Semantic Web

Ein Vortrag von Holger Szillat

Betreut von Jochen Hipp

Seminar “Konzepte von Informationssystemen”
Sommersemester 2002
Universität Tübingen

szillat@informatik.uni-tuebingen.de

3. Juli 2002

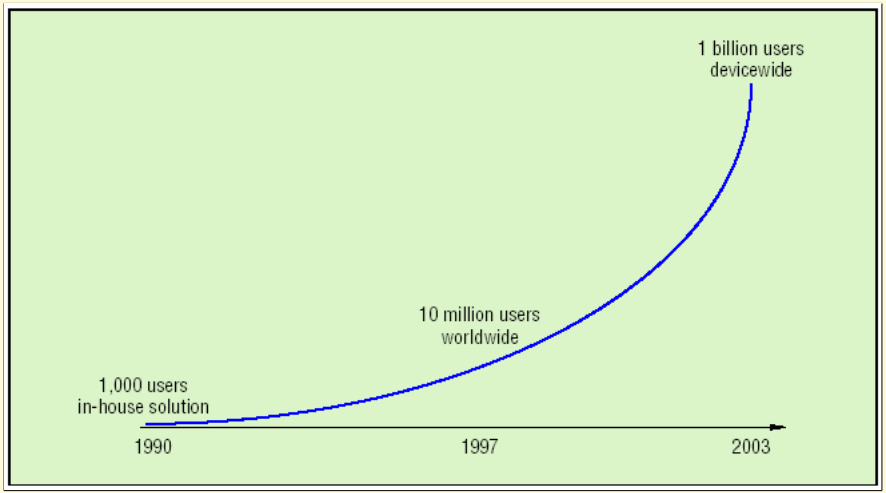
Version 1.19

Übersicht

- *Was ist Semantic Web überhaupt?*
  - *Was ist falsch mit dem World Wide Web heute?*
  - *WWW — The Next Generation*
- *Wie macht man Semantic Web?*
  - *SHOE*
  - *RDF und RDFSchema*
  - *OIL und DAML*
- *Warum Semantic Web?*
  - *Für wen? Wirklich “The Next Generation”??*
- *Fazit und Ausblick*

Was ist falsch mit dem World Wide Web heute?

- *“Eyeballweb”.*
- *Informationen sind für Maschinen “unverständlich”.*
- $\Rightarrow$ *Informationen schwer zu finden.*



Problem: Wie findet man Informationen?

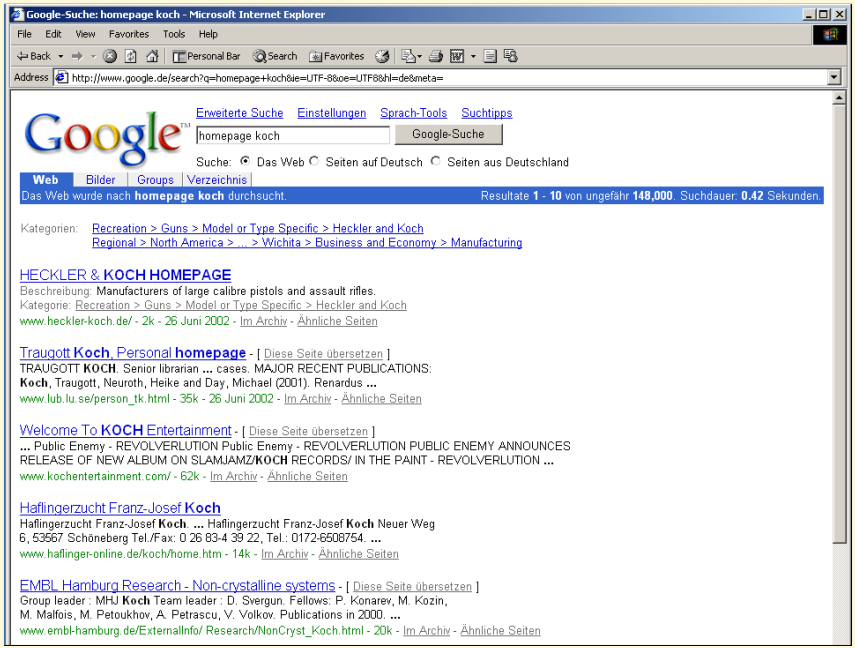
Klassische Ansätze: Suchmaschinen.

- *Altavista, Google. . . : (Kleveres) Text Retrieval.*
  Aber: Sucherfolge manchmal unbefriedigend.
- *Yahoo, web.de. . . : Auch anderer Ansatz: Gelbe Seiten.*
  Aber: Manuell erstellt, teuer, Zentrale Instanz.

Warum? *World Wide Web* = “Eyeballweb” und zu groß!

Wie findet man Informationen?

