



**Forschungszentrum
Informatik**



**Universität
Karlsruhe (TH)**



Information Process Engineering

Ontologien für die Informationsintegration und das Semantic Web

Andreas Schmidt

WS 2011/2012

- Hauptherausforderung bei der Informationsintegration ist die Überwindung der semantischen Heterogenität
- Dabei steht die prinzipielle Machbarkeit außer Frage.
 - vgl. die letzten beiden Vorlesungen
- Aber wie bekommt semantische Integration auch effizient (i.S.d. menschlichen Aufwandes), wartbar und skalierbar in den Griff?
 - kann nur durch einen höheren Grad der Automatisierung erreicht werden
 - und durch Spezifikationen auf höherer Abstraktionsebene

Semantische Technologien...



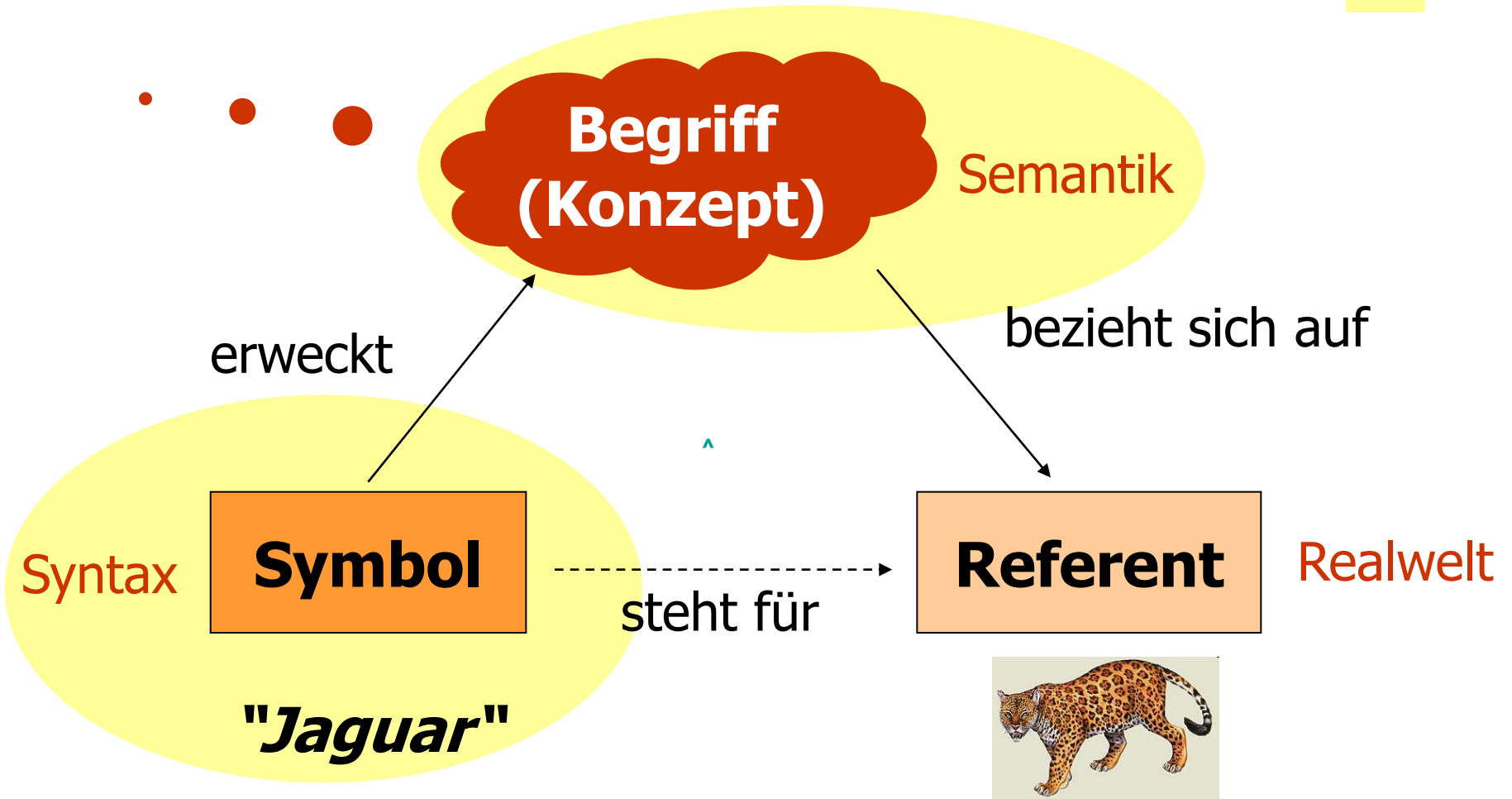
- ... sind ein Sammelbegriff für aktuelle technische Ansätze, um dies zu realisieren.
- Überwiegend bedienen sie sich Ontologien als Instrument zur expliziten und maschinenverarbeitbaren Repräsentation von Semantik.
- Prominenteste Vision dieser Technologieansätze ist das Semantic Web.

- Was sind Ontologien?
- Ontologiesprachen
 - speziell OWL (OWL-Lite, OWL-DL, OWL-Full)
- Einsatzmöglichkeiten von Ontologien
 - Verbesserung der Suche
 - Ontologiebasierte Informationsintegration
 - „Intelligente“ Unterstützung
- Vision des Semantic Web



**Grundidee:
Was sind Ontologien?**

Bedeutungsdreieck



Syntax vs. Semantik

■ Syntaxebene

- erfordert nur geringe Vereinbarungen („Syntaxregeln“)
- kann nicht mit mehrdeutigen Symbolen (Homonymie)
- und nur schwer mit mehrfachen Symbolen (Synonymie) für ein und dasselbe Realweltobjekt umgehen
 - wie sie typischerweise in der realen Welt vorkommen...
- oft Annahme einer impliziten Semantik

■ Semantikebene

- hoher Grad an Vereinbarungen („gemeinsames Modell“)

■ Derzeit real:

- Unterstellen einer impliziten Semantik

■ Ziel: explizite Semantik

Arten von Semantik



Shared human
consensus.

