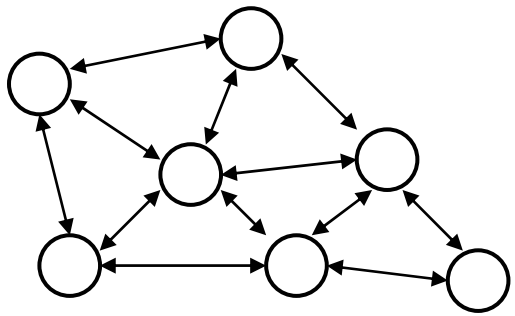
Künstliche Intelligenz

Sebastian Iwanowski
FH Wedel

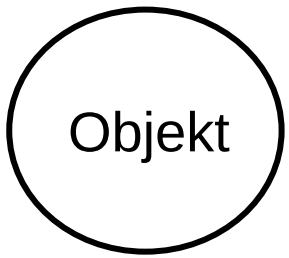
Kap. 6:
Agentenorientierte Systeme
(mit Anwendungsbeispiel Touristeninformationssystem)

Basistechnologie: Agentenorientierte Software

Multiagentensystem:



Softwareagent:



autonom

soziale Kompetenz

reaktiv

proaktiv

Aufgaben für einen Agenten im Multiagentenkontext:

Planung

Kommunikation

Entscheidungsfindung → für individuellen Vorteil → Auktionen, Verhandlungen

Entscheidungsfindung → für Gruppenvorteil → Schwarmintelligenz

wird vertieft im Informatik-Seminar SS 2010

Schwarmintelligenz → Kapitel 7

Touristeninformationssystem als Beispiel für eine Diensteplattform mit weiter gehenden Ansprüchen

Ziele:

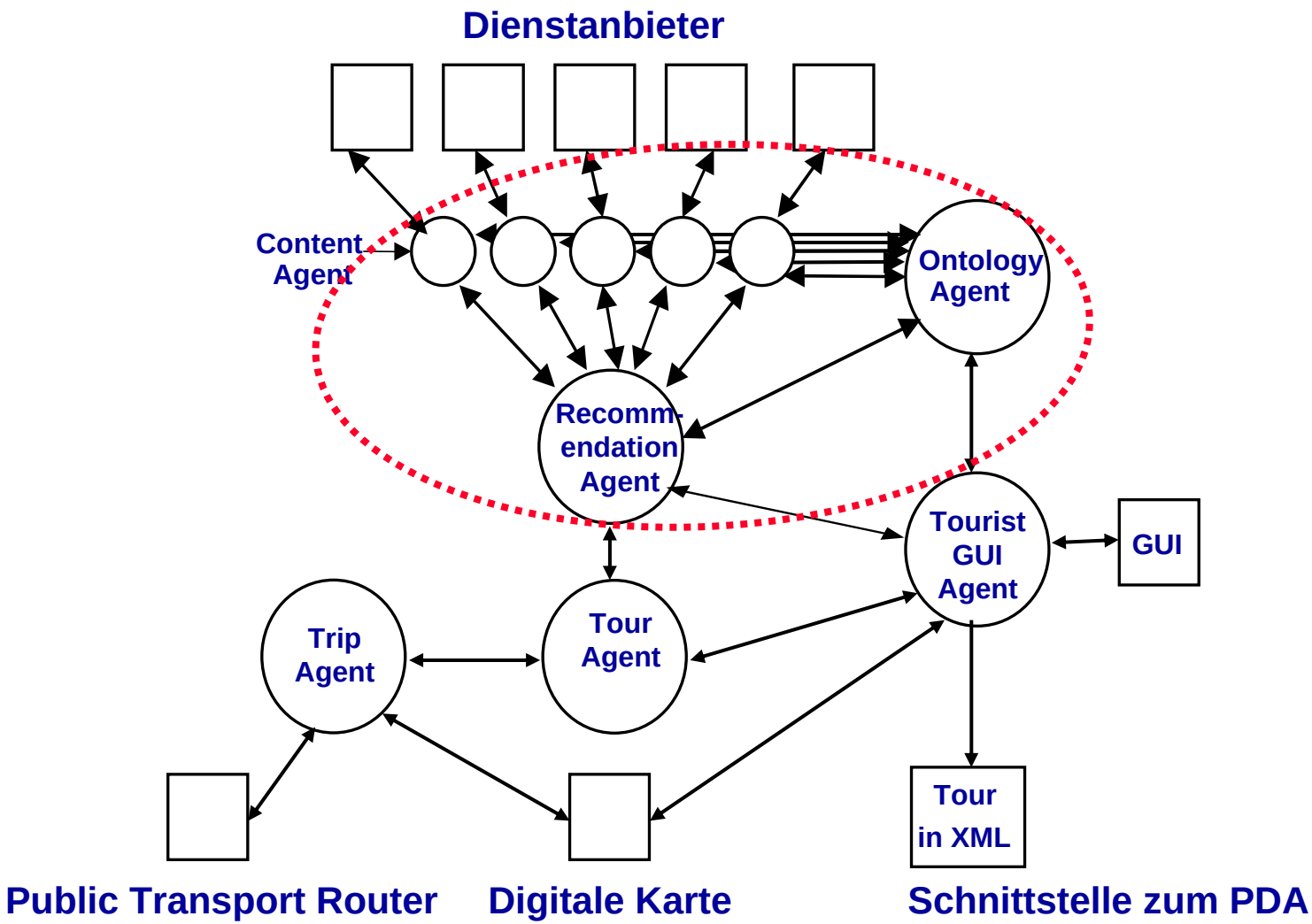
- **Unterstützung plattform- und softwareunabhängiger Dienste**