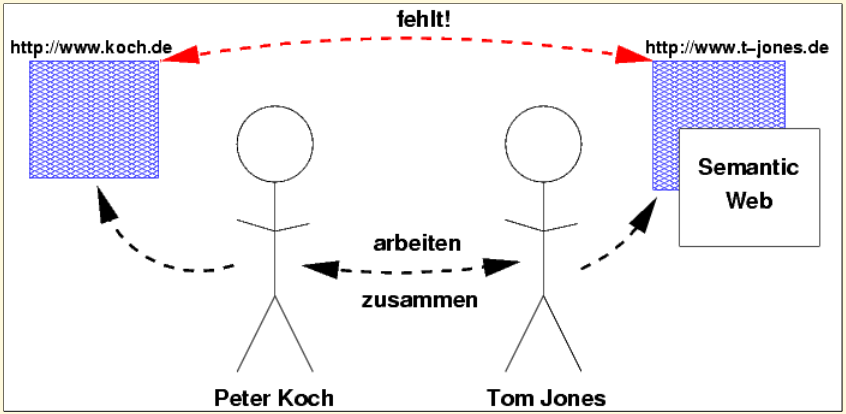
Beispiel: Bei `www.google.de` die Homepage von Herrn Koch suchen:



Ungefähr 148.000 Resultate…

Problem: Wie findet man Informationen, die nicht direkt zusammenhängen?

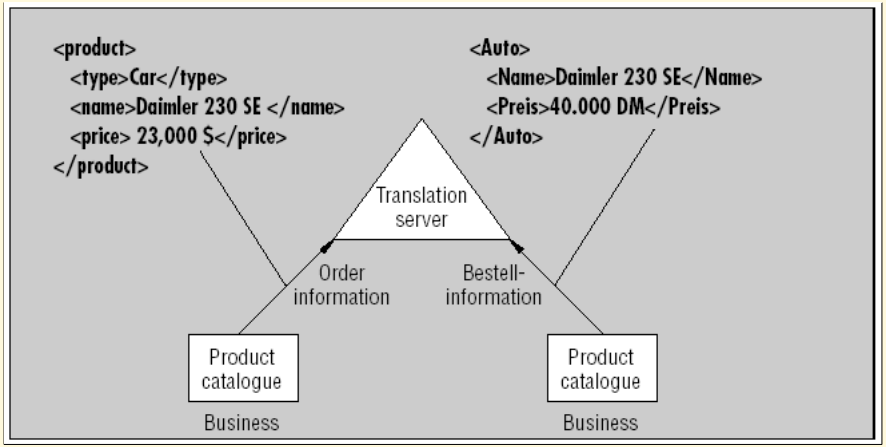
Problem: Herr Koch hat einen Kollegen. Dieser hat Informationen über *Semantic Web*. Aber: Herr Koch hat keinen Link auf seiner Seite…



Lösung mit dem heutigen *World Wide Web*: Unmöglich!

Wie kann man Informationen “vergleichen”?

- *Beispiele: Preise, Testergebnisse . . .*
- *Bisher: Preisagenturen, Hand-made Software.*
  Aber: Teuer und aufwendig.
- *In Zukunft: Intelligente Software und Agenten.*



Vision: Preise vergleichen und Produkt bestellen!

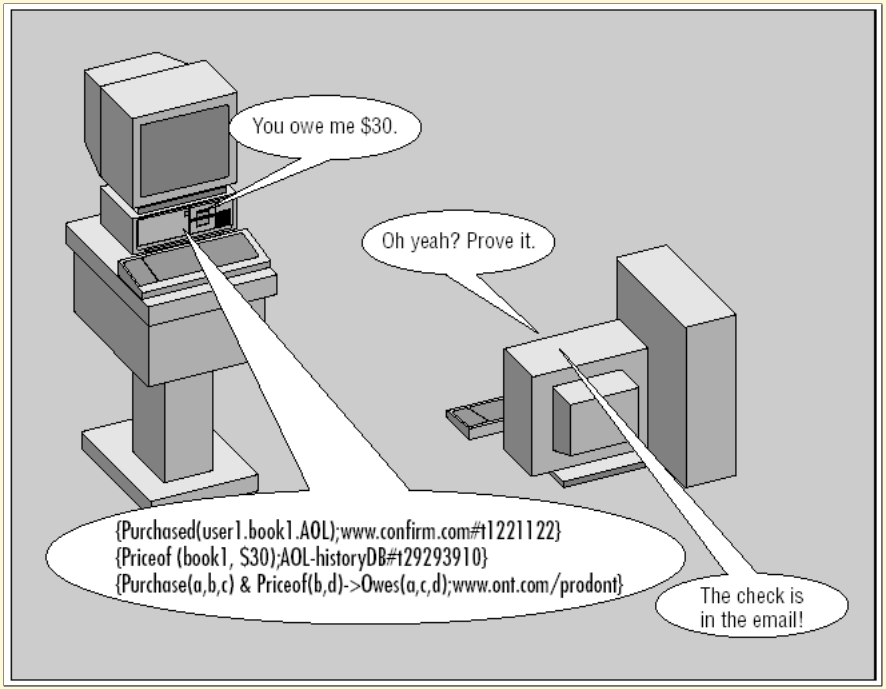
Ein Agent…

- *vergleicht Preise in Bezug auf Testergebisse.*
- *sucht günstigestes Angebot inkl. Versand, Steuer, Zoll…*
- *bestellt Produkt auf Webseite, die man u.U. vorher noch nie besucht hat.*

Mit *Semantic Web* und Ontologien leicht machbar!

Vision: "Proof-Checking"

Semantic Web ermöglicht es, eine Transaktion zu "beweisen."



"World Wide Web — The Next Generation"

- *WWW bleibt "Eyeballweb".*
- *Aber: Meta-Informationen müssen hinzukommen!*
- *$\Rightarrow$ Das nächste WWW muß auch von Maschinen "verstanden" werden können.*
- *Vision: "Semantic Web"*
  Begriff von Tim Berners-Lee
  im Buch "Weaving the Web" von 1999.