Pump : "a device for
moving a gas or liquid
from one place or
container to another"

Text descriptions.



(pump has
(superclasses (...))

Semantics hardwired;
used at runtime.



Semantics processed
and used at runtime.

Implicit

Informal
(explicit)

Formal
(for humans)

Formal
(for machines)

[Uschold (2002), "Where are the Semantics in the Semantic Web?"]

Was ist eine Ontologie?



Ontologien in der Informatik

- Eine **Ontologie** [in der Informatik] ist eine
 - **explizite Spezifikation** einer **Konzeptualisierung**
[explicit specification of a conceptualization] (Gruber 1993)
 - ein **gemeinsames** Verständnis einer **bestimmten** Domäne
[shared understanding of a domain of interest]
(Uschold & Grüninger 1996)

Was sind Ontologien?

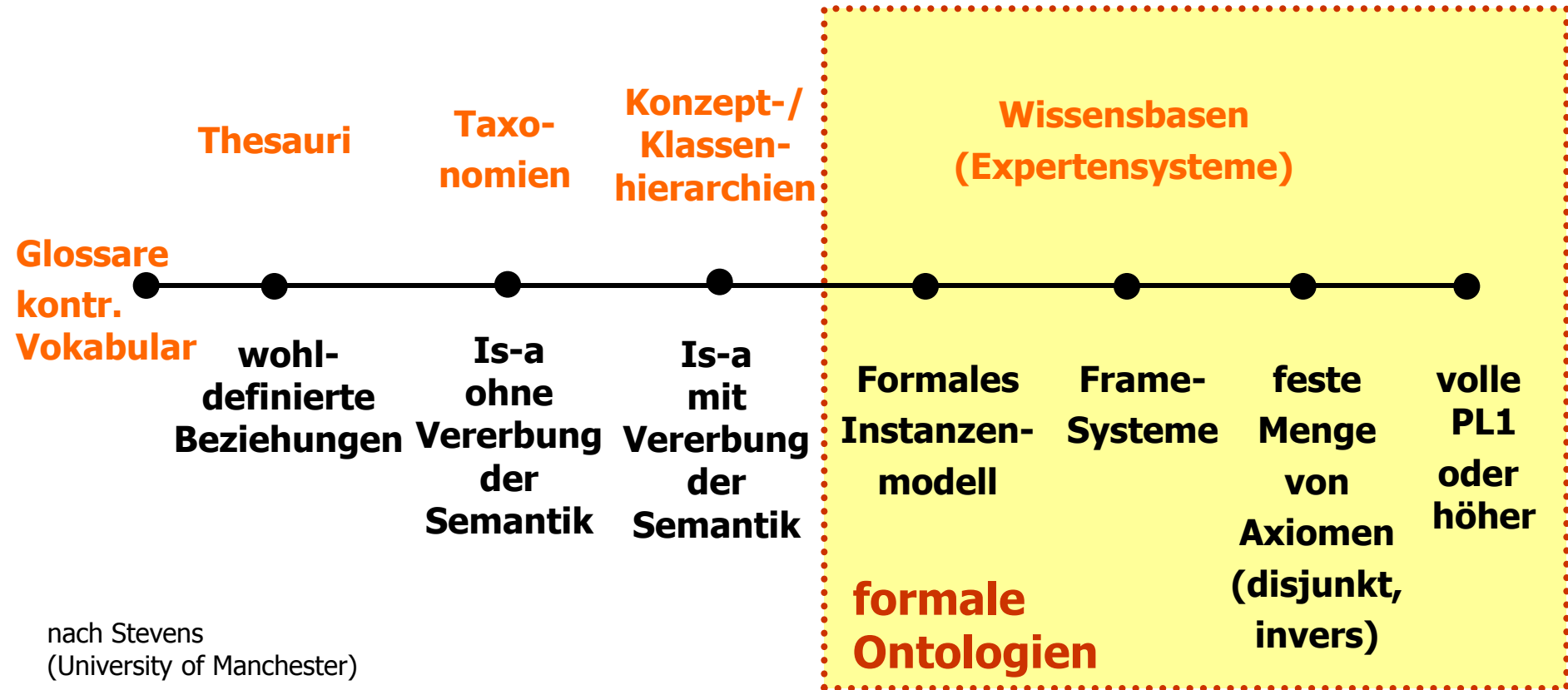
- ... die Wikipedia?
- ... ein ER-Diagramm oder ein UML-Modell?
- ... ein Datenbankschema?
- ... ein Thesaurus?
- ... eine [gemeinsame] Ordnerstruktur?
- ... eine Folksonomy?

- Gibt es persönliche Ontologien?

Ontologiekontinuum

Ausdrucksmächtigkeit

Berechnungskomplexität



nach Stevens
(University of Manchester)

Ontologien im engeren Sinne



- Oft: **Ontologie im engeren Sinne** ist eine
 - gemeinsame
 - **formale** Spezifikation
 - einer bestimmten Domäne
- Ziel: maschinenverarbeitbare Semantik
- meistens logikbasiert, aber nicht zwangsläufig!

