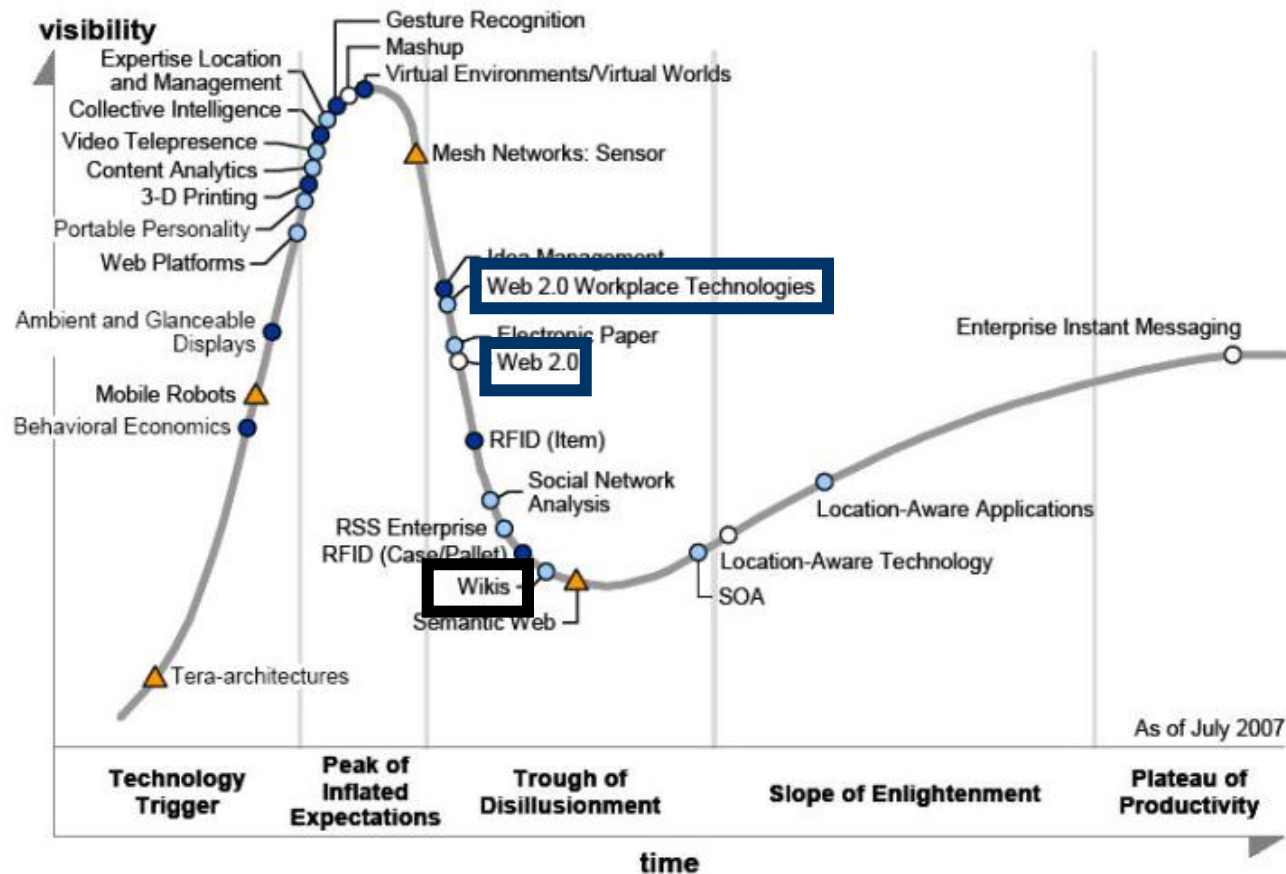
# Gartner Hype Cycle 2005 - Wikis



Quelle: Gartner (2005)

# Gartner Hype Cycle 2007 - Wikis

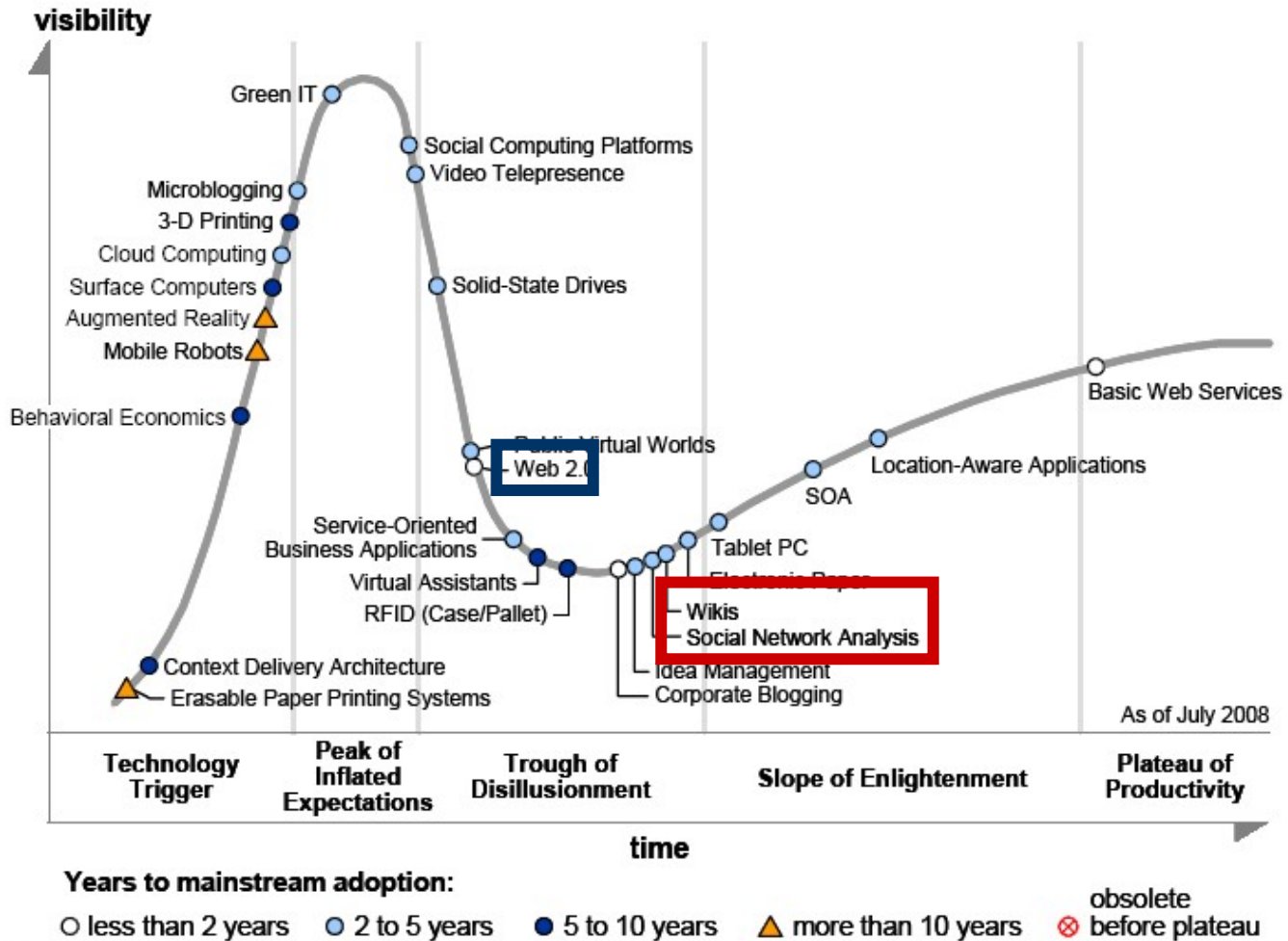
Hype Cycle for Emerging Technologies, 2007



