Semantic Web Technologies II

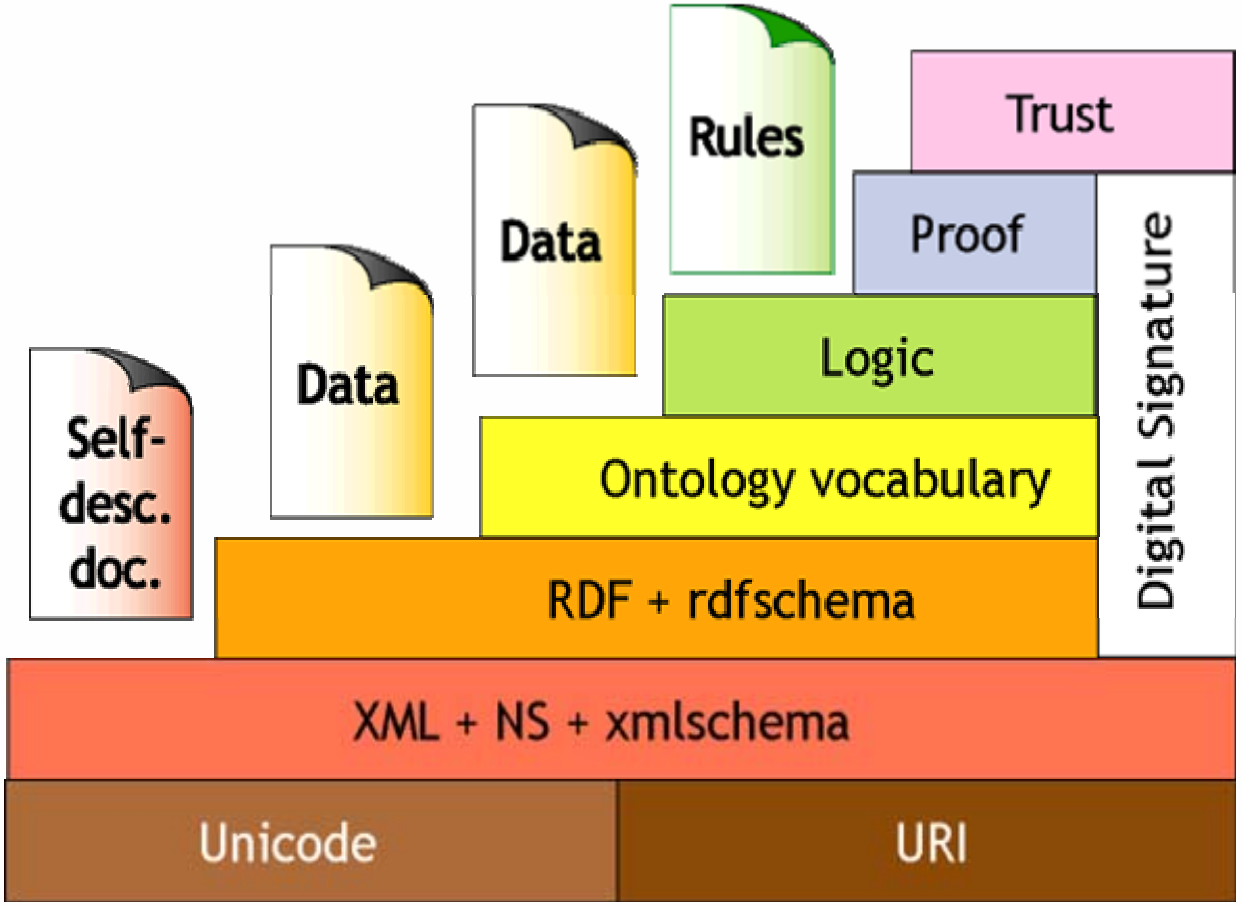
SS 2008

23.04.2008

OWL Syntax

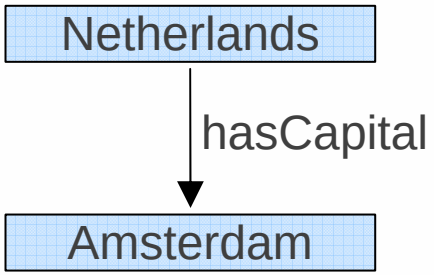
Dr. Peter Haase
PD Dr. Pascal Hitzler
Dr. Steffen Lamparter
Denny Vrandecic

Die Semantic Web Schichttorte



RDF: Das Datenmodel

- *statements* sind (subject, predicate, object) *triple*:
  - (Netherlands, hasCapital, Amsterdam)