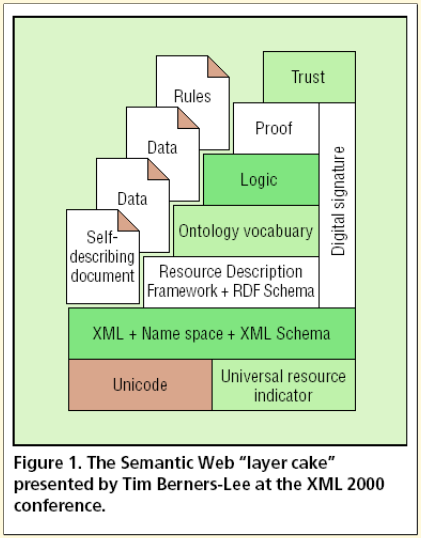


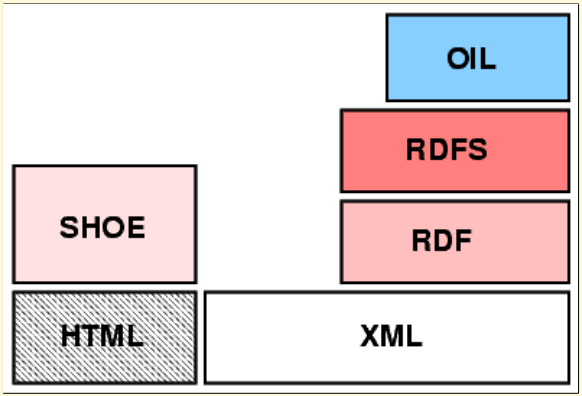
Figure 1. The Semantic Web "layer cake" presented by Tim Berners-Lee at the XML 2000 conference.

Definition: “Ontologie”

- *Oxford English Dictionary: “Study or Science of Being”.*
- *KI: “[An Ontology is a] representation of a shared conceptualization of a particular domain”*
- *Einfach: “Wissen”, Gesamtheit von Fakten, Regeln, Logik…*

SHOE

- **S**imple **H**TML **O**ntology **E**xtensions
- Idee: HTML + Ontologie



SHOE

Beispiel: Ontologien mittels Tags im HTML-Dokument eingebettet.

```
<HTML>
<HEAD>
 <TITLE>Ontologien von Peter Koch</TITLE>
 <META HTTP-EQUIV="SHOE" CONTENT="VERSION=1.0">
</HEAD>
<BODY>
 <H1>Ontologien von Peter Koch</H1>
 ...
 <ONTOLOGY ID="koch-ontology" VERSION="1.0">
  <USE-ONTOLOGY ID="base-ontology" VERSION="1.0" PREFIX="b"
   URL="http://www.cs.umd.edu/projects/plus/SHOE/base.html">
  <DEF-CATEGORY NAME="Person" ISA="b.SHOEentity">
  <DEF-CATEGORY NAME="Webpage" ISA="b.SHOEentity">
  ...
 </ONTOLOGY>
</BODY>
</HTML>
```

SHOE

Beispiel: Ontologien mittels Tags im HTML-Dokument eingebettet.

```
<HTML>
<HEAD>
 <TITLE>Ontologien von Peter Koch</TITLE>
 <META HTTP-EQUIV="SHOE" CONTENT="VERSION=1.0">
</HEAD>
<BODY>
 <H1>Ontologien von Peter Koch</H1>
 ...
 <ONTOLOGY ID="koch-ontology" VERSION="1.0">
  <USE-ONTOLOGY ID="base-ontology" VERSION="1.0" PREFIX="b"
   URL="http://www.cs.umd.edu/projects/plus/SHOE/base.html">
  <DEF-CATEGORY NAME="Person" ISA="b.SHOEentity">
  <DEF-CATEGORY NAME="Webpage" ISA="b.SHOEentity">
  ...
 </ONTOLOGY>
</BODY>
</HTML>
```

SHOE

Beispiel: Ontologien definieren.

```
 ...
 <ONTOLOGY ID="koch-ontology" VERSION="1.0">
  <USE-ONTOLOGY ID="base-ontology" VERSION="1.0" PREFIX="b"
   URL="http://www.cs.umd.edu/projects/plus/SHOE/base.html">
  <DEF-CATEGORY NAME="Person" ISA="b.SHOEentity">
  <DEF-CATEGORY NAME="Webpage" ISA="b.SHOEentity">
  ...
  <DEF-RELATION NAME="createdBy">
   <DEF-ARG POS="1" TYPE="Person">
   <DEF-ARG POS="2" TYPE="Webpage">
  </DEF-RELATION>
  ...
 </ONTOLOGY>
 ...
</BODY>
</HTML>
```

SHOE

Beispiel: Inferenzen (für neue Ontologien.)

```
...
 <DEF-INFERENCE>
  <INF-IF>
   <RELATION NAME="createdBy">
    <ARG POS="1" VALUE="x" VAR>
    <ARG POS="2" VALUE="y" VAR>
   </RELATION>
  </INF-IF>
  <INF-THEN>
   <RELATION NAME="hasCreated">
    <ARG POS="1" VALUE="y" VAR>
    <ARG POS="2" VALUE="x" VAR>
   </RELATION>
  </INF-THEN>
 </DEF-INFERENCE>
...
```