Source: Gartner (July 2007)

# Gartner Hype Cycle 2008 - Wikis

Figure 1. Hype Cycle for Emerging Technologies, 2008



Source: Gartner (July 2008)

## 1. SOA im Überfluss

- Unternehmen setzen SOA-Services mit technischen Funktionen gleich:
  - Repositories voller Services
  - umfangreiche Dokumentationen
  - breite Palette neuer Tools und Middleware
  - SOA-Ziele (Agilität, inkrementelle Software-Versionen, Mehrfachverwendung) lassen sich nicht erreichen
- **Empfehlung**
  - Entwurf von SOA-Services - eigenständigen Schritt im Softwaredesign-Prozess
  - Business-Funktionen statt technische Module

## 2. Vergessene Daten

- Servicemodell zu entwerfen ähnelt dem Design eines Datenmodells.
- Empfehlung
  - beim Entwerfen von Services das Designmodell der zugrundeliegenden Datenbank berücksichtigen

## 3. SOA den Technies überlassen

- SOA-Vorhaben überwiegend in der IT-Organisation an
- Gefahr:
  - Services vor allem mit Blick auf Leistungsoptimierung und Ausfallsicherheit konzipiert werden
  - Anforderungen der Fachabteilungen unberücksichtigt
- **Empfehlung**
  - SOA Design - gemeinsame Herausforderung für Business & IT

## 4. Kulturelle Hürden übersehen

- "Not-Invented-Here"-Syndrom
- Empfehlung
  - Mehrfachverwendung von Software belohnen
  - Förderung der entsprechenden Umfeld: Wiederverwendung als anzustrebendes Charakteristikum exzellenter Softwareentwicklung

## 5. Zu groß in die SOA einsteigen

- breit angelegte SOA-Initiative ohne gründliche Vorbereitung und Planung birgt erhebliche Risiken
- Empfehlung
  - Strategisch Denken & Handeln
  - Entwicklung einer langfristigen SOA-Vision aber Umsetzung in kleinen Schritten

## 6. Am falschen Ort beginnen

- opportunistisches Vorgehen
- Empfehlung
  - kurzfristig entwickelte Services aus opportunistischen Motiven vermeiden
  - geschlossenes Set aus grundlegenden Services

## 7. Das eigene SOA-Verständnis zum Maßstab machen

- Unterschiedliche Sichten auf SOA
- Empfehlung
  - Unterschiede bei der Kommunikation der SOA-Strategie berücksichtigen

## 8. Anarchie mit Diktatur bekämpfen

- Abteilungen und Projektgruppen werden gezwungen, sich an zentrale Vorgaben zu halten
- Empfehlung
  - SOA Centre of Excellence (COE) gründen, um einzelne Projekte zu koordinieren

## 9. Technische Probleme unterschätzen

- SOA-Verantwortliche müssen die komplexe Welt der Middleware verstehen
- Empfehlung
  - Punkt-zu-Punkt-Verbindungen mit Web-Services nur für kleine, experimentelle SOA-Projekte

## 10. Service ohne Wiederverwendung zulassen

- konsumiert eine Anwendung im Durchschnitt deutlich mehr als 20 Services oder
- werden weniger als zehn Prozent der Services mehrfach verwendet
  - der Grad der Wiederverwendung nicht optimal
- Empfehlung
  - formaler Prozess für die Servicedefinition und –validierung

## 11. Übertriebene Zentralisierung

- alles in einem, alles zentral: eine zentrale Registry, ein vorgegebenes Set aus Prozessen, usw.  
→ zahlreiche technische, organisatorische und politische Probleme erwachsen
- Empfehlung
  - föderierter SOA-Ansatz

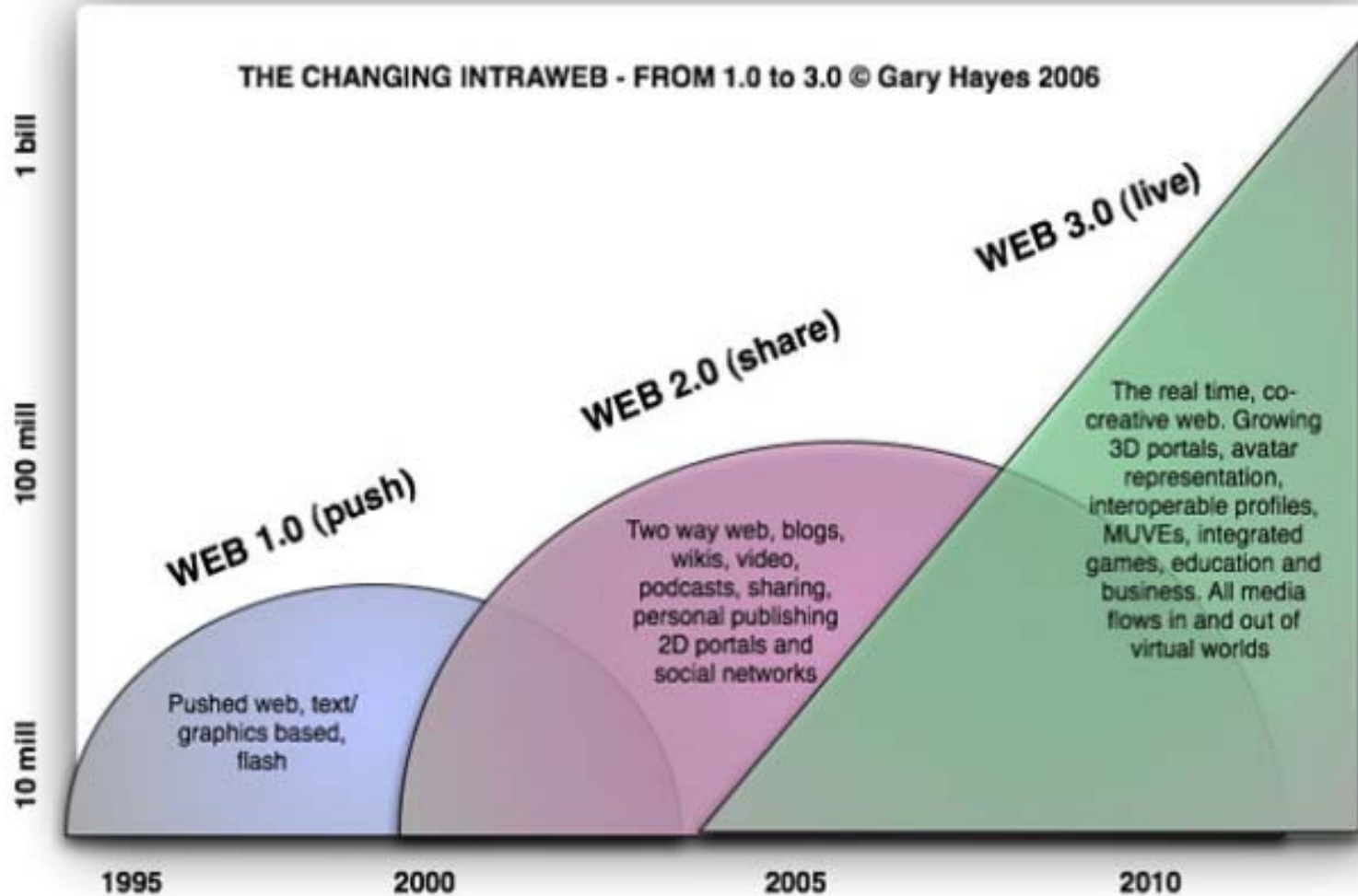
## 12. SOA zu früh verkaufen

- um eine SOA unternehmensweit einzuführen, bedarf es der Unterstützung aus dem Topmanagement
  