Ontologiesprachen



Anforderungen an eine Ontologiesprache

- wohldefinierte Syntax
- formale Semantik
- adäquate Ausdrucksmittel
 - Ausdrucksmittel für eine konzeptuelle Modellierung
 - Klassen, Beziehungen, Vererbung etc.
 - hinreichende Ausdrucksmächtigkeit
- effiziente Verarbeitung
 - Durchsetzung der formalen Semantik
 - Zugriffsmethoden
- Unterstützung offener (Web-)Umgebungen
 - verteilte Ontologien
 - Autonomie und ihre Konsequenzen

weiche, aber
entscheidende Kriterien

Beispiele für qualifizierende Ontologiesprachen

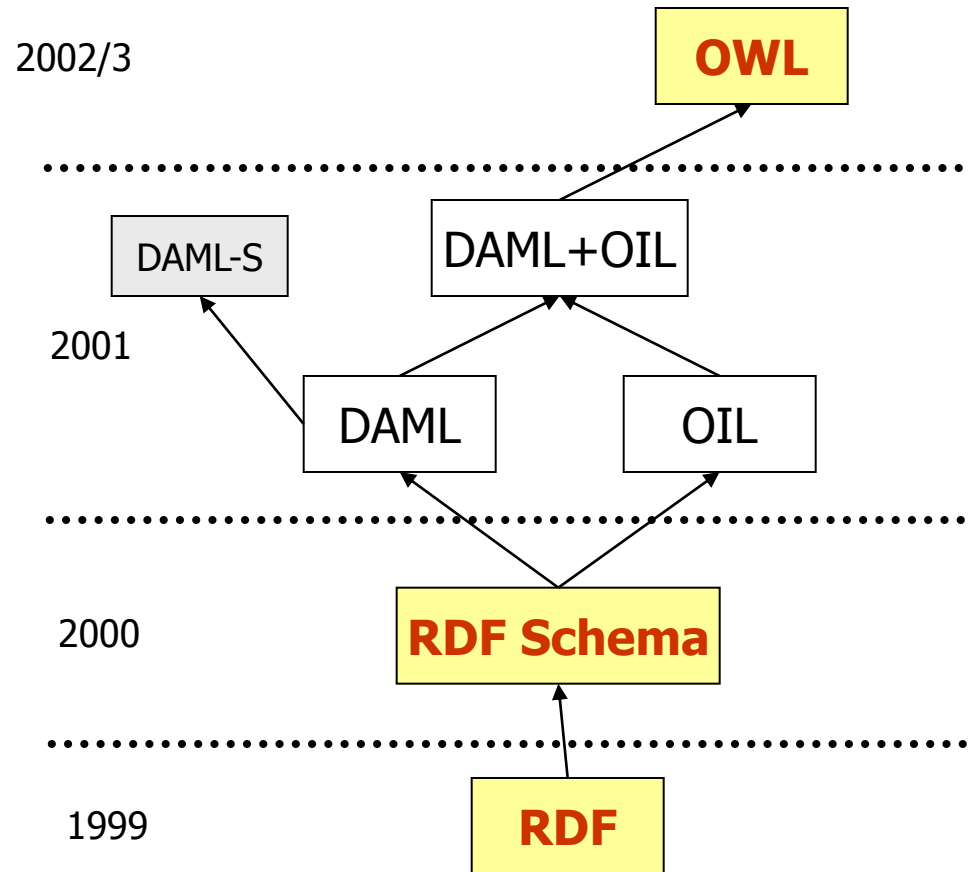
- UML mit OCL
 - sehr ausdrucksmächtig, aber effizient?
 - offene Umgebungen?

- Topic Maps

- F-Logic

- **RDFS und OWL als W3C-Standards**

Ontologiesprachen für das Web



OWL: Drei Ausprägungen



- OWL-Lite
- OWL-DL
- OWL-Full
- OWL-Lite und OWL-DL sind noch entscheidbar; OWL-Full ist es nicht.

■ Klassen (`owl:Class`)

- Subklassen (Mehrfachvererbung) (`rdfs:subclassOf`)
- Äquivalenz (`equivalentClass`)
- Schnitt (`intersectionOf`)

■ Instanzen

- Äquivalenz

■ Warum brauchen wir Aussagen zur Äquivalenz?

- keine „Unique Names Assumption“
 - zwei verschiedene Namen heißt nicht automatisch, dass Dinge verschieden sind
- für Integrationsszenarien wichtig!

OWL-Lite: Properties

- Eigenschaften/Properties
 - Definitionen sind unabhängig von Klassen!
 - zwei Typen: Attribute (**DatatypeProperty**) und komplexe Eigenschaften (**ObjectProperty**)
- Definitionsbereich (**rdfs:domain**) und Wertebereich (**rdfs:range**)
- spezielle Eigenschaftstypen:
 - funktional (**FunctionalProperty**)
 - eindeutig (**InverseFunctional**)
 - symmetrisch (**SymmetricProperty**)
 - transitiv (**TransitiveProperty**)

OWL-Lite: Properties (2)

- Vererbung von Properties (**rdfs:subpropertyOf**)
 - Dies ist ein wesentlicher Unterschied zu klassischen Modellierungssprachen wie ER oder UML
 - erlauben die Repräsentation von Beziehungen auf unterschiedlichen Abstraktionsebenen

- Property Restrictions: in Klassen lokale Einschränkungen von Wertebereichen, Kardinalitäten (nur 0 oder 1)

- Annotationseigenschaften (**AnnotationProperty**)
 - erlauben eine (sehr beschränkte) Metamodellierung
 - Annotation von Properties und Klassen
 - keine Vererbung o.ä.