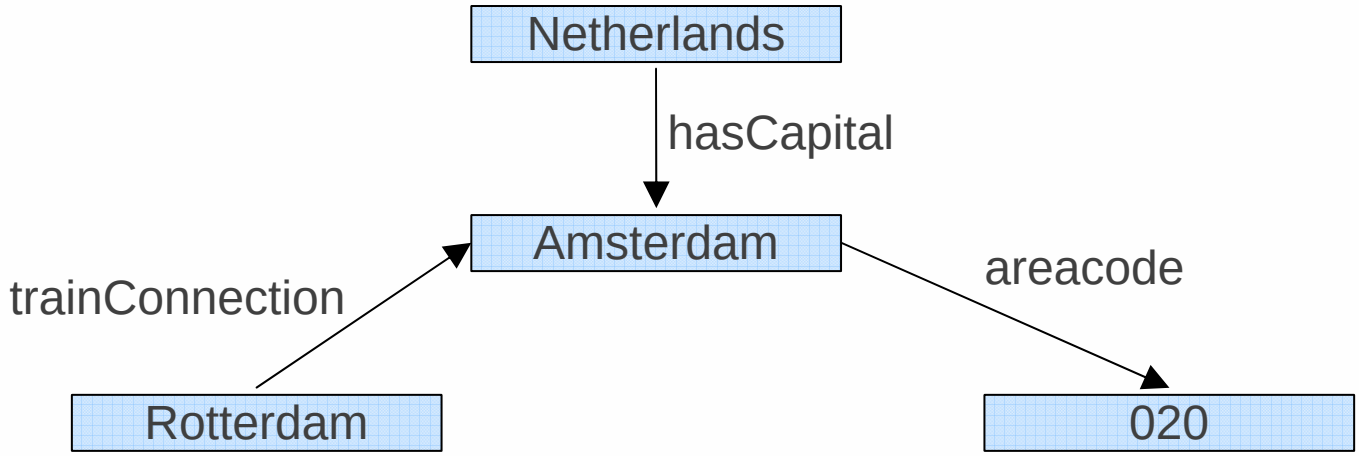
- statements beschreiben *resources (Object)*
- *Eine resource kann alles sein, was eine URI besitzt:*
  - *Ein Dokument, ein Bild, einTeil eines XML Dokuments*
    - *http://www.cs.vu.nl/index.html*
  - *ein Buch in einer Bibliothek:*
    - *isbn://5031-4444-3333*

URIs

- URI = Uniform Resource Identifier
- “The generic set of all names/addresses that are short strings that refer to resources”
- URLs (Uniform Resource Locators) sind spezielle URIs, die Addressen von Webseiten angeben.
- In RDF werden häufig URLs verwendet – oft mit direkten Verweisen auf Teile des Dokuments:
  - http://somedomain.com/some/path/to/file#fragmentID

Verknüpfung von Statements

- Das subject eines Statements kann das object (oder auch predicate) eines anderen statements sein
- Mengen solcher Statements bilden semantische Netzwerke:



RDF Syntax: XML

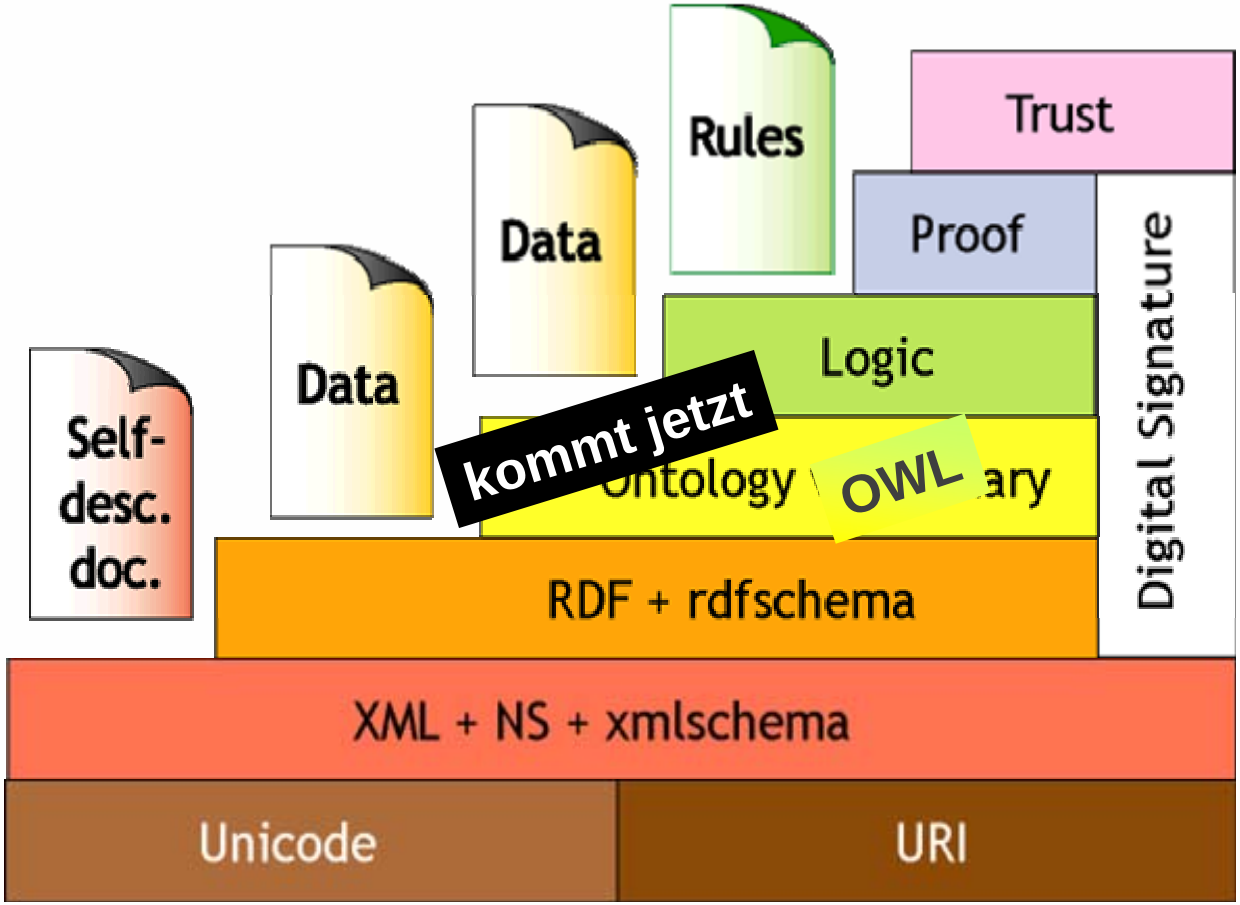
- RDF hat eine spezielle XML-basierte Syntax
  - Jedes Description Element beschreibt eine resource
  - Jedes geschachtelte Element bezeichnet eine property
  - Das Attribut des property elements bezeichnet das object:

```
<rdf:Description
rdf:about="http://www.countries.org/countries#Netherlands">
    <hasCapital rdf:resource="http://www.cities.org/cities#Amsterdam"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.cities.org/cities#Amsterdam">
    <areacode>020</areacode>
</rdf:Description>

<rdf:Description rdf:about="http://www.cities.org/cities#Amsterdam">
    <rdf:type rdf:resource="http://www.cities.org/cities#City"/>
</rdf:Description>
```

Die Semantic Web Schichttorte



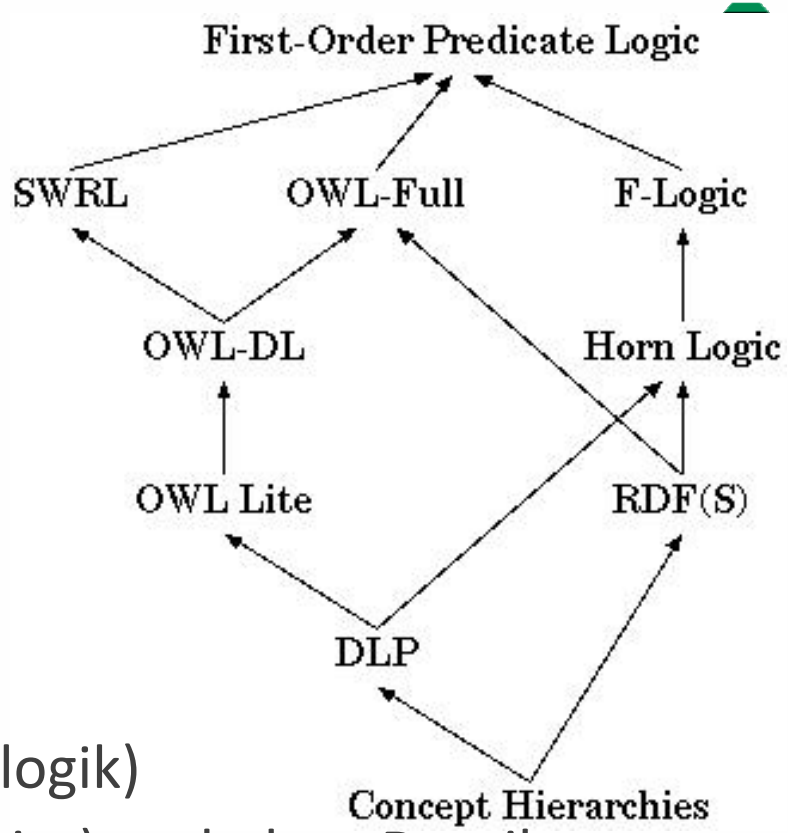
Ontologie in der Informatik

Gruber 93:

An Ontology is a

formal specification	⇒ machine-understandable
of a shared	⇒ group of people
conceptualization	⇒ about concepts
of a domain of interest	⇒ between general description and individual use

OWL – Allgemeines



- W3C Recommendation seit 2004
- Semantisches Fragment von FOL
- Drei Varianten:
  OWL Lite < OWL DL < OWL Full
- Keine Reifikation in OWL DL
  RDFS ist Fragment von OWL Full
- OWL DL ist entscheidbar
- OWL DL = SHOIN(D) (Beschreibungslogik)
- W3C-Dokumente (Vorlesungswebseite) enthalten Details, die hier nicht alle angesprochen werden können.

OWL Varianten

- OWL Full
  - Enthält OWL DL und OWL Lite
  - Enthält als einzige OWL-Teilsprache ganz RDFS
  - Semantik enthält einige Aspekte, die aus logischem Blickwinkel problematisch sind.
  - Unentscheidbar.
  - Wird durch aktuelle Softwarewerkzeuge nur bedingt unterstützt.
- OWL DL
  - Enthält OWL Lite und ist Teilsprache von OWL Full.
  - Entscheidbar.
  - Wird von aktuellen Softwarewerkzeugen fast vollständig unterstützt.
  - Komplexität NExpTime (worst-case).
- OWL Lite
  - Ist Teilsprache von OWL DL und OWL Full.
  - Entscheidbar.
  - Wenig ausdrucksstark.
  - Komplexität ExpTime (worst-case).

Inhalt

I. OWL – Syntax und allgemeines Verständnis
   a. Was ist „Ontologie“?
   b. Bausteine von OWL Dokumenten
      - **Kopf**
      - **Klassen, Rollen und Individuen**
      - **Klassenbeziehungen**
      - **komplexe Klassendefinitionen**
         - **logische Konstruktoren**
         - **Rolleneinschränkungen**
      - **Rolleneigenschaften**
   c. OWL Varianten
   d. Weitere Ressourcen

OWL Dokumente

- sind RDF Dokumente
- bestehen aus
  - Kopf mit allgemeinen Angaben
  - Rest mit der eigentlichen Ontologie

Der Kopf eines OWL Dokumentes

- Definition von Namespaces in der Wurzel

```
<rdf:RDF
   xmlns      ="http://www.semanticweb-
   grundlagen.de/beispielontologie#"
   xmlns:rdf  ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd  ="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl  ="http://www.w3.org/2002/07/owl#">
...
</rdf:RDF>
```

Der Kopf eines OWL Dokumentes

- Allgemeine Informationen

```
<owl:Ontology rdf:about=
   "http://www.semanticweb-grundlagen.de/beispielontologie#">
   <rdfs:comment
        rdf:datatype="http://www.w3.org/2001/XMLSchema#string“>
   Eine Beispielontologie
   </rdfs:comment>
   <owl:versionInfo>v0.5</owl:versionInfo>
   <owl:imports rdf:resource="http://www.semanticweb-
        grundlagen.de/foo"/>
   <owl:priorVersion
        rdf:resource="http://ontoware.org/projects/swrc"/>
  </owl:Ontology>
```

Der Kopf eines OWL Dokumentes

von RDFS geerbt

- `rdfs:comment`
- `rdfs:label`
- `rdfs:seeAlso`
- `rdfs:isDefinedBy`

außerdem

- `owl:imports`

für Versionierung

- `owl:versionInfo`
- `owl:priorVersion`
- `owl:backwardCompatibleWith`
- `owl:incompatibleWith`
- `owl:DeprecatedClass`
- `owl:DeprecatedProperty`

Inhalt

I. OWL – Syntax und allgemeines Verständnis
   a. Was ist „Ontologie“?
   b. Bausteine von OWL Dokumenten
      - **Kopf**
      - Klassen, Rollen und Individuen
      - **Klassenbeziehungen**
      - **komplexe Klassendefinitionen**
        - **logische Konstruktoren**
        - **Rolleneinschränkungen**
      - **Rolleneigenschaften**
   c. OWL Varianten
   d. Weitere Resourcen

Klassen, Rollen und Individuen

- Die drei Bausteine von Ontologieaxiomen.
- Klassen
  - Vergleichbar mit Klassen in RDFS
- Individuen
  - Vergleichbar mit Objekten in RDFS
- Rollen
  - Vergleichbar mit Properties in RDFS