- Empfehlung
  - Konzentration auf kleine Vorhaben

*„Gartner nennt die zwölf SOA-Todsünden“, Wolfgang Herrmann, 07.12.2007*  
Quelle: [http://www.computerwoche.de/knowledge\\_center/soa\\_bpm/1849999/](http://www.computerwoche.de/knowledge_center/soa_bpm/1849999/)



## Evolution des Webs – Semantic Web



*Quellen:*

- „Virtual Worlds, Web 3.0 and Portable Profiles“, <http://www.personalizemedia.com/virtual-worlds-web-30-and-portable-profiles>, 2006
- „The Virtual Worlds Hype Cycle for 2009“, <http://www.muvedesign.com/the-virtual-worlds-hype-cycle-for-2009>, 2009

# 10 Future Web Trends (2007)

1. Semantic Web
2. Künstliche Intelligenz
3. Virtuelle Welten
4. Mobile
5. Attention Economy
6. Web Seiten als Web Services
7. Online Video/Internet TV
8. Reichhaltige Internet Applikationen
9. Internationales Web
10. Personalisierung

Quelle: [http://www.readwriteweb.com/archives/10\\_future\\_web\\_trends.php](http://www.readwriteweb.com/archives/10_future_web_trends.php)

- **Syntax** – die Art und Weise, wie Worte in einem Satz zusammengesetzt wurden.
- **Semantik** – Informationen, die in diesem Sinne kodiert wurden.
- **Pragmatik** – Implikationen aus den Informationen in einem Kontext.

Quelle: [http://www.web2open.org/presentations/OASIS\\_SOA\\_Adobe\\_Semantics.pdf](http://www.web2open.org/presentations/OASIS_SOA_Adobe_Semantics.pdf)

“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“



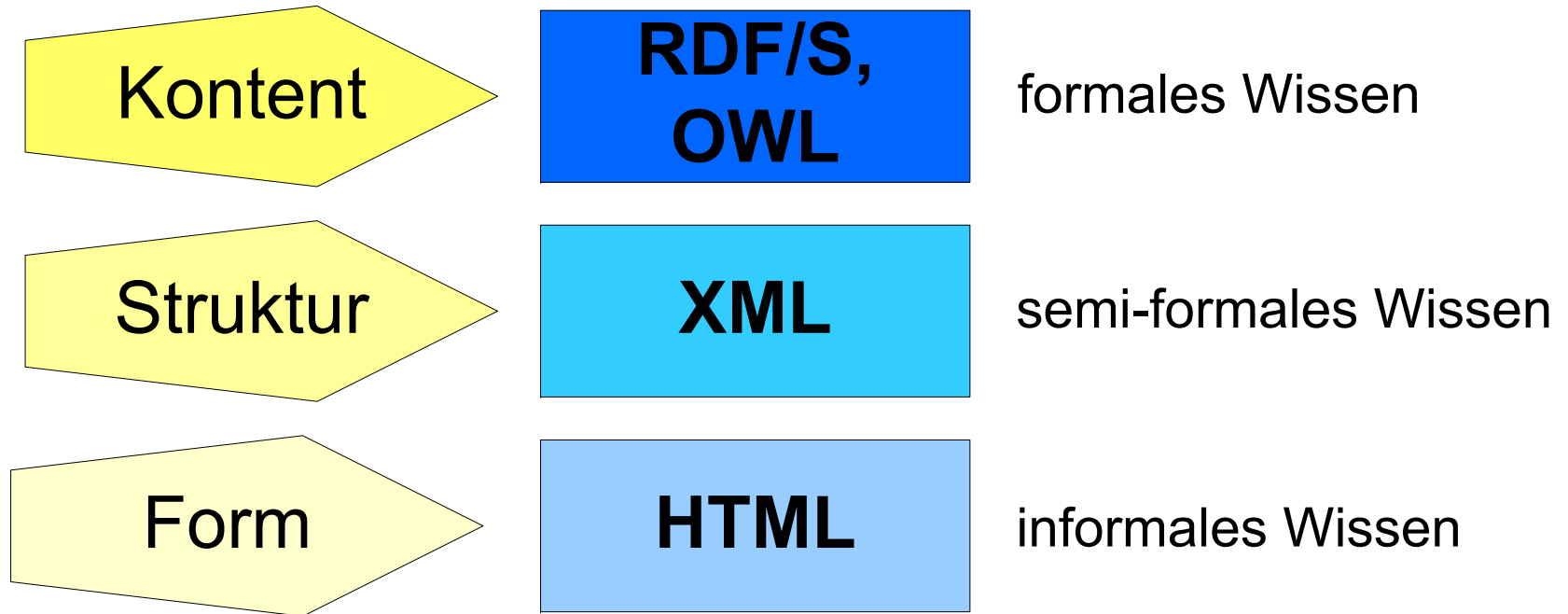
- Maschinen fehlt dieser Kontext aus Begriffen und Zusammenhängen
- Kontext muss Maschinen zusätzlich bereitgestellt werden