Konsistenzsicherung vs. logische Schlussfolgerungen

- Definitions- und Wertebereiche von Properties dienen nicht der Konsistenzsicherung (wie UML oder ER), sondern der logischen Schlussfolgerung
 - **Beispiel:** `hat-autor`, `domain: Publikation`, `range: Person`
 - *Interpretation Konsistenzsicherung:*
Property `hat-autor` darf nur auf `Publikation` verwendet und als Werte Instanzen von `Person` haben
 - *Interpretation logische Schlussfolgerung:*
Wenn eine Ausprägung einer Eigenschaft zwischen einer `Publikation` und einer `Person` vorkommt, dann ist sie eine Instanz der `hat-autor`-Beziehung

Konsistenzsicherung vs. logische Schlussfolgerungen (2)

- Ähnliches gilt auch bei Klassendefinitionen mit Hilfe von Property Restrictions
 - Beispiel
 - Klasse `Wissenschaftler` subclassOf `Person`
 - Klasse `WissenschaftlichePublikation`
 - `ist-autor` allValuesFrom `Wissenschaftler`
 - *Interpretation Konsistenzsicherung:*
Für `WissenschaftlichePublikation` dürfen nur `Wissenschaftler` als Autoren angegeben werden.
 - *Interpretation logische Schlussfolgerung:*
Wenn eine `Publikation` einen `Wissenschaftler` als Autor hat, dann ist sie eine `WissenschaftlichePublikation`

Konsistenzsicherung vs. logische Schlussfolgerungen (3)

- Weiteres Beispiel: Eigenschaften von Properties:
 - hat-verlag als FunctionalProperty
- *Interpretation Konsistenzsicherung:*
Für eine Instanz (z.B. Publikation) darf es nur einen Verlag geben.
- *Interpretation logische Schlussfolgerungen:*
 $P \text{ hat-verlag } A, P \text{ hat-verlag } B \Rightarrow A = B$

```
<owl:Ontology rdf:about="...">
  <owl:Class rdf:ID="Publikation" />
  <owl:Class rdf:ID="Person" />
  <owl:Class rdf:ID="Wissenschaftler">
    <rdfs:subClassOf rdf:resource="#Person" />
  </owl:Class>
  <owl:Class rdf:ID="WissenschaftlichePublikation">
    <rdfs:subClassOf rdf:resource="#Publikation" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hat-autor" />
      <owl:allValuesFrom rdf:resource="#Wissenschaftler" />
    </owl:Restriction>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="hat-autor">
    <rdfs:domain rdf:resource="#Publikation" />
  </owl:ObjectProperty>
  <owl:FunctionalProperty rdf:ID="hat-verlag" />
</owl:Ontology>
```

- basiert auf formalen Beschreibungslogiken, die einen Kompromiss aus größtmöglicher Ausdrucksmächtigkeit bei gleichzeitig noch (einigermaßen) effizienten Algorithmen darstellt
- Zusätzliche Klassenkonstruktoren
 - `unionOf`, `complementOf`
 - Aufzählungen `oneOf`
 - Disjunktheit (`disjointWith`)
- Property Restrictions
 - allgemeine Kardinalitäten
 - `hasValue`

■ Wesentlicher Unterschied:

- keine Trennung zwischen Klassen, Eigenschaften und Instanzen
- dadurch Metamodellierung möglich:
 - eine Klasse ist Instanz einer Klasse, die wiederum Instanz einer anderen Klasse ist
 - Beispiel:
Buchexemplar instanceOf Buch [als Werk] instanceOf Publikation
- manchmal für die Modellierung ganz sinnvoll, weil Realweltzusammenhänge präziser ausgedrückt werden können

■ Allerdings: Sprache ist nicht mehr entscheidbar

Anwendung von OWL auf RDF-Daten

RDF-Daten:

```
<http://datenbanken-und-xml.de> <hat-autor> <http://andreas.schmidt.name>  
<http://andreas.schmidt.name> <rdf:type> <Wissenschaftler>  
<Wissenschaftler> <rdf:type> <owl:Class>  
<Publikation> <rdf:type> <owl:Class>
```

SPARQL-Anfrage direkt:

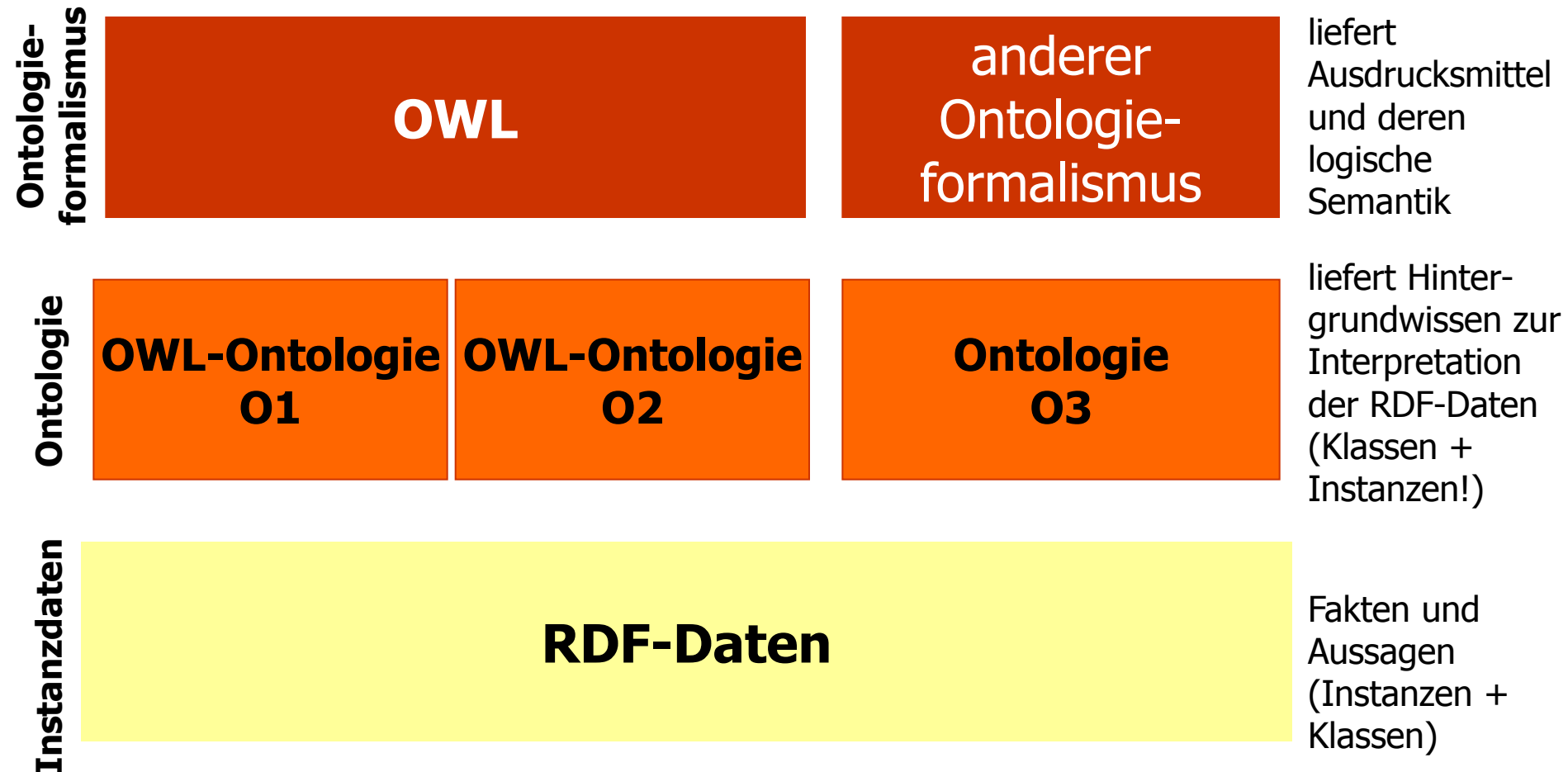
```
SELECT ?p  
WHERE (?p rdf:type Publikation)  
=> liefert nichts zurück
```