SHOE

Beispiel: Inferenzen (für neue Ontologien.)

```
...
 <DEF-INFERENCE>
  <INF-IF>
   <RELATION NAME="createdBy">
    <ARG POS="1" VALUE="x" VAR>
    <ARG POS="2" VALUE="y" VAR>
   </RELATION>
  </INF-IF>
  <INF-THEN>
   <RELATION NAME="hasCreated">
    <ARG POS="1" VALUE="y" VAR>
    <ARG POS="2" VALUE="x" VAR>
   </RELATION>
  </INF-THEN>
 </DEF-INFERENCE>
...
```

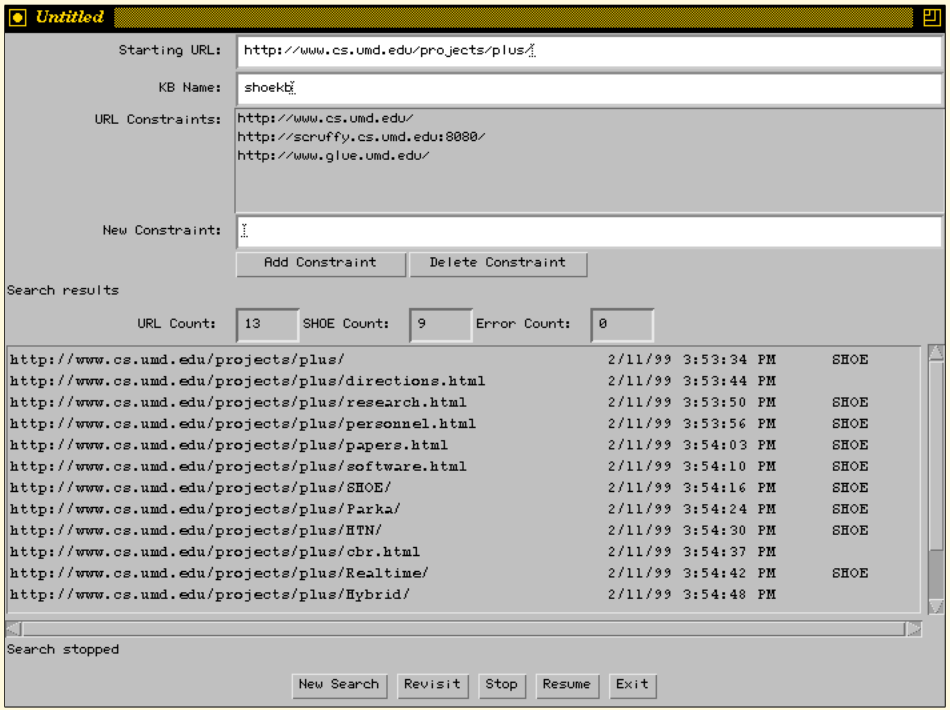
SHOE

Beispiel: Instanzen definieren = Ontologien nutzen.

```
<HTML>
<HEAD>
 <TITLE>Homepage von Peter Koch</TITLE>
 <META HTTP-EQUIV="SHOE" CONTENT="VERSION=1.0">
</HEAD>
<BODY>
 <H1>Homepage von Peter Koch</H1>
 ...
 <INSTANCE ID="http://www.koch.de/index.html" VERSION="1.0">
  <USE-ONTOLOGY ID="koch-ontology" VERSION="1.0" PREFIX="my"
   URL="http://www.koch.de/ontos.html">
  <RELATION NAME="createdBy">
   <ARG POS="1" VALUE="Peter Koch">
   <ARG POS="2" VALUE="http://www.koch.de">
  </RELATION>
  ...
 </INSTANCE>
</BODY>
</HTML>
```

SHOE-Tools

Beispiel: Exposé — hilft Ontologien zu suchen und zu nutzen.

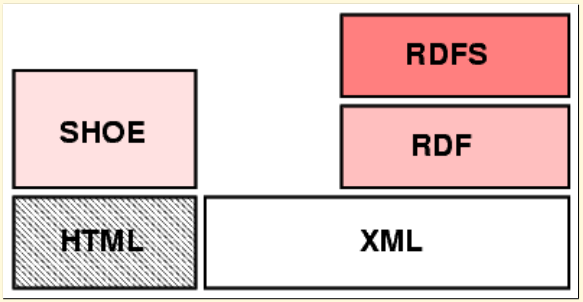


Löst SHOE das Problem?

- *Vorteile:*
  - *SHOE "kann" Variablen, Inferenzen, n-stellige Relationen...*
  - *SGML und XML DTD Spezifikation.*
  - *Schon lange da (seit 1997).*
  - *Es gibt einige Tools: Knowledge Annotator, Exposé ...*
  - *Eingebettet im HTML-Dokument.*
- *Nachteile:*
  - *Eingebettet im HTML-Dokument: Wie andere Ressourcen beschreiben?*
  - *HTML-Editoren können sich an den Tags verschlucken.*
  - *Sackgasse? Schon veraltet??*

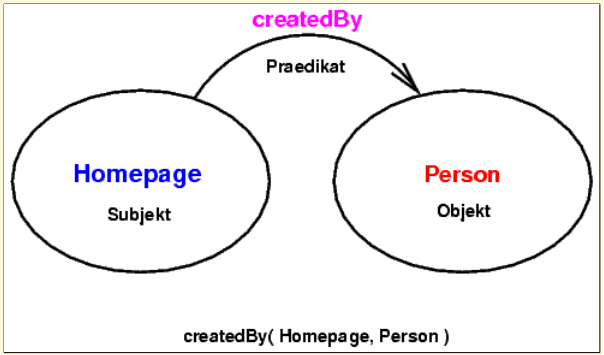
Lösung des Problems: RDF

- ***R**esource **D**escription **F**ramework.*
- *Standard vom W3C seit 1999; momentan noch “Working Draft”.*
- *Ziel: Alle “Web-erreichbaren” Resourcen beschreiben.*
- *Theoretisch auch alle anderen!*