- **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

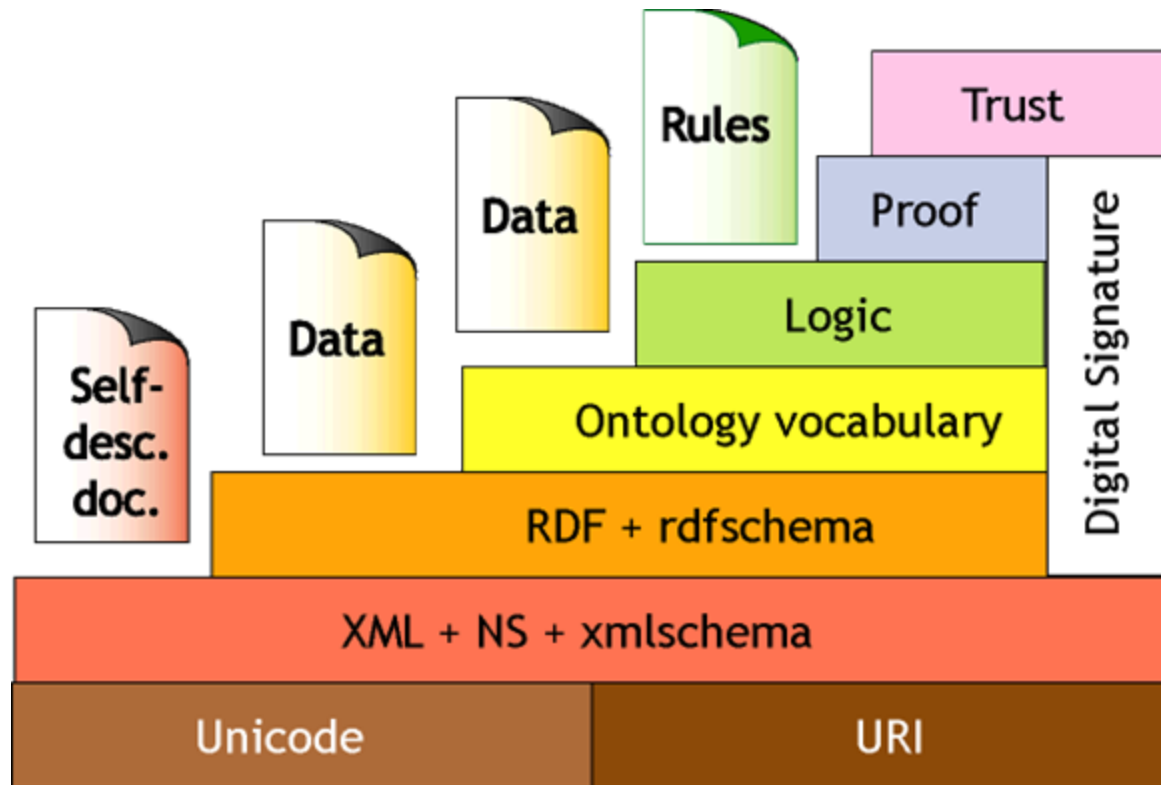
- Webinhalte und ihre Vernetzung werden für Maschinen verständlich.
- Auch komplexe Anfragen können ans Web gestellt werden.
- Beispiel: Mit welchen Kollegen arbeitet der Autor eines bestimmten Dokumentes zur Zeit zusammen?



# 3 Levels von Markup im Web



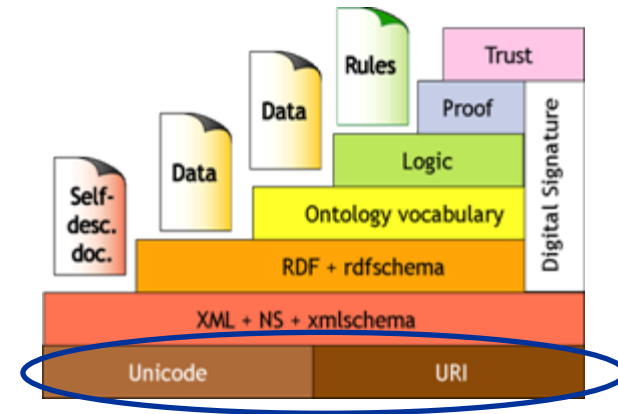
# Semantic Web Stack (W3C, 2000)



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

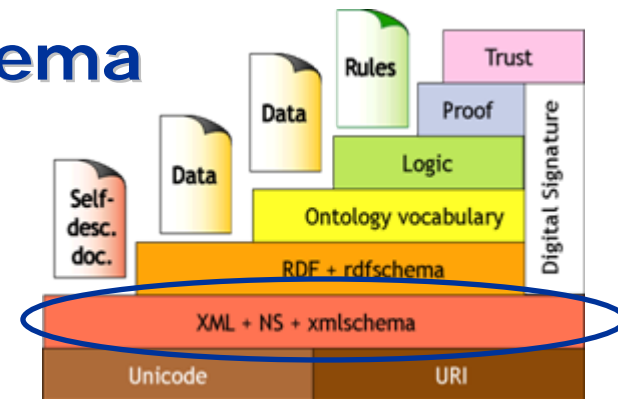


## URI – Uniform Resource Identifier

- 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

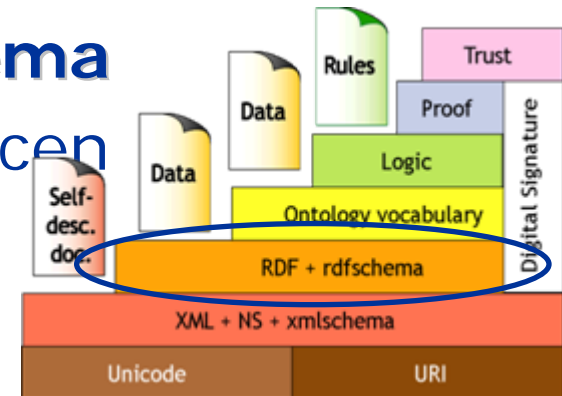
- hierarchisch strukturierte, medienneutrale Daten
- Vokabular kann mit XML-Schema definiert werden
- Bedeutung des Vokabulars kann mit Namensräumen festgelegt werden
- XML-Daten können mit XLink verlinkt werden: Links können Namen, aber keinen Namensraum haben



⇒ 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>

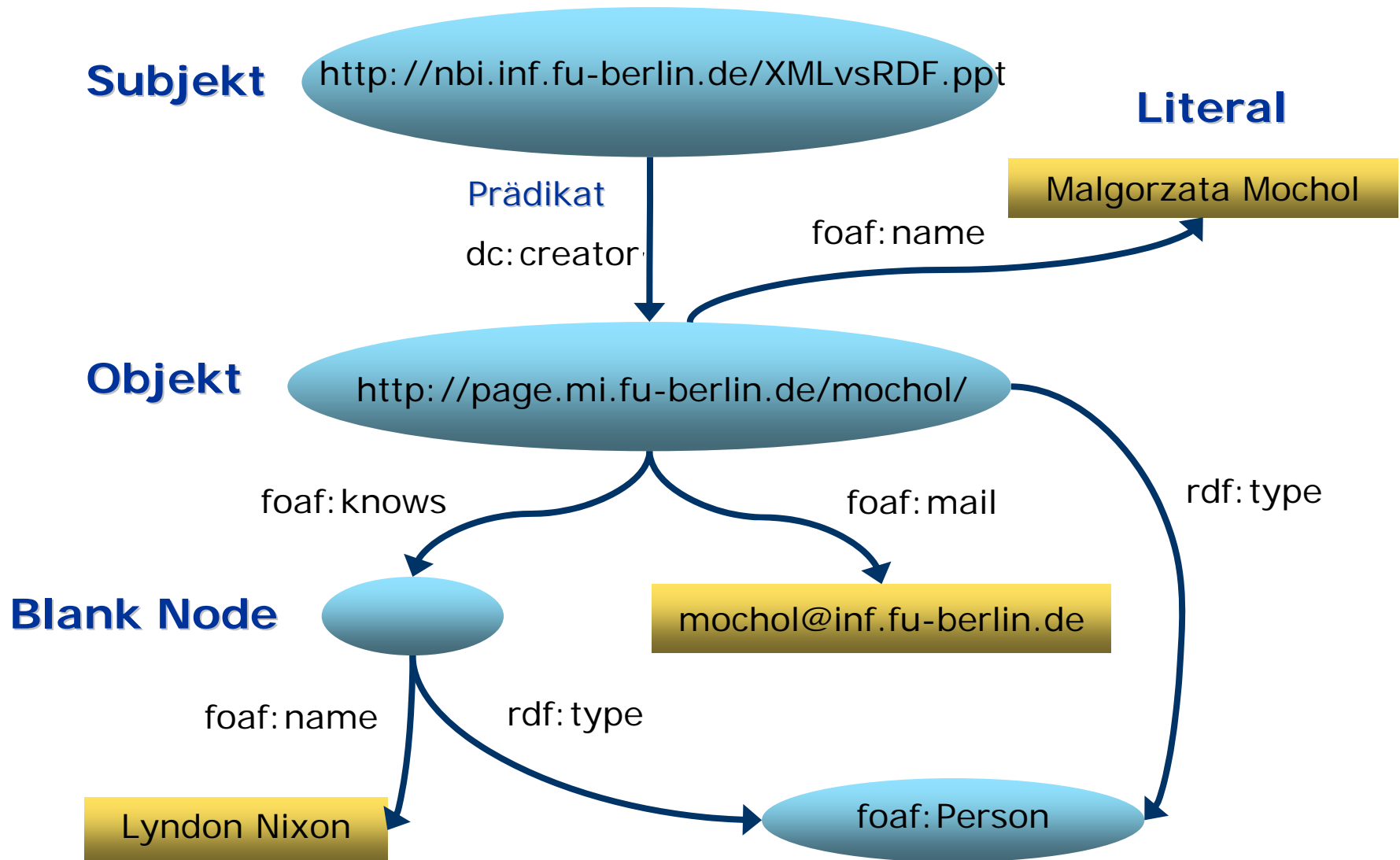
URI-s steht zu URI-o in der Beziehung URI-p