SPARQL mit der obigen OWL-Ontologie und Reasoning:

```
=> liefert <http://datenbanken-und-xml.de> zurück
```

Wiederum: Schemaformalismen verändern das Datenmodell!

Schichtung



travel Protégé 3.1 (file:\C:\protege-owl\owl\travel.pprj, OWL Files (.owl or .rdf))

File Edit Project OWL Code Window Tools Help

owl:Classes Properties Forms Individuals Metadata

SUBCLASS RELATIONSHIP CLASS EDITOR

For Project: travel For Class: FamilyDestination (instance of owl:Class)

Asserted Hierarchy

- owl:Thing
 - Accommodation
 - BedAndBreakfast
 - BudgetAccommodation
 - Campground
 - Hotel
 - AccommodationRating
 - Activity
 - Contact
 - Destination
 - BackpackersDestination
 - Beach
 - BudgetHotelDestination
 - FamilyDestination**
 - QuietDestination
 - RetireeDestination
 - RuralArea
 - UrbanArea
 - City
 - Town

CLASS EDITOR

Name: FamilyDestination

SameAs DifferentFrom

rdfs:comment

A destination with at least one accommodation and at least 2 activities.

Annotations

Property	Value	Lang
rdfs:comment	A destination with at le...	

Asserted Conditions

Asserted Inferred

Destination — NECESSARY & SUFFICIENT

- hasAccommodation ≥ 1
- hasActivity ≥ 2

— NECESSARY

Properties

- hasAccommodation (multiple Accommodation)
 - 1
- hasActivity (multiple Activity)
 - 2
- hasPart (multiple Destination)

Disjoints

- RetireeDestination

SWOOP v2.2b
File View Bookmarks Resource Holder

Address: <http://sweet.jpl.nasa.gov/ontology/space.owl#DistanceCategory>

Collapse New Remove

space.owl
process.owl
units.owl
property.owl
earthrealm.owl
material_thing.owl
numerics.owl
human_activities.owl
time.owl

Add Class Add Property Remove Rename

Show Imports QNames RDFS

Class Tree Property Tree Alphabetical List

- numerics:Variable
- phenomena:SeaFloor
- substance:SeaIce
- human_activities:Sense
 - human_activities:Hearing
 - human_activities:Vision
- space:SpatialDistribution
 - space:Broken
 - space:Overcast
 - space:Patch
- property:SpatialExtent
- space:SpatialExtentCategory*
 - space:DepthCategory
 - space:Deep
 - space:Shallow
 - space:DistanceCategory*
 - space:Vicinity
 - space:SizeCategory*
 - space:Big
 - space:Small
- space:SpatialReferenceSystem
- space:SpatialScale
 - space:Mesoscale
 - space:SynopticScale

Concise Format Abstract Syntax N3 view RDF/WML Turtle Format

OWL-Class: [space:DistanceCategory](#)

Intersection of: (Add)
[space:SpatialExtentCategory](#) (Delete)
 (≥ 1 [space:hasDirection](#)) (Delete)

Disjoint with: (Add)
[space:SizeCategory](#) (Undo)

Subclass of: (Add)
[space:SpatialExtentCategory](#) (Undo)
 (Yspace:hasAssociatedQuantity , property:Distance) (Delete)

Superclass of: (Add)
[space:Vicinity](#) (Delete)

Annotations: (Add)

Union of: (Add)

One of: (Add)

Equivalent to: (Add)

Domain of: (Add)

Range of: (Add)

Show Inherited Show Changes Editable

Changes Checkpoints

Scope: Entity Ontology Workspace

Ontology Change Logging: **enabled** (Press **Ctrl-L** to disable)

Uncommitted Changes:

[2004-09-28 21:57:48]
 REMOVE SUPER CLASS ([space:SpatialExtentCategory](#)) (Undo)

[2004-09-28 21:58:16]
 ADD DISJOINT CLASSES (Undo)
 ([space:SizeCategory](#) [space:DistanceCategory](#))

Committed Changes:

[2004-09-28 21:56:58]
 ADD INTERSECTION ELEMENT ([space:SpatialExtentCategory](#))

[2004-09-28 21:57:33]
 ADD INTERSECTION ELEMENT (≥ 1 [space:hasDirection](#))

Apply Changes Undo Changes

Lookup

Zwischenfazit: Ontologiesprachen

- Grundphilosophie von Ontologiesprachen wie OWL ist die maschinenverarbeitbare Kodierung von Hintergrundwissen, das nicht explizit gegebene Zusammenhänge logisch erschließbar macht
 - wesentlicher Unterschied zu UML/ER und Schemasprachen, wo viele Modellkonstrukte zur Konsistenzsicherung dienen
- Können also unvollständig angegebene Daten „vervollständigen“
- Deshalb können Ontologiesprachen auch nur sinnvoll mit einem sog. Reasoner eingesetzt werden, der die logischen Schlussfolgerungen ausführt.