Klassen

- Definition

```
<owl:Class rdf:ID="Professor"/>
```

  - vordefiniert:
    - **owl:Thing**
    - **owl:Nothing**

Individuen

- Definition durch Klassenzugehörigkeit

```
<rdf:Description rdf:ID="RudiStuder">
<rdf:type rdf:resource="#Professor"/>
</rdf:Description>
```

  - gleichbedeutend:

```
<Professor rdf:ID="RudiStuder"/>
```

Abstrakte Rollen

- abstrakte Rollen werden definiert wie Klassen

```
<owl:ObjectProperty
                rdf:ID="Zugehoerigkeit"/>
```

- Domain und Range abstrakter Rollen

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Organisation"/>
</owl:ObjectProperty>
```

Konkrete Rollen

- konkrete Rollen haben Datentypen im Range

```
<owl:DatatypeProperty rdf:ID="Vorname"/>
```

- Domain und Range konkreter Rollen

```
<owl:DatatypeProperty rdf:ID="Vorname">
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range  rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

- Viele XML Datentypen können verwendet werden. Im Standard vorgeschrieben sind `integer` und `string`.

Individuen und Rollen

```
<Person rdf:ID="RudiStuder">
  <Zugehoerigkeit rdf:resource="#AIFB"/>
  <Zugehoerigkeit
  rdf:resource="#ontoprise"/>
  <Vorname
  rdf:datatype="&xsd;string">Rudi</Vorname>
</Person>
```

- Rollen sind im allgemeinen nicht funktional.

Inhalt

I. OWL – Syntax und allgemeines Verständnis
   a. Was ist „Ontologie“?
   b. Bausteine von OWL Dokumenten
      - **Kopf**
      - **Klassen, Rollen und Individuen**
      - Klassenbeziehungen
      - **komplexe Klassendefinitionen**
        - **logische Konstruktoren**
        - **Rolleneinschränkungen**
      - **Rolleneigenschaften**
   c. OWL Varianten
   d. Weitere Ressourcen

Einfache Klassenbeziehungen

```
<owl:Class rdf:ID="Professor">
  <rdfs:subClassOf
     rdf:resource="#Fakultaetsmitglied"/>
</owl:Class>
<owl:Class rdf:ID="Fakultaetsmitglied">
  <rdfs:subClassOf
  rdf:resource="#Person"/>
</owl:Class>
```

Es folgt durch Inferenz, dass `Professor` eine Subklasse von `Person` ist.

Einfache Klassenbeziehungen

```
<owl:Class rdf:ID="Professor">
  <rdfs:subClassOf
   rdf:resource="#Fakultaetsmitglied"/>
</owl:Class>
<owl:Class rdf:ID="Buch">
  <rdfs:subClassOf rdf:resource="#Publikation"/>
</owl:Class>
<owl:Class rdf:about="#Fakultaetsmitglied">
  <owl:disjointWith rdf:resource="#Publikation"/>
</owl:Class>
```

Es folgt durch Inferenz, dass `Professor` und `Buch` ebenfalls disjunkte Klassen sind.

Einfache Klassenbeziehungen

```
<owl:Class rdf:ID="Buch">
  <rdfs:subClassOf
  rdf:resource="#Publikation"/>
</owl:Class>

<owl:Class rdf:about="#Publikation">
  <owl:equivalentClass
     rdf:resource="#Publication"/>
</owl:Class>
```

Es folgt durch Inferenz, dass `Buch` eine Subklasse von `Publication` ist.

Individuen und Klassenbeziehungen

```
<Buch rdf:ID="SemanticWebGrundlagen">
  <Autor rdf:resource="#YorkSure"/>
  <Autor rdf:resource="#PascalHitzler"/>
</Buch>
<owl:Class rdf:about="#Buch">
  <rdfs:subClassOf
  rdf:resource="#Publikation"/>
</owl:Class>
```

Es folgt durch Inferenz, dass `SemanticWebGrundlagen` eine `Publikation` ist.

Beziehungen zwischen Individuen

```
<Professor rdf:ID="RudiStuder"/>
<rdf:Description rdf:about="#RudiStuder">
  <owl:sameAs
     rdf:resource="#ProfessorStuder"/>
</rdf:Description>
```

Es folgt durch Inferenz, dass `ProfessorStuder` ein `Professor` ist.

Verschiedenheit von Individuen mittels

`owl:differentFrom`.