- RDF Statement – die kleinste Informationseinheit, die ein Fakt darstellt

- Beispiel:

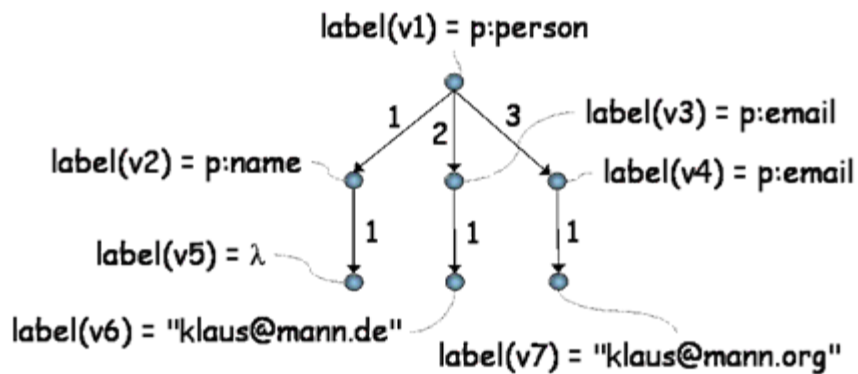
*This presentation was created by Malgorzata Mochol*

- Subject (Ressource): *This presentation*
  - Predicate (Property): *creator*
  - 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/>



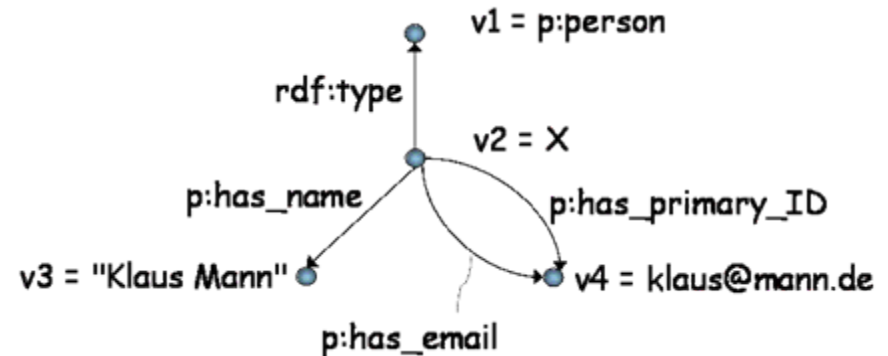
## XML

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

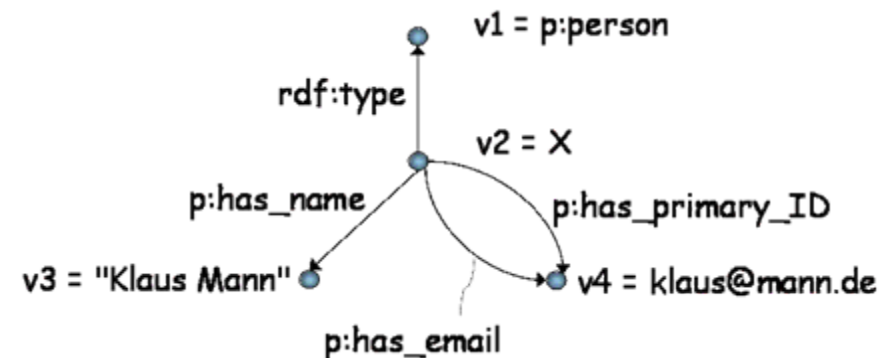
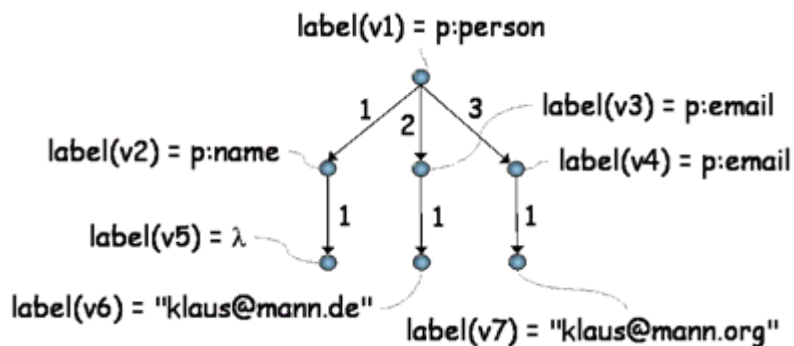


## RDF

```
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
```



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



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

CWA:

explizit erlaubt

nicht zulässig

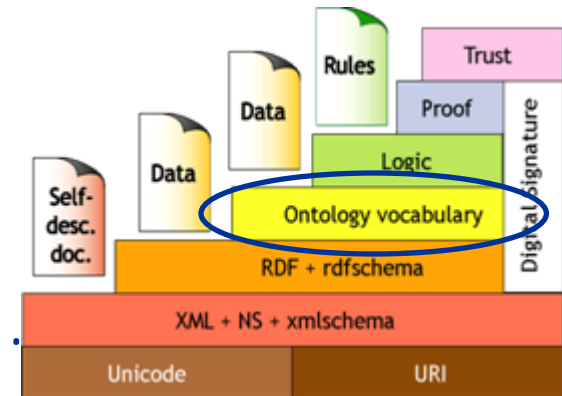
OWA:

Randbedingungen nicht erfüllbar

zulässig

## 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)



## Logik

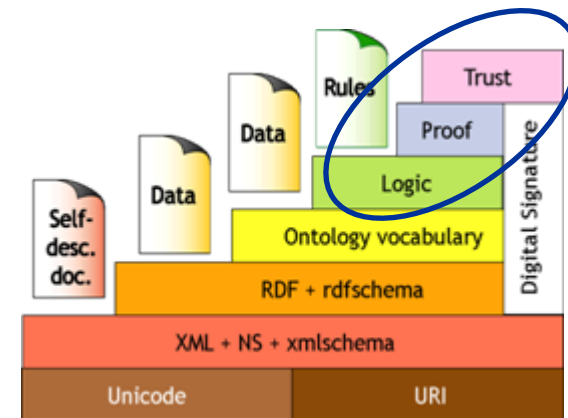
- Semantik auf logischer Basis
- Ableitungsregeln

## Proof

- Konsistenz
- Ableitung (Inferenz)

## Trust

→ Immer noch in der Forschung



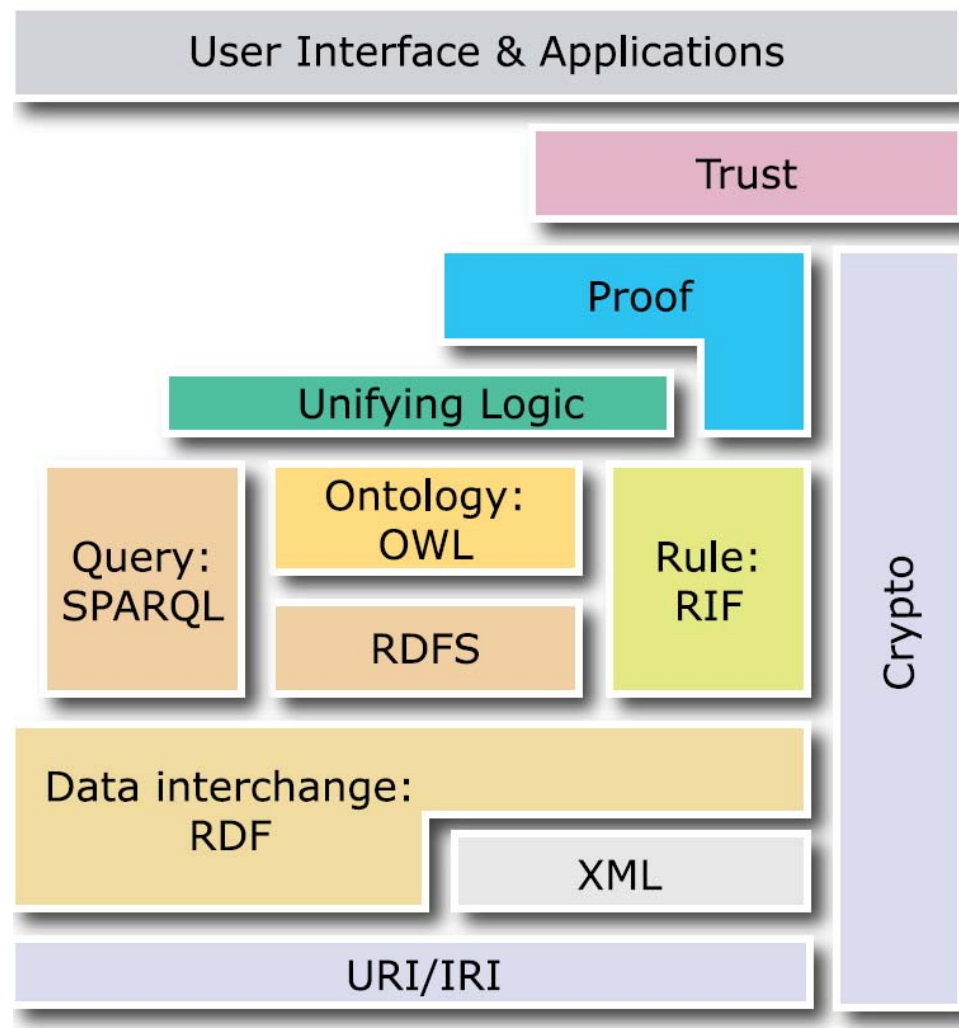
## XML

- XML heute omnipräsent, wenn auch nicht immer sichtbar

## RDF

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

# Semantic Web layer cake 2007



Quelle: Steve Bratt. Semantic Web, and Other Technologies to Watch. <http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb>, 2007



**Semantic Web → Beispiel**

# Beispiel: e-Recruitment Szenario

## 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:JobPositionPosting
        rdf:about="http://www.example.org/jp1.html"/>
    </rdf:RDF>
  </head>
  <body>
    ...Job posting in free text...
  </body>
</html>
```

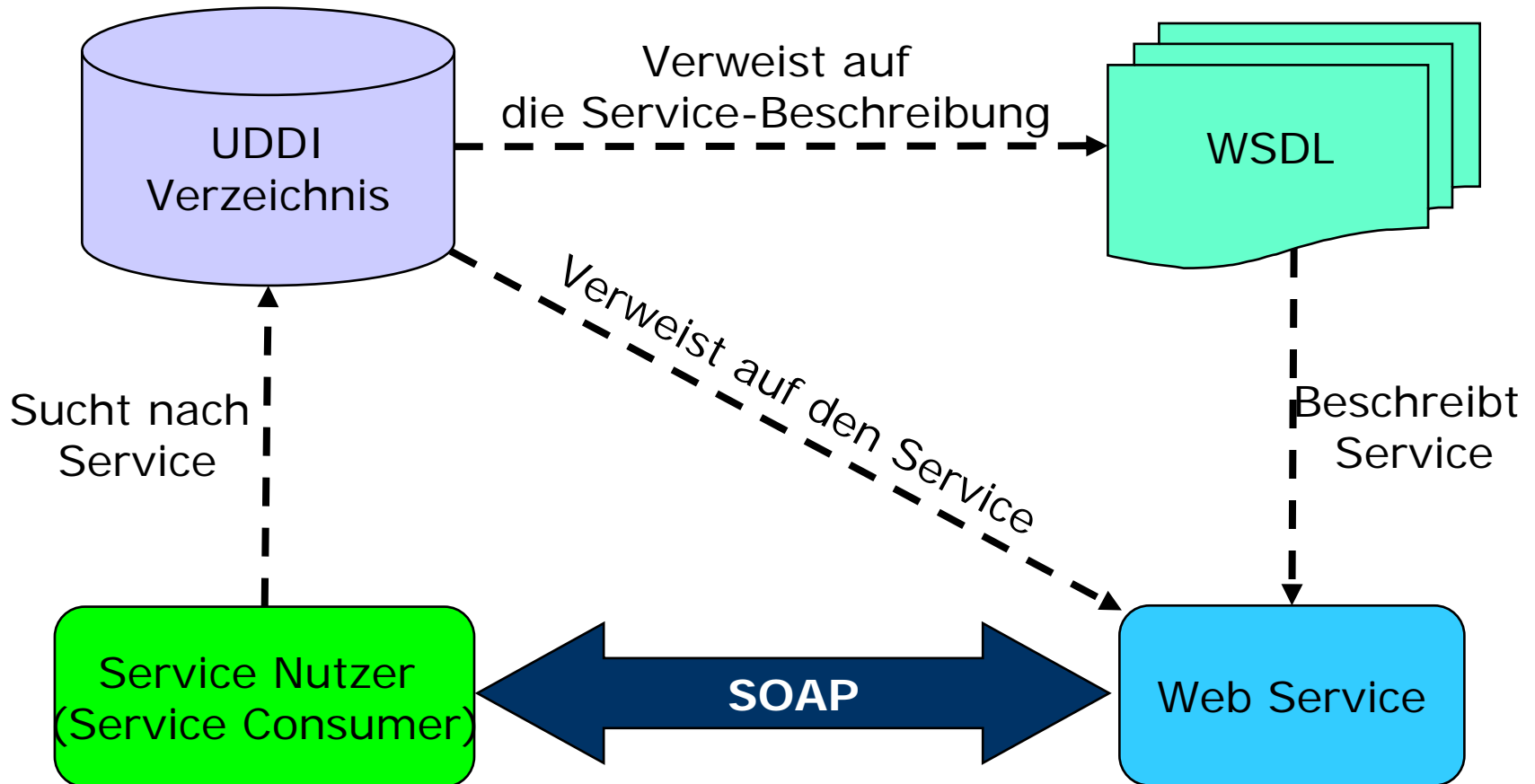
- Suchmaschinen können so Stellenangebote identifizieren