Zwischenfazit: Ontologiesprachen (2)

- Typische Schlussfolgerungen sind:
 - auf Instanzebene (sog. A-Box-Reasoning)
 - Klassifikation von Instanzen
(Berechnung der instanceOf-Beziehung in beide Richtungen)
 - Typisierung von Property-Instanzen
 - Äquivalenz von Instanzen
 - auf Klassenebene (sog. T-Box-Reasoning)
 - Subklassenbeziehung (subClassOf-Berechnung)
 - Äquivalenz oder Disjunktheit von Klassen
 - Konsistenz von Klassendefinitionen

Zwischenfazit: Ontologiesprachen (3)

- Grundlage ist meistens eine offene Welt (z.B. OWL)
- Geschlossene Welt (Closed World Assumption, CWA)
 - wahr ist, was explizit angegeben wurde
 - alles andere ist falsch („negation as failure“)
 - Semantik von SQL, XQuery, ...
- Offene Welt (Open World Assumption)
 - wahr ist, was nicht explizit ausgeschlossen wurde
 - keine Unique Names Assumption

Einsatzmöglichkeiten für Ontologien



- Verbesserung der Suche
 - Informationsintegration
 - „Intelligente“ Unterstützung
 - (Dienstorientierte Architekturen
 - Dienstfindung
 - Dienstkopplung
 - Traceability)
- => kommt übernächstes Mal

Verbesserung der Suche

- Wesentlich: Trennung von Syntax (die verwendeten Wörter) und der allgemeinen Konzepte
 - erlaubt mehrsprachige Suche
 - erlaubt die Berücksichtigung von Synonymen=> automatisches „Query Rewriting“

- Beziehungen zwischen Konzepten
 - erlauben die Verfeinerung von Anfragen
 - erlauben die Generierung von Navigationsmöglichkeiten=> Ausnutzen der typisierten Graphenstruktur

Verbesserung der Suche (2)

- Ideal: automatische Indexierung der Dokumente
 - würde aber automatische Übersetzung von der Syntax in die Semantikebene bedeuten
 - insbesondere Disambiguierung
 - deshalb: meist nur eingeschränkt tauglich
 - Verfahren nutzen Hintergrundwissen
- Oft: Manuelle Annotationen erforderlich
 - semi-automatische Verfahren derzeit am vielversprechendsten

Ontologiebasierte Informationsintegration: (Naive) Grundidee

- Erstellung einer globalen Ontologie
- Einordnung der Datenquellen in die Ontologie
 - Schemakonstrukte als abgeleitete oder äquivalente Konzepte/Properties
 - Instanzen als Extensionen von Konzepten
- Anfragen als Reasoning-Problem
 - Anfrage ist eine Konzeptdefinition
 - Reasoner berechnet, welche Instanzen Instanz dieser Konzeptdefinition sind

Ontologiebasierte Informationsintegration: Vergleich mit „klassischen“ Ansätzen

■ Vorteil:

- derselbe Mechanismus wird für die Spezifikation der Schemaabbildung und die Anfrageübersetzung verwendet

■ Nachteil:

- skaliert in dieser Form überhaupt nicht
- derzeitige Reasoningansätze erfordern eine Materialisierung
 - bis auf wenige Ausnahmen (Datalog-basierte Systeme wie KAON2) erfordern sie sogar alle Instanzen im Hauptspeicher!

Ontologiebasierte Informationsintegration: abgewandelte Anfragebearbeitung

- Beschränkung auf den T-Box-Teil
(also Klassen- und Propertydefinitionen)
- Nutzung der Ontologie für
 - die Quellenauswahl
 - Konsistenz von Quellenkonzept und Anfragekonzept
 - die Anfrageübersetzung
 - Reformulieren der Anfrage, so dass nur verstandene Konzepte
 - die Schema- und Instanzintegration
 - Nutzung von Hintergrundwissen
- Integrationsproblem wird nicht komplett im Ontologieformalismus gelöst!
- „Ontology Mapping“ und „Ontology Merging“ als altbekannte Integrationsdisziplinen

- Ontologiebasierte Verfahren für die Informationsintegration versprechen mehr als sie zu halten im Stande sind
 - insbesondere Probleme der Skalierbarkeit
- Aber: gezielt und eingeschränkt eingesetzt sind sie elegante (da auf deskriptiven Spezifikationen aufbauend) Bausteine für typische Integrationsprobleme, vor allem wenn
 - viel Domänenwissen benötigt wird
 - mit unvollständigen Informationen umgegangen wird
- Schritt auf dem Weg zu mehr modellgetriebenen Integrationsansätzen

Vision des Semantic Web



Semantic Web: Die Vision (1)



“The Semantic Web is a vision: the idea of having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for **automation, integration** and **reuse** of data across various applications. ” [W3C 2001]