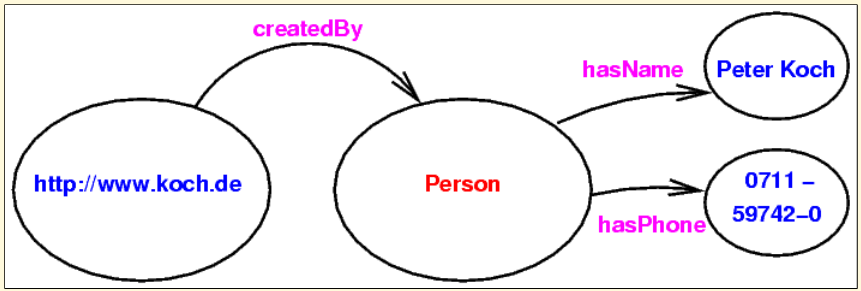
RDF Beispiel:

- *RDF ist nur ein Datenmodell.*
- *RDF-Statements: Subjekt + Prädikat + Objekt*
- *Alle Elemente mit* `URI`.



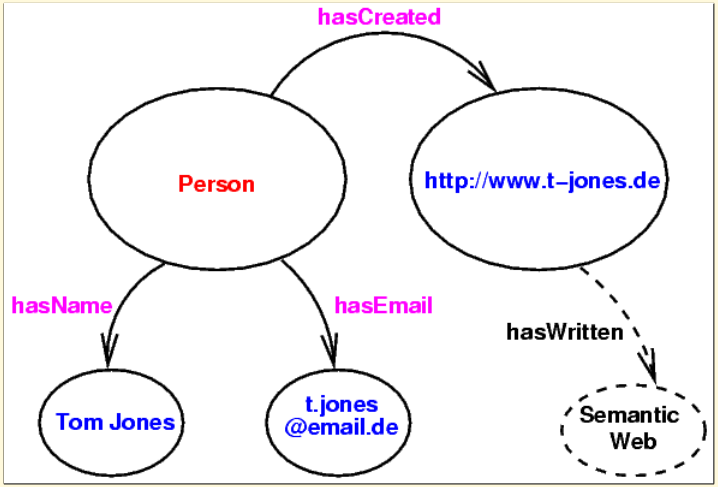
RDF Beispiel, Teil 2.

Eine Ontologie von Peter Koch's Homepage:



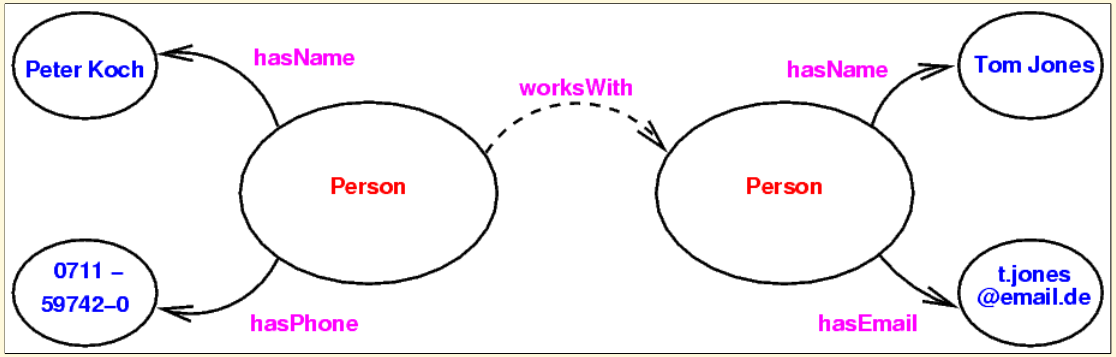
RDF Beispiel, Teil 2.

Eine Ontologie von Tom Jones' Homepage:



