Intelligente Agenten für das Internet der Dienste

PD. Dr. Matthias Klusch

Zur Person

- 2009 Privatdozent an U Saarland (Habilitation)
- 2008 a.o. Professor an TU Swinburne, Australien
- 2003 DFKI Research Fellow
- 1999 Senior Researcher am DFKI (FG Multiagentensysteme)
  mit FT Intelligente Informationssysteme (14; PhD/MSc/BSc).
- 2000 – 2002 Assistant Professor (TUD) an VU Amsterdam.
- 1998 Postdoc Researcher an Carnegie Mellon University, USA.
  1997 - 1999 Assistenzprofessor an TU Chemnitz.
- 1997 Promotion an CAU Kiel. Studium der Informatik an CAU Kiel.

@DFKI: Theorie und Anwendung von intelligenten Agenten,
semantischen Technologien, dienstbasierten Systemen.

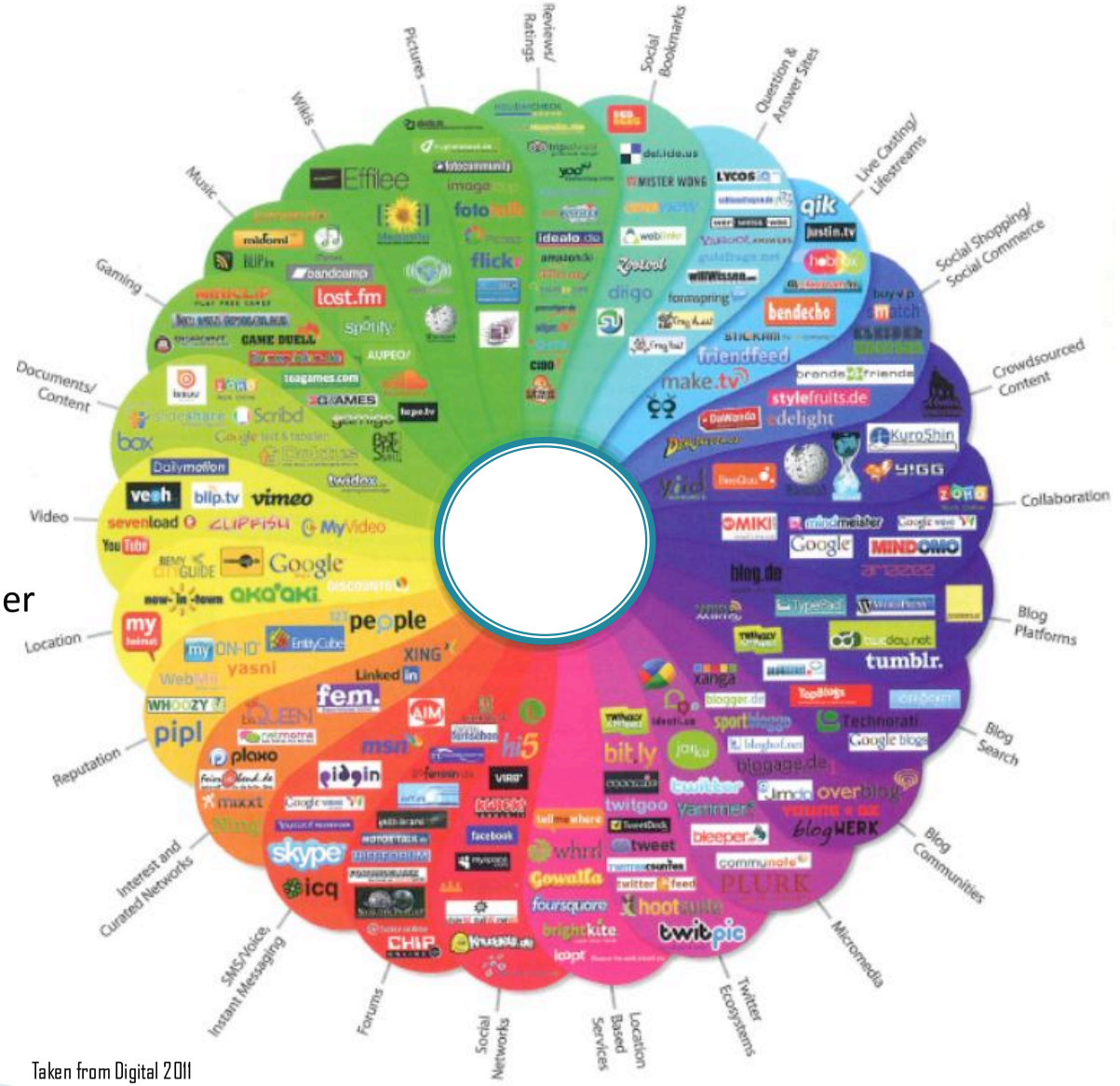
www.dfki.de/~klusch www.dfki.de/~klusch/i2s

Vielfalt von Dienstleistungen im Web ….

Millionen Webseiten,
Tausende Webdienste,
Milliarden Inhalte ….

4 Mrd Bilder auf flickr,
2 Mrd YouTube vids d/l pro Tag,
50M tweets pro Tag,
500M facebook Mitglieder

Viele Dienstleistungen auch als mobile App verfügbar.



Taken from Digital 2011

Was ist das Internet der Dienste ?

"Das Internet der Dienste umfaßt

alle Dienstleistungen im Internet, die von jedem Nutzer

als Anbieter, Vermittler oder Konsument allgegenwärtig

verwendet, zusammengestellt und gehandelt werden können."

Jorge Villasante, European Commission, 2008

Beispiel: Persönliche Reiseplanung überall

- Erstelle einen Reiseplan, der für mich persönlich in der gegebenen Situation optimal ist.
- Berücksichtige individuelle, gewichtete Präferenzen, Lokation, Wetter, Zeitplan, Verfügbarkeit, Bewertungen, Preis, Navigation, etc.

„Koordiniere Rundreise in Australien“

Geg.: Reisezeiten, -ziele, -budget;
Nutzerprofil.

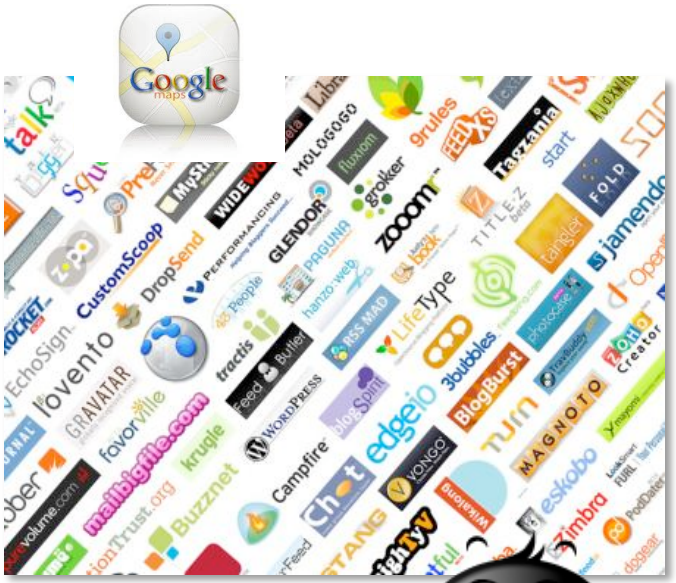
→ **Zeige mir die besten persönlichen Reiseplanoptionen.**

Zuhause und unterwegs**.**

Problem

Es gibt eine Vielzahl von Dienstleistungen für einzelne relevante Planungsaufgaben als (mobile) Apps …

Kartografische Ansicht von Hotels, POIs, Routen, …

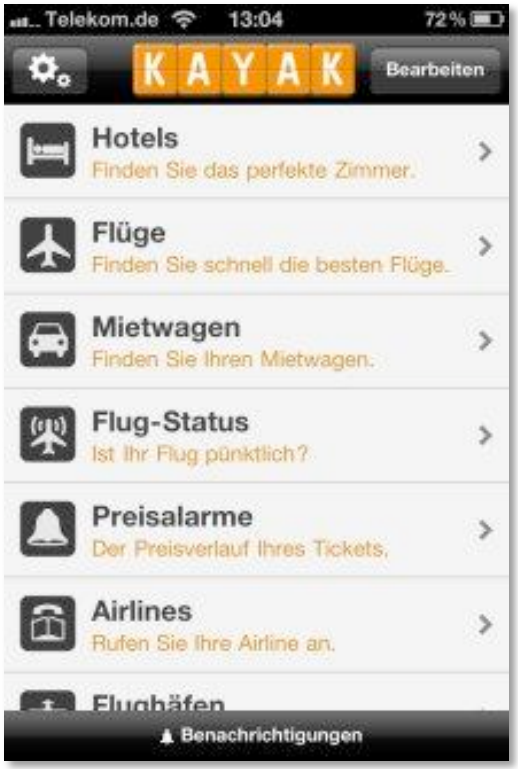


… wie finde und kombiniere ich diese Dienste für meine Reise am besten ?

*Gibt es **Programme**, die das leisten können ?!*

Beispiel: Persönliche Reiseplanung – Mashups

- Erster Ansatz: *Suche Dienst-Mashup für gewünschte Art der Reiseplanung*
- **Mashup**: Bietet feste oder interaktive Komposition (Plan) von vorselektiert relevanten Diensten für vorgegebene Menge von Aufgaben. *Für Reisebuchung: nur interaktiv*.

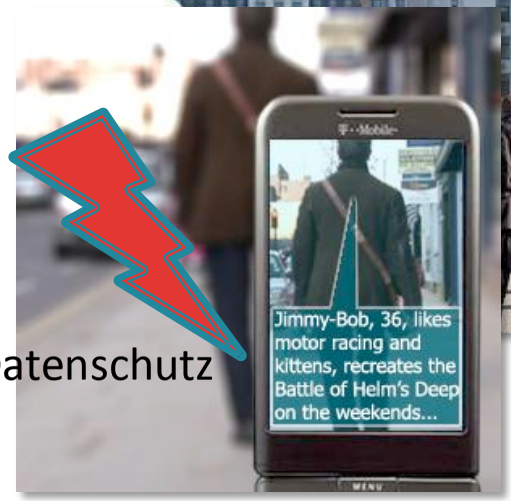


Kaum intelligent:

- Kein *automatisches Planen* nach Nutzerprofil
- Kein *proaktives Umplanen für geänderten Kontext*
- kein *implizites Lernen* von Profil und Planen

Beispiel: Persönliche Reiseplanung – Mashups (2)

- *Echtzeitanzeige ortsbasierter Informationen über Bildanalytisch klassifizierte Dinge* für gegebene Kategorien auf Aufnahmen des Nutzers vor Ort (AR-Mashups).



Datenschutz

Kaum intelligent:

- Kein *automatisches Planen* nach Nutzerprofil
- Kein *proaktives Umplanen für geänderten Kontext*
- kein *implizites Lernen* von Profil und Planen

Beispiel: Persönliche Reiseplanung – Mashups (3)

- *Echtzeitanzeige ortsbasierter Informationen über Bildanalytisch klassifizierte Dinge* für gegebene Kategorien auf Aufnahmen des Nutzers vor Ort (AR-Mashups).

in 3D

Kaum intelligent:

- Kein *automatisches Planen* nach Nutzerprofil
- Kein *proaktives Umplanen für geänderten Kontext*
- kein *implizites Lernen* von Profil und Planen

Beispiel: Persönliche Reiseplanung – Mashups (3)

- *Echtzeitanzeige ortsbasierter Informationen über Bildanalytisch klassifizierte Dinge* für gegebene Kategorien auf Aufnahmen des Nutzers vor Ort (AR-Mashups).

in 3D

Problem:

„Es gibt kein intelligentes Mashup für persönliche Reiseplanung.“

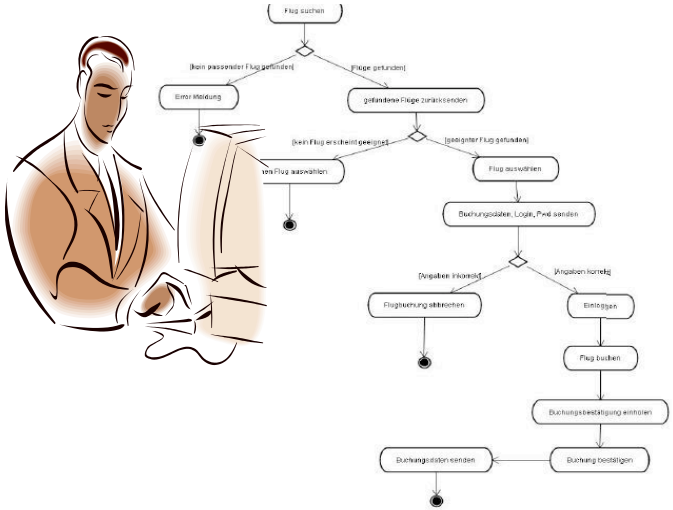
➢ Intelligente, automatische Suche und Komposition von Diensten ?

Kaum intelligent:

- Kein *automatisches Planen* nach Nutzerprofil
- Kein *proaktives Umplanen für geänderten Kontext*
- kein *implizites Lernen* von Profil und Planen

Beispiel: Dienste für komplexe Geschäftsprozesse

Geschäftsprozess mit Funktionen von verschiedenen Unternehmensbereichen (Wareneinkauf, etc.)



Produktion

Finanzen

Einzelhandel

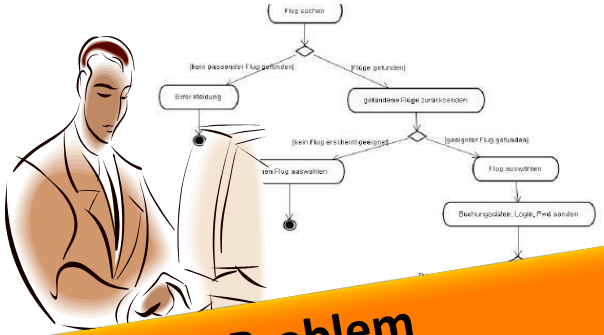
Logistik

Kontraktpartner für Dienste

B2B Service Marktplätze

Beispiel: Dienste für komplexe Geschäftsprozesse

Geschäftsprozess mit Funktionen von verschiedenen Unternehmensbereichen (Wareneinkauf, etc.)



Produktion

Finanzen

Problem

„Manuelle Suche, Orchestrierung und Verhandlung von prozessrelevanten Diensten."

➢ *Automatische, intelligente Unterstützung ?*

Einzelhandel

Logistik

Kontraktpartner für Dienste

B2B Service Marktplätze

SAP IBM

Herausforderung im Internet der Dienste

- Intelligente Suche, Komposition und Verhandlung

  von relevanten Diensten für beliebige Arbeitsprozesse.

- „Intelligent“: Automatisch, adaptiv und optimal

  in Bezug auf die jeweilige Situation

  - persönliches Profil (Präferenzen, soziales Netze, Aufgaben, etc.)
  - Umwelt (Verfügbarkeit von Diensten, Netzwerk, etc.)

Intelligente Programme = „Intelligente Softwareagenten“
für das Internet der Dienste

Agenda

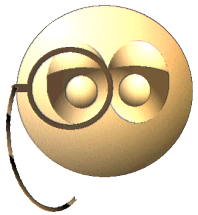
- Intelligente Agenten
- Agenten und Webdienste
- Agenten und Semantische Dienste

Agenda

- Intelligente Agenten
- Agenten und Webdienste
- Agenten und Semantische Dienste

Was ist ein intelligenter Agent?

"Intelligente Agenten sind Programme, die zu einer im gegebenen Kontext optimalen Problemlösung durch eine autonome, reaktive und zielgerichtete Anwendung von geeigneten Methoden der Künstlichen Intelligenz (KI) fähig sind." [George Luger]

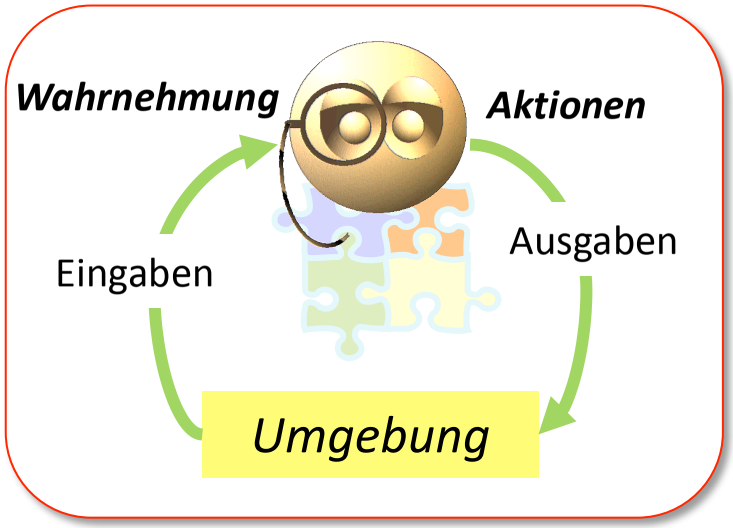


Einige klassische Technologien für Entwicklung intelligenter Agenten:

- Wissensrepräsentation (Wissen über Umgebung, Aktionen, Ziele, …)
- Logische Planung (Planen von Aktionen)
- Maschinelle Lernverfahren (Adaption an veränderlichen Kontext)
- Sprachverarbeitung (Interaktion mit Nutzer)
- Mikroökonomik / Entscheidungstheorie (Nutzenorientiertes Handeln)

Eigenschaften von Agenten

- **Autonom:** Eigenverantwortliches Handeln
- ***Reaktiv***
  - kontinuierlich angepaßt (lernend) an Benutzer und Umgebung
- ***Proaktiv***
  - Eigeninitiativ und zielgerichtet planend und entscheidend
- ***Kooperativ***
  - Interaktion und Zusammenarbeit mit anderen Agenten



- Programm: **Softwareagent**
- mit physischer Ausprägung: **Roboter**

Arten von Agenten

- **Reaktive Agenten**

  (1) Reaktives Planen von Aktionen für unmittelbare oder sehr zeitnahe Reaktion auf Ereignisse in der Umgebung:

  - **Reiner Reflex** - Ereignisorientierte „Reiz-Reaktion“ Regeln mit „Vergessen“ (ohne internen Zustand)
  - **Reflex mit Zustand:** Reiz-Reaktion-Regeln mit kontinuierlich gespeichertem Wissen über Umgebung (mit internem Zustand & Weltmodell; “Erinnerung”)
  - **Ausführung von Planmustern** über Zustand („Planning from 2nd Principles“)

  (2) Lernen der Nützlichkeit von ausgeführten Aktionen für individuelle Lernziele über internem Zustand und (partiell) beobachteter Umgebung.

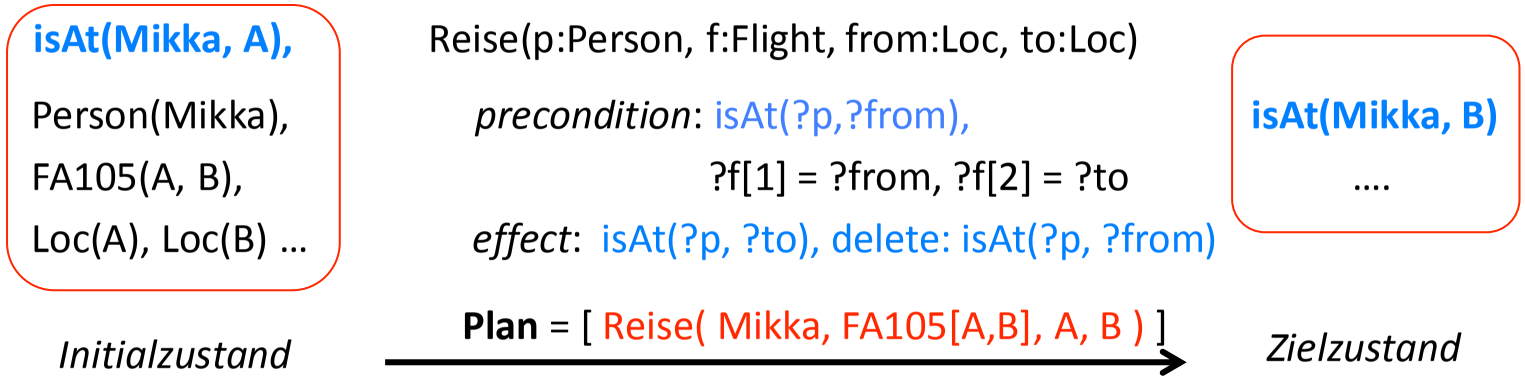
V **Schwarm mit reinen Reflexen ohne Kommunikation** (Animationsfilmvorstudie von BioGraphics)

Arten von Agenten (2)

- **Deliberative Agenten**

  Eigenständige, logisch wissensbasierte (deliberative) Erzeugung von optimalen Aktionsplänen zur Erreichung eines gegebenen Ziels. („Planning from 1st principles“)

  Einfaches Beispiel: Logisches *zustandsbasiertes Vorwärtsplanen* von Aktionen



Gemeinsames Planen zwischen Agenten ?