```
<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>
```

```
<?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



## Schnittstellen

- Schnittstellen ausgelegt um mit einem und demselben Programm zu kommunizieren

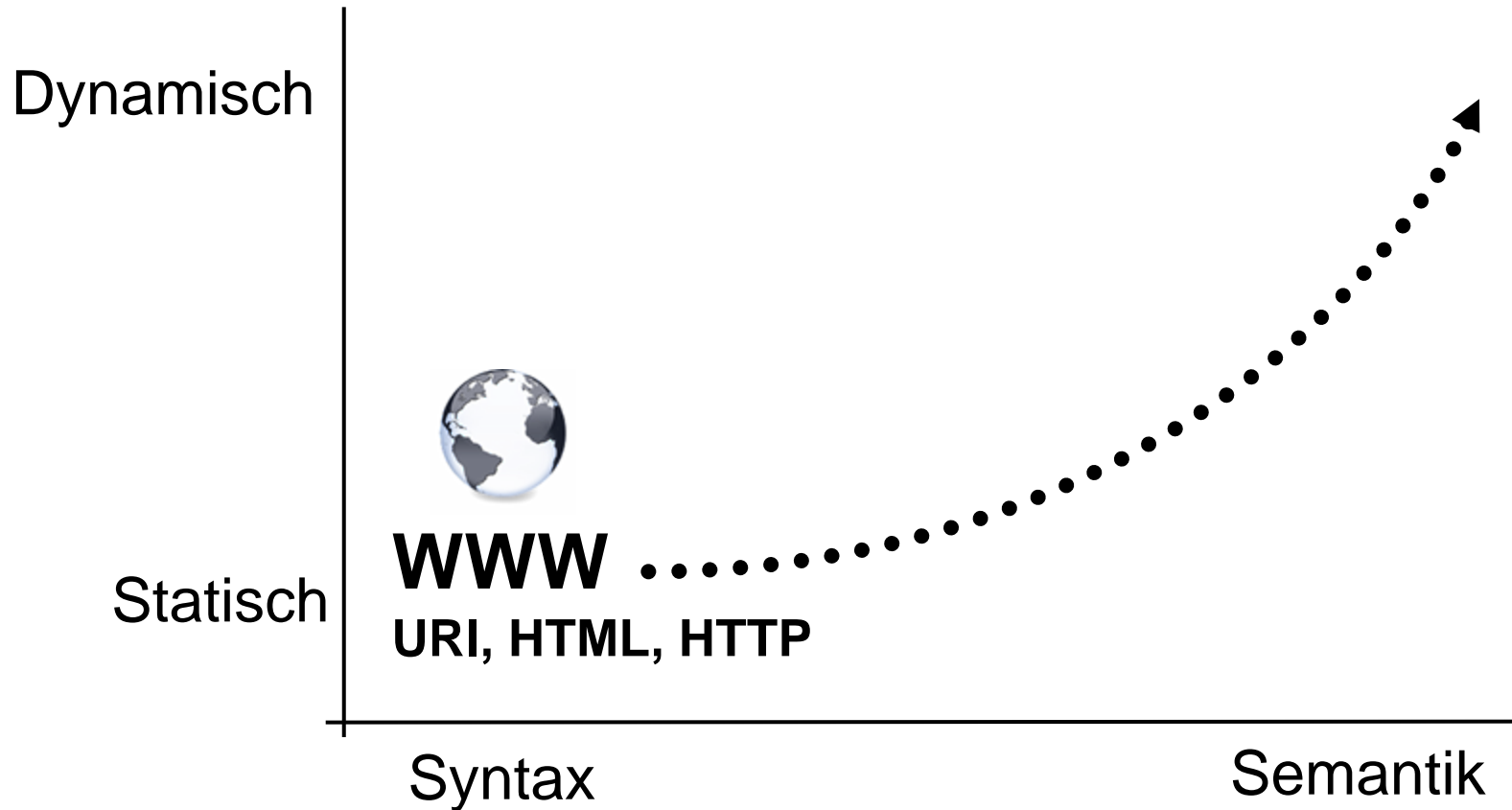
## Standards

- 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 → Anwendungen können Services auch nicht ohne weiteres finden
- ein Service kann einen anderen Service meist nur mit erheblichem implementations-technischem Aufwand nutzen
- heutige Technologien erlauben **keine Automatisierung der Web Services Prozesse**

Semantic Web Services (SWS)  
=  
Semantic Web Technology  
+  
Web Service Technology