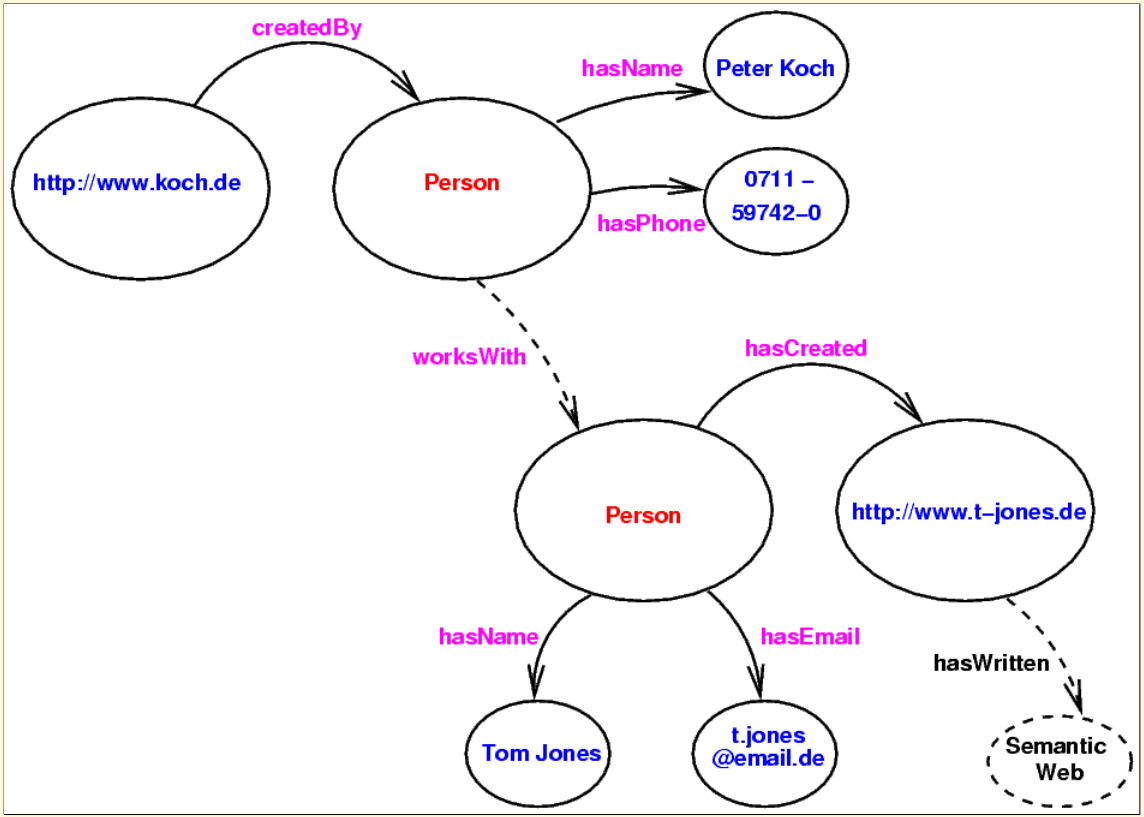
RDF Beispiel, Teil 2.

Eine Ontologie von Peter Koch's Firma:



RDF Beispiel, Teil 2.

RDF-Statements können reiifiziert werden!



RDF Beispiel, Teil 3.

Eine mögliche Darstellung:

```
<?xml version='1.0' encoding='UTF-8'?>
<rdf:RDF xmlns:rdf='http://www.w3c.org/rdf-syntax-ns#'
         xmlns:dc='http://purl.org/elements/1.0/'
         xmlns:NS0='http://www.w3c.org/...' >
 <rdf:Description rdf:about='http://www.koch.de'>
  <dc:createdBy rdf:resource='#A0'/>
 </rdf:Description>
 <rdf:Description rdf:about='#A0'>
  <NS0:hasName>Peter Koch</NS0:hasName>
  <NS0:hasPhone>0711 - 59742-0</NSO:hasPhone>
 </rdf:Description>
</rdf:RDF>
```

RDF Beispiel, Teil 3.

Eine mögliche Darstellung:

```
<?xml version='1.0' encoding='UTF-8'?>
<rdf:RDF xmlns:rdf='http://www.w3c.org/rdf-syntax-ns#'
         xmlns:dc='http://purl.org/elements/1.0/'
         xmlns:NS0='http://www.w3c.org/...' >
 <rdf:Description rdf:about='http://www.koch.de'>
  <dc:createdBy rdf:resource='#A0'/>
 </rdf:Description>
 <rdf:Description rdf:about='#A0'>
  <NS0:hasName>Peter Koch</NS0:hasName>
  <NS0:hasPhone>0711 - 59742-0</NSO:hasPhone>
 </rdf:Description>
</rdf:RDF>
```

RDF Beispiel, Teil 3.

Eine mögliche Darstellung:

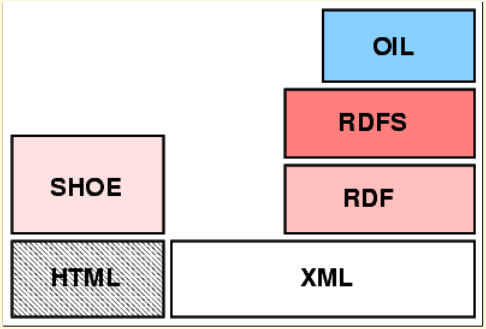
```
<?xml version='1.0' encoding='UTF-8'?>
<rdf:RDF xmlns:rdf='http://www.w3c.org/rdf-syntax-ns#'
         xmlns:dc='http://purl.org/elements/1.0/'
         xmlns:NS0='http://www.w3c.org/...' >
 <rdf:Description rdf:about='http://www.koch.de'>
  <dc:createdBy rdf:resource='#A0'/>
 </rdf:Description>
 <rdf:Description rdf:about='#A0'>
  <NS0:hasName>Peter Koch</NS0:hasName>
  <NS0:hasPhone>0711 - 59742-0</NSO:hasPhone>
 </rdf:Description>
</rdf:RDF>
```

RDF-Beispiel unter der Lupe

+ *Man kann jede Ressource beschreiben!*
+ *RDF nutzt XML und XMLSchema.*
+ *RDF läßt sich mit RDFSchema erweitern.*
− *Keine Variablen, nur binäre Relationen.*

RDFSchema — RDF erweitern

- *RDF um eigene Klassen und Eigenschaften erweitern.*
- *Objektorientierte Idee: Klassen + Unterklasen + Eigenschaften.*