Multiagentensysteme – Gemeinsame Problemlösung

- **Verteilte Problemlösung**
  - Hierarchische Aufgabenverteilung
  - Aufgabenbearbeitung im Schwarm (Emergente Lösung)

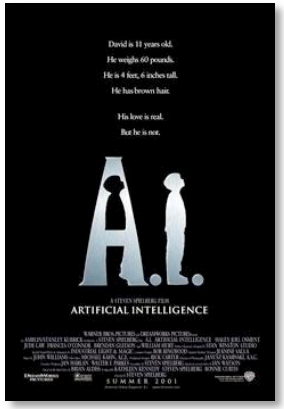
- **Koordination für Problemlösung**
  - Gemeinsames Kooperationsmodell (SharedPlans, Koalition, etc.)
  - mit/ohne Kommunikation zwischen Agenten

Problemlösung im Schwarm *mit* Kommunikation
(„Mundpropaganda-Effekt“)

V

- **Standards und Werkzeuge für Agentenentwicklung**
  - FIPA AgentUML, FIPA-ACL; Agentenprogrammierung mit JACK, JADE, JADEX, …

Vision: Intelligent wie ein Mensch … ?



"Can machines think? … We can only see a short distance ahead, but we can see plenty there that needs to be done."
Alan Turing, 1950

- Ethische Richtlinien für intelligente Agenten im Internet der Dienste? Verantwortlichkeiten?

(1) Ein Roboter (= physischer Agent) darf durch sein Verhalten keinem Menschen direkt oder indirekt Schaden zufügen.

(2) Ein Roboter muß allen ihm von Menschen gegebenen Befehlen gehorchen, sofern sie nicht im Konflikt mit dem 1. Gesetz stehen.

(3) Ein Roboter muß sich selbst schützen sofern dies nicht gegen Gesetz 1 & 2 verstößt.

Issac Asimov

Klassische Anwendungsbereiche von Agenten

Transport & Logistik

Dezentrale Optimierung
von Transportrouten,
Warenlagerverwaltung.

DFKI: Flottenmanagement für
Speditionen Konz, Fixemer (SL)

Produktionssteuerung

Dezentrale Optimierung
von Produktionsabläufen.

DFKI: Stahlproduktion bei Saarstahl AG

Elektronischer Handel

Verhandlung von Dienstleistungen,
Meta-Shopbots

DFKI: Einkaufsassistenten;
Holz-/Getreideauktionen

Edutainment

Intelligente Agenten in 3D Welten

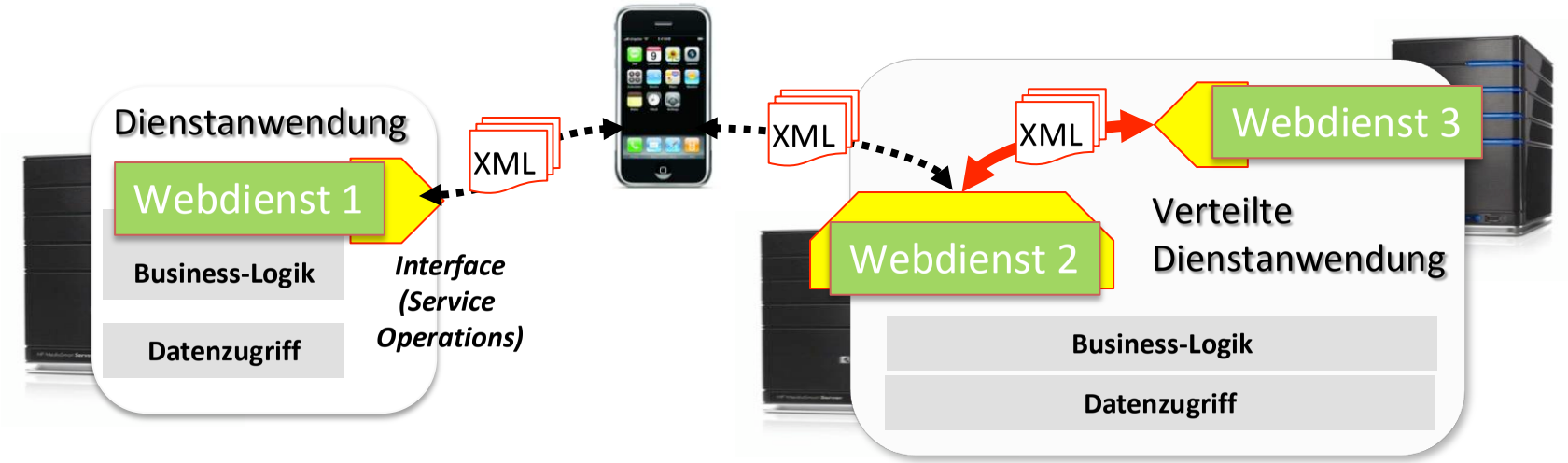
DFKI: ISReal-Projekt

Agenda

- Intelligente Agenten
- Agenten und Webdienste
- Agenten und Semantische Dienste

Was sind Webdienste?

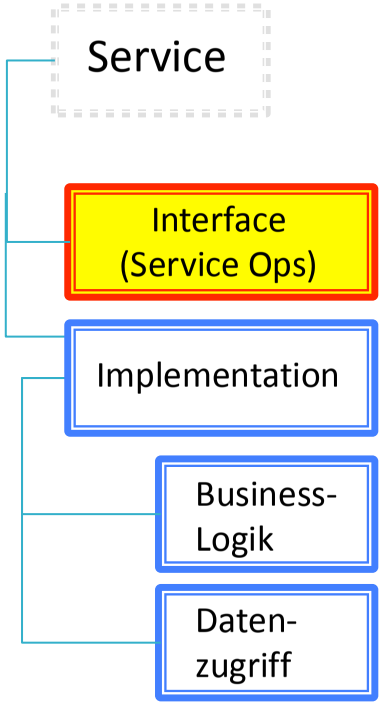
Ein Webdienst ist eine eindeutig im Internet addressierte Softwarekomponente, die ihre Funktionalität über eine veröffentlichte Standard-XML-basierte Schnittstelle anbietet und für beliebige Anwendungen über ein im Internet verwendetes Protokoll XML-nachrichtenorientiert und programmatisch zugreifbar ist. [Jeckle, 2004]

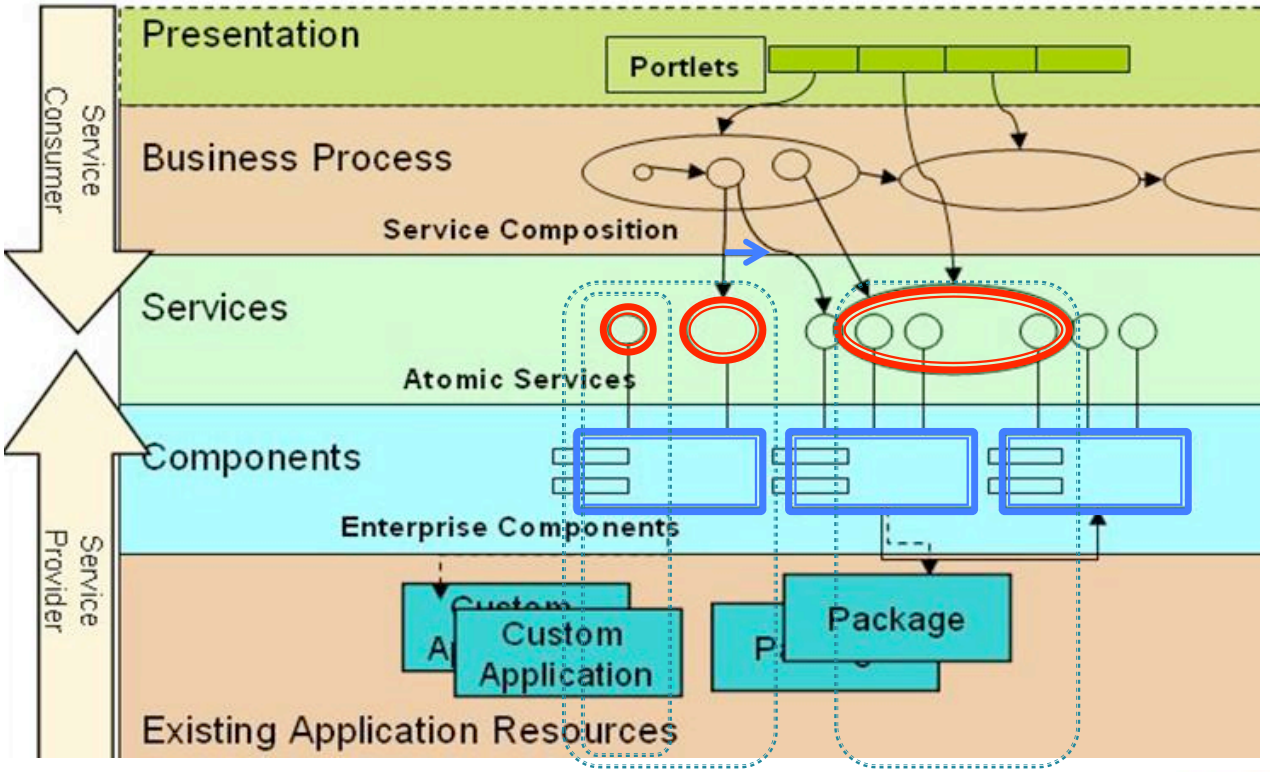


Ein Dienst bietet eine oder mehrere Funktionen über sein Interface an:
Für verschiedene Anwendungen wiederverwendbar; interoperabel; plattformunabhängig. → Skalierbar verteilte Dienstanwendungen

Dienstorientierte Architektur (SOA)

- Keine allgemein akzeptierte Definition von SOA
- „SOA ist ein ***Paradigma*** für die Strukturierung und Nutzung verteilter Funktionalität, die von unterschiedlichen Besitzern verantwortet wird.“ [OASIS, 2006]





(Non-)atomic service: 1 (n>1) operation from one component
Composite service: n>1 operations from m>1 components (services)

SOA 2006

Wie werden Webdienste beschrieben?

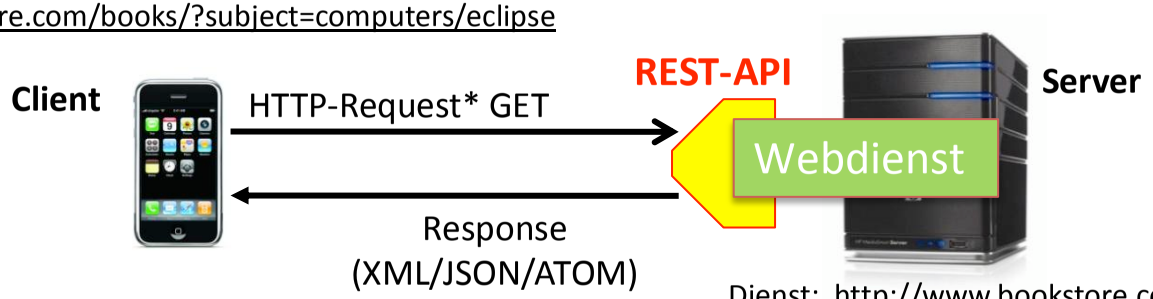
Prominente Sprachen zur Beschreibung von Webdiensten und Webanwendungen

- **WSDL** – Web Service Description Language
- SML – Service Modeling Language
- **REST** – Representational State Transfer
- WADL - Web Application Description Language

  XML-basiertes Dateiformat für REST-Dienstbasierte Anwendungen, seit 2006; kein Standard (http://java.net/projects/wadl)

REST-basierter Webdienst

Dienstanfrage:
http://www.bookstore.com/books/?subject=computers/eclipse



Dienst: http://www.bookstore.com/books
Param: subject, authors, …

REST - Representational State Transfer

- Jeder Dienst, jeder Teil einer Dienstanfrage und -antwort ist eine per URI (URL) referenzierbare Ressource im Web.
- Dienstzugriff nur über Methoden des zustandslosen HTTP (GET, PUT, POST, DELETE).
- Transfer von verschiedenen Repräsentationen einer Ressource: Zustände von Client-Server Dienstinteraktionen werden als in XML, JSON, ATOM kodierte Dienstausgaben vom Server an den Client transferiert. (HTTP content negotiation)

Beispiel: REST-basierter Dienst „Book_List“

- Buchladenportal http://www.bookstore.com
  REST-basierte Dienstanwendung “Bookstore” mit mehreren REST-Diensten.
- **REST-Dienst**: Book_list (Buchinventarliste) http://www.bookstore.com/books/
  Anfrageparameter: author, language, publisher, subject, title
- **Dienstanfrage**: http://www.bookstore.com/books/?subject=computers/eclipse
- **Dienstantwort** in XML:

```
<booklist:bookList xmlns:booklist="http://www.bookstore.org/booklist/xsd"  xmlns:book="http://www.bookstore.org/book/xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.bookstore.org/booklist/xsd booklist.xsd
http://www.bookstore.org/book/xsd book.xsd">
<booklist:book url="http://www.bookstore.com/books/0321442598"  title="BIRT: A Field Guide to Reporting"/>
<booklist:book url="http://www.bookstore.com/books/0321205758"  title="Contributing to Eclipse: Principles, Patterns, and Plug-Ins"/>
<booklist:book url="http://www.bookstore.com/books/0321245873"  title="Eclipse AspectJ: Aspect-Oriented Programming with AspectJ and the..."/>
<booklist:book url="http://www.bookstore.com/books/0321288157"  title="Eclipse Distilled"/>
<booklist:book url="http://www.bookstore.com/books/0131425420"  title="Eclipse Modeling Framework"/>
<booklist:book url="http://www.bookstore.com/books/0321334612"  title="Eclipse Rich Client Platform: Designing, Coding, and Packaging Java..."/>
 <booklist:book url="http://www.bookstore.com/books/0321396855"  title="Eclipse Web Tools Platform: Developing Java Web Applications"/>
<booklist:book url="http://www.bookstore.com/books/032142672X"  title="Eclipse: Building Commercial-Quality Plug-Ins (2nd Edition)"/>
<booklist:book url="http://www.bookstore.com/books/0321443853"  title="Integrating and Extending BIRT"/>
<booklist:book url="http://www.bookstore.com/books/0321268385"  title="Official Eclipse 3.0 FAQs"/>
<booklist:book url="http://www.bookstore.com/books/0321256638"  title="Official Eclipse 3.0 FAQs"/>
</booklist:bookList>
```

Beispiel: REST-basierter Dienst „PartsDepot“

Service: *PartsDepot*

Operation: *PartsDepot*.GetPartsList

URL: www.parts-depot.com/parts

Method: GET

Parameters: cid - Engine component ID

Returns: XML parts_list

Operation: *PartsDepot*.Delet

URL: …, Method: DELETE

REST-basierter Dienst „*PartsDepot*“
mit mehreren Operationen (Unterdienste)

Aufruf einer REST-Dienstoperation
aus einer XHTML-Seite des Clients (mit **hREST**)

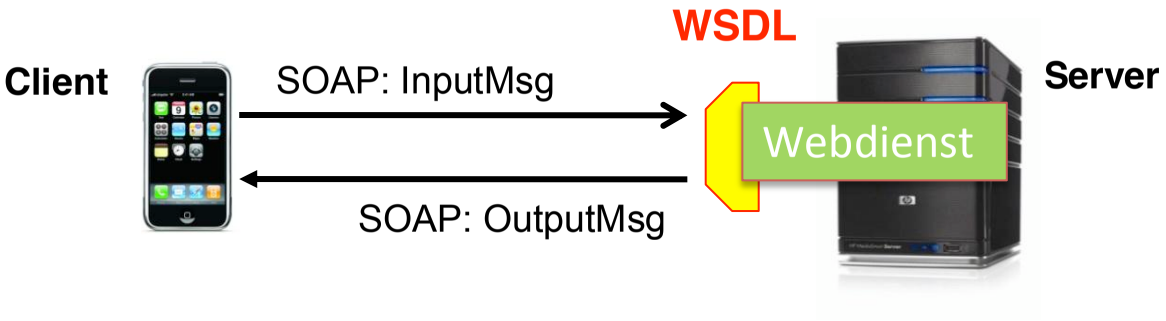
```
<div class="service" id = “PartsDepot”> The service operation
<div class="operation" id = “parts“> named
<code class="label"> GetPartsList </code> will be invoked through the
<span class="method"> GET </span> method at
<code class="address“> http://www.parts-depot.com/{id}?cid=123
</code> <span class="input"> for component 123 </span>.
Returns <span class="output">parts list </span></div> </div>
```

XML-Antwortnachricht
des Webservers

```
<?xml version="1.0"?>
<p:Parts xmlns:p=http://www.parts-depot.com xmlns:xlink="http://www.w3.org/1999/xlink">
    <Part id="00345" xlink:href="http://www.parts-depot.com/parts/00345"/>
    <Part id="00346" xlink:href="http://www.parts-depot.com/parts/00346"/>
    <Part id="00347" xlink:href="http://www.parts-depot.com/parts/00347"/> </p:Parts>
```

Teile der Komponente „123“

Webdienst in WSDL



WSDL - Web Service Description Language

- XML-basierte Schnittstellenbeschreibung
  - Funktionale Parameter: XML-Ein/Ausgabenachrichten; Nicht-funktionale Parameter (Name, Preis, usw.)
- Aufruf von Dienstoperationen ist nicht auf HTTP-Methoden beschränkt: Unabhängigkeit vom Transportprotokoll (HTTP, TCP, SMTP, POP3) - Methodennamen
- Diverse Erweiterungen (WS-*): WS-Security, WS-Adressing, usw.
- W3C Standard: WSDL 2.0 (2007)

Simple Object Access Protocol

- Standardprotokoll für XML-Nachrichtenbasierte Interaktion mit Webdiensten
  - SOAP-Nachricht: XML Header + Body (kapselt E/A-Nachrichtenbezogene Inhalte)
  - Interaktionsmuster (request/response)