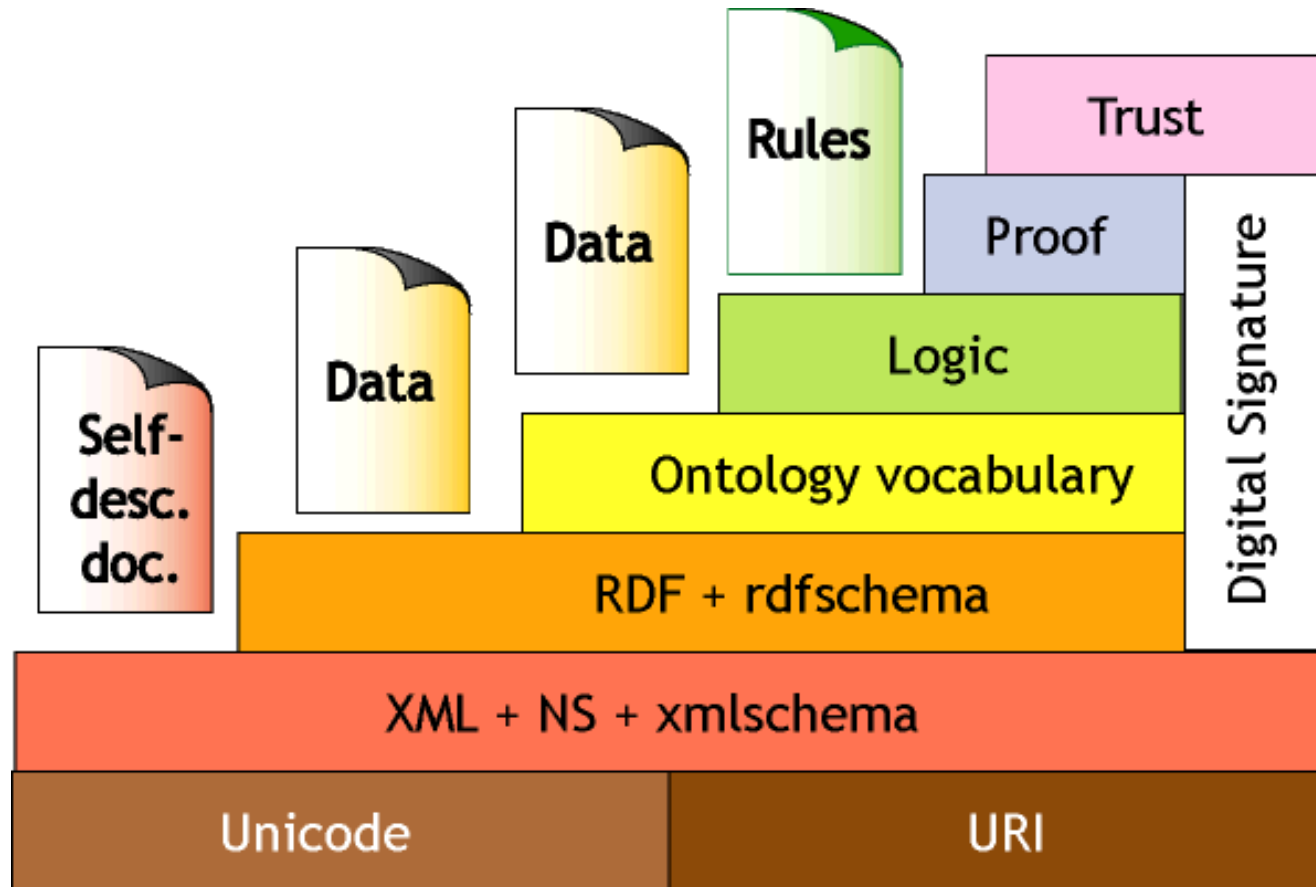
“The Semantic Web is an extension of the current Web in which information is given well-defined meaning, **better enabling computers and people to work in cooperation.**” [Berners-Lee et al 2001]

Semantic Web: Die Vision (2)