Beziehungen zwischen Individuen

```
<owl:AllDifferent>
  <owl:distinctMembers
          rdf:parseType="Collection">
    <Person rdf:about="#RudiStuder"/>
    <Person rdf:about="#YorkSure"/>
    <Person rdf:about="#PascalHitzler"/>
  </owl:distinctMembers>
</owl:AllDifferent>
```

Abgekürzte Schreibweise anstelle der Verwendung von mehreren `owl:differentFrom`.

Der Einsatz von `owl:AllDifferent` und `owl:distinctMembers` ist nur dafür vorgesehen.

Abgeschlossene Klassen

```
<owl:Class rdf:about=#SekretaerinnenVonStuder>
  <owl:oneOf rdf:parseType="Collection">
    <Person rdf:about="#GiselaSchillinger"/>
    <Person rdf:about="#AnneEberhardt"/>
  </owl:oneOf>
</owl:Class>
```

Dies besagt, dass es nur **genau diese beiden** `SekretaerinnenVonStuder` gibt.

Inhalt

I. OWL – Syntax und allgemeines Verständnis
   a. Was ist „Ontologie“?
   b. Bausteine von OWL Dokumenten
      - **Kopf**
      - **Klassen, Rollen und Individuen**
      - **Klassenbeziehungen**
      - komplexe Klassendefinitionen
        - **logische Konstruktoren**
        - **Rolleneinschränkungen**
      - **Rolleneigenschaften**
   c. OWL Varianten
   d. Weitere Ressourcen

Logische Konstruktoren auf Klassen

- logisches Und (Konjunktion):
  `owl:intersectionOf`
- logisches Oder (Disjunktion):
  `owl:unionOf`
- logische Negation:
  `owl:complementOf`

- Werden verwendet, um komplexe Klassen aus einfachen Klassen zu konstruieren.

Konjunktion

```
<owl:Class rdf:about="#SekretaerinnenVonStuder">
  <owl:equivalentClass>
    <owl:intersectionOf
          rdf:parseType="Collection">
      <owl:Class rdf:about="#Sekretaerinnen"/>
      <owl:Class
          rdf:about="#AngehoerigeAGStuder"/>
    </owl:intersectionOf>
  </owl:equivalentClass>
</owl:Class>
```

Es folgt z.B. durch Inferenz, dass alle `SekretaerinnenVonStuder` auch `Sekretaerinnen` sind.

Disjunktion

```
<owl:Class rdf:about="#Professor">
  <owl:subClassOf>
    <owl:unionOf
  rdf:parseType="Collection">
      <owl:Class
  rdf:about="#aktivLehrend"/>
      <owl:Class
  rdf:about="#imRuhestand"/>
    </owl:unionOf>
  </owl:subClassOf>
</owl:Class>
```

Negation

```
<owl:Class rdf:about="#Fakultaetsmitglied">
  <owl:subClassOf>
    <owl:complementOf
  rdf:resource="#Publikation"/>
  </owl:subClassOf>
</owl:Class>
```

Sehr komplizierte Art, das Folgende auszudrücken:

```
<owl:Class rdf:about="#Fakultaetsmitglied">
  <owl:disjointWith
  rdf:resource="#Publikation"/>
</owl:Class>
```

Rolleneinschränkungen (allValuesFrom)

- Dienen der Definition komplexer Klassen durch Rollen.

```
<owl:Class rdf:ID="Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:allValuesFrom rdf:resource="#Professor"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

D.h. *alle* Prüfer einer Prüfung müssen Professoren sein.

Rolleneinschränkungen (someValuesFrom)

```
<owl:Class rdf:about="#Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
  rdf:resource="#hatPruefer"/>
      <owl:someValuesFrom
  rdf:resource="#Person"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

D.h. jede Prüfung muss *mindestens einen* Prüfer haben.

Rolleneinschränkungen (cardinalities)

```
<owl:Class rdf:about="#Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:maxCardinality
            rdf:datatype="&xsd;nonNegativeInteger">
      2
      </owl:maxcardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Eine Prüfung kann *höchstens zwei* Prüfer haben.

Rolleneinschränkungen (cardinalities)

```
<owl:Class rdf:about="#Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatThema"/>
      <owl:minCardinality
            rdf:datatype="&xsd;nonNegativeInteger">
      3
      </owl:mincardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Eine Prüfung muss sich über *mindestens drei* Themengebiete erstrecken.