Beispiel: WSDL-Dienst „PartsDepot“

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl" … >
<types>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
             xmlns:pdc="http://www.parts-depot.com/schemas/pdc" …>
  <xs:element name="cid" type="xs:string"/> …  </types>
<interface name="PartsListInterface">
   <operation name="GetPartsList"
        pattern="http://www.w3.org/ns/wsdl/in-out"…>
          <input messageLabel="In" element="xs:cid" />
          <output messageLabel="Out„ element="pdc:parts_list" />
    </operation> </interface>
 <binding name="PListHTTPBinding"
     type="http://www.w3.org/ns/wsdl/http" interface="tns:PartsListInterface">
    <operation ref="tns:GetPartsList" whttp:method="GET"/> </binding>
 <service name="PartsDepot" interface="tns:PartsListInterface">
 <endpoint name="PListHTTPEndpoint binding="tns:PListHTTPBinding"
           address="http://www.parts-depot.com/parts/"> </endpoint>
</description>
```

Datentypen der E/A-Nachrichten

Operation(en)

E/A-Nachrichten

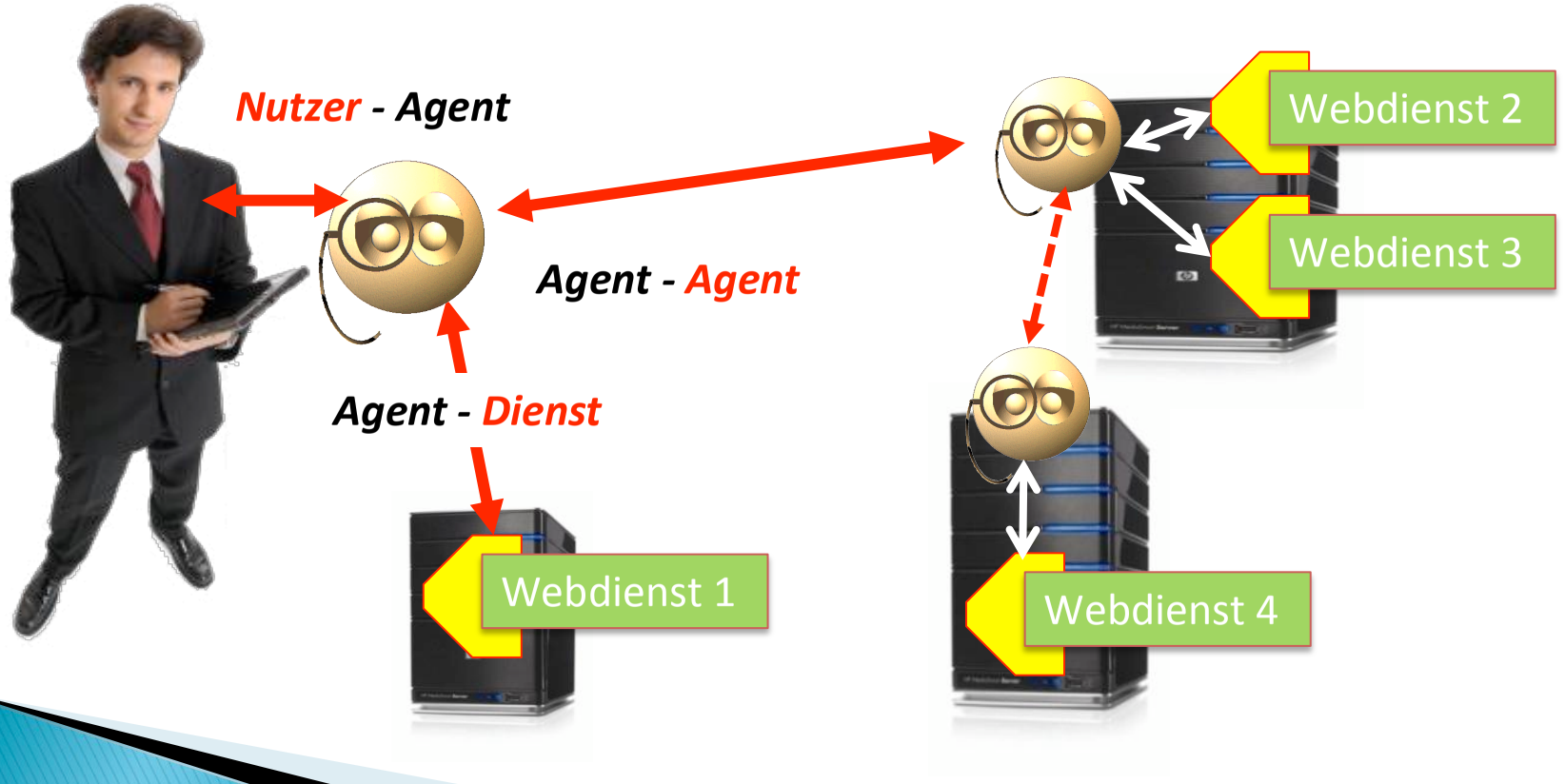
Bindung an Transportprotokoll

Webadresse der gebundenen Dienstoperation

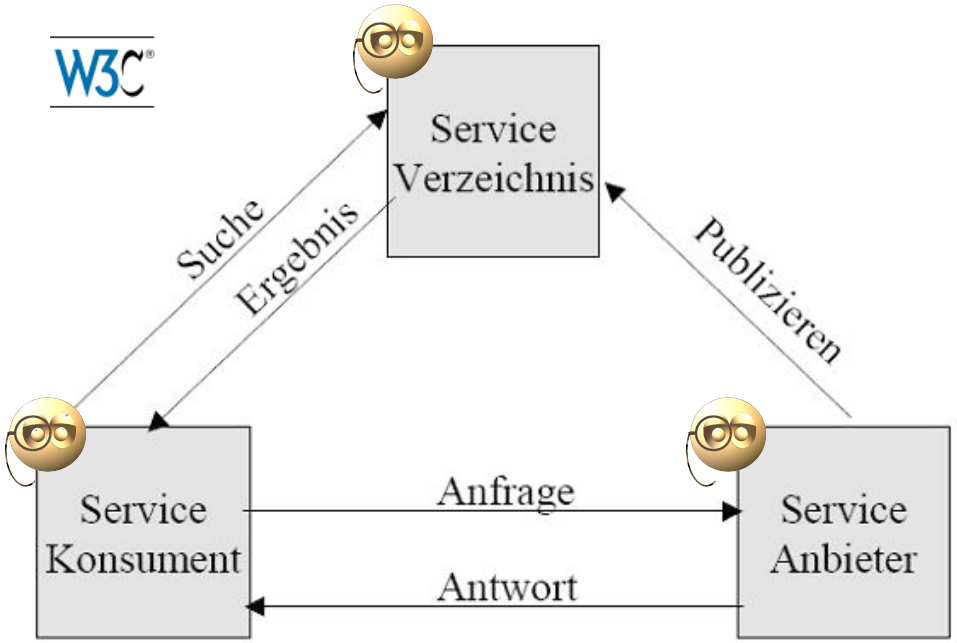
Service --- Ops --- Bindings --- URLs

Agentenbasierte Dienste

- Intelligente Agenten sind keine Webdienste
  - Dienste suchen, komponieren, verhandeln sich nicht selbst
  - Agenten repräsentieren und koordinieren Dienste für ihre Benutzer
- Intelligente Koordination: Persönlich, adaptiv, proaktiv



Suche nach relevanten Diensten



- **Verzeichnisse**

Android Market App-Store

Webmashup.com 4906 (114 Kat.)

WSDL: Xmethods.net, webservicelist.com, wsindex.org, UDDI Verzeichnisse

- **Suchmaschinen**
  - Google, Bing
  - Speziell: seekda.com (WSDL)

7.739 providers 28.606 services seekda! Nov 17, 2011

Matchmaker-Agenten

Selektion und Vermittlung von relevanten Diensten:

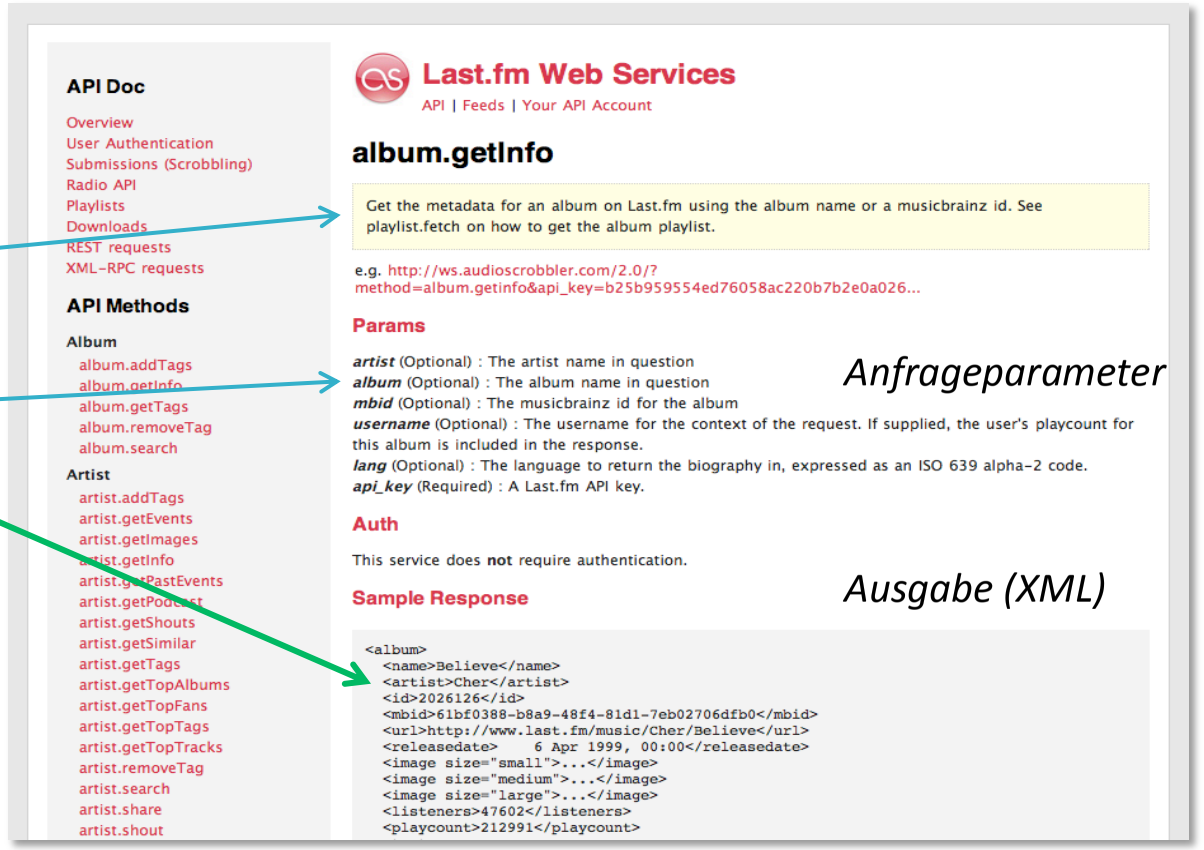
(a) Paarweiser Abgleich von Dienstanfrage mit -angeboten (Relevanz)

(b) Rangliste von relevanten Diensten mit Anbietern

Beispiel: Relevanz von REST-Diensten

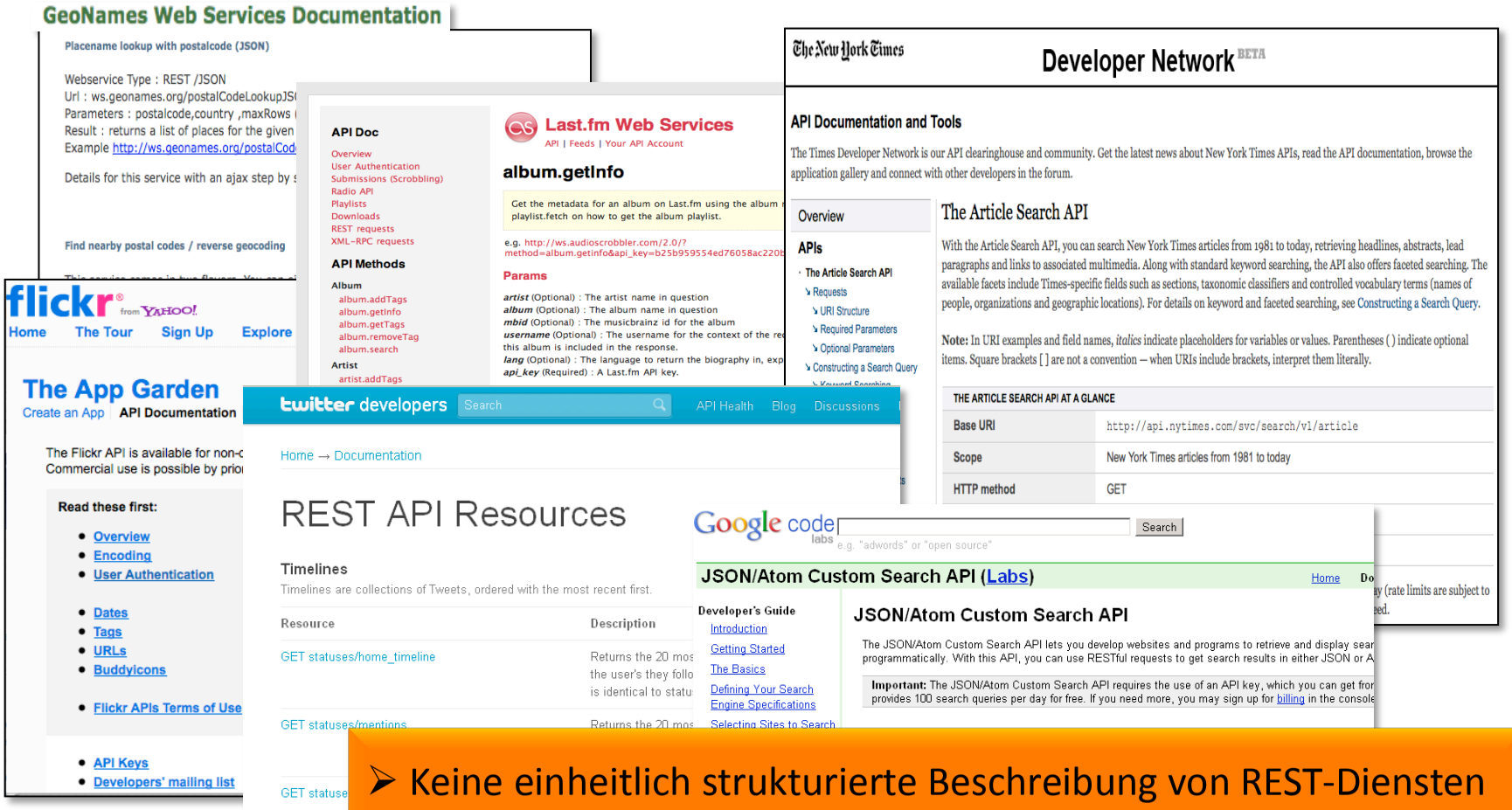
Service
returns artists of
music album

Input: album
Output: artists



- Dienstanfragen: Unstrukturierter Text; strukturiert in XML/WADL
- Relevanz von REST-Dienst: Aggregierte **Text - und Strukturähnlichkeiten**
- Methoden des Text-/XML-Retrieval; spez. Graphalgorithmen

Problem: Heterogene APIs von REST-Diensten



- Keine einheitlich strukturierte Beschreibung von REST-Diensten und Anfragen für automatische Selektion aus Dokumentationen
- WADL ist (noch) kein Standard und kaum verbreitet

Beispiel: Selektion von WSDL-Diensten mit WSDLAnalyzer

Input: WSDL Dienste S, Q

Output: Struktureller Ähnlichkeitswert $sim(S,Q) \in [0,1]$

- **Rekursive Berechnung struktureller Ähnlichkeit von XML-Graphen**

T_S, T_Q von S, Q: $sim(T_S,T_Q) = sim(\ root(T_S),root(T_Q)\ )_{norm}$

$$sim(a,b) = \begin{cases} \omega_1 * sim_{name}(l(a), l(b)) + \\ \omega_2 * \max\{\sum_k sim(n_i, m_j) \mid (a,n_i),(b,m_j) \in E, m_j = \varphi_k(n_i), \varphi_k \text{inj.}\}, & \text{if } l(a), l(b) \notin D \\ sim_{type}(l(a), l(b)), & \text{if } l(a), l(b) \in D \end{cases}$$

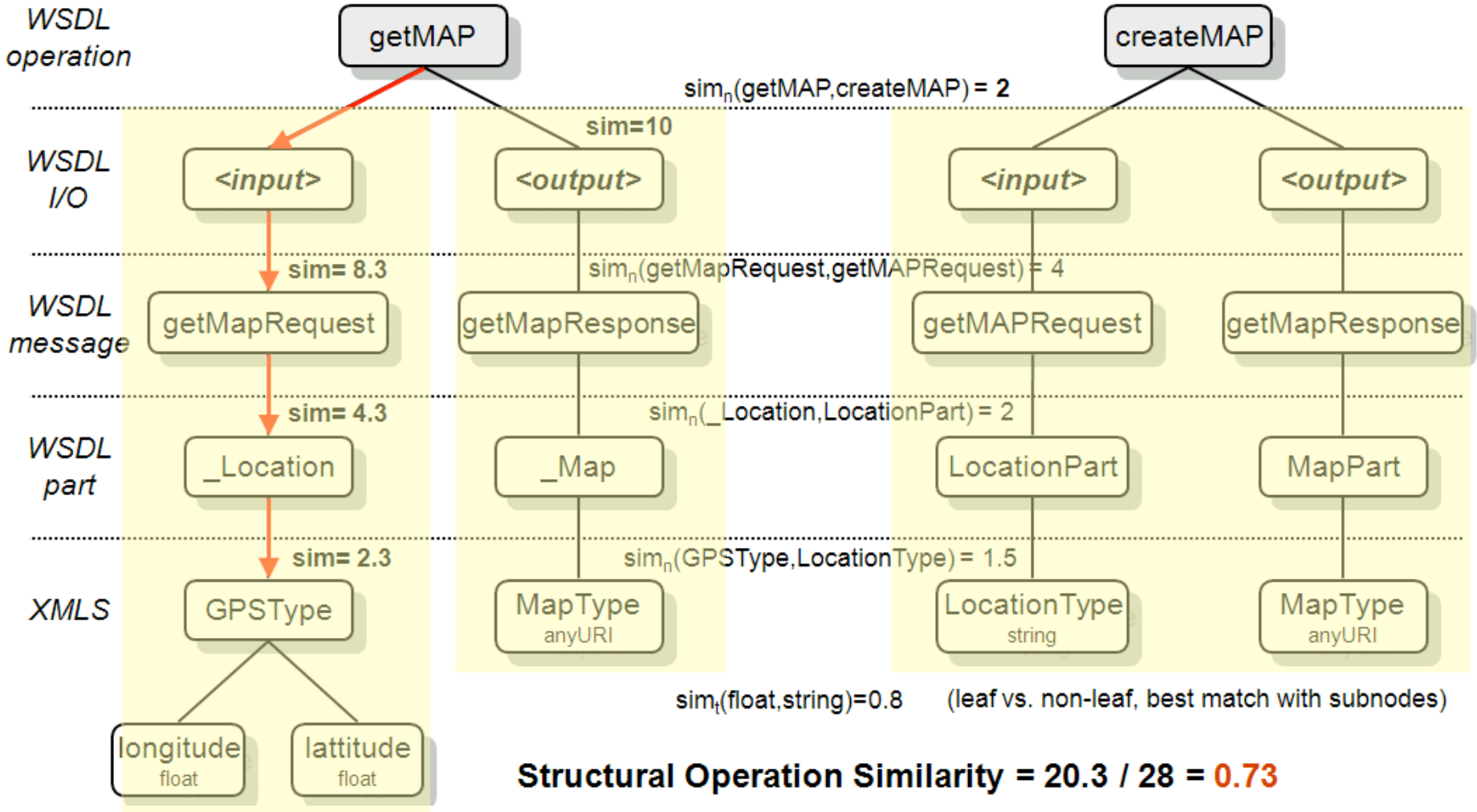
sim_{name} : **Textähnlichkeit der Bezeichner von Graphknoten**

(WordNet-distance + keyword string matching)

sim_{type} : **Datentypkompatibilität** $\in [0,1]$ (Look-up table); Gewichte $w_1+w_2=1$

[Zinnikus, Rupp & Fischer, Proceedings 2nd WS Web Services and Interoperability, 06]

Beispiel: Selektion von WSDL-Diensten mit WSDLAnalyzer

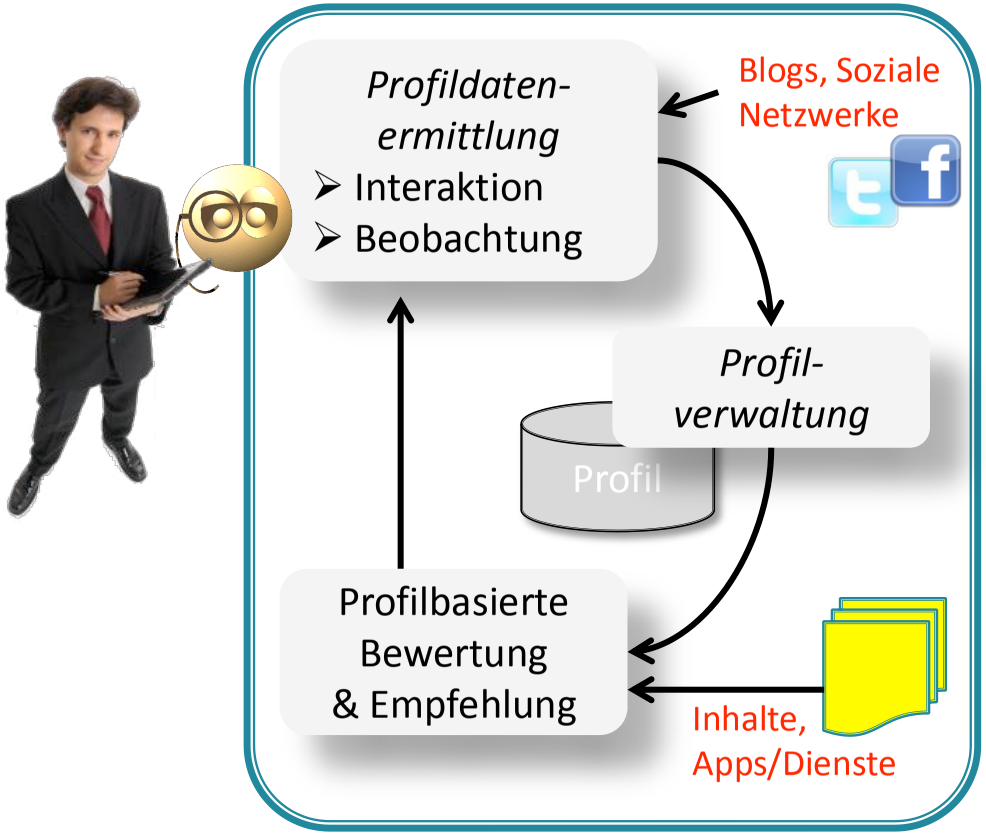


Persönliche Selektion von Diensten

- **Persönliches Benutzerprofil**
  - Personaldaten, aktueller Standort
  - Benutzerpräferenzen zu verschiedenen Aufgabenarten
  - Langzeit- und Kurzzeitpräferenzen
  - Soziale Netzwerke
  - Historie von Nutzeranfragen, -aktionen mit Bewertung (Nutzer, Freunde)
  - Eigenschaften relevanter Dienstleistungen: Inhalt, Verfügbarkeit, Qualität; usw.
- **Erlernen des Benutzerprofils**
  - Explizit durch Nutzerinteraktion
  - Implizit durch Nutzerbeobachtung
- **Persönliche Selektion und Empfehlung**
  - Individuell Präferenzbasiert
  - Inhaltsbasierte Empfehlung: Ähnlichkeit von neuen Inhalten mit zuvor präferierten Inhalten
  - Kollaborative (soziale) Empfehlung

Recommender-Agenten

Lernen von Benutzerprofilen



Ermittlung von Nutzerpräferenzen

von Inhalten und Diensten:

- **Explizit** (Nutzerinteraktion)
  - Frage nach Eingabe von persönlichen Daten und Bewertungen von genutzten Inhalten/Diensten, kategorischen Beispielen
- **Implizit** (Nutzerverhalten)
  - Suchanfragen, Infos zu genutzten Diensten/Inhalten (Themen, Häufigkeit, etc.), Clickstream-Analyse
  - Positive, negative Bewertungen (eigene und die seiner Freunde) in sozialen Netzwerken, Blogs
  - Vorherige Bewertung von ähnlichen, genutzten Diensten /Inhalten

Individuell Präferenzbasierte Dienstselektion

„Finde ein für mich passendes Hotel in AnyCity am 22.2.2012“

Ausschnitt aus Nutzerprofil:

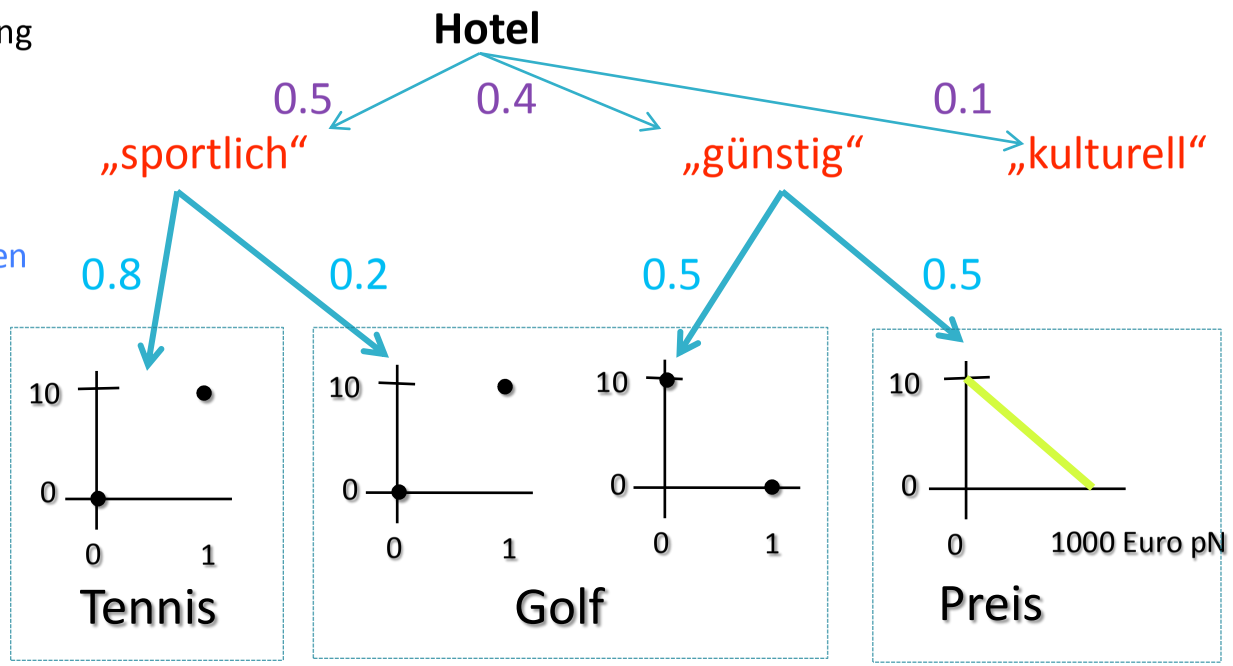
Nutzer-profil (Teil)

Persönliche Gewichtung von aktuellen **Präferenzen:**

Mit individuell relativen Gewichtungen

und absoluter Bewertung [0,10]

von **Eigenschaften:** (von Hotels)



- **Suche** Hotelreservierungsdienste mit *verfügbaren Hotels nach gegebenem Ort, Datum*