RDFSchema-Beispiel

```
...
<rdfs:Class rdf:ID='Herbivore'>
 <rdfs:type rdf:resource='.../#DefinedClass'/>
 <rdfs:subClassOf rdf:resource='#Animal'/>
</rdfs:Class>
...
<rdfs:Property rdf:ID='hasName'>
 <rdfs:label>The Name of an animal</rdfs:label>
 <rdfs:domain rdf:resource='#Animal'/>
 <rdfs:range rdf:resource='http://www.w3c.org/rdf-schema#Literal'/>
</rdfs:Property>
...
<Herbivore rdf:about='http://www.farm.de/Cow/Sieglinde'>
 <hasName>Sieglinde</hasName>
</Herbivore>
...
```

RDFSchema-Beispiel

```
...
<rdfs:Class rdf:ID='Herbivore'>
 <rdfs:type rdf:resource='.../#DefinedClass'/>
 <rdfs:subClassOf rdf:resource='#Animal'/>
</rdfs:Class>
...
<rdfs:Property rdf:ID='hasName'>
 <rdfs:label>The Name of an animal</rdfs:label>
 <rdfs:domain rdf:resource='#Animal'/>
 <rdfs:range rdf:resource='http://www.w3c.org/rdf-schema#Literal'/>
</rdfs:Property>
...
<Herbivore rdf:about='http://www.farm.de/Cow/Sieglinde'>
 <hasName>Sieglinde</hasName>
</Herbivore>
...
```

RDFSchema-Beispiel

```
...
<rdfs:Class rdf:ID='Herbivore'>
 <rdfs:type rdf:resource='.../#DefinedClass'/>
 <rdfs:subClassOf rdf:resource='#Animal'/>
</rdfs:Class>
...
<rdfs:Property rdf:ID='hasName'>
 <rdfs:label>The Name of an animal</rdfs:label>
 <rdfs:domain rdf:resource='#Animal'/>
 <rdfs:range rdf:resource='http://www.w3c.org/rdf-schema#Literal'/>
</rdfs:Property>
...
<Herbivore rdf:about='http://www.farm.de/Cow/Sieglinde'>
 <hasName>Sieglinde</hasName>
</Herbivore>
...
```

RDFSchema — RDF erweitern

Beispiel unter der Lupe:

\+ *Benutzt XML zur Beschreibung.*

+− *Kann nur Klassen- und Unterklassen-Beziehungen…*

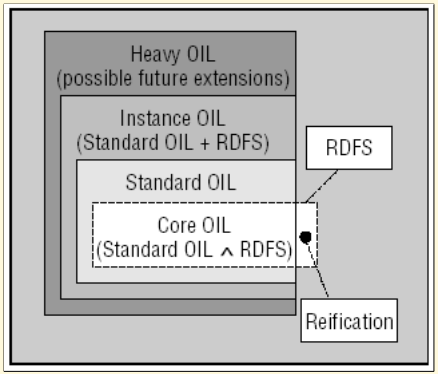
+− *…und nur Eigenschaften und Untereigenschaften modellieren.*

− *Keine Variablen!*

− *Keine Transitivität, Negation, Inversion…*

OIL

- ***O**ntology **I**nference **L**ayer*
- *OIL erweitert RDF/RDFSchema um Logik.*
- *In Schichten gegliedert: Eine Applikation kann die höheren Schichten u.U. ignorieren.*
- *Maschinen- und Menschenlesbar.*



OIL — Maschinenlesbares Beispiel

Modellierung: “Herbivoren” und “Carnivoren”?

```
 ...
<rdfs:Class rdf:ID='Herbivore'>
 <rdfs:type rdf:resource='...'/>
 <rdfs:subClassOf rdf:resource='#Animal'/>
 <rdfs:subClassOf>
  <oil:NOT>
   <oil:hasOperand rdf:resource='#Carnivore'/>
  </oil:NOT>
 </rdfs:subClassOf>
</rdfs:Class>
...
```

Vorteil: Nur *RDFSchema*-fähige Applikation kann trotzdem Informationen nutzen.

OIL — Maschinenlesbares Beispiel

Modellierung: “Herbivoren” und “Carnivoren”?

```
 ...
<rdfs:Class rdf:ID='Herbivore'>
 <rdfs:type rdf:resource='...'/>
 <rdfs:subClassOf rdf:resource='#Animal'/>
 <rdfs:subClassOf>
  <oil:NOT>
   <oil:hasOperand rdf:resource='#Carnivore'/>
  </oil:NOT>
 </rdfs:subClassOf>
</rdfs:Class>
...
```

Vorteil: Nur *RDFSchema*-fähige Applikation kann trotzdem Informationen nutzen.

OIL — Menschenlesbares Beispiel

Modellierung: Was sind “Herbivoren” und “Carnivoren”?

```
...
class-def defined carnivore
  subclass-of animal
  slot-constraint eats value-type animal

class-def defined herbivore
  subclass-of animal
  slot-constraint eats
    value-type plant
    (OR slot-constraint is-part-of has-value plant)

disjoint carnivore herbivore
...
```

OIL — Menschenlesbares Beispiel

Modellierung: Was sind “Herbivoren” und “Carnivoren”?