Rolleneinschränkungen (cardinalities)

```
<owl:Class rdf:about="#Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatThema"/>
      <owl:cardinality
      rdf:datatype="&xsd;nonNegativeInteger">
      3
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Eine Prüfung muss sich über *genau drei* Themengebiete erstrecken.

Rolleneinschränkungen (hasValue)

```
<owl:Class rdf:ID="PruefungBeiStuder">
  <rdfs:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:hasValue rdf:resource="#RudiStuder"/>
    </owl:Restriction>
  </rdfs:equivalentClass>
</owl:Class>
```

`owl:hasValue` verweist immer auf eine konkrete Instanz. Dies ist äquivalent zum Beispiel auf der nächsten Folie.

Rolleneinschränkungen (hasValue)

```
<owl:Class rdf:ID="PruefungBeiStuder">
  <rdfs:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:someValuesFrom>
        <owl:oneOf rdf:parseType="Collection">
          <owl:Thing rdf:about=#RudiStuder/>
        </owl:oneOf>
      </owl:someValuesFrom
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Inhalt

I. OWL – Syntax und allgemeines Verständnis
   a. Was ist „Ontologie“?
   b. Bausteine von OWL Dokumenten
      - **Kopf**
      - **Klassen, Rollen und Individuen**
      - **Klassenbeziehungen**
      - **komplexe Klassendefinitionen**
        - logische Konstruktoren
        - Rolleneinschränkungen
      - Rolleneigenschaften
   c. OWL Varianten
   d. Weitere Ressourcen

Rollenbeziehungen

```
<owl:ObjectProperty rdf:ID="hatPruefer">
  <rdfs:subPropertyOf
            rdf:resource="#hatAnwesenden"/>
</owl:ObjectProperty>
```

Ebenso: `owl:equivalentProperty`

Rollen können auch invers zueinander sein:

```
<owl:ObjectProperty rdf:ID="hatPruefer">
  <owl:inverseOf rdf:resource="#prueferVon"/>
</owl:ObjectProperty>
```

Rolleneigenschaften

- Domain
- Range
- Transitivität, d.h.
  r(a,b) und r(b,c) impliziert r(a,c)
- Symmetrie, d.h.
  r(a,b) impliziert r(b,a)
- Funktionalität
  r(a,b) und r(a,c) impliziert b=c
- Inverse Funktionalität

Domain und Range

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">
  <rdfs:range rdf:resource="#Organisation"/>
</owl:ObjectProperty>
```

Ist gleichbedeutend mit dem Folgenden:

```
<owl:Class rdf:about="owl:Thing">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#Zugehoerigkeit"/>
      <owl:allValuesFrom rdf:resource="#Organisation"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Domain und Range: Vorsicht!

```
<owl:ObjectProperty
  rdf:ID="Zugehoerigkeit">
  <rdfs:range
  rdf:resource="#Organisation"/>
</owl:ObjectProperty>
<Zahl rdf:ID="Fuenf">
  <Zugehoerigkeit
  rdf:resource="#Primzahlen"/>
</Zahl>
```

Es folgt nun, dass `Primzahlen` eine Organisation ist!

Rolleneigenschaften

```
<owl:ObjectProperty rdf:ID="hatKollegen">
  <rdf:type rdf:resource="owl:TransitiveProperty"/>
  <rdf:type rdf:resource="owl:SymmetricProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hatProjektleiter">
  <rdf:type rdf:resource="owl:FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="istProjektleiterFuer">
  <rdf:type rdf:resource="owl:InverseFunctionalProperty"/>
</owl:ObjectProperty>
<Person rdf:ID="YorkSure">
  <hatKollegen rdf:resource="#PascalHitzler"/>
  <hatKollegen rdf:resource="#AnupriyaAnkolekar"/>
  <istProjektleiterFuer rdf:resource="#SEKT"/>
</Person>
<Projekt rdf:ID="SmartWeb">
  <hatProjektleiter rdf:resource="#PascalHitzler"/>
  <hatProjektleiter rdf:resource="#HitzlerPascal"/>
</Projekt>
```

Folgerungen aus dem Beispiel

- `AnupriyaAnkolekar hatKollegen YorkSure`
- `AnupriyaAnkolekar hatKollegen PascalHitzler`
- `PascalHitzler owl:sameAs HitzlerPascal`

Inhalt

I. OWL – Syntax und allgemeines Verständnis
   a. Was ist „Ontologie“?
   b. Bausteine von OWL Dokumenten
      - **Kopf**
      - **Klassen, Rollen und Individuen**
      - **Klassenbeziehungen**
      - **komplexe Klassendefinitionen**
        - logische Konstruktoren
        - Rolleneinschränkungen
      - **Rolleneigenschaften**
   c. OWL Varianten
   d. Weitere Ressourcen

OWL Varianten

- OWL Full
  - Enthält OWL DL und OWL Lite
  - Enthält als einzige OWL-Teilsprache ganz RDFS
  - Semantik enthält einige Aspekte, die aus logischem Blickwinkel problematisch sind.
  - Unentscheidbar.
  - Wird durch aktuelle Softwarewerkzeuge nur bedingt unterstützt.
- OWL DL
  - Enthält OWL Lite und ist Teilsprache von OWL Full.
  - Entscheidbar.
  - Wird von aktuellen Softwarewerkzeugen fast vollständig unterstützt.
  - Komplexität NExpTime (worst-case).
- OWL Lite
  - Ist Teilsprache von OWL DL und OWL Full.
  - Entscheidbar.
  - Weniger ausdrucksstark.
  - Komplexität ExpTime (worst-case).