Individuell Präferenzbasierte Dienstselektion (2)

- **Extrahiere Eigenschaften** der verfügbaren Hotels nach Präferenzen aus den Dienstbeschreibungen oder Antworten auf Dienstanfragen:

Hotel h1 “Rabenstein”

- **Tennis**
- **Golf**
- **Preis: 500 Euro**

Hotel h2 “Goldstar”

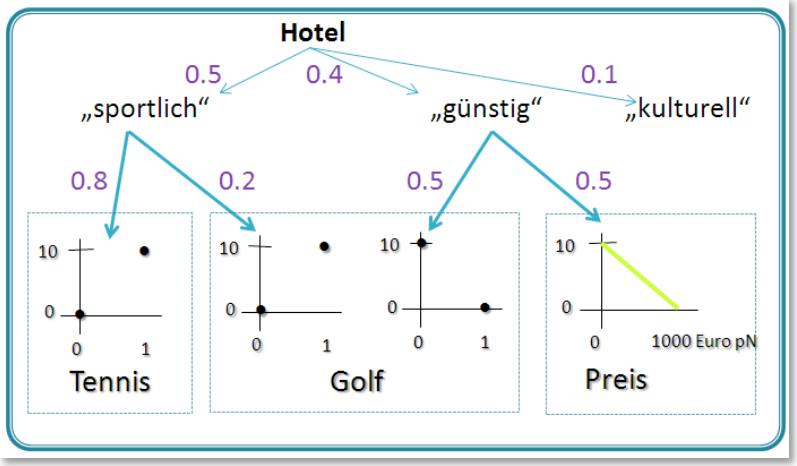
- **Tennis**
- **No golf**
- **Preis: 100 Euro**

- **Ermittle das individuell maximal präferierte Hotel** (Präferenzwert v):

$$v(h1, P) = 0.5 \cdot (0.8 \cdot 10 + 0.2 \cdot 10) + 0.4 \cdot (0.5 \cdot 0 + 0.5 \cdot 5) = 5.1$$

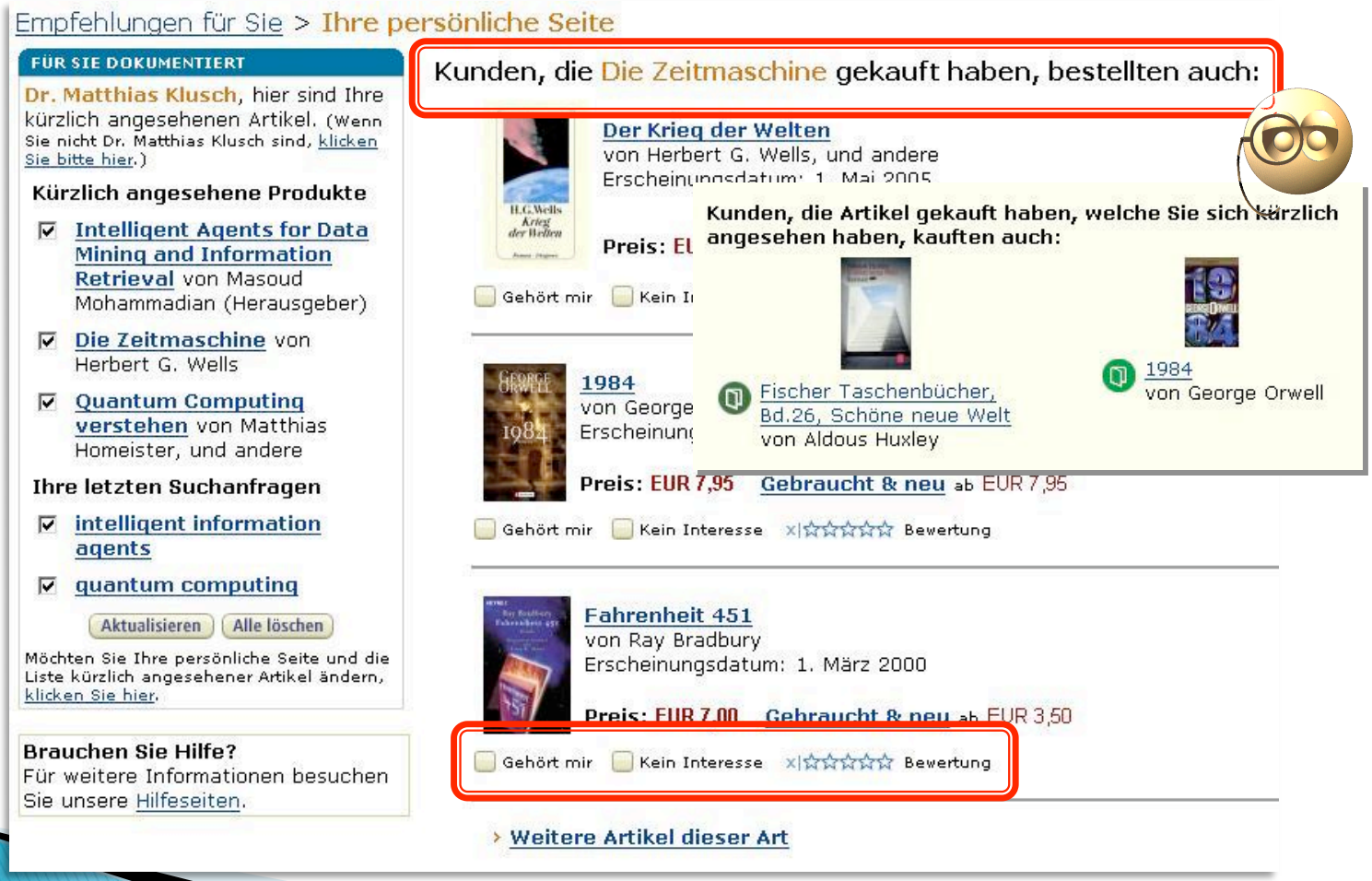
$$v(h2, P) = 0.5 \cdot (0.8 \cdot 10 + 0.2 \cdot 0) + 0.4 \cdot (0.5 \cdot 10 + 0.5 \cdot 9) = 7.8$$

- **Empfehlung** (Dienstselektion) von Hotels mit hinreichend hohen Präferenzwerten.



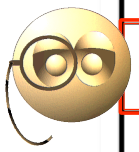
Kollaborativ persönliche Dienstselektion

- Selektion und Empfehlung von Dienstleistungen, die von Nutzern mit ähnlichem Profil präferiert wurden. Beispiele: Amazon, eBay, Barnes&Nobles, iTunes, MovieLens….



Beispiel: Kollaborative Empfehlung

- Bereitstellung von Filmen durch einen oder verschiedene gefundene Mediendienste
- Bewertung der Filme durch Nutzer impliziert Relevanz der entsprechenden Dienste
- Problem: *Vorhersage der Bewertung von noch unbekannten Filmen durch Nutzer*
- **Nutzerpräferenzmatrix:**



	Ran	Casablanca	Ben Hur	Tomb Raider	MI -II	Air Force One
Jane	5	5			?	2
Bill		2		3	4	
Tom	2		2		5	5
Cathy	3	3			1	1

From: Ramanathan, Manjunath, Banerjee (HP Labs) 2006

Intuitiv: Jane‘s Präferenzen sind ähnlich zu denen ihrer Freundin Cathy [> Bill > Tom].

→ Welche Bewertung würde Jane dem Film „Mission Impossible II“ geben?

Beispiel: Kollaborative Empfehlung

- **Nutzerbasiert soziale Vorhersage einer Präferenz** *p(a,j)* von Nutzer „a“ einer unbekannten Dienstleistung „j“ basierend auf seiner Ähnlichkeit w(a,u) zu anderen Nutzern „u“ :

$$p(a,j) = \frac{\sum_{u \in K} (r_{uj} - r_u^{avg}) \cdot w(a,u)}{\sum_{u \in K} |w(a,u)|} + r_a^{avg}$$

→ *Gewichtete, durchschnittliche Bewertungen von „j“ durch die zu Nutzer „a“ (top-k) ähnlichsten Nutzer „u“*

Nutzerähnlichkeit: Kosinus-Korrelation

$$w(a,u) = \cos(\vec{r}_a, \vec{r}_u) = \frac{\sum_{i \in I} r_{ai} \cdot r_{ui}}{\sqrt{\sum_{i \in I} (r_{ai})^2 \cdot \sum_{i \in I} (r_{ui})^2}}$$

Pearson-Korrelation

$$w(a,u) = \frac{\sum_{i \in I} [r_{ai} - r_a^{avg}] \cdot [r_{ui} - r_u^{avg}]}{\sqrt{\sum_{i \in I} [r_{ai} - r_a^{avg}]^2 \cdot [r_{ui} - r_u^{avg}]^2}}$$

I *Menge von durch beide Nutzer a, u bewertete Items*

r_{ui} *Bewertung von Item i durch Nutzer u*

r_u^{avg} *Mittelwert der Bewertungen von Nutzer u*

Beispiel: Kollaborative Empfehlung

- Nutzerpräferenzmatrix normalisiert $(r_{ui} - r_u^{avg})$

  Nicht alle Nutzer geben gleiche Werte bei gleichen Präferenzen

	Ran	Casablanca	Ben Hur	Tomb Raider	MI -II	Air Force One
Jane	1	1				-2
Bill		-1		0	1	
Tom	-1.5		-1.5		1.5	1.5
Cathy	1	1			-1	-1

	Ran	Casablanca	Ben Hur	Tomb Raider	MI -II	Air Force One
Jane	5	5			?	2
Bill		2		3	4	
Tom	2		2		5	5
Cathy	3	3			1	1

- **Nutzerähnlichkeiten** *w(Jane,u)* nach Kosinus-Korrelationen:

	Bill	Tom	Cathy
Jane	-0.289	-0.612	0.816

Beispiel: Kollaborative Empfehlung

- **Nutzerbasiert sozial ermittelte Präferenz** des für Jane unbekannten Films (k=3):

$$p(Jane, MI-II) = 4 + \frac{(-0.289 \cdot 1) + (-0.612 \cdot 1.5) + (0.816 \cdot (-1))}{0.289 + 0.612 + 0.816} = 2.82$$

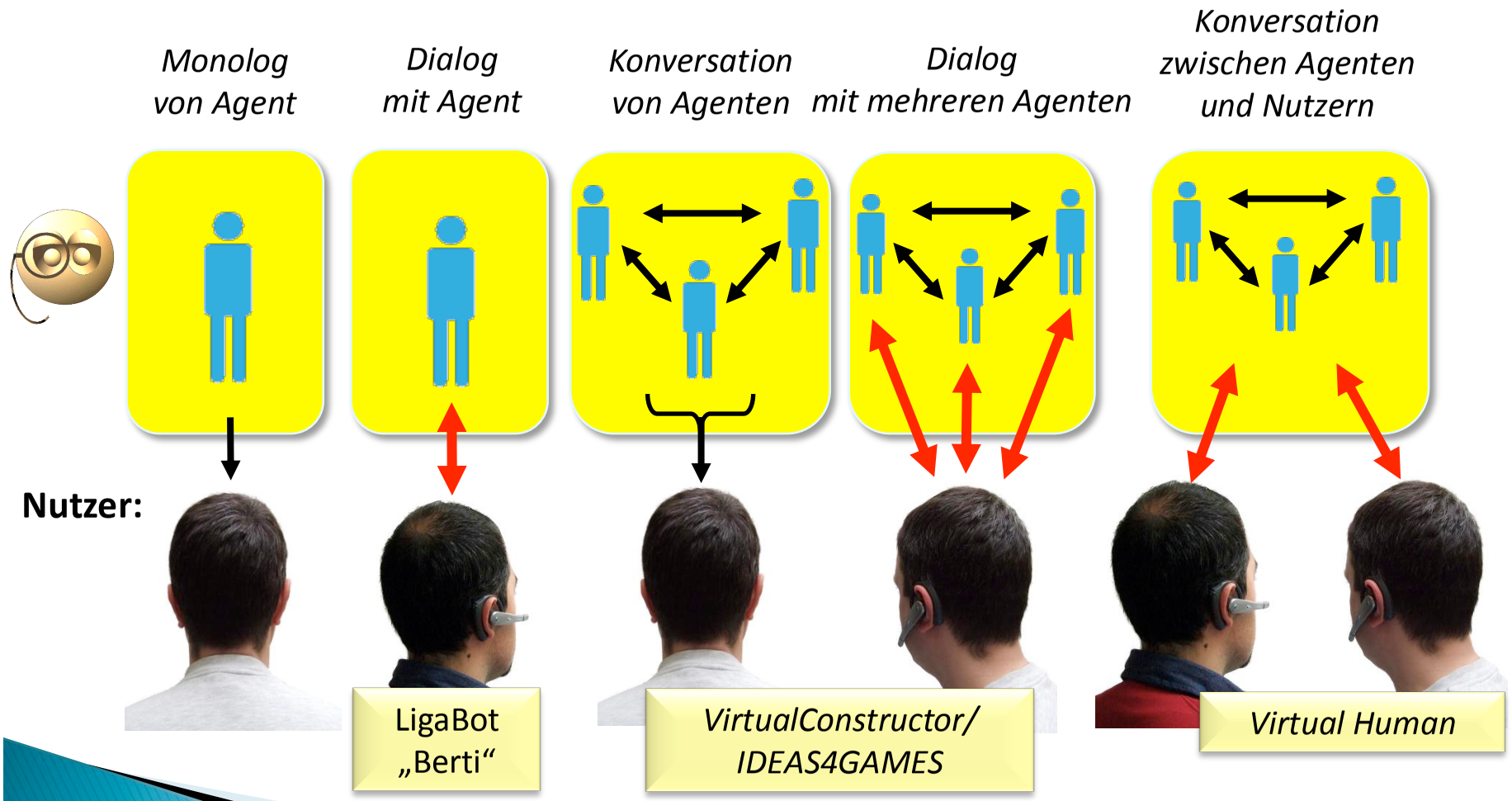
	Ran	Casablanca	Ben Hur	Tomb Raider	MI -II	Air Force One
Jane	5	5			**2.8**	2
Bill		2		3	4	
Tom	2		2		5	5
Cathy	3	3			1	1

- **Empfehlung an Jane**, falls die vermutete Präferenz p hinreichend hoch ist, d.h. über dem aktuellen, kurz- oder langfristigen Interessensschwellwert von Jane liegt.

Probleme von CF: Cold Start (→externe Quellen), Popularity Bias

Interaktive Dienste mit Avataren

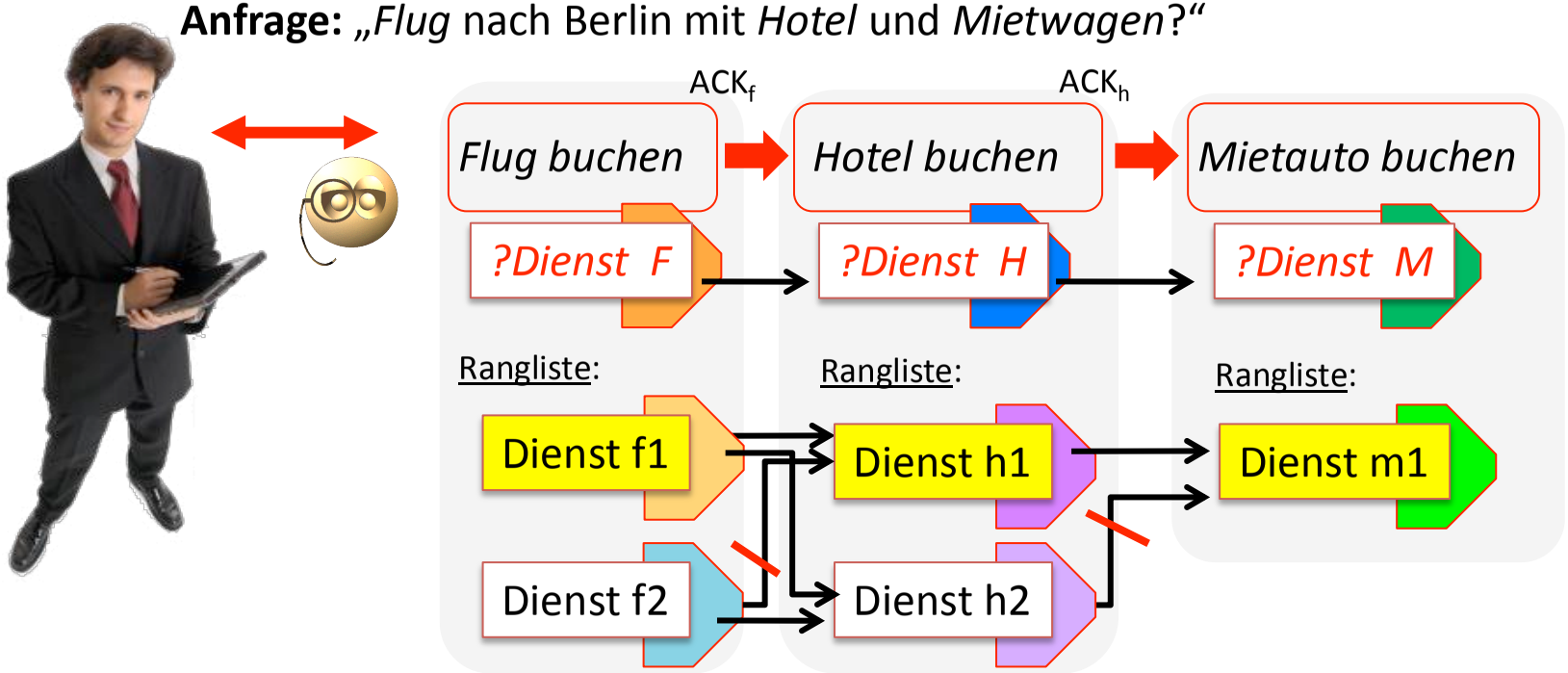
- Arten der Interaktion zwischen Nutzer und Agent für explizite Ermittlung von Nutzerprofildaten und/oder persönlich interaktive Koordination von Inhalten/Diensten:



Kein passender Dienst vorhanden? Komposition!

- Beantwortung einer Anfrage mit einem geeigneten **Dienstplan**.
- **Plan**: *Ausführbare Folge von relevanten, kompatiblen Diensten, die für gegebene Eingabe die gewünschte Ausgabe (Ziel) liefert*.

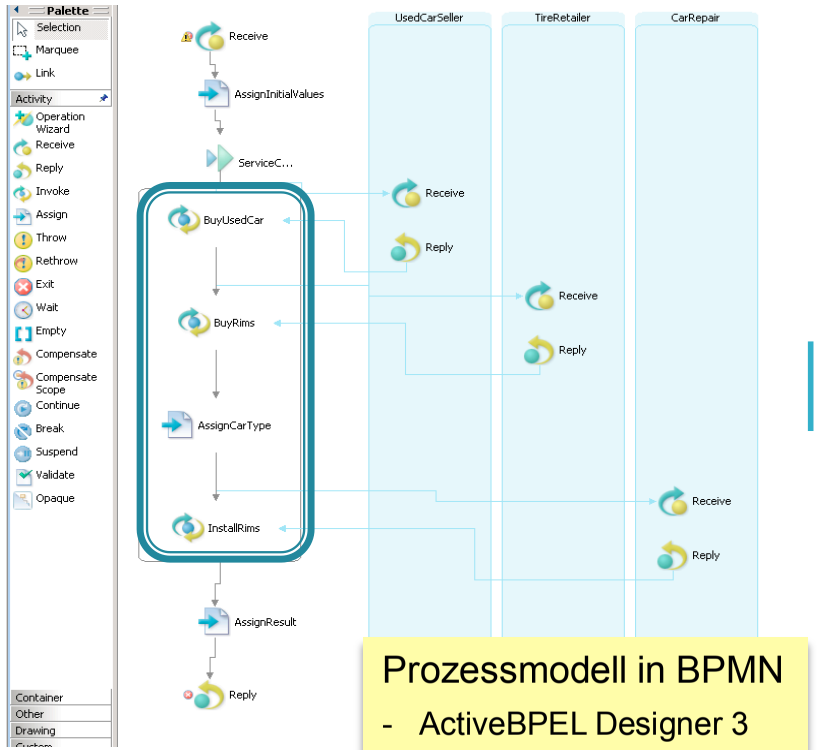
Anfrage: „*Flug* nach Berlin mit *Hotel* und *Mietwagen*?“



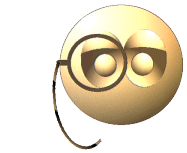
- Prinzipielle Folge (Kontrollfluss) ist vorgegeben oder wird vom Agenten (semi-)automatisch aus Anfrage und Kontext erzeugt.

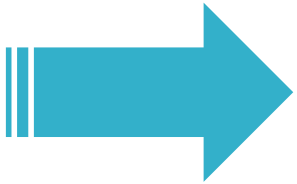
Beispiel: Dienstkomposition für Prozessmodell

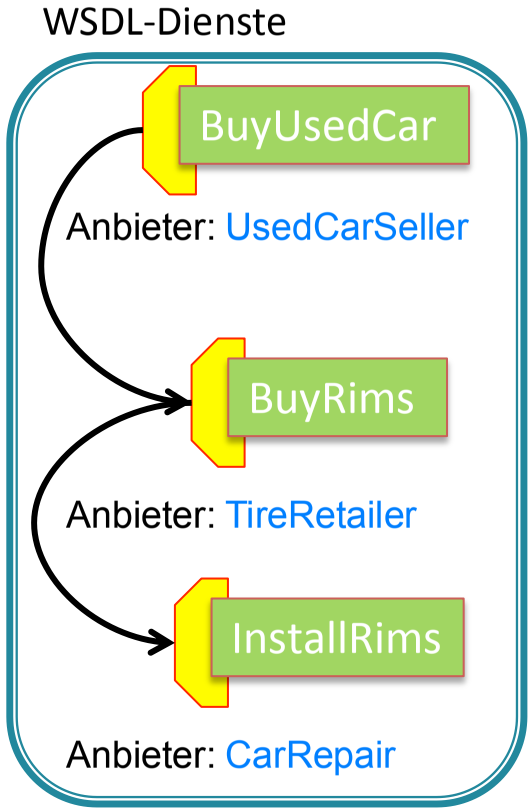
- **Geschäftsprozessmodell** in BPMN: Prinzipieller Kontrollfluss für einen Dienstplan in OMG-Standard BPEL (Business Process Execution Language; XPDL) gegeben.
- **Text-/strukturähnlichkeitsbasierte Selektion** von kompatiblen Diensten für Teilprozesse:



Prozessmodell in BPMN
- ActiveBPEL Designer 3
- ARIS Business Analyst

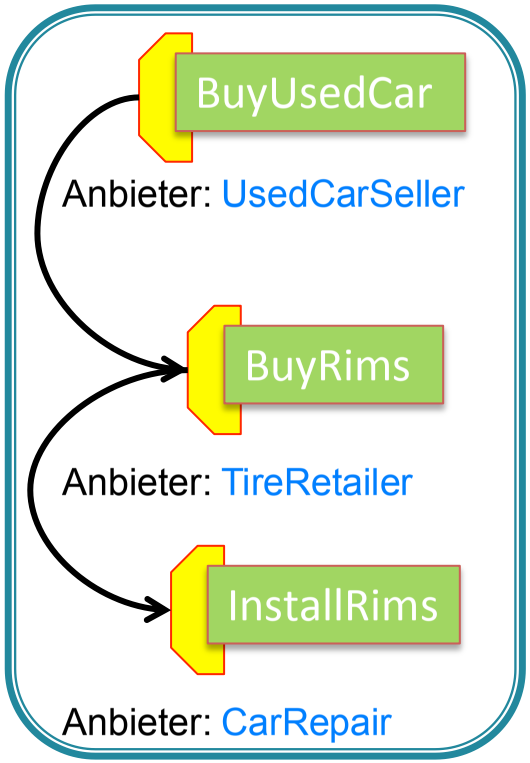


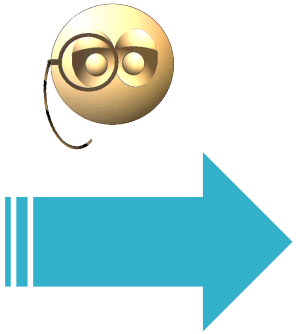




Beispiel: Dienstkomposition für Prozessmodell

- **Dienstorientierte Realisierung des Gesamtprozesses** in ausführbarem BPEL-Skript mit Angabe von Datenfluss zwischen relevanten WSDL-Diensten