Dynamisch



**WWW**  
URI, HTML, HTTP



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

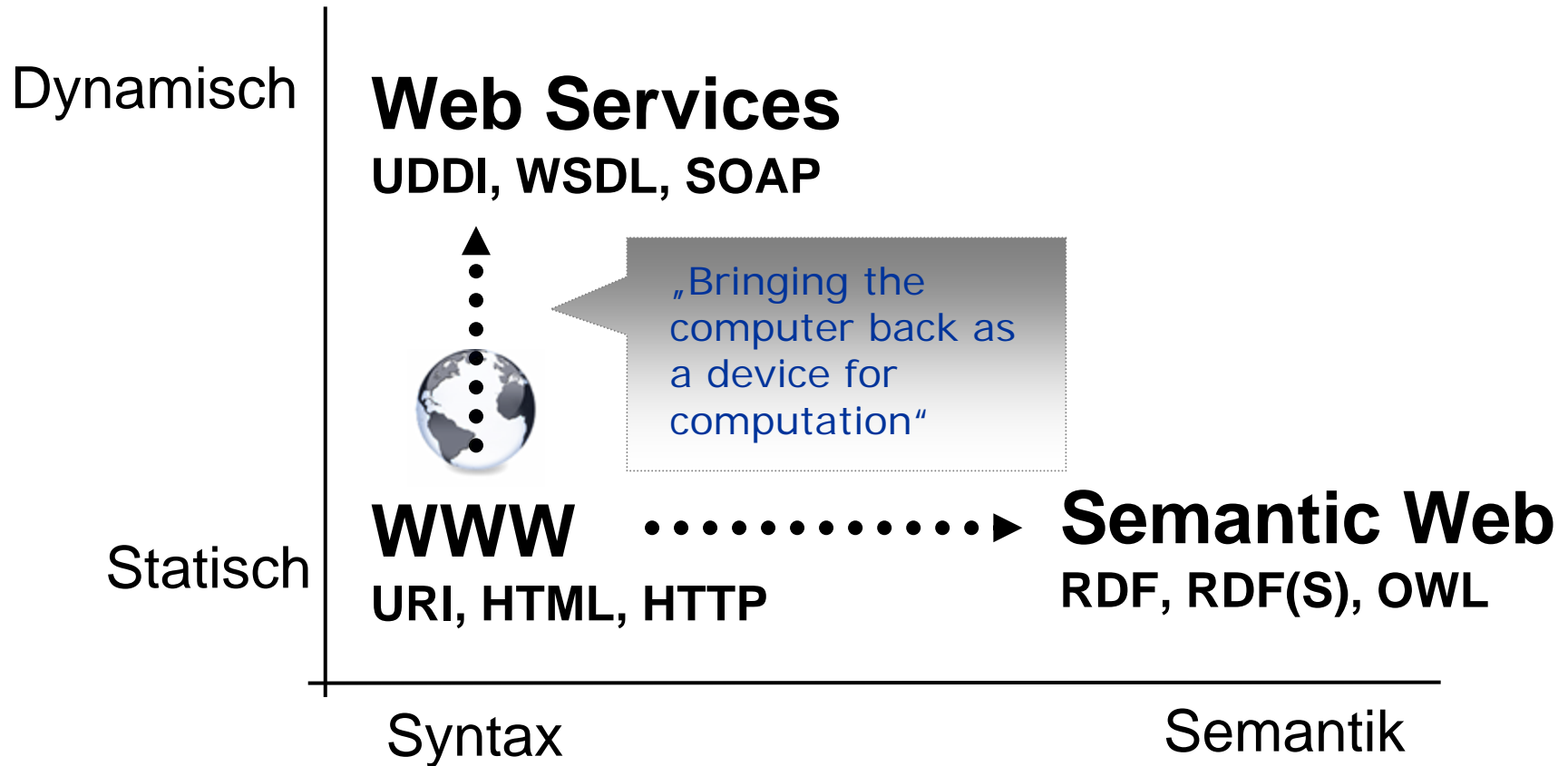
Statisch

Syntax

Semantik

## Probleme mit

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





- teilweise/vollständige Automatisierung der Bildung einer SOA durch das Hinzufügen von Semantik
- dynamische Bindung von Nutzer & 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 & Anbieter auf dem Daten-, Protokoll- und Prozesslevel
- automatischer Service Aufruf

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

- nur syntaktische Standards
- keine semantik-basierten Repositories
- keine Standardframeworks um Discovery, Composition and Execution zu erleichtern
- keine Tools/Plattformen, die semantische Anreicherung 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 Prozessfehlpassung (mismatch) zwischen den kombinierten Services auf
- **Execution** → spricht ein Service an

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

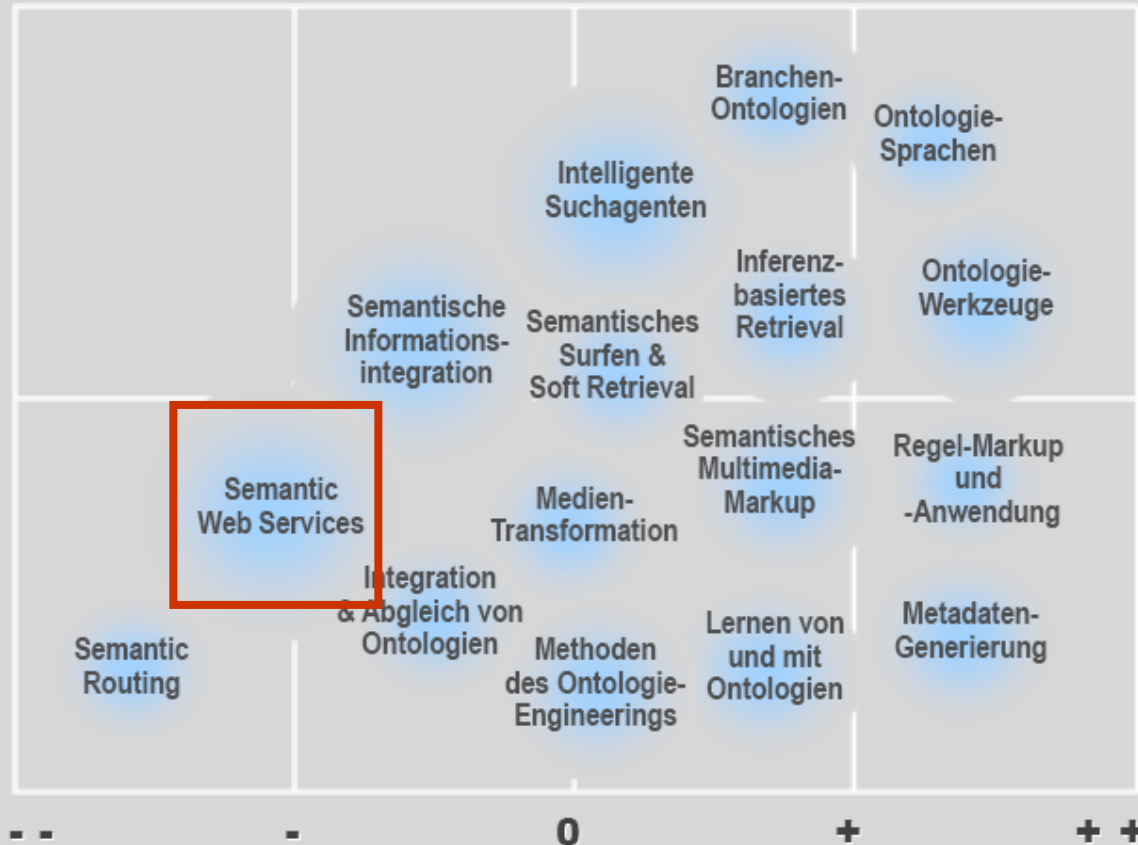
Der Durchmesser der Kreise entspricht der erwarteten wirtschaftlichen Bedeutung der Themen



**Anwendungsreife**  
hoch

mittel

gering



**Wettbewerbsstellung von Deutschland**

Quelle: „Forschen für die Internet-Gesellschaft: Trends, Technologien, Anwendungen“  
[http://w4.siemens.de/ct/de/activities/inet\\_symp/downloads/ergebnisse.pdf](http://w4.siemens.de/ct/de/activities/inet_symp/downloads/ergebnisse.pdf)

# Wie geht es weiter?

## Heute/morgen

- Letzte Übung

## Mi. 08.07.

- Rückblick (Klausurvorbereitung)

## Mi. 15.07. 14:00-16:00

- Klausur