→ Verteilte Systeme

- **Unterstützung der Generierung neuer Diensttypen**
- **Unterstützung der Generierung von Mehrwertdiensten**

→ Künstliche Intelligenz

Touristeninformationssystem



Ontologien

Was ist eine Ontologie ?

- Begriff „Ontologie“ stammt aus agentenorientierter Programmierung
- Ontologie bezeichnet Begriffswelt
- Ontologien dienen zur Beschreibung von Semantik
- Neueste Entwicklung: Semantic Web

Syntax zur Beschreibung von Ontologien

- Objektorientierte Klassifizierungsbäume (UML)
- KQML (Knowledge Query Manipulation Language)
- XML

Ontologien

Was bezweckt man mit der Definition von Ontologien ?

- **Vereinheitlichung der Begriffswelt zwischen autonomen Dienst- oder Informationsanbietern**
- **Möglichkeit der Einbindung neuer Anbieter in ein bestehendes verteiltes Dienst- oder Informationssystem**

Ontologien: Beispiel einer Tripbeschreibung

KQML:

```
<Trip> ::= (Trip
            :startingPlace <GeoPlace>      :departure <TimeStamp>
            :destinationPlace <GeoPlace>   :arrival <TimeStamp>          :numberOfChanges <number>
            :rides (list {<Ride>}+))

<Ride> ::= <PublicRide> | <PrivateRide>

<PublicRide> ::= (PublicRide
                        :lineName <string>
                        :startingPlace <StopPlace>      :departure <TimeStamp>
                        :destinationPlace <StopPlace>   :arrival <TimeStamp>
                        :directionPlace <StopPlace>     :rideTime <number>)

<PrivateRide> ::= (PrivateRide
                        :transportMedium <string>
                        :startingPlace <Place>          :departure <TimeStamp>
                        :destinationPlace <Place>       :arrival <TimeStamp>
                        :rideLength <number>            :rideTime <number>)
```

Ontologien: Beispiel einer Tripbeschreibung

XML: (informell)

```
⟨Trip⟩ ::= <trip language="⟨String⟩" orderNr="⟨Integer⟩">
        <starting> ⟨GeoPlace⟩ ⟨Date⟩ ⟨Time⟩ </starting>
        <ending> ⟨GeoPlace⟩ ⟨Date⟩ ⟨Time⟩ </ending>
        <rides> {⟨Ride⟩}+ </rides>
    </trip>

⟨Ride⟩ ::= ⟨PublicRide⟩ | ⟨PrivateRide⟩

⟨PublicRide⟩ ::=   <public-ride orderNr="⟨Integer⟩">
            <line> ⟨String⟩ </line>
            <starting>
                <station-name> ⟨String⟩ </station-name>
                ⟨GeoPlace⟩
                ⟨Date⟩
                ⟨Time⟩
            </starting>
            <ending>
                <station-name> ⟨String⟩ </station-name>
                ⟨GeoPlace⟩
                 ⟨Date⟩
                ⟨Time⟩
            </ending>
        </public-ride>

⟨PrivateRide⟩ ::=   <private-ride orderNr="⟨Integer⟩">
            <starting> ⟨GeoPlace⟩ ⟨Date⟩ ⟨Time⟩ </starting>
            <ending> ⟨GeoPlace⟩ Date⟩ ⟨Time⟩ </ending>
        </private-ride>
```