```
... <bpel:flow> <bpel:sequence name="ServiceComposition"> ...
<bpel:invoke
  inputVariable="orderCarMessage" name="BuyUsedCar" operation="OrderCar"
  outputVariable="orderResponse"  partnerLink="UsedCarSeller" portType="ns4:CarProvider"/>

<bpel:invoke
  inputVariable="orderRimsRequest" name="BuyRims" operation="orderRims"
 outputVariable="orderRimsResponse" partnerLink="TireRetailer" portType="ns2:TireRetailer"/>
 <bpel:assign name="AssignCarType">
 <bpel:copy> <bpel:from part="orderedCar" variable="orderResponse"/>
             <bpel:to part="car" variable="installMessage"/></bpel:copy>
 <bpel:copy> <bpel:from part="orderResponseDetails" variable="orderRimsResponse"/>
            <bpel:to part="rims" variable="installMessage"/> </bpel:copy> ...

<bpel:invoke
  inputVariable="installMessage" name="InstallRims" operation="install"
  outputVariable="installResponse" partnerLink="CarRepair" portType="ns3:installRims"/>
</bpel:sequence>
... <bpel:copy> <bpel:from part="car" variable="executeRequest"/>
              <bpel:to part="car" variable="orderCarMessage"/></bpel:copy>
   <bpel:copy>  <bpel:from part="rims" variable="executeRequest"/>
              <bpel:to part="orderDetails" variable="orderRimsRequest"/></bpel:copy>
... <bpel:copy>  <bpel:from part="installAck" variable="installResponse"/>
              <bpel:to part="carAck" variable="executeResponse"/></bpel:copy>
   <bpel:copy> <bpel:from part="installAck" variable="installResponse"/>
              <bpel:to part="rimsAck" variable="executeResponse"/></bpel:copy>
 </bpel:flow>
</bpel:process>
```

BPEL-Skript des Prozessmodells realisiert mit WSDL-Diensten

→ **Ausführbar** mit BPEL-Engines

Beispiel: Dienstkomposition für Anfrage

- **Anfrage**: „Kauf von gebrauchtem Ferrari mit Spezialfelgen R1 !“
- **Text-/strukturähnlichkeitsbasierte Selektion mit regelbasierter Komposition**:

(1) Selektiere Dienst mit Eigenschaft: Verkauf von gebrauchten Ferraris?

Findet *BuyUsedCars*: „Verkaufe u.a. rote, gebrauchte Ferraris …. “

Keine Erwähnung von R1-Felgen in Dienstbeschreibung.

→ Semantisch erweiterte Interpretation „mit [installierten, montierten,..] Spezialfelgen R1“

→ Selektiere Dienst mit Eigenschaft: Installation von Felgen R1?

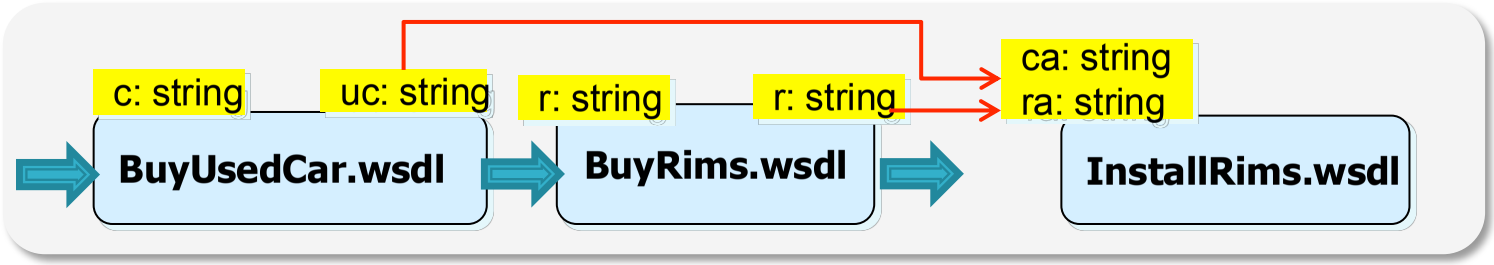
Findet *InstallRims: „*Installiert Felgen (u.a. R1) auf Autos (u.a. Ferraris)“.

Erfordert Felgen als Eingabe, verkauft aber keine. Profil: *Nutzer besitzt keine Felgen R1.*

→ Selektiere Dienst für Kauf von Felgen R1: Findet *BuyRims* „Nur Verkauf von Felgen (u.a. R1)..“

Beispiel: Dienstkomposition für Anfrage (2)

- **Anfrage**: „Kauf von gebrauchtem Ferrari mit Spezialfelgen R1 !“
- **Text-/strukturähnlichkeitsbasierte Selektion mit regelbasierter Komposition**
- **Prüfung der Planausführbarkeit**: Erforderliche Datentypkompatibilitäten (Datenfluss)



Problem

- Allgemein relativ geringe Präzision der Dienstselektion nur mit Methoden für Text- und Strukturähnlichkeit (z.B. WSDLAnalyzer: 0.42 AP, 4sec AQRT).
- WSDL / REST bietet keine formale Beschreibung der Funktionalität (Ausführung).
- Entwurf von expliziten Kompositionsregeln bereichsabhängig und mühsam.

Herausforderung im Internet der Dienste

- **Höhere Präzision** der Suche nach **relevanten** Diensten!
- **Automatische, logische Komposition** von Diensten mit klassischen KI-Planverfahren!
- **Bessere Erkennung von semantischen Relationen** zwischen relevanten Diensten und Daten für Anfragen!

Besser maschinenverständliche Beschreibung der Bedeutung (Semantik) von Daten und Diensten!

Agenda

- Intelligente Agenten
- Agenten und Webdienste
- Agenten und Semantische Dienste

Semantisches Web

„To make the semantics of Web resources machine-understandable.“

2001
Tim Berners-Lee,
Ora Lassila, Jim Hendler

Seit der IBM-Computer „Watson“ seine menschlichen Mitspieler beim US-Quiz „Joepardy“ haushoch besiegte, sind **semantische Technologien** einer breiteren Öffentlichkeit bekannt.

Handelsblatt, 23.03.2011

Semantisches Web: Prinzipien

(1) Hinreichend machinenverständliche Repräsentation der Bedeutung von Ressourcen im Web:

- **Formale Ontologien**

  mit logisch definierten Konzepten, Relationen, Objekten eines Bereichs

- **Semantische Annotation von Ressourcen (Daten, Dienste)**

  - Referenzen (URI) auf Konzepte/Objekte in geeigneten Ontologien
  - Interoperabel: Standards für formale Ontologiesprachen (OWL2, RDFS)

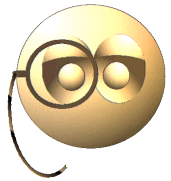
(2) Maschinelles Verständnis der Semantik und ihre Verwendung

- **Logikbasiert semantische Inferenzen über Annotationen**

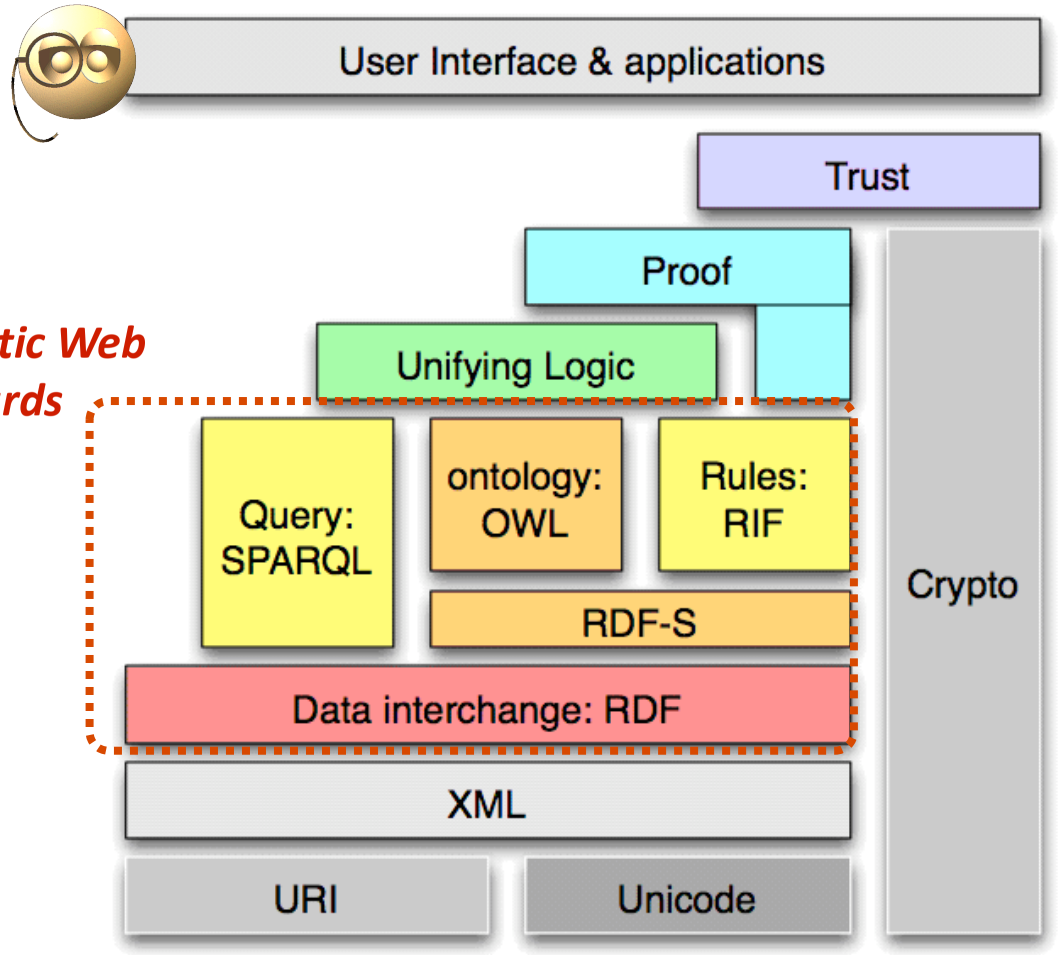
  für Exploration von (implizitem) Wissen und Relationen zwischen Ressourcen

- Automatische, semantische Koordination von Ressourcen

  für intelligente Webdienstanwendungen **durch intelligente Agenten**



Semantisches Web: Architektur



Semantic Web Standards

Semantic Web „light“
a.k.a *Web of Linked Data*

→ **RDF, RDFS, SPARQL**

Tim Berners-Lee (W3C), 2005

Beispiel: Semantische Annotation in OWL

Retailer web site:
www.mercedes-retailer.com

<html xmlns="http://www.mo.com/**O1**">

[M37] Rotes Mercedes Coupé:
BJ 2009, Guter Zustand, 2.Hd,
Preis: …
 … </html>

Mercedes-Ontologie **O1** in OWL:

Konzepte: ….

Coupe = (and **O2:Car** (=3 hDoors))

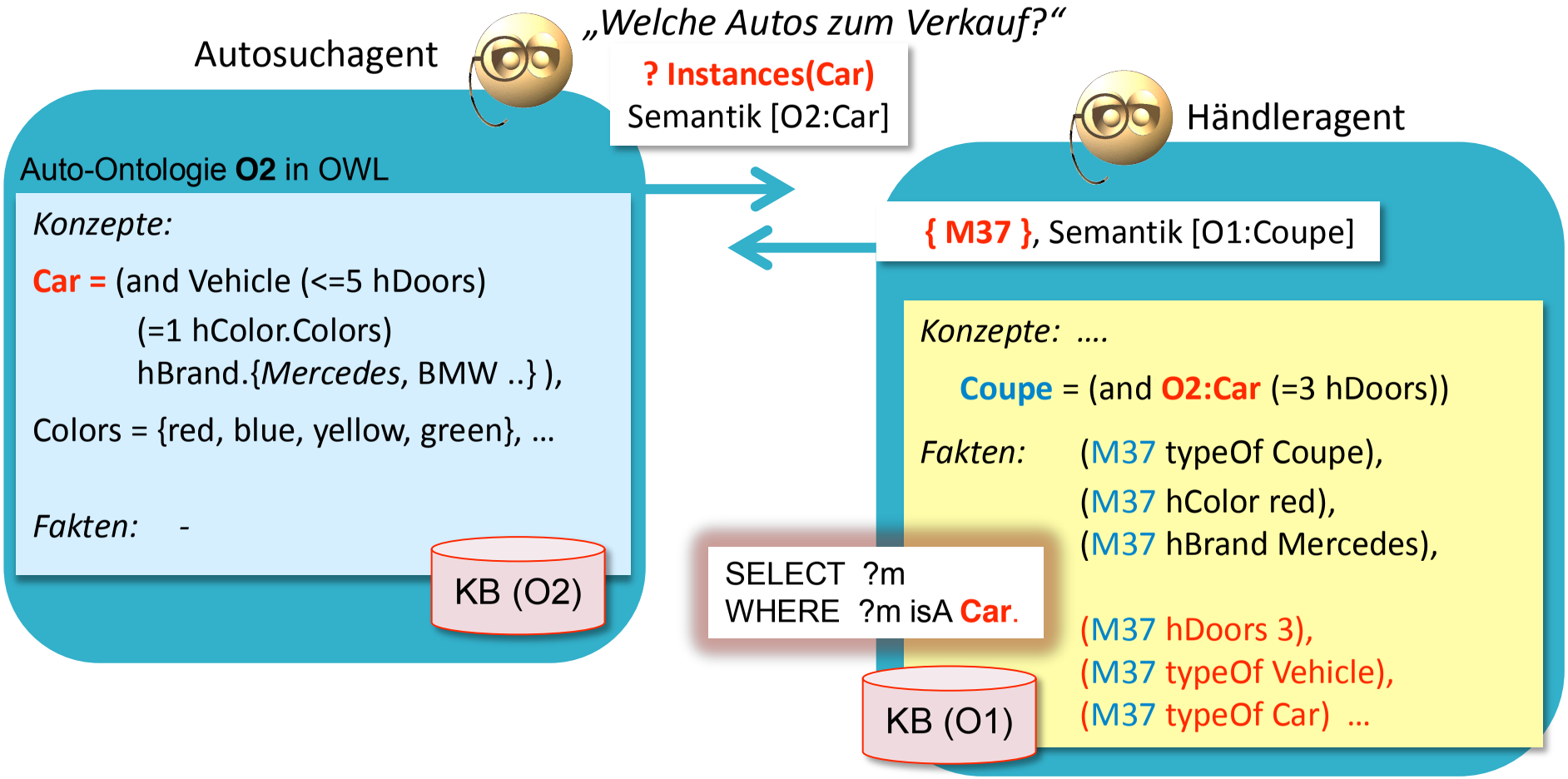
Fakten: (M37 typeOf Coupe),
(M37 hColor red),
(M37 hBrand Mercedes),

(M37 hDoors 3),
(M37 typeOf Vehicle),
(M37 typeOf Car)

KB (O1)

- OWL ist Standardontologiesprache mit formaler (modelltheoretischer) Semantik
- Wissensbasis: Explizites und automatisch abgeleitetes implizites Wissen
- Fakten in Tripel-Form (Datenmodell RDF): (Subjekt, Prädikat, Objekt)

Beispiel: Agentenbasierte semantische Datensuche



Beispiel: Agentenbasierte semantische Datensuche

Autosuchagent

Auto-Ontologie **O2** in OWL

Konzepte:

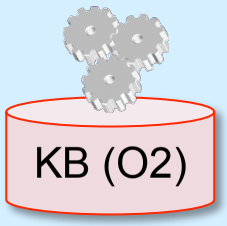
Car = (and Vehicle (<=5 hDoors)
(=1 hColor.Colors)
hBrand.{*Mercedes*, BMW ..}),

Colors = {red, blue, yellow, green}, …

O1:Coupe = (and Car (=3 hDoors))
[www.mercedes-retailer.com]

Fakten:

(M37 typeOf Coupe),
(M37 hColor red),
(M37 hBrand Mercedes)
(M37 hDoors 3),
(M37 typeOf Vehicle),
(M37 typeOf Car)



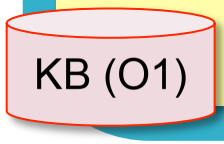
Händleragent

Konzepte: ….

Coupe = (and **O2:Car** (=3 hDoors))

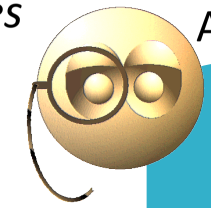
Fakten: (M37 typeOf Coupe),
(M37 hColor red),
(M37 hBrand Mercedes),

(M37 hDoors 3),
(M37 typeOf Vehicle),
(M37 typeOf Car) …



Beispiel: Agentenbasierte semantische Datensuche

„Red colored Mercedes with 3 doors?“



Autosuchagent

M37 (Coupe)
Offered for sale by:
www.mercedes-retailer.com

Auto-Ontologie **O2** in OWL

Konzepte:

Car = (and Vehicle (<=5 hDoors)
(=1 hColor.Colors)
hBrand.{*Mercedes*, BMW ..}),

Colors = {red, blue, yellow, green}, …

O1:Coupe = (and Car (=3 hDoors))
[www.mercedes-retailer.com]

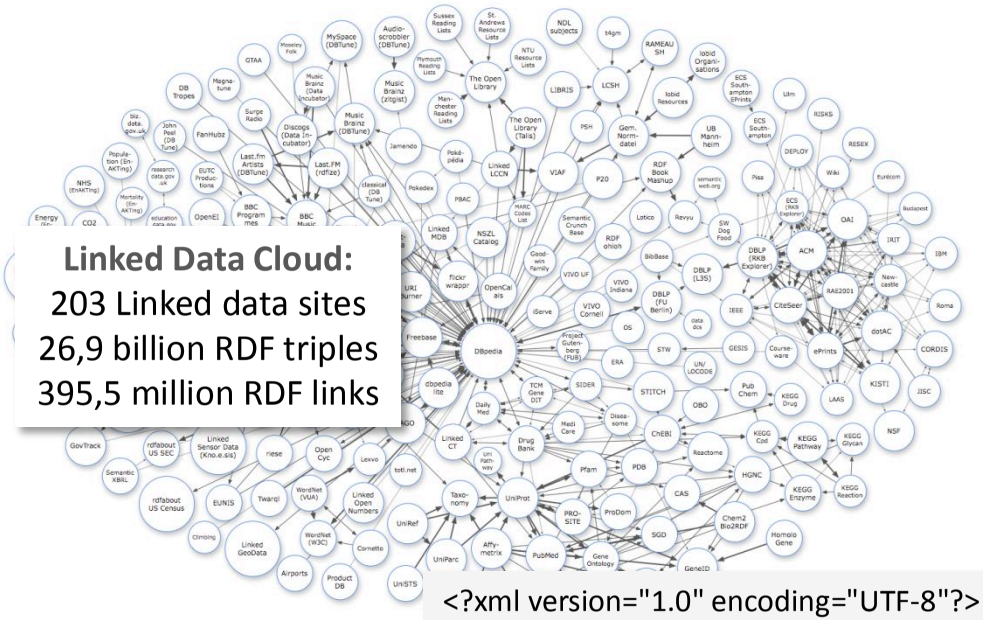
Fakten:

(M37 typeOf Coupe),
(M37 hColor red),
(M37 hBrand Mercedes)
(M37 hDoors 3),
(M37 typeOf Vehicle),
(M37 typeOf Car)

KB (O2)