OWL Full

- Uneingeschränkte Nutzung aller OWL und RDFS-Sprachelemente (muss gültiges RDFS sein).
- Schwierig z.B.: nicht vorhandene Typentrennung (Klassen, Rollen, Individuen), dadurch:
  - `owl:Thing` dasselbe wie `rdfs:resource`
  - `owl:Class` dasselbe wie `rdfs:Class`
  - `owl:DatatypeProperty` Subklasse von `owl:ObjectProperty`
  - `owl:ObjectProperty` dasselbe wie `rdf:Property`

Beispiel für Typendurchmischung in OWL Full

```
<owl:Class rdf:about="#Buch">
  <englischerName rdf:datatype="&xsd;string">
    book
  </englischerName>
  <franzoesischerName rdf:datatype="&xsd;string">
    livre
  </franzoesischerName>
</owl:Class>
```

Inferenzen über solche Konstrukte werden oft nicht wirklich benötigt.

OWL DL

- Nur Verwendung von explizit erlaubten RDFS Sprachelementen (z.B. die in unseren Beispielen). Nicht erlaubt: `rdfs:Class`, `rdfs:Property`
- Typentrennung. Klassen und Rollen müssen explizit deklariert werden.
- Konkrete Rollen dürfen nicht als Transitiv, Symmetrisch, Invers oder Invers Funktional deklariert werden.
- Zahlenrestriktionen dürfen nicht mit transitiven Rollen, deren Subrollen, oder inversen davon verwendet werden.

OWL Lite

- Alle Einschränkungen für OWL DL gelten.
- Nicht erlaubt: `oneOf`, `unionOf`, `complementOf`, `hasValue`, `disjointWith`
- Zahlenrestriktionen nur mit 0 und 1 erlaubt.
- Einige Einschränkungen zum Auftreten von anonymen (komplexen) Klassen, z.B. nur im Subjekt von `rdfs:subClassOf`.

Inferenzanfragen an OWL (Klassen und Rollen)

- Klassenäquivalenz
- Subklassenbeziehung
- Disjunktheit von Klassen
- globale Konsistenz (Erfüllbarkeit, Widerspruchsfreiheit)
- Klassenkonsistenz: Eine Klasse ist *inkonsistent*, wenn sie äquivalent zu `owl:Nothing` ist.
  - Dies deutet oft auf einen Modellierungsfehler hin.

```
<owl:Class rdf:about="#Buch">
  <owl:subClassOf rdf:resource="#Publikation"/>
  <owl:disjointWith rdf:resource="#Publikation"/>
</owl:Class>
```

Inferenzanfragen an OWL (mit Individuen)

- Suche nach allen Individuen, die in einer Klasse enthalten sind.

- Instanzüberprüfung: Gehört gegebenes Individuum zu gegebener Klasse?

Inhalt

I. OWL – Syntax und allgemeines Verständnis
   a. Was ist „Ontologie“?
   b. Bausteine von OWL Dokumenten
      - **Kopf**
      - **Klassen, Rollen und Individuen**
      - **Klassenbeziehungen**
      - **komplexe Klassendefinitionen**
        - logische Konstruktoren
        - Rolleneinschränkungen
      - **Rolleneigenschaften**
   c. OWL Varianten
   d. Weitere Resourcen

OWL Werkzeuge

- Editoren
  - Protegé, http://protege.stanford.edu
  - SWOOP, http://www.mindswap.org/2004/SWOOP/
  - NeOn Toolkit, http://www.neon-toolkit.org/
- Inferenzmaschinen
  - Pellet, http://www.mindswap.org/2003/pellet/index.shtml
  - KAON2, http://kaon2.semanticweb.org
  - FACT++, http://owl.man.ac.uk/factplusplus/
  - Racer, http://www.racer-systems.com/
  - Cerebra, http://www.cerebra.com/index.html

OWL Sprachelemente

Kopf

- `rdfs:comment`
- `rdfs:label`
- `rdfs:seeAlso`
- `rdfs:isDefinedBy`
- `owl:versionInfo`
- `owl:priorVersion`