Softwareunabhängige Dienstevermittlung

Problem:

- **Woher weiß ein Client, welche Fragen der Agent versteht ?**
- **Woher weiß ein Agent, welche Antwort ein Client zu einer gegebenen Frage erwartet ?**

Lösung:

1. **Agent gibt seine Ontologie bekannt**
2. **Client stellt seine Frage als unvollständigen Begriff aus der Server-Ontologie**
3. **Agent vervollständigt den Begriff mit Daten, die er generiert, und sendet sie als Antwort zurück.**
    - Client kann in Frage angeben, wie viele Antworten er erwartet und wie detailliert die Antworten sein sollen.

Softwareunabhängige Dienstevermittlung

Beispiel für TripAgent als Agent mit der eben definierten Ontologie (KQML):

Frage des TourAgents:

```
(Trip       :startingPlace (GeoPlace :x 4591300 :y 5822100)
            :departure (TimeStamp :date “29.02.2000” :time “12:00”)
            :destinationPlace (GeoPlace :x 4593100 :y 5825850))
```

Antwort des TripAgents:

```
(Trip       :startingPlace (GeoPlace :x 4591300 :y 5822100)
            :departure (TimeStamp :date “29.02.2000” :time “12:05”)
            :destinationPlace (GeoPlace :x 4593100 :y 5825850)
            :arrival (TimeStamp :date “29.02.2000” :time “12:17”)
```

Detailliertere Frage des TourAgents:

```
(Trip       :startingPlace (GeoPlace :x 4591300 :y 5822100)
            :departure (TimeStamp :date “29.02.2000” :time “12:00”)
            :destinationPlace (GeoPlace :x 4593100 :y 5825850)
            :rides *)
```

Detailliertere Antwort des TripAgents:

```
(Trip
    :startingPlace (GeoPlace :x 4591300 :y 5822100)
    :departure (TimeStamp :date “29.02.2000” :time “12:05”)
    :destinationPlace (GeoPlace :x 4593100 :y 5825850)
    :arrival (TimeStamp :date “29.02.2000” :time “12:17”)
    :numberOfChanges 2
    :rides
        ((PrivateRide
            :transportMedium “footwalk”
            :startingPlace (GeoPlace :x 4591300 :y 5822100)          :departure (TimeStamp :date “29.02.2000” :time “12:05”)
            :destinationPlace (GeoPlace :x 4591207 :y 5822200)       :arrival (TimeStamp :date “29.02.2000” :time “12:07”)
            :rideLength 137                                          :rideTime “00:02”)
        (PublicRide
            :lineName “U9”
            :startingPlace (StopPlace :stopArea “U Turmstraße”)      :departure (TimeStamp :date “29.02.2000” :time “12:08”)
            :destinationPlace (StopPlace :stopArea “U Osloer Straße”) :arrival (TimeStamp :date “29.02.2000” :time “12:16”)
            :directionPlace (StopPlace :stopArea “U Osloer Straße”)  :rideTime “00:08”)
        (PrivateRide
            :transportMedium “footwalk”
            :startingPlace (GeoPlace :x 4593092 y: 5825754)          :departure (TimeStamp :date “29.02.2000” :time “12:16”)
            :destinationPlace (GeoPlace :x 4593100 :y 5825850)       :arrival (TimeStamp :date “29.02.2000” :time “12:17”)
            :rideLength 96                                           :rideTime “00:01”))
```

Einklinken eines neuen Anbieters in ein bestehendes verteiltes Informationssystem

Problem:

- **Wie kann sich ein neuer Anbieter so einbinden, dass er von potentiellen Clients auch gefunden wird ?**
- **Wie kann der Anbieter erreichen, dass ein potentieller Client seine Dienste versteht ?**

Lösung:

1. **Neuer Anbieter meldet sich bei zentraler Stelle an, die von vielen Clients besucht wird.**
2. **Neuer Anbieter benutzt eine im bestehenden Informationssystem bereits verwendete Ontologie**
3. **Neuer Anbieter hängt von ihm benötigte neue Begriffe als Unterbegriffe an bestehende Begriffe an.**

Einklinken eines neuen Anbieters in ein bestehendes verteiltes Informationssystem

Beispiel:

- **Eine Menge von Ontologien für POIs ist dem System bereits bekannt**
- **Ein neuer ShoppingAgent für Einkaufsmöglichkeiten will sich einbinden**
- **Ontologie des ShoppingAgents besteht aus einem neuen Begriff ShoppingPOI mit seinen Unterbegriffen und Attributen**
- **ShoppingAgent beschließt, seinen Begriff als Unterbegriff des bereits bestehenden Begriffs POI einzuklinken**

Einklinken eines neuen Anbieters in ein bestehendes verteiltes Informationssystem

Beispiel:

Bereits bekannte Ontologie:

```
(POI (DictEntry POI)
        :subClasses (<SightseeingPOI> <DiningPOI> <EntertainmentPOI>)
        :name (DictEntry Name) <string>
        :address (DictEntry Address) <AddressPlace>
        :description (DictEntry Description) <DictEntry>
        :place (DictEntry Dummy) ({<GeoPlace>}+))
```

Einklinken eines neuen Anbieters in ein bestehendes verteiltes Informationssystem

Beispiel:

Ontologie des neuen ShoppingAgents:

```
(ShoppingPlace (DictEntry ShoppingPlace)
      :subClasses
         ((DepartmentStore (DictEntry DepartmentStore) :subClasses ())
          (GroceryStore (DictEntry GroceryStore) :subClasses ())
          (FashionShop (DictEntry FashionShop) :subClasses ())
          (SouvenirShop (DictEntry SouvenirShop) :subClasses ())
          (SpecialShop (DictEntry SpecialShop) :subClasses ()))
      :sellingItems (DictEntry SellingItems) ({<DictEntry>}))
```

Einklinken in bereits bekannte Ontologie:

```
(POI (DictEntry POI)
        :subClasses (<SightseeingPOI> <DiningPOI> <EntertainmentPOI> ShoppingPlace)
        :name (DictEntry Name) <string>
        :address (DictEntry Address) <AddressPlace>
        :description (DictEntry Description) <DictEntry>
        :place (DictEntry Dummy) ({<GeoPlace>}+))
```

Anbieten eines Mehrwertdienstes

Problem:

- **Einzelne Anbieter können verschiedene Dienste anbieten, aber keiner kann die Dienste kombinieren**
- **Client möchte mehrere Dienste in Anspruch nehmen, fordert aber, dass die Dienste koordiniert werden**

Lösung:

- **Ein Mehrwertanbieter bietet sich als Koordinator an.**
- **Der Mehrwertanbieter bietet keine originären Dienste an, sondern vermittelt Anfragen an die Teilanbieter.**
- **Der Mehrwertanbieter kennt alle Ontologien der Teilanbieter sowie die Beziehungen zwischen ihnen.**

Anbieten eines Mehrwertdienstes

Beispiel:

- **Der RecommendationAgent bietet die Möglichkeit, nach Restaurants eines bestimmten Typs zu fragen, in denen ein Konzert eines bestimmten Typs stattfindet**
- **Ein RestaurantAgent bietet Ontologie an, in der Restaurants katalogisiert sind**
- **Ein EntertainmentAgent bietet Ontologie an, in der Entertainments katalogisiert sind**
- **Der RecommendationAgent fragt diese beiden Agent getrennt und übermittelt die zusammengesetzte Antwort, wobei er die jeweilige Herkunft nicht verbirgt**

Anbieten eines Mehrwertdienstes

Beispiel:

Ontologie des RestaurantAgents (Auszug):

```
(DiningPlace (DictEntry DiningPlace)
   :subClasses
      ((Bistro (DictEntry Bistro)
         :subClasses ()
         :seatingFacilities (DictEntry SeatingFacilitiesExist) <boolean>
      (Restaurant (DictEntry Restaurant)
         :subClasses ()
         :seats (DictEntry NumberOfSeats) <number>
         :stars (DictEntry CookingHats) <number>))
   :cuisine (DictEntry Cuisine) <DictEntry>
```