```
SELECT  ?m
WHERE   ?m isA Car.
        ?m.hBrand  = 'Mercedes'
        ?m.hColor  = 'red'.
        ?m.#hDoors = 3
```

Beispiel: Semantisches „Web of Linked Data“



Linked Data Cloud:
203 Linked data sites
26,9 billion RDF triples
395,5 million RDF links

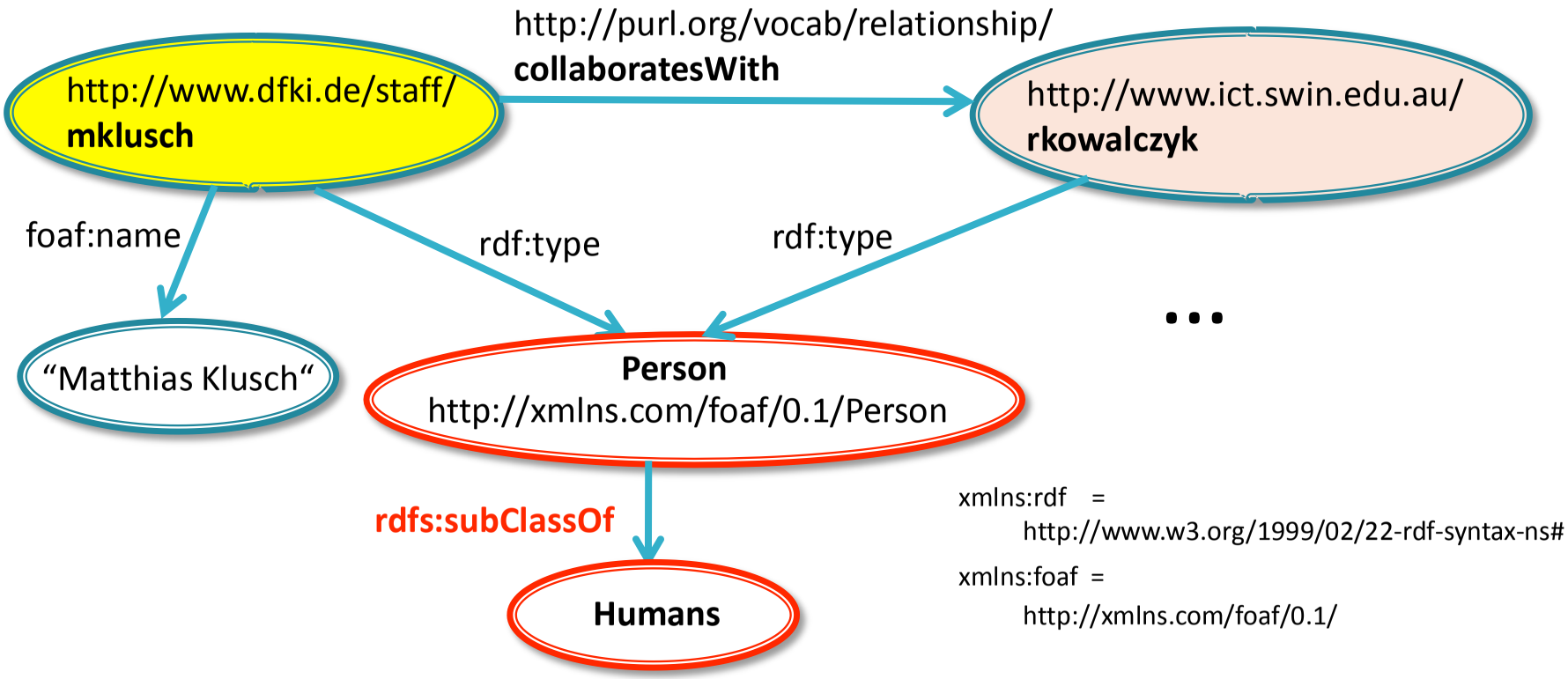
- **Referenzierung von Dingen mit URIs.**
  Bsp. einer “*Thing*-URI”:
  http://www.dfki.de/staff/mklusch
- **Beschreibung der Semantik von Dingen mit Datenmodell RDF.**
- Semantik **enthält Referenzen zu relevanten Dingen.**

Semantik von „Ding“ mklusch in RDF

mklusch.rdf

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf=http://xmlns.com/foaf/0.1/, xmlns:rel=http://purl.org/vocab/relationship/>
 <rdf:Description rdf:about="http://www.dfki.de/mklusch">
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
    <foaf:name> Matthias Klusch </foaf:name>
    <foaf:isPrimaryTopicOf> http://www.dfki.de/staff/mklusch.rdf </foaf:isPrimaryTopicOf>
    <rel:collaboratesWith> http://www.ict.swin.edu.au/rkowalczyk </rel:collaboratesWith>
</rdf:Description>  </rdf:RDF>
```

Beispiel: Semantik einer verlinkten Ressource in RDF …



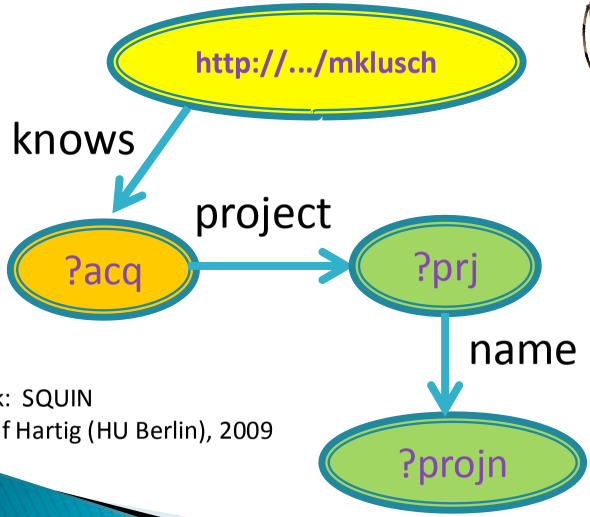
- **RDF** – Resource Description Framework:
  Für strukturierte Fakten über Ressourcen im Web in Tripelform (S, P, O)
- **RDFS** – RDF Schema: Für Teilklassen- und Rollenbeziehungen
- W3C Standards; Modelltheoretische bzw. regelbasierte Semantik

Beispiel: Anfrageverarbeitung im Web of Linked Data

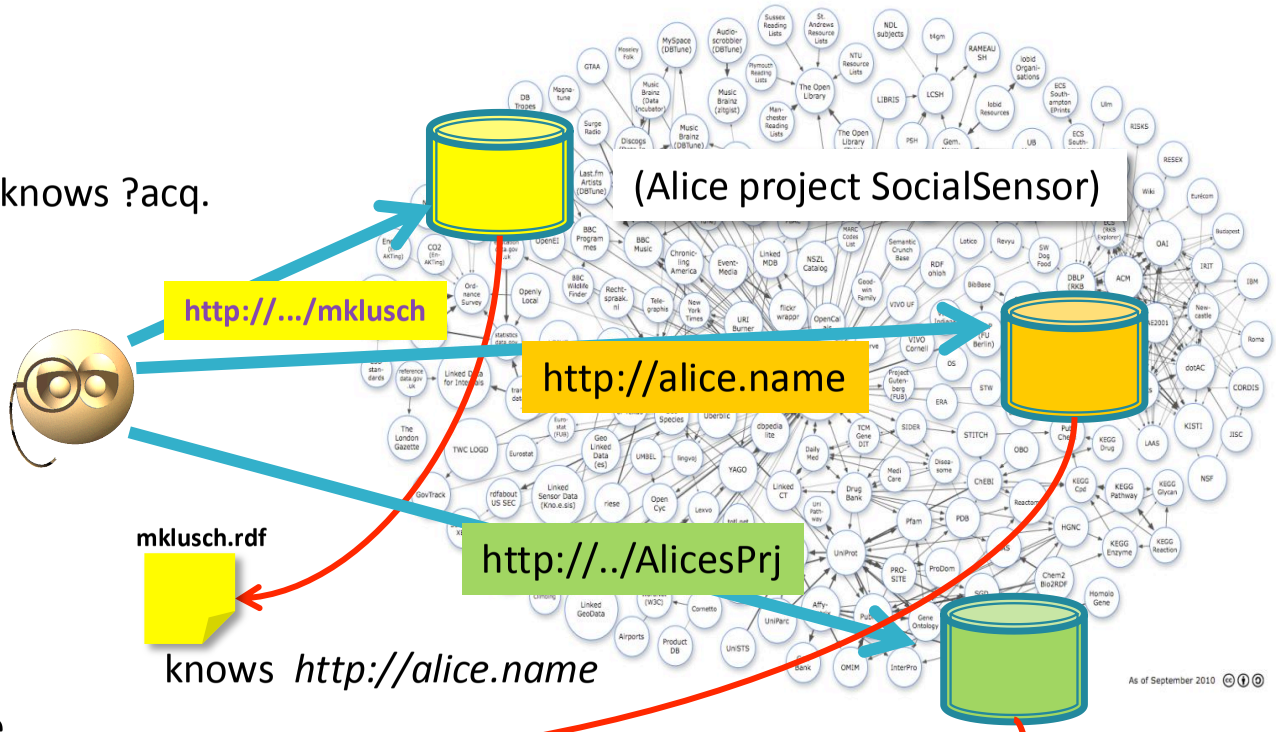
Können wir Anfragen an das Web wie an eine einzelne große Datenbank stellen ?

„In welchen Projekten arbeiten Freunde von mklusch?“

```
SELECT ?acq ?projn
WHERE
   http://.../mklusch  knows ?acq.
   ?acq project ?prj.
   ?prj  name  ?projn.
```



Ack: SQUIN
Olaf Hartig (HU Berlin), 2009



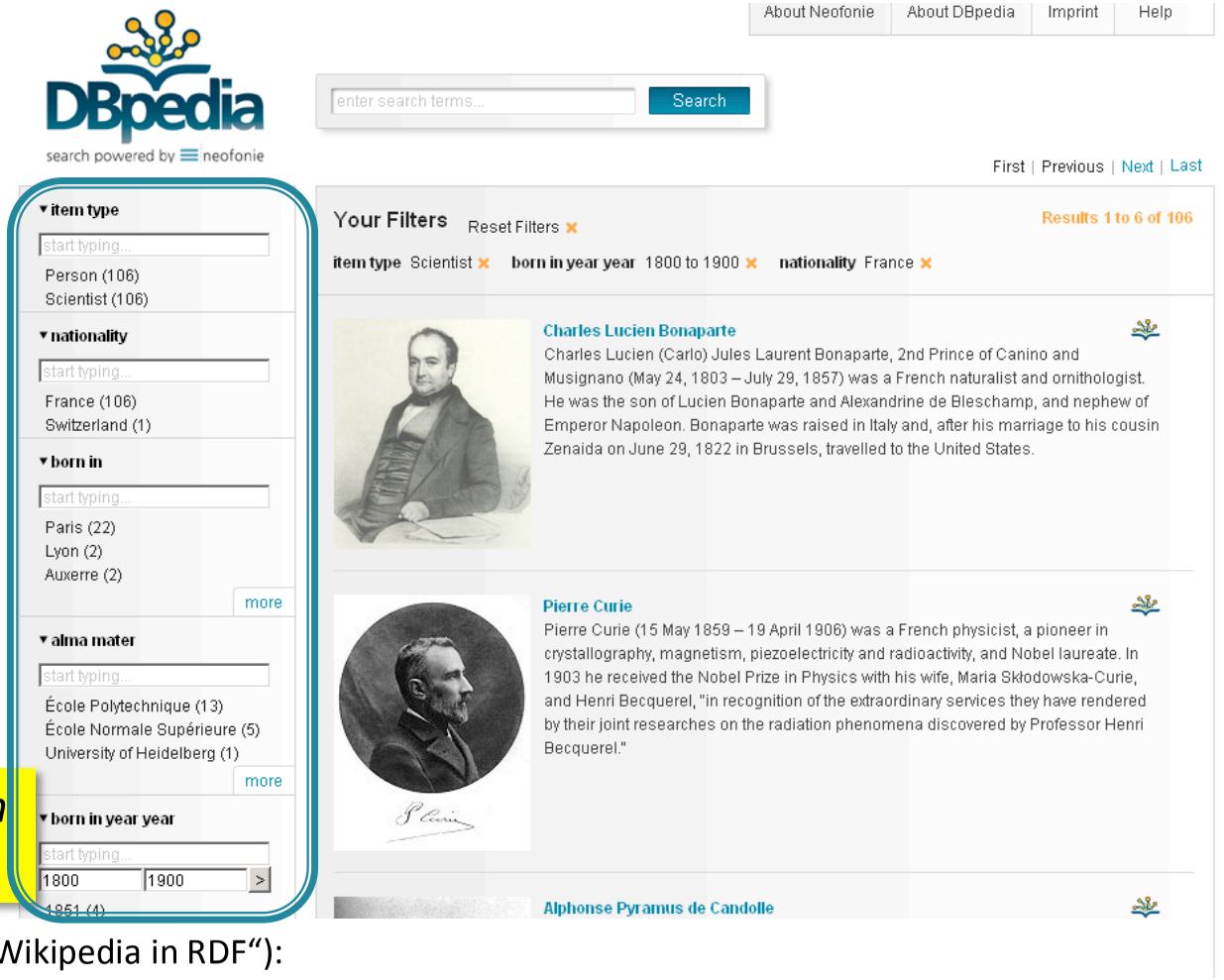
Anwendung: Semantisch „facettierte“ Suche

Strukturiert semantische Suche

- Jede RDF/S-Semantik einer Ressource bietet u.a *Menge relevanter Ressourcen mit Eigenschaften* (**Facetten**) und Links...
- Sukzessive Filterung von Suchresultaten mit *semantisch relevanten* Facetten.