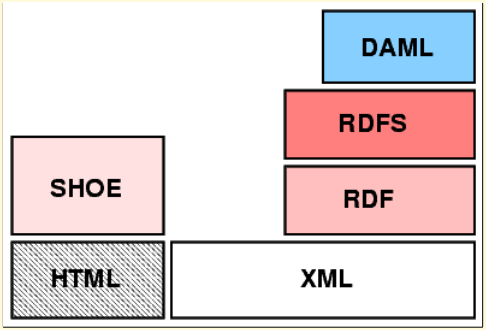
```
...
class-def defined carnivore
  subclass-of animal
  slot-constraint eats value-type animal

class-def defined herbivore
  subclass-of animal
  slot-constraint eats
    value-type plant
    (OR slot-constraint is-part-of has-value plant)

disjoint carnivore herbivore
...
```

OIL unter der Lupe

+ *OIL baut auf RDF und RDFSchema auf.*
+ *Erweitert RDF und RDFSchema um mächtigere Konstrukte.*
+ *Applikationen können OIL auch ignorieren.*
− *Wird wohl in DAML aufgehen...*

DAML

- ***D**arpa **A**gent **M**arkup **L**anguage.*
- *Viele große Namen hinter dem Projekt: DARPA, Tim Berners-Lee…*
  (Viele Leute vom W3C)
- *Ziel: Das Beste von SHOE, RDF, RDFSchema und OIL.*



DAML Beispiel

```
 ...
<daml:Class rdf:ID='Animal'>
 <rdfs:label>Animal</rdfs:label>
 ...
</daml:Class>

<daml:Class rdf:ID='Herbivore'>
 <rdfs:subClassOf rdf:resource='#Animal'/>
 <daml:disjointWith rdf:resource='#Carnivore'/>
</daml:Class>

<daml:Class rdf:ID='Carnivore'>
 <rdfs:subClassOf rdf:resource='#Animal'/>
 <rdfs:subClassOf>
  <daml:Restriction>
   <daml:onProperty rdf:resource='#eats'/>
   <daml:toClass rdf:resource='#Herbivore'/>
  </daml:Restriction>
 </rdfs:subClassOf>
</daml:Class>
...
```

DAML Beispiel

```
 ...
<daml:Class rdf:ID='Animal'>
 <rdfs:label>Animal</rdfs:label>
 ...
</daml:Class>

<daml:Class rdf:ID='Herbivore'>
 <rdfs:subClassOf rdf:resource='#Animal'/>
 <daml:disjointWith rdf:resource='#Carnivore'/>
</daml:Class>

<daml:Class rdf:ID='Carnivore'>
 <rdfs:subClassOf rdf:resource='#Animal'/>
 <rdfs:subClassOf>
  <daml:Restriction>
   <daml:onProperty rdf:resource='#eats'/>
   <daml:toClass rdf:resource='#Herbivore'/>
  </daml:Restriction>
 </rdfs:subClassOf>
</daml:Class>
...
```

DAML Beispiel

```
 ...
<daml:Class rdf:ID='Animal'>
 <rdfs:label>Animal</rdfs:label>
 ...
</daml:Class>

<daml:Class rdf:ID='Herbivore'>
 <rdfs:subClassOf rdf:resource='#Animal'/>
 <daml:disjointWith rdf:resource='#Carnivore'/>
</daml:Class>

<daml:Class rdf:ID='Carnivore'>
 <rdfs:subClassOf rdf:resource='#Animal'/>
 <rdfs:subClassOf>
  <daml:Restriction>
   <daml:onProperty rdf:resource='#eats'/>
   <daml:toClass rdf:resource='#Herbivore'/>
  </daml:Restriction>
 </rdfs:subClassOf>
</daml:Class>
...
```

DAML Beispiel

```
 ...
<daml:Class rdf:ID='Animal'>
 <rdfs:label>Animal</rdfs:label>
 ...
</daml:Class>

<daml:Class rdf:ID='Herbivore'>
 <rdfs:subClassOf rdf:resource='#Animal'/>
 <daml:disjointWith rdf:resource='#Carnivore'/>
</daml:Class>

<daml:Class rdf:ID='Carnivore'>
 <rdfs:subClassOf rdf:resource='#Animal'/>
 <rdfs:subClassOf>
  <daml:Restriction>
   <daml:onProperty rdf:resource='#eats'/>
   <daml:toClass rdf:resource='#Herbivore'/>
  </daml:Restriction>
 </rdfs:subClassOf>
</daml:Class>
...
```

DAML unter der Lupe

+ *Mächtig! Kann viel!!*
+ *Noch sehr jung (seit 2001) $\Rightarrow$ Viel Potential.*
+ *Nutzt die Erfahrungen von SHOE, RDF, RDFSchema und OIL.*
+ *Wird wahrscheinlich zum Standard werden!*
− *Vielleicht doch zu groß?*

Semantic Web — Warum?

Für Endbenutzer…

- als “Client”? Ja!
- als “Server”? Eher nicht.
- “The Next Generation” — Bestimmt!

Semantic Web — Warum?

Für Firmen…

- als “Client”? Ja, auch.
  (Testergebnisse, Produkte leichter zu finden…)
- als “Server”? Vielleicht!
  (Ausreichend Geld und Personal ermöglichen proprietäre Lösung?)
- “The Next Generation”? Tja…

Fazit und Ausblick

- Manche Probleme mit dem Web von heute nicht lösbar!
- "*Semantic Web* macht das Leben leichter…"
- *SHOE* wohl Sackgasse?
- *RDF* und *RDFSchema* sind gute Ansätze.
- *OIL* hat gute Ideen.
- *DAML* vielleicht zu mächtig?
- *Semantic Web* kann sich nur durchsetzen bei "ausreichend Masse!"
- Die nächste Generation des Webs wird kommen!
  Wann? In welcher Form??

Surf long and prosper...