Anbieten eines Mehrwertdienstes

Beispiel:

Ontologie des EntertainmentAgents (Auszug):

```
(EntertainmentPOI (DictEntry EntertainmentPOI)
    :subClasses ()
    :events (DictEntry Dummy) ({<CulturalEvent>}+)

<CulturalEvent> ::=
(CulturalEvent (DictEntry CulturalEvent)
    :subClasses (<Theater> <Concert>)
    :description (DictEntry Description) <dictEntry>
    :admissionFee (DictEntry AdmissionFee) <float>
    :startingTime (DictEntry StartingTimeGeneral) <TimeStamp>
    :endingTime (DictEntry EndingTimeGeneral) <TimeStamp>)))

<Concert> ::=
(Concert (DictEntry Concert)
    :subClasses
        ((Classic (DictEntry Classic) :subClasses ())
         (Jazz (DictEntry Jazz) :subClasses ())
         (Folk (DictEntry Folk) :subClasses ())
         (Pop (DictEntry Pop) :subClasses ())
         (Rock (DictEntry Rock) :subClasses ())))
```

Anbieten eines Mehrwertdienstes

Beispiel:

Frage an den RecommendationAgent:

(POI ((Restaurant :cuisine Italian) AND
(EntertainmentPOI :events (Jazz :startingTime (TimeStamp :date „29.02.2000“)))))

Frage des RecommendationAgents an den RestaurantAgent:

(POI (Restaurant :cuisine Italian))

Frage des RecommendationAgent an den EntertainmentAgent:

(POI (EntertainmentPOI :events (Jazz :startingTime (TimeStamp :date „29.02.2000“))

Anbieten eines Mehrwertdienstes

Beispiel:

Antwort des RecommendationAgents nach Erhalt der Teilantworten:

```
((Restaurant    :seats 50 :stars 3 :cuisine (DictEntry Italian) :name “Piazza del Jazz”
                :address (AddressPlace :postalCode 10400 :city “Berlin”
                                       :streetName “Käthe-Kollwitz-Platz” :number 3)
                :description (DictEntry PiazzaJazzEntry)
                :place ((GeoPlace :x 4000 :y 5500))))
AND
(EntertainmentPOI     :seats 50 :name “Piazza del Jazz”
                      :address (AddressPlace :postalCode 10400 :city “Berlin”
                                             :streetName “Käthe-Kollwitz-Platz” :number 3)
                      :description (DictEntry PiazzaJazzEntry)
                      :place ((GeoPlace :x 4000 :y 5500))
                      :events ((Jazz
                               :description (DictEntry AckermanConcertEntry)
                               :startingTime (TimeStamp :date „29.02.2000“ :time „20:00“)
                               :endingTime (TimeStamp :date „29.02.2000“ :time „23:00“)))))
```

Semantic Web

Definition des Erfinders

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.

It is based on the idea of having data on the Web defined and linked such that it can be used for more effective
discovery, automation, integration, and reuse across various applications.

aus Hendler, Berners-Lee, Miller: http://www.w3.org/2002/07/swint

Ontologien im Semantic Web

Main components of an ontology:

- **Classes**: concepts of the domain tasks, usually organized in taxonomies and contain attributes
- **Relations**: express relationship between concepts in the domain
- **Functions**: Special case of relations in which the n-element of the relationship is unique for the n-1 preceding elements
- **Axioms**: model sentences that are always true
- **Instances**: represent specific elements of the concepts, in contrast with general concepts or classes

aus Alonso / Bussler: EDBT-WebService-Tutorial,
http://www.inf.ethz.ch/personal/alonso/teaching.html

Semantic Web

RDF: Resource Description Framework

Beschreibungsmodell

Eine Aussage besteht aus

- Subjekt
  - sagt aus, worüber die Aussage gemacht wird
  - ist meist eine Webseite, eine Person, ein Buch…
- Prädikat
  - ist die Eigenschaft des Subjektes
  - Autor, Erstelldatum, Sprache
- Objekt
  - ist der Wert einer Eigenschaft

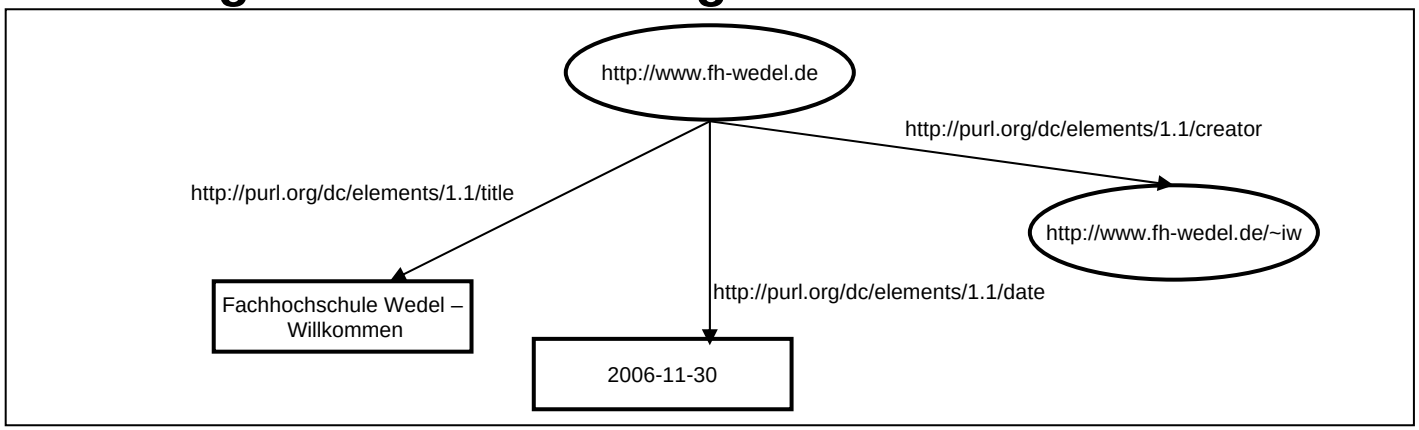
http://www.fh-wedel.de hat einen **Titel** mit dem Wert **Fachhochschule Wedel - Willkommen**

aus Imken Chitralla: RDF, Vortrag 8 im SOA-Seminar WS 2006/2007

Semantic Web

RDF: Resource Description Framework

XML-Darstellung des Beschreibungsmodells




```
<?xml version="1.0"?>
<rdf:RDF  xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://www.fh-wedel.de">
    <dc:title>Fachhochschule Wedel - Willkommen</dc:title>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.fh-wedel.de">
    <dc:date>2006-11-30</dc:date>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.fh-wedel.de">
    <dc:creator rdf:resource="http://www.fh-wedel.de/~hs"/>
  </rdf:Description>
</rdf:RDF>
```

aus Imken Chitralla: RDF, Vortrag 8 im SOA-Seminar WS 2006/2007

Semantic Web

RDFS: Resource Description Framework Schema

Klassen und Eigenschaften

- Ressource ist Instanz (einer oder mehrerer Klassen)
- Beschreibung durch
  - RDF Schema resources
    - rdfs:Class
    - rdfs:Resource
  - RDF Schema properties
    - rdf:type
    - rdfs:subClassOf
    - rdfs: range
    - rdfs: subpropertyOf

aus Imken Chitralla: RDF, Vortrag 8 im SOA-Seminar WS 2006/2007

Semantic Web

RDFS: Resource Description Framework Schema

Beispiel:
(Teil 1)

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
   xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
   xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
   xml:base="http://example.org/schemas/vehicles">

<rdfs:Class rdf:ID="MotorVehicle"/>
<rdfs:Class rdf:ID="PassengerVehicle">
   <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Truck">
   <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Van">
   <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>

<rdfs:Class rdf:ID="MiniVan">
   <rdfs:subClassOf rdf:resource="#Van"/>
   <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdfs:Class>
```

aus http://www.w3.org/TR/rdf-primer/

Semantic Web

RDFS: Resource Description Framework Schema

Beispiel:
(Teil 2)

```
<rdfs:Class rdf:ID="Person"/>

<rdfs:Datatype rdf:about="&xsd;integer"/>

<rdf:Property rdf:ID="registeredTo">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property rdf:ID="rearSeatLegRoom">
  <rdfs:domain rdf:resource="#PassengerVehicle"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>

<rdf:Property rdf:ID="driver">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
</rdf:Property>

<rdf:Property rdf:ID="primaryDriver">
  <rdfs:subPropertyOf rdf:resource="#driver"/>
</rdf:Property>

</rdf:RDF>
```

aus http://www.w3.org/TR/rdf-primer/

Semantic Web

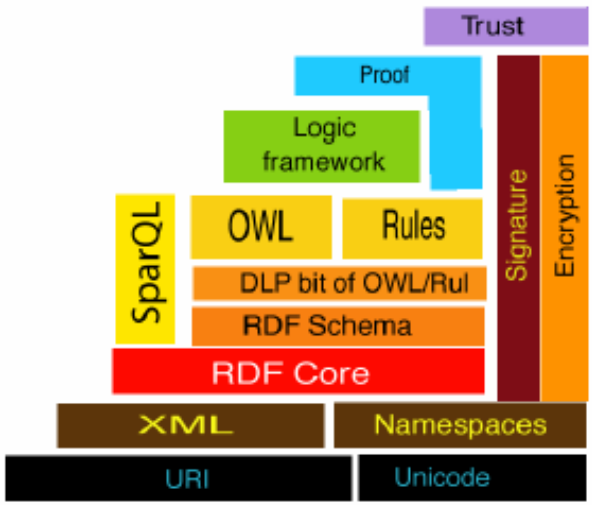
OWL: Web Ontology Language

Ziele:

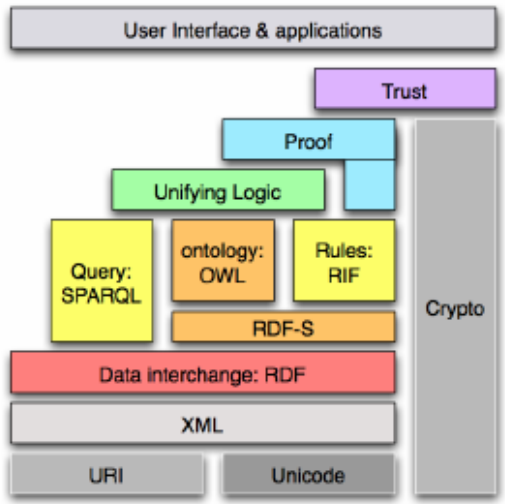
- Entscheidung der Klassenzugehörigkeit
- Äquivalenz von Klassenbeschreibungen
- Ausschluss von Klassenzugehörigkeiten
- Hinreichende Kriterien für Klassenzugehörigkeit

Semantic Web

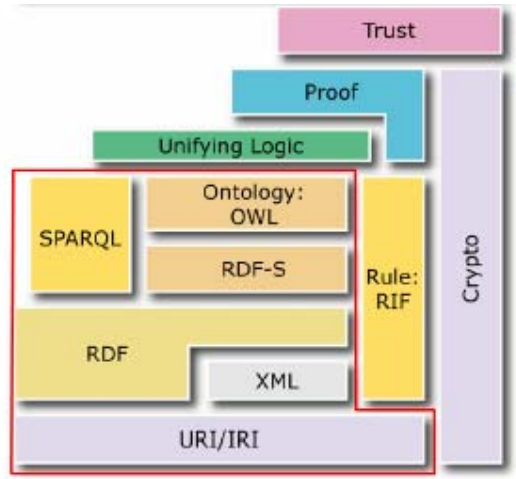
Semantic Web Stack (Berners-Lee)



(a) 2005



(b) 2006



(c) 2007

Semantic Web Services

Web Services

- ermöglichen automatischen Aufruf von Services ohne Notwendigkeit menschlicher Transformationsarbeit
- dokumentieren die Bedeutung des Aufrufs sowie die Interpretation der Antwort nur in menschenlesbarer (nicht maschinenlesbarer) Form

Semantic Web Standards

- kümmern sich nicht um maschinenlesbare Aufrufmöglichkeiten
- standardisieren die Bedeutung von Beschreibungen in maschinenlesbarer Form

Semantic Web Services

- versuchen die Vorteile von Web Services und Semantic Web Services zu vereinen

Semantic Web Services

- Maschinenlesbare Ontologien
- Bisher nur im prototypischen Stadium
- Standards: WSMO (Web Service Modelling Ontology)
  WSMX (Web Service Modelling Execution Environment)

Details in Masterarbeit Max Herold 2008:
http://www.fh-wedel.de/fileadmin/mitarbeiter/iw/Abschlussarbeiten/MasterarbeitHerold.pdf