Keine reine Schlagwortsuche… relationale Anfragen!

“French scientists who were born in the 19th century?”



→ REST-API von Dbpedia („Wikipedia in RDF“):

http://dbpedia.neofonie.de/browse/**rdf-type:Scientist/birthDate-year~:1800~1900/nationality:France**/

Was ist ein semantischer Webdienst ?

Formale Ontologien

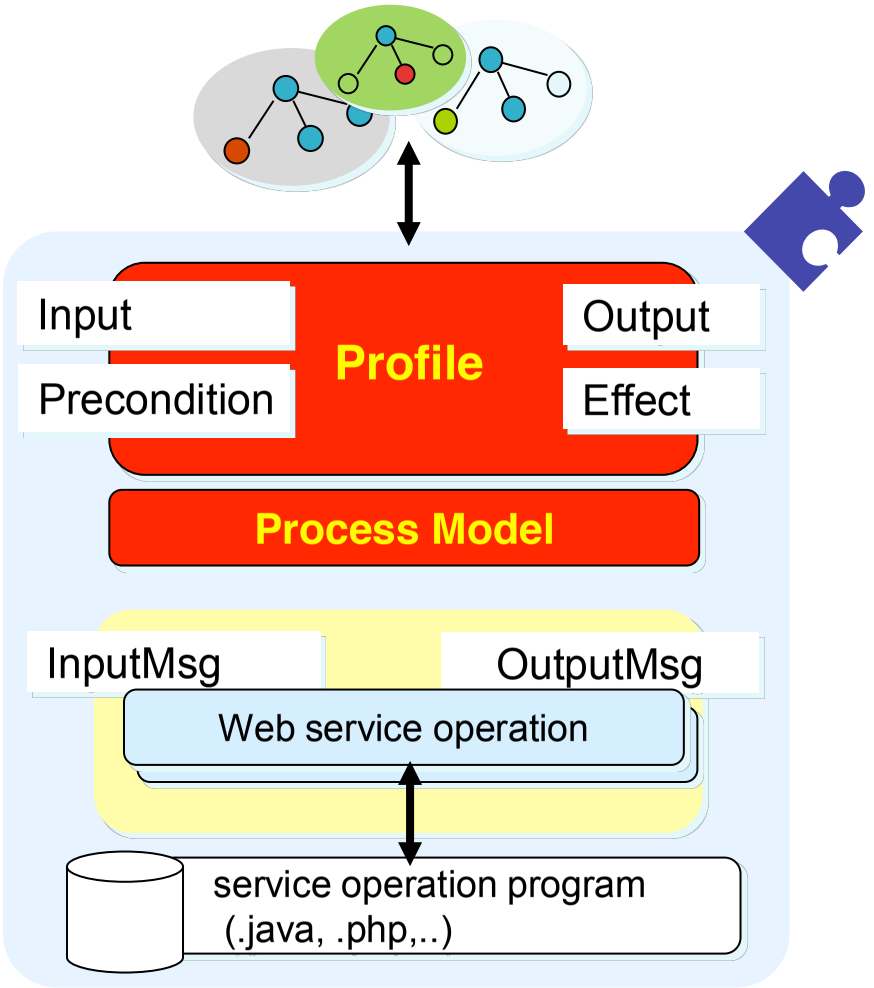
(in RDFS, OWL2, WSMO, SWRL etc)

Semantischer Dienst

(in OWL-S, WSML, SAWSDL, SAREST)

Webdienst

(in WSDL, REST)



Beispiel: Semantischer Dienst „BookFlight“ in OWL-S

Ontology O1:

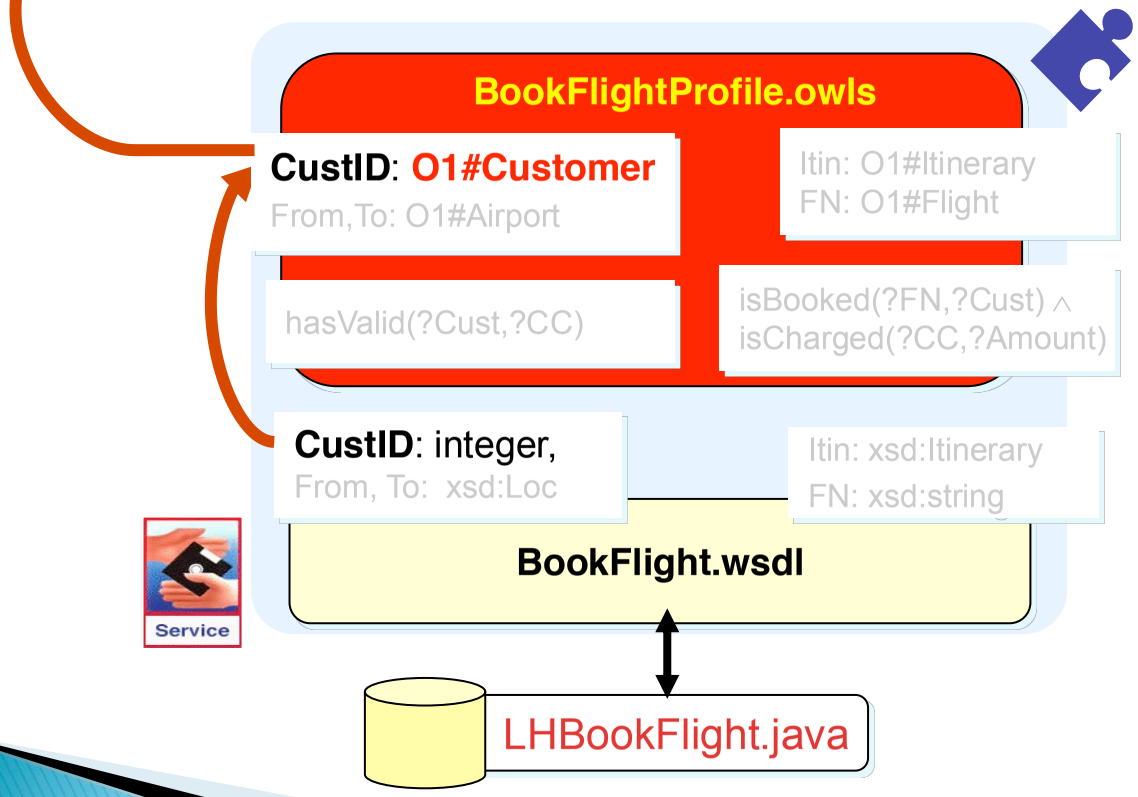
Konzepte und Relationen:

Definitionen in Logik OWL2-DL

Person = (and Human), …

Customer = (and Person (≥1 hasValid.CreditCard) …)

Fakten: Customer(**Hans**), Flight(LH951), CreditCard(VISA), hasValid(Hans,VISA**7)..



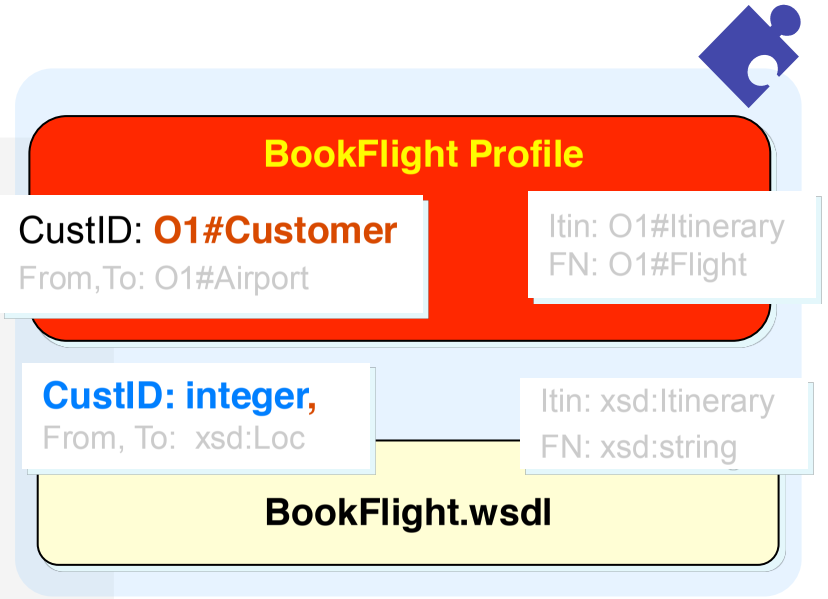
Dienstsprache OWL-S:

- Signatur (I/O)
- Funktion (P/E)
- Annotationen nur in OWL2 definiert
- Kein Standard

… derselbe Dienst im Standard SAWSDL

BookFlight.sawsdl

```
<wsdl xmlns:sawsdl="http://www.w3.org/2002/ws/sawsdl"
<interface name="BookFlightInterface">
<operation name="BookFlightOp" …>
<input messageLabel="CustID" element="xsd:integer"
      sawsdl:modelReference = "O1#Customer"/>
<output messageLabel="FN" element="xsd:string" /> …
</operation> </interface> <binding … …. </binding>
<service name="BookFlight" interface="xs:BookFlightInterface">
<endpoint … address="http://www.airlineX.com/booking/"> …
</wsdl>
```



SAWSDL: Semantic Annotations for WSDL and XML Schema.

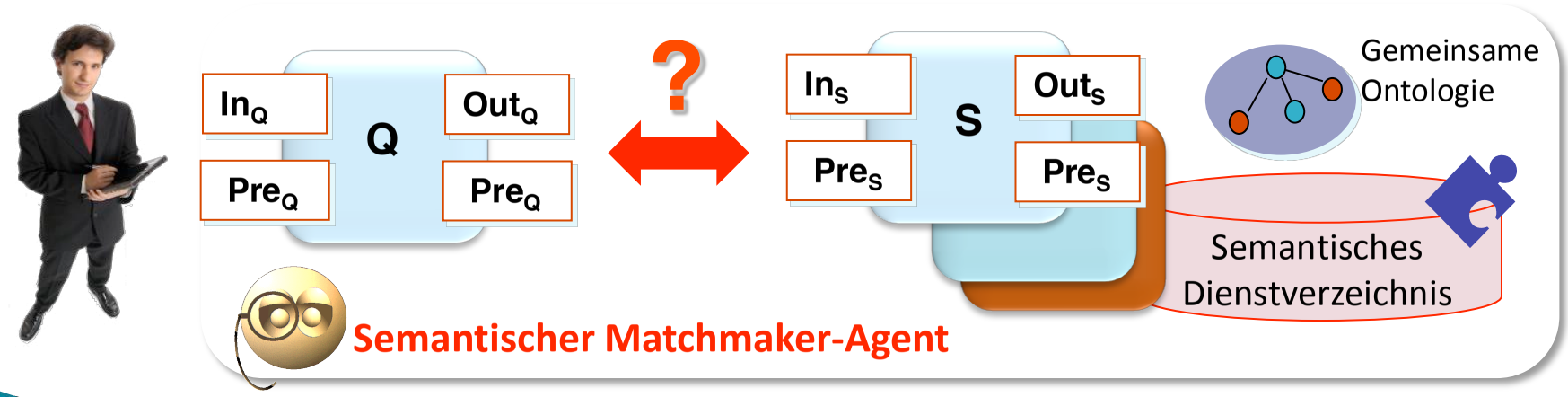
Dienstsprache SAWSDL:

- Nur Signatur
- Beliebige Ontologien
- W3C Standard 2007

Semantische Suche und Selektion von Diensten

- **Suche nach semantischen Diensten S im Web**
  - Spezielle Suchmaschinen-APIs: Sousuo, iSmoogle (DFKI), S3E
  - Suchmaschinen-APIs: Google, Bing

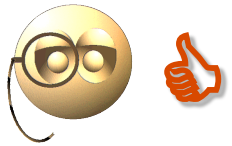
    z.B: Suche nach Dateiendungen .owls, .sawsdl, .wsml
  - Indizierung und Verwaltung im lokalen Dienstverzeichnis
- **Selektion von semantisch relevanten Diensten S für Anfrage Q**



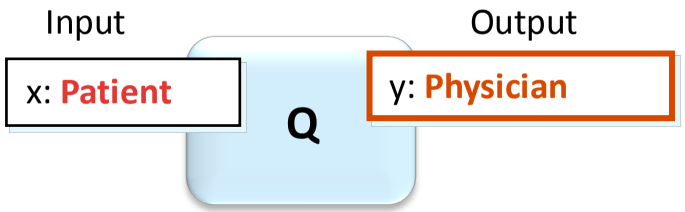
Beispiel: Logikbasiert semantische Selektion

- Dienst S substitutiert semantisch (plugs in) Anfrage Q

PlugIn(S,Q) =

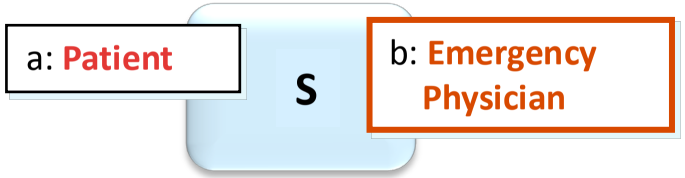


Dienstanfrage durch Nutzer:

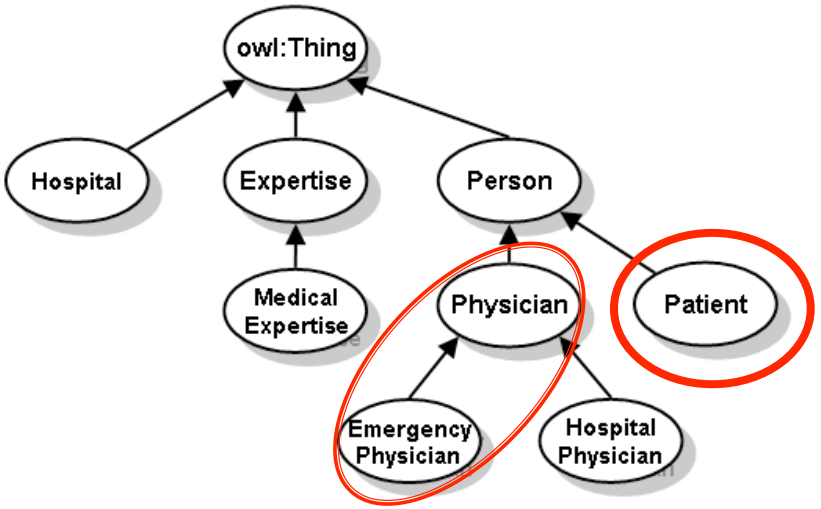


„Returns physician for given patient.“

Für Nutzer relevanter Dienst:



„Returns emergency physician for given patient.“



$\text{Physician} \equiv \text{Person} \sqcap \exists \text{trainedAt.MedicalExpertise}$

$\text{worksFor} \equiv \text{employed}^{-}$

$\top \sqsubseteq \forall \text{worksFor.Hospital}$

$\text{MedicalExpertise} \sqsubseteq \text{Expertise}$

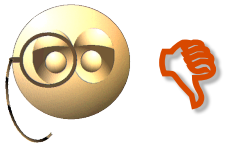
$\text{Patient} \equiv \text{Person} \sqcap =1\ \text{hospitalizedAt.Hospital}$

$\text{EmergencyPhysician} \equiv \text{Physician} \sqcap \geq 2\ \text{worksFor}$

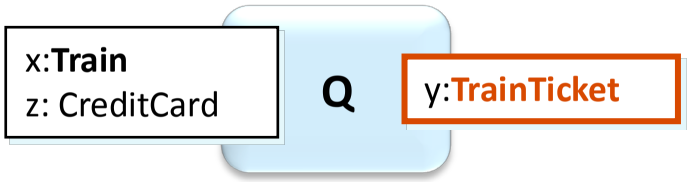
Beispiel: Logische Selektion kann irren …

- Dienst S subsumiert semantisch die Anfrage Q:

  subsumedBy(Q,S) =

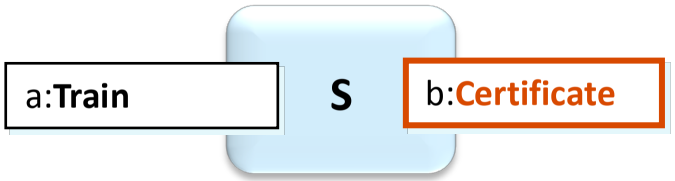


Dienstanfrage

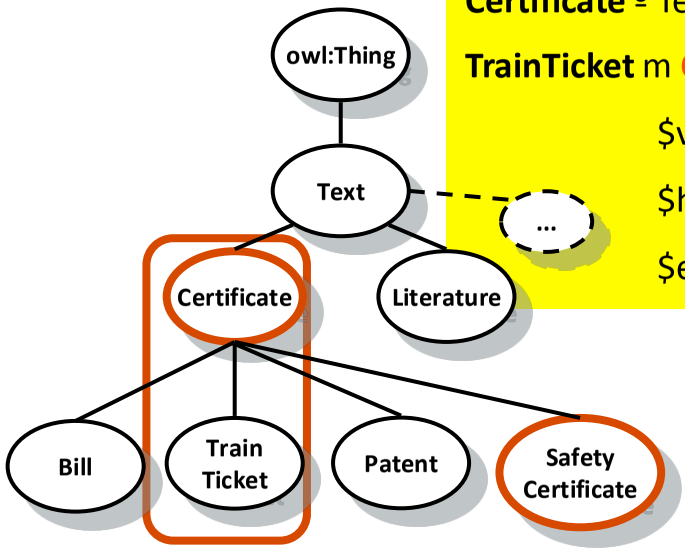


„Online purchase of train ticket with credit card.“

Nicht relevanter Dienst



„Issues **safety certificate** for given train.“



Certificate º Text * $issuedFor

TrainTicket m Certificate *

$validFor.Train *

$hasReservation.Seat *

$expiresAt.Date

TrainTicket < Certificate

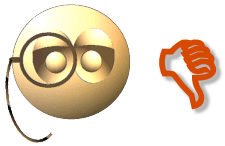
T[Certificate] = „and Certificate Text“

T[TrainTicket] = „and TrainTicket Certificate Text ex issuedFor ex validFor.Train ex hasReservation.Seat ex expiresAt.Date“

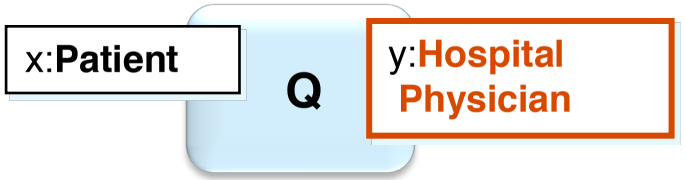
Geringe Textähnlichkeit → Geringe Relevanz

Beispiel: Logische Selektion kann irren … (2)

- Dienst S ist semantisch nicht relevant zur Anfrage Q: **Fail(S,Q)**

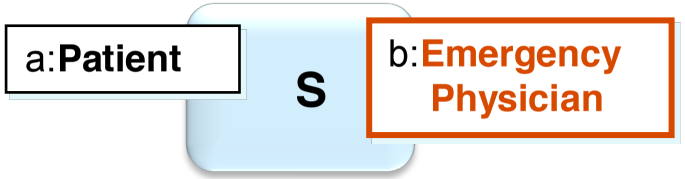


Dienstanfrage

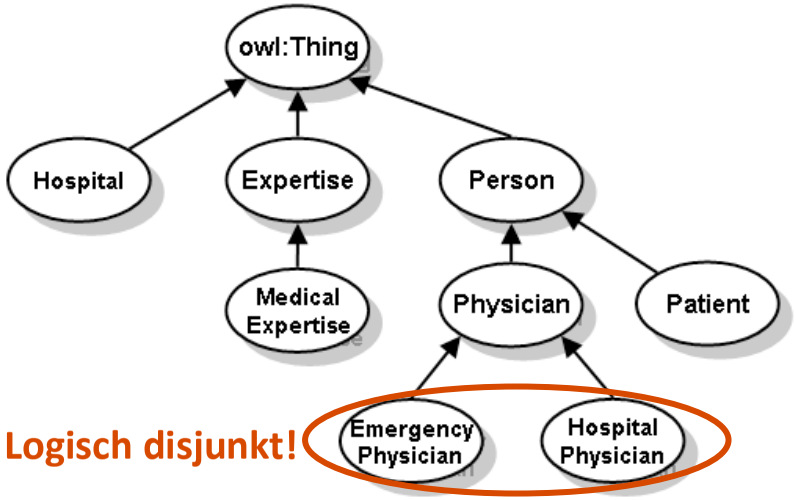


„Returns hospital physician for given patient.“

Relevanter Dienst



„Returns emergency physician for given patient.“



Logisch disjunkt!