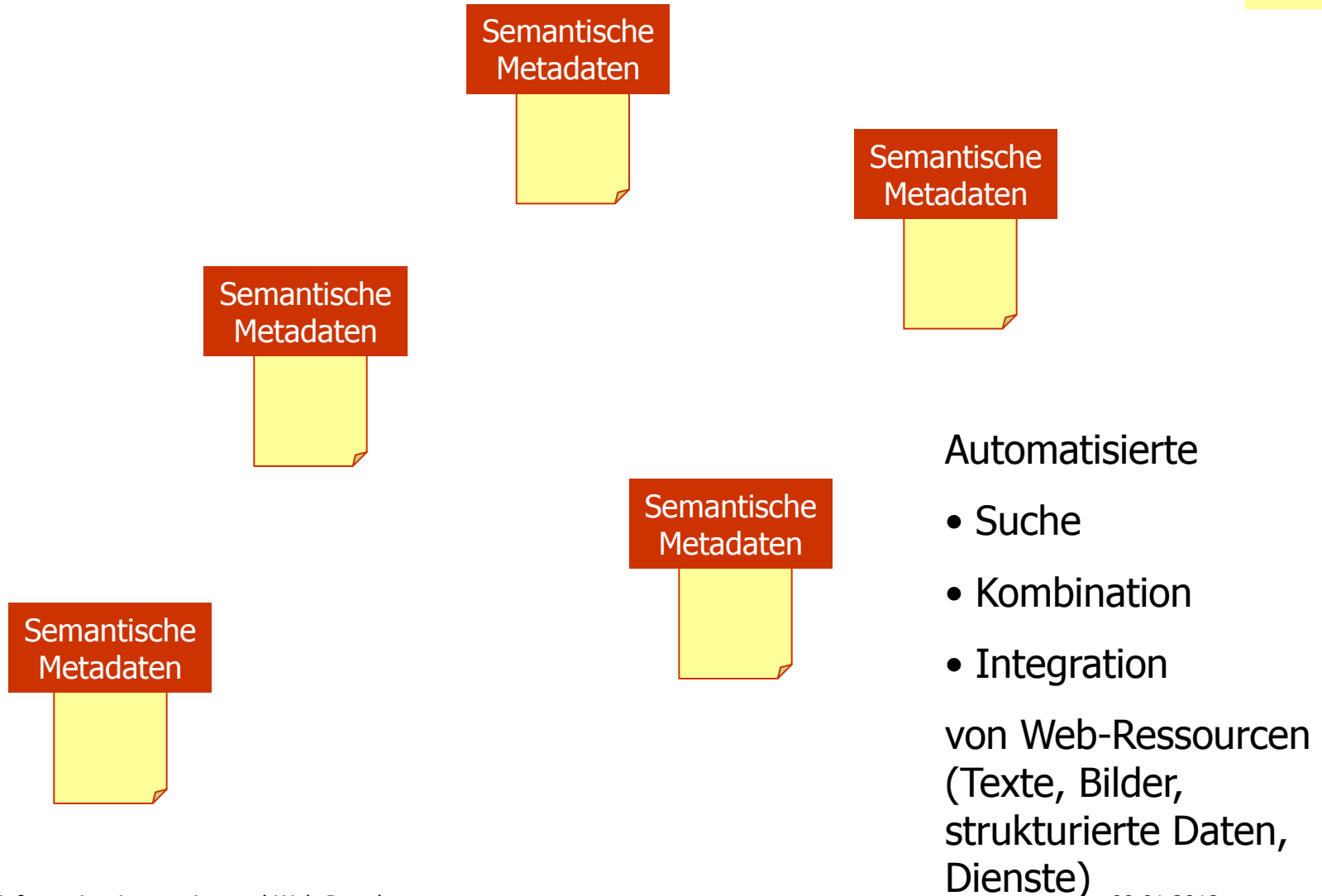


- Semantic Web ersetzt nicht, sondern ergänzt das Web,
- um Inhalte auf dem Web maschinenverarbeitbar zu machen,
- indem die Semantik der Ressourcen und ihrer Beziehungen auf dem Web explizit gemacht wird.

Das Semantic Web im W3C



Quelle: Berners-Lee (1999)



Warum ist das Semantic Web interessant?



- Mächtige Langfrist-Vision
- Aus technischer Sicht liefert das (Semantic) Web natürlich wichtige Anforderungen
 - Skalierbarkeit („web-scale“)
 - Autonomie
 - dezentral
 - niemals *eine* einheitliche Ontologie
 - unvollkommen

Semantic Web und Web 2.0

- Semantic Web wird in der allgemeinen Wahrnehmung eher als „schwergewichtiger“ Ansatz gesehen
 - Grund hierfür ist sicherlich die starke logische Fundierung
- Web 2.0 dagegen gilt als leichtgewichtig
 - sehr wenig formale Ansätze („tagging“)
- Konstruktion eines Gegensatzes zwischen beiden geht am Problem vorbei
 - Web 2.0 erreicht nur wenig Maschinenverarbeitbarkeit
 - s. übernächstes Mal
 - Semantic Web hat bislang die Nutzer vernachlässigt
 - wichtige Impulse und Gegengewicht zur Formalisierungstendenz in der Semantic-Web-Forschung

Fazit



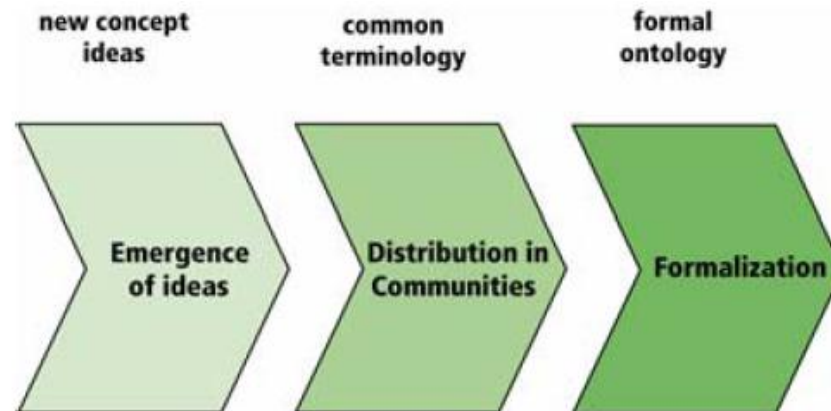
Fazit: Ontologien

- Ontologien sind derzeit **das** Mode-Thema im Bereich der Informationsverarbeitung
 - an sich bieten sie in vielen Anwendungsbereichen wenig, was noch nicht da war
 - aber:
 - standardisierte Technologien (Sprachen, Werkzeuge) für die breite Anwendung
 - Ontologien als „soziale Kontrakte“ realistischere Grundhaltung
 - bieten Vorteile in Richtung modellgetriebener Entwicklung
- Allerdings sollte ihr Einsatz wohldosiert erfolgen
 - Berücksichtigung von Skalierbarkeit
 - Klassische Anfrageverarbeitung vs. Reasoning
 - Nutzen von Ontologien liegt vor allem im Umgang mit
 - unvollständigen Informationen
 - hohem Maß an Hintergrundwissen

Herausforderung: Ontology Engineering

- Achilles-Ferse der ontologiebasierten Verfahren ist die Entwicklung der Ontologien
 - aufwendiger als Schemamodellierung und Schemaintegration durch höhere Ausdrucksmächtigkeit
 - einige methodische Ansätze existieren
- Kann man zwischenmenschliche, arbeitsintegrierte Aushandlungsprozesse von Semantik nutzen:
 - Kombination mit Tagging-Ansätzen aus dem Web 2.0

Ontology Maturing



- Generell empfehlenswert (auch für andere Teile der VL):
Leser/Naumann: Informationsintegration, dpunkt, 2006

- **Ontologie-Begriff**
 - Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2):199-220, 1993.
 - Mike Uschold, Michael Grüninger: Ontologies: Principles, Methods and Applications, Knowledge Engineering Review 11(2), 1996

■ OWL

- <http://www.w3.org/2004/OWL/>

■ Semantic Web

- Tim Berners-Lee, James Hendler, Ora Lassila: The Semantic Web, Scientific American, Mai 2001
<http://www.sciam.com/linktous.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21>