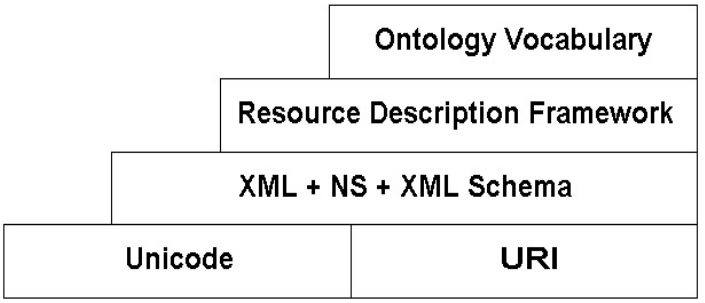
OWL Web Ontology Language

Minh Tuan Nguyen, 03.06.2003

Inhalt

- XML & RDF
- Ontologien & DAML+OIL
- OWL
- Tools

Übersicht + Motivation



RDFS und Ontologien

- RDF Schema hat begrenzte Ausdruckskraft
- Z.B. Disjunkte Klassen, Inverse Rollen?
- Für komplexere Schemas −>

(komplexere) Ontologien.

Beispiel

motherOf **subProperty** parentOf
if Mary motherOf Bill
then Mary parentOf Bill

Beispiel (cont‘d)

motherOf **inverseOf** childOf

if Mary motherOf Bill
then Bill childOf Mary

Ontologien

- Ursprünglich: Begriff aus der Philosopie (Metaphysik)
- **Die Lehre vom Sein**: Existenz von Dingen im Universum?

- Hier: Menge von Begriffen (z.B Dinge, Ereignisse, Beziehungen), mit einer Beschreibung, die ein geregeltes Vokabular für den Informationsaustausch darstellen

Bestandteile einer Ontologie

- Classes der Ontologie
- Taxonomische Anordnung der classes
- Eigenschaften und erlaubte Werte
- “Inference Rules“ (Ableitungsregeln)

Classes und Properties

- **Classes** beschreiben die Konzepte in der Domäne
- **Properties** beschreiben die Eigenschaften der Klassen und deren “instances“
- **Inference Rules** :

  sibling ist eine symmetrische Beziehung

  If sibling(x,y) then sibling(y,x).

  childOf und motherOf inverse Beziehungen

  if childOf(x,y) then motherOf(y,x)

DAML+OIL

- August 2000: DAML-ONT, von der Regierung gesponsertes Projekt, um detailliertere RDF class Definitionen auszudrücken
- OIL liefert komplexere Klassifizierungen durch die Verwendung von Konstrukten aus der KI
- DAML und OIL = DAML+OIL
- DAML+OIL, komplexere Klassifizierung und Properties von Ressourcen als durch RDFS
- DAML+OIL verwendet RDF

DAML+OIL einige Sprachelemente

- TransitiveProperty:
  if P is TransitiveProperty, then if P(x, y) and P(y, z) then P(x, z).
- UniqueProperty: if P is a UniqueProperty, then if P(x, y) and P(x, z) then y=z
- sameIndividualAs:
  sameIndividualAs(a, b) =
  a is the same individual as b

Beispiel

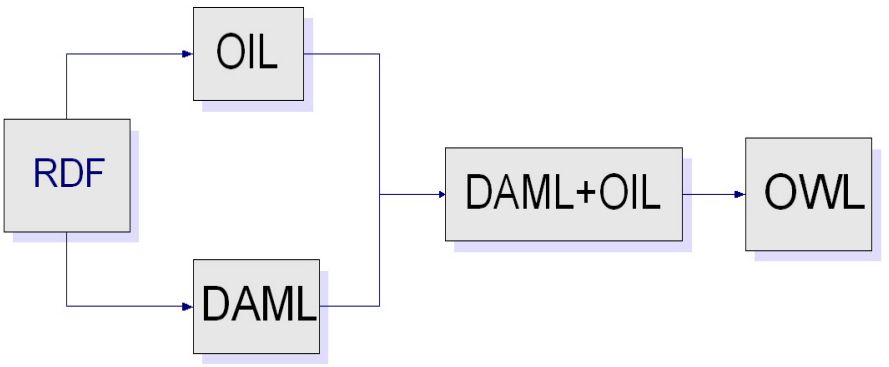
Eine Person hat **genau eine** Mutter

```
<daml:Class rdf:ID="Person">
<rdfs:subClassOf>
     <daml:Restriction>
       <daml:onProperty rdf:resource="#hasMother"/>
       <daml:minCardinality>1</daml:minCardinality>
       <daml:maxCardinality>1</daml:maxCardinality>
     </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

OWL

- DAML+OIL und RDF Semantik nicht 100% compatible
- W3C verwendet DAML+OIL als Grundlage für OWL
- DAML+OIL vergleichbar mit OWL DL!
- OWL (Full) ist eine Vokabularerweiterung von RDF
- Eine OWL Ontologie ist ein RDF Graph.
  (Menge von RDF Tripel)

OWL



OWL

- **Zweck** : eine Methode bereitstellen, um (in XML Syntax) Klassen, ihre Eigenschaften und Beziehungen zwischen Klassen zu definieren,
- **Vorteil** von OWL: Folgerungen (Aussagen über Klassen und Properties) in besserer Art und Weise möglich (mehr als RDF Schema)

OWL ist Sprache zum Definieren und Instanziieren von Web Ontologien!

Was kommt jetzt?

- Klassen mit OWL definieren
- Properties mit OWL definieren
- Relationships mit OWL definieren

Namespaces

- Präzise Angabe des benutzten Vokabulars

```
<rdf:RDF
    xmlns ="http://www.w3.org/2001/sw/WebOnt/guide-src/wine#"
    xmlns:vin ="http://www.w3.org/2001/sw/WebOnt/guide-src/wine#"
    xmlns:food="http://www.w3.org/2001/sw/WebOnt/guide-src/food#"
    xmlns:owl ="http://www.w3.org/2002/07/owl#"
    xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd ="http://www.w3.org/2000/10/XMLSchema#">
```

Ontology Header

```
<owl:Ontology rdf:about="">
   <rdfs:comment>An example OWL ontology</rdfs:comment>
   <owl:priorVersion rdf:resource=
            "http://www.w3.org/2001/sw/WebOnt/guide-src/wine-112102.owl"/>
   <owl:imports rdf:resource="http://www.w3.org/2001/sw/WebOnt/guide-src/food.owl"/>
   <rdfs:label>Wine Ontology</rdfs:label>
  ...
</owl:Ontology>
```

OWL Features

- **Basic Elements**
  - Class
  - Property,
  - Property Restrictions
- **Mapping**
  - Äquivalenz
  - sameIndividualAs
  - differentFrom
- **Complex Classes**
  - Vereinigung
  - Schnitt
  - Komplement

Class

```
<rdfs:Class rdf:ID="River">
    <rdfs:subClassOf rdf:resource="#Stream"/>
</rdfs:Class>
```

RDFS

```
<owl:Class rdf:ID="River">
    <rdfs:subClassOf rdf:resource="#Stream"/>
</owl:Class>
```

OWL

Properties

- RDF Schema stellt drei Methoden zur Verfügung Eigenschaften zu charakterisieren:
  - **range**: mögliche Werte für eine Property
  - **domain**: assoziiert eine Property mit einer Klasse
  - **subPropertyOf**: Spezialisierung einer Property

Properties

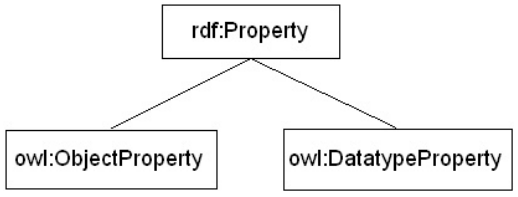
- OWL liefert zusätzliche Methoden Properties zu charakterisieren

  -> verbesserte Schlussfolgerungen

Properties

- Bei RDF Schema: rdf:Property bei
  - Beziehung einer Resource zu einer anderen
  - Beziehung einer Resource zu einem rdfs:Literal oder Datentyp
- Bei OWL sind dies zwei Klassen von Properties:
  - ***owl:ObjectProperty*** für Resource - Resource
  - ***owl:DatatypeProperty*** für *Resource - rdfs:Literal* oder Resource - XML Schema built-in Datentyp

Property Hierarchie



owl:ObjectProperty und *owl:DatatypeProperty* sind Subklassen von *rdf:Property*

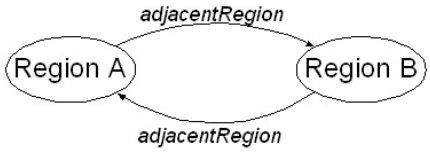
Symmetric Properties

P(x,y) ⇔ P(y,x)

adjacentRegion ist eine symmetrische Property

Region A *adjacentRegion* Region B

⇔ Region B *adjacentRegion* Region A



Symmetric Property

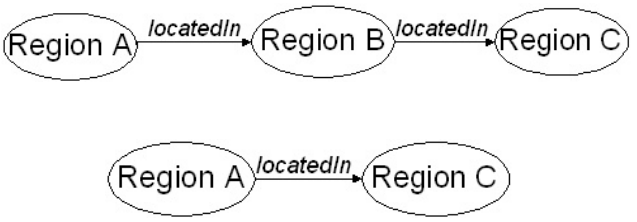
```
<owl:ObjectProperty rdf:ID="adjacentRegion">
    <rdf:type
        rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty" />
    <rdfs:domain rdf:resource="#Region" />
    <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>

<Region rdf:ID="MendocinoRegion">
    <locatedIn rdf:resource="#CaliforniaRegion" />
    <adjacentRegion rdf:resource="#SonomaRegion" />
</Region>
```

Transitive Properties

P(x,y) and P(y,z) => P(x,z)

locatedIn ist eine transititive Property



Transitive Property

```
<owl:ObjectProperty rdf:ID="locatedIn">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
  <rdfs:domain rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
  <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>
```

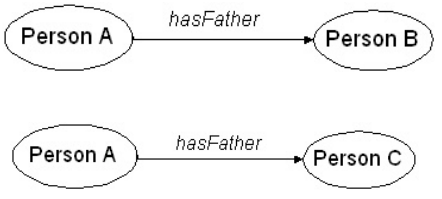
Transitive Property

```
<Region rdf:ID="SantaCruzMountainsRegion">
  <locatedIn rdf:resource="#CaliforniaRegion" />
</Region>

<Region rdf:ID="CaliforniaRegion">
  <locatedIn rdf:resource="#USRegion" />
</Region>
```

Functional Property

P(x,y) and P(x,z) => y = z



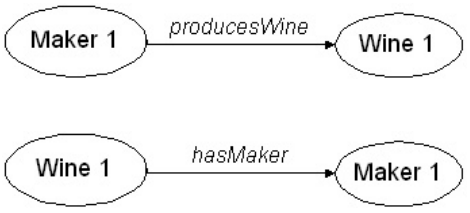
Daraus folgern wir: **Person C = Person B**

Functional Property

```
<owl:Class rdf:ID="VintageYear" />
<owl:ObjectProperty rdf:ID="hasVintageYear">
  <rdf:type
      rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty" />
  <rdfs:domain rdf:resource="#Vintage" />
  <rdfs:range rdf:resource="#VintageYear" />
</owl:ObjectProperty>
```

Inverse Property

P1(x,y) ⇔ P2(y,x)



producesWine ist die inv. Property von *hasMaker*

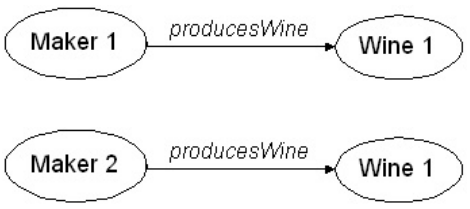
Inverse Property

```
<owl:ObjectProperty rdf:ID="hasMaker">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty“ />
  </owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="producesWine">
  <owl:inverseOf rdf:resource="#hasMaker" />
</owl:ObjectProperty>
```

Inverse Functional Property

P(y,x) and P(z,x) => y = z



Falls *producesWine* eine inv. func. Property,

Maker 1 = Maker 2

Inverse Functional Property

```
<owl:ObjectProperty rdf:ID="producesWine">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <owl:inverseOf rdf:resource="#hasMaker" />
</owl:ObjectProperty>
```

Property Restriction

- Manchmal nützlich Bereich (range) von Properties einzuschränken

Local Restrictions

- rdfs:range stellt globale Einschränkung der Property dar
- Wir wollen den Bereich der Property einschränken.
- Wir brauchen eine *lokale Definition der Property*.

Local Restrictions

- *Wine* ist eine trinkbare Flüssigkeit und wird in einem Weinkeller (*Winery*) hergestellt (*hasMaker*)

Local Restrictions

(1) *Wine* ist subclass von *PotableLiquid*, und subclass einer anonymen Klasse, die eine Property *hasMaker* enthält, und alle Werte für *hasMaker* müssen "*Winery*" sein

(2) *Wine* ist subclass von *PotableLiquid*, und hat eine Property *hasMaker*. Alle Werte für *hasMaker* müssen instances von *Winery* sein

allValuesFrom

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource=
                "http://www.w3.org/2001/sw/WebOnt/guide-src/food#PotableLiquid" />
   ...
  <rdfs:subClassOf>
     <owl:Restriction>
        <owl:onProperty rdf:resource="#hasMaker" />
        <owl:allValuesFrom rdf:resource="#Winery" />
     </owl:Restriction>
  </rdfs:subClassOf>
   ...
</owl:Class>
```

Property Restrictions

```
<Wine rdf:ID="someWine"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        …
<hasMaker rdf:resource="  <someURI>  "/>
</Wine>
```

Jetzt wissen wir : This value is a winery

Property Restrictions

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource=
            "http://www.w3.org/2001/sw/WebOnt/guide-src/food# PotableLiquid" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
      <owl:someValuesFrom rdf:resource="#Winery" />
    </owl:Restriction>
  </rdfs:subClassOf> ...
</owl:Class>
```

allValuesFrom - someValuesFrom

- ***allValuesFrom*** : Wenn *Wine* einen Maker hat, dann ist Maker eine "*instance"* von Winery
- ***someValuesFrom***: Für alle "*instances*" von wine gilt: **mindestens** ein Maker ist eine "*instance"* von *Winery*

hasValue

- Alle Burgunder sind trocken

```
<owl:Class rdf:ID="Burgundy">
 ...
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#hasSugar" />
         <owl:hasValue rdf:resource="#Dry" />
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Kardinalität

- Kardinalität: Anzahl der Vorkommnisse.
- 3 Möglichkeiten Kardinalität auszudrücken:
  - cardinality
  - minCardinality
  - maxCardinality

Kardinalität

Vintage **hat genau** ein *VintageYear*

```
<owl:Class rdf:ID="Vintage">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasVintageYear"/>
      <owl:cardinality rdf:datatype=
                    "...#nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Kardinalität

Definition eines Bereiches

```
…
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#name"/>
    <owl:minCardinality rdf:datatype="…#nonNegativeInteger">0</owl:minCardinality>
    <owl:maxCardinality rdf:datatype="…#nonNegativeInteger">10</owl:maxCardinality>
</owl:Restriction>
</rdfs:subClassOf>
…
```

Mapping

- "shared Ontologies"
- Ontologien aus unterschiedlichen Quellen "kombinieren" -> **Wiederverwendbarkeit**

equivalentProperty

```
<owl:DatatypeProperty rdf:ID="name">
   <owl:equivalentProperty rdf:resource="…/dublin-core#Title"/>
</owl:DatatypeProperty>
```

P1(x,y) ⇔ P2(x,y)

Property Extension ist identisch

equivalentClass

```
<owl:Class rdf:ID="TexasThings">
  <owl:equivalentClass>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#locatedIn" />
         <owl:allValuesFrom rdf:resource="#TexasRegion" />
      </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

Complex Classes

- zusätzliche Methoden, Klassen zu bilden:
- Vereinigung (*owl:unionOf*)
- Schnitt (*owl:intersectionOf*)
- Komplement (*owl:complementOf*)
- Enumeration (*owl:oneOf*)

intersectionOf

- Weißwein = weißer Wein

```
<owl:Class rdf:ID="WhiteWine">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine" />
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasColor" />
        <owl:hasValue rdf:resource="#White" />
      </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

unionOf

Frucht = süße Frucht ∪ nicht-süße Frucht

```
<owl:Class rdf:ID="Fruit">
  <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#SweetFruit" />
      <owl:Class rdf:about="#NonSweetFruit" />
  </owl:unionOf>
</owl:Class>
```

complementOf

- $\bar{A} = \{x \mid x \notin A\}$
- NonFrenchWine = Wine not located in France
- NonFrenchWine =

  Wine $\cap$ {x | x not located in France}

complementOf

```
<owl:Class rdf:ID="NonFrenchWine">
  <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#Wine"/>
      <owl:Class>
          <owl:complementOf>
              <owl:Restriction>
                  <owl:onProperty rdf:resource="#locatedIn" />
                  <owl:hasValue rdf:resource="#FrenchRegion" />
              </owl:Restriction>
          </owl:complementOf>
      </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```

Enumerated Classes

- Definition einer Klasse durch Aufzählen seiner Mitglieder (*instances*)
- Class extension somit vollständig beschrieben, d.h. kein anderes Individuum gehört dieser Klasse an

oneOf

```
<owl:Class rdf:ID="WineColor">
    <rdfs:subClassOf rdf:resource="#WineDescriptor"/>
    <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#White"/>
        <owl:Thing rdf:about="#Rose"/>
        <owl:Thing rdf:about="#Red"/>
    </owl:oneOf>
</owl:Class>
```

Disjoint Classes

- Individuum ist Mitglied einer Klasse, nicht Mitglied einer best. anderen Klasse

disjointWith

```
<owl:Class rdf:ID="Pasta">
      <rdfs:subClassOf rdf:resource="#EdibleThing"/>
      <owl:disjointWith rdf:resource="#Meat"/>
      <owl:disjointWith rdf:resource="#Fowl"/>
      <owl:disjointWith rdf:resource="#Seafood"/>
      <owl:disjointWith rdf:resource="#Dessert"/>
      <owl:disjointWith rdf:resource="#Fruit"/>
</owl:Class>
```

Mapping - sameIndividualAs

- Declares two **individuals** to be identical

```
<Wine rdf:ID="MikesFavoriteWine">
   <owl:sameIndividualAs rdf:resource="#StGenevieveTexasWhite" />
</Wine>
```

Mapping - differentFrom

- Umgekehrte Eigenschaft von *sameIndividualAs*

```
<WineSugar rdf:ID="Dry" />

<WineSugar rdf:ID="Sweet">
    <owl:differentFrom rdf:resource="#Dry"/>
</WineSugar>

<WineSugar rdf:ID="OffDry">
   <owl:differentFrom rdf:resource="#Dry"/>
   <owl:differentFrom rdf:resource="#Sweet"/>
</WineSugar>
```

AllDifferent

- Define a set of mutually distinct individuals

```
<owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
        <vin:WineColor rdf:about="#Red" />
        <vin:WineColor rdf:about="#White" />
        <vin:WineColor rdf:about="#Rose" />
    </owl:distinctMembers>
</owl:AllDifferent>
```

- *owl:distinctMembers* nur zusammen mit *owl:AllDifferent*

OWL Class Properties

- rdf:Class :

  Properties:
  subClassOf
  domain
  range

- owl:Class:

  Properties:
  intersectionOf
  unionOf
  complementOf
  oneOf
  equivalentClass
  disjointWith

OWL Class Properties

- owl:Restriction:

  Properties:
  onProperty
  allValuesFrom
  hasValue
  someValuesFrom
  cardinality
  minCardinality
  maxCardinality

Versionen von OWL

- 3 Versionen von OWL:
  - OWL Full
  - OWL DL
  - OWL Lite

Full

- Enthält alle OWL Sprachkonstrukte
- Erlaubt alle Freiheiten von RDF (Metaklassen)
- owl:ObjectProperty = rdf:Property

  => DatatypeProperty $\subset$ ObjectProperty

- Keine Garantie!

DL

- DL Ontology kann nicht mit OWL Full Ontology verwendet werden
- Trennung zwischen classes, datatypes, datatype properties, etc…

  (keine Metaklassen).
- inverse of, inverse functional,symmetric, und transitive können nicht mit Datatype Properties verwendet werden

DL

- *owl:cardinality* mit *TransitiveProperty* nicht erlaubt
- Maximale Menge, bei der noch gilt: Folgerungen sind entscheidbar (und berechenbar)

Lite

- Einschränkungen, die für DL gelten + mehr:
- Nur Kardinalitäten 0 und 1
- ¬ *owl:oneOf,*
- ¬ *owl:unionOf,*
- ¬ *owl:complementOf,*
- ¬ *owl:hasValue,*
- ¬ *owl:disjointWith,*
- ¬ *owl:Nothing*

Lite

- minimale brauchbare Teilmenge der Sprache
- Sprachkonstrukte von Lite erlauben einfache Klassenhierarchien (subclass, value und cardinality)
- Properties optional oder erforderlich (mit cardinality).

Full/DL/Lite

OWL Lite ⊂ OWL DL ⊂ OWL Full

Full / DL / Lite

- Full:
  - Volle Ausdruckskraft
  - Schwierig, ein Tool für Full zu konstruieren.
- DL/Lite:
  - Tools können leichter und schneller entwickelt werden
  - Haben nicht die volle Ausdruckskraft der Sprache

RDF/OWL

- Jedes OWL (Full/DL/Lite) ist ein RDF Dokument
- Jedes RDF Dokument ist ein OWL Full Dokument
- **Nicht alle** RDF Dokumente sind OWL Lite oder OWL DL Doumente

OWL Tools

- Noch wenige Tools für OWL
- Zahlreiche Tools für DAML
- Vorhandene Tools, v.a. DAML, werden nach OWL migriert

OWL Tools

- **RDF Instance Creator** (RIC)
  - Tool zum Erstellen von semant. Mark-Up durch Einlesen von Ontologien.
  - begrenzte OWL Funktionalität
  - http://www.mindswap.org/~mhgrove/RIC/RIC.shtml
- **OWL Validator**:
  - Web-based utility
  - check OWL markup for problems beyond simple syntax errors
  - http://owl.bbn.com/validator/

OWL Tools

- **Dumpont**:
  - http://www.daml.org/2001/03/dumpont/
  - program to display the class and property hierarchies present in a DAML ontology, (also works with OWL)
- **OWL Ontology Validator**:
  - http://phoebus.cs.man.ac.uk:9999/OWL/Validator
  - a prüft, die Korrektheit eines Dokuments in Bezug auf OWL Lite, OWL DL, OWL Full
- **DAML Tools** Page - http://www.daml.org/tools/

Zusammenfassung

- Web Ontology Language mächtiger als RDFS
- Komplexe Ontologien definierbar
- Drei Varianten unterschiedlicher Ausdruckskraft
- Tools vermehrt verfügbar

Quellen

- **OWL Overview**
  http://www.w3c.org/TR/owl-features/
- **OWL Guide**
  http://www.w3c.org/TR/owl-guide/
- **OWL Reference**
  http://www.w3c.org/TR/owl-ref/
- **Introduction To DAML: Part I**
  http://www.xml.com/pub/a/2002/01/30/daml1.html
- **Wolgang Hesse: Ontologie(n)**
  Informatik Spektrum – 16. Dezember 2002