EmergencyPhysician ≡ Physician ⊓ ≥2 worksFor
HospitalPhysician ≡ Physician ⊓ =1 worksFor

T[EmergencyPhysician] = “and EmergencyPhysician Physician Person trainedAt MedicalExpertise worksFor atleast 2 Hospital”

T[HospitalPhysician] = “and HospitalPhysician Physician Person trainedAt MedicalExpertise worksFor exactly 1 Hospital)”

Hohe Textähnlichkeit → Hohe Relevanz

Semantischer Matchmaker-Agent OWLS-MX2

- **Eingabe**: Dienstanfrage Q in OWL-S
  Geg.: Verzeichnis V von Diensten S in OWL-S mit Ontologie in OWL2,
  Textindex
- F.a Dienste S aus V: Bestimme semantische Relevanz von S für Q
  - 5 Abgleichsfilter: Kombination von logischer und Textähnlichkeit von semantischen Annotationen der Signaturen (E/A) von S und Q
- **Ausgabe**: Rangliste von relevanten Diensten: [(S, r, Anbieter)].
  Semantische Relevanzgrade r sortiert nach (1) Logisch (2) Textähnlichkeit

[Klusch, Fries, Sycara: Journal of Web Semantics, 7(3), 2009]

@ projects.semwebcentral.org/projects/owlsmx
(> 42.500 downloads)

International
Semantic Service Selection Contest:
www.dfki.de/~klusch/s3

Best Service Matchmaker

Herausforderung im semantischen Internet der Dienste

Wie kann ein Vermittleragent *selbsttätig*

beliebige Arten von semantischen Filtern

für maximale Präzision kombinieren?

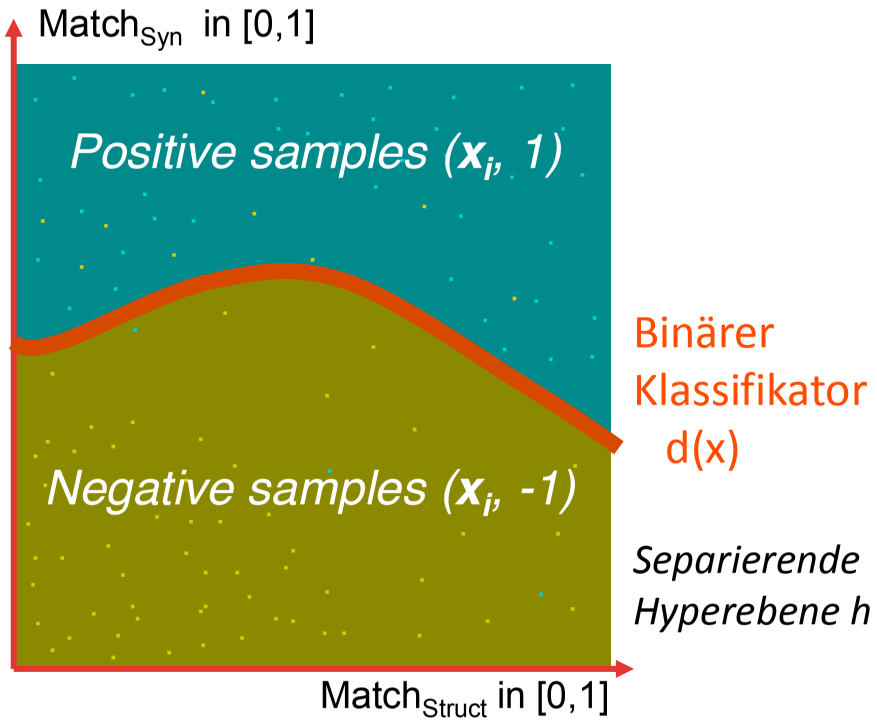
Beispiel: Lernender Matchmaker-Agent iSeM …

[Klusch, M.; Kapahnke, P. (2010): 4th IEEE International Conference on Semantic Computing]

(1) Lernen der besten Filterkombination über geg. Trainingskollektion:

F.a. (Anfrage q, Dienst $s)_i$: (1 / -1; x_i)

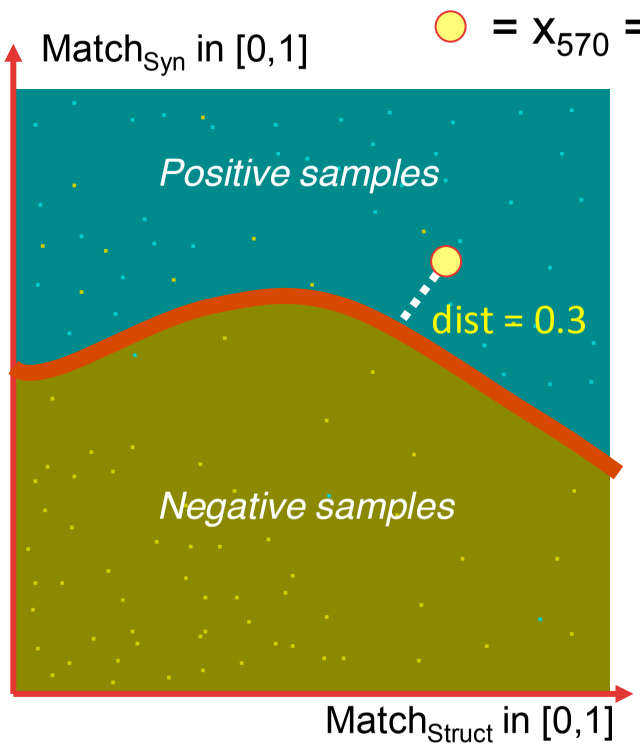
mit x_i = Vektor von Filterresultaten für (q,s)



Binärer Klassifikator d(x)

Separierende Hyperebene h

(2) Selektion von Diensten s für neue Dienstanfrage q mit gelerntem Klassifikator d:



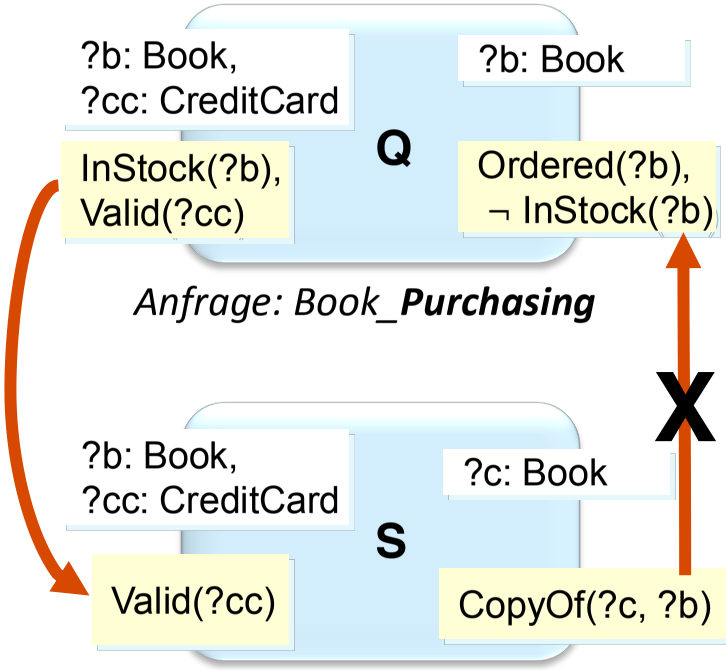
Klassifikation
d(x) = 1

Relevanzwert
dr = 1 – dist(x,h)

Rangliste: [… (10., s_{284}, 0.7) …]

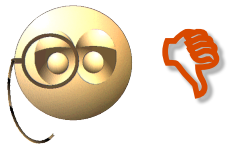
… mit vollständigem Dienstprofilabgleich …

Semantische Dienste in OWL-S enthalten logische Spezifikationen von **Vor- und Nachbedingungen** ihrer Ausführung.



- Logischer Signaturabgleich:

  IO(S,Q) = *Exact(S,Q)*

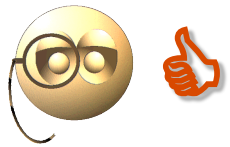


- Logischer Spezifikationsabgleich:

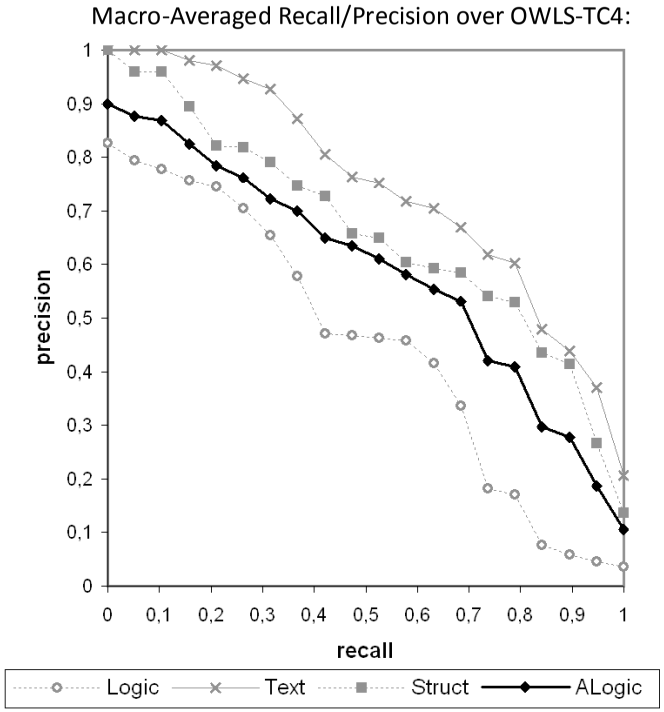
  $PE(S,Q) = (Pre_Q \rightarrow Pre_S) \;\&\; (Eff_S \rightarrow Eff_Q)$

$IOPE(S,Q) = IO(S,Q)$ **&** $PE(S,Q)$

= **False** „S ist irrelevant“



… bietet die (noch) präziseste Art der Dienstselektion!



Präzision der Filtervarianten von iSeM:

- Logikbasierte Ähnlichkeit: 0.41
- Textähnlichkeit: 0.74
- Ontologisch strukturelle Ähnlichkeit: 0.65
- Logic & Text & Struct: 0.76
- **Adaptiv gewichtet kombiniert: 0.92**

Antwortzeit: 2.34 sec (OWLS-MX2: 5.4 sec)

@ projects.semwebcentral.org/projects/isem

Best Service Matchmaker

2010

Herausforderung im semantischen Internet der Dienste

Wie können für eine gegebene Anfrage semantische Dienste automatisch zusammengesetzt werden ?

Arten der Komposition von semantischen Diensten

- **Automatisch:**
  - Reduktion auf KI-Planungsproblem
  - Anwendung von KI-Planverfahren
    - State-based planning [Klusch 05, et al.]
    - Model-checking [Traverso+ 01, et al.]
    - HTN-planning [SHOP2: Sirin+ 02]
- **Interaktiv:**
  - Vorwärts- oder Rückwärtskombination von Diensten durch Benutzer mit Hilfe von Matchmaker-Agenten

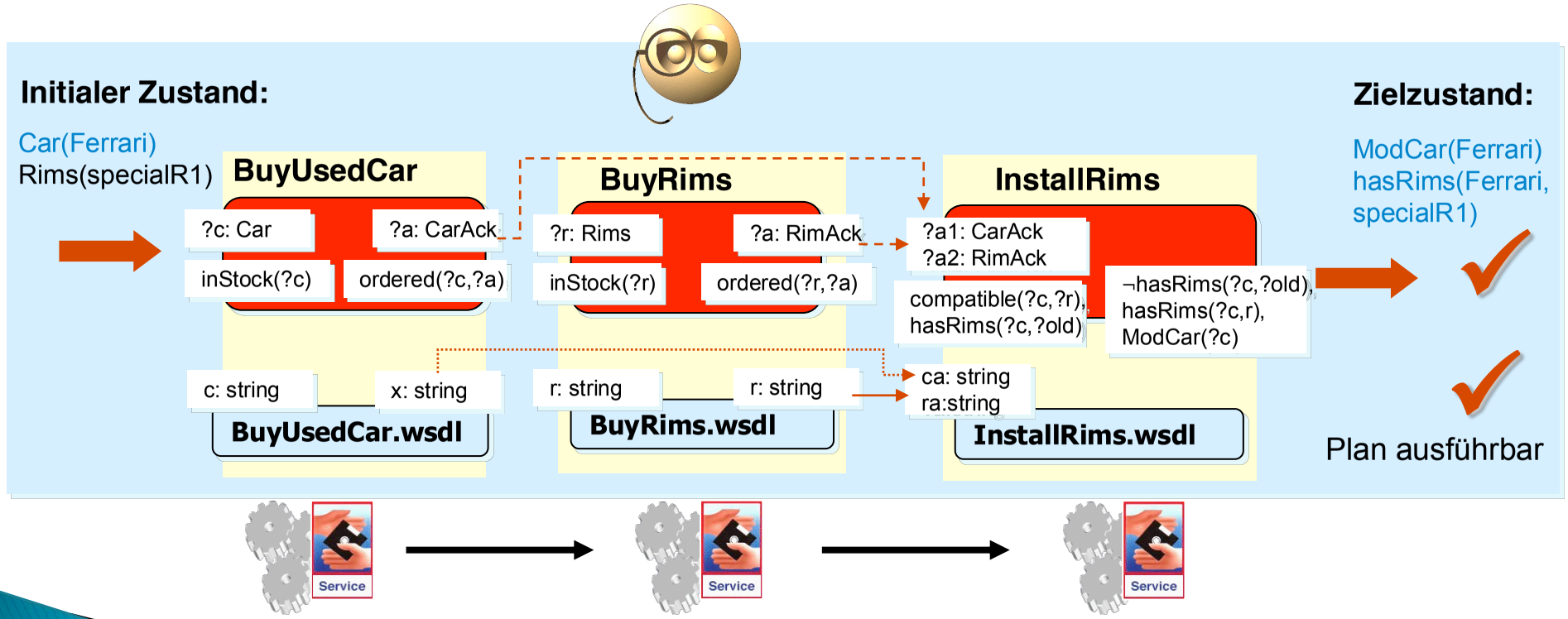
    [Sirin & Hendler 01, Henocque & Kleiner 07]

Beispiel: Semantische Komposition mit OWLS-Xplan

- **Reduktion der Dienstkomposition auf zustandsbasierte Aktionsplanung**
- **Anwendung von KI-Planer FF (GRAPHPLAN) und HTN-Planer**

[Klusch, Gerber (2006): 4th European Conference on Web Services (ECOWS)]

OWLS-Xplan @semwebcentral.org (> 5.000 downloads)



Einige Anwendungen semantischer Dienste

- Auswahl von Raumfahrtinformationsdiensten

  OWLS-MX in der NASA Suchmaschine SPASE (seit 2007)

- Auswahl von Werbungsdiensten für Nutzeranfragen

  OWLS-MX für Google: Pilotprojekt der U Zagreb mit Google (2006)

- Auswahl und Planung von mobilen medizinischen Notfalldiensten

  OWLS-MX und OWLS-Xplan in System CASCOM (EU Projekt), Firma EMA

- Koordination von Polizei-/Feuerwehr-/Ambulanzdiensten in Katastrophenszenarien (BMB+F Projekt)

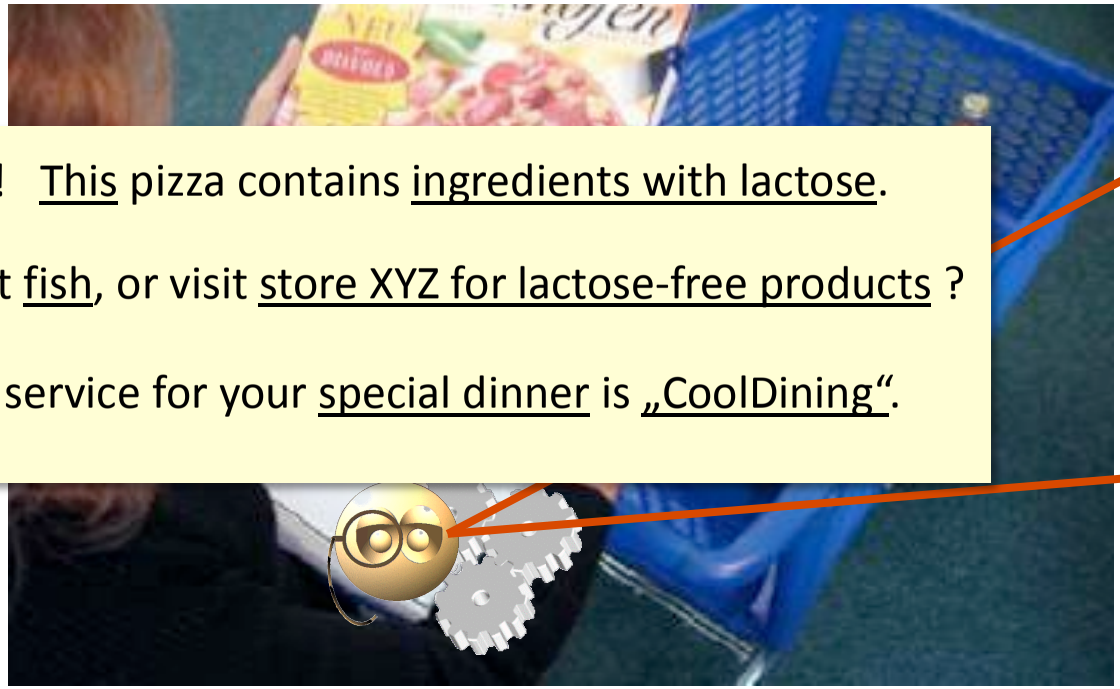
- Planung der Sequenzierung (Komposition) von Proteinen

  OWLS-Xplan an der U Rennes, Faculty of Medicine (France) seit 2008

Intelligenter Einkaufsagent

„Is this pizza good for my health? I am suffering from lactose intolerance !

Who offers the cheapest party service for my dinner? “



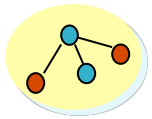
Attention! This pizza contains ingredients with lactose.

Wanna try out fish, or visit store XYZ for lactose-free products ?

Best party service for your special dinner is „CoolDining“.

Semantische Dienste:

- Allergy
- PartyService
- Pricing



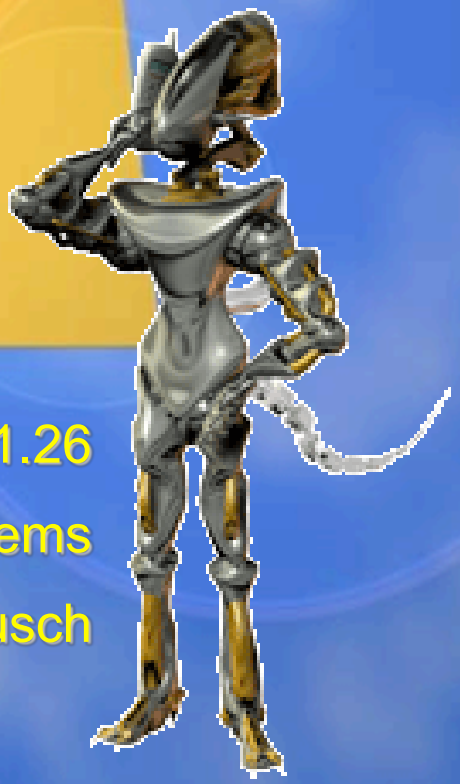
RDF/S Ontology

Produkte sind semantisch annotiert und per RFID / Barcode identifizierbar.

Suche nach relevanten semantischen Diensten für Anfrage zu Produkten.

@ DFKI Intelligent Retail Lab Saarbrücken

Thank you
for your attention !



Uni Saarland Campus, Bldg. D3.2, R +1.26
DFKI Research Group Multiagent Systems
klusch@dfki.de, www.dfki.de/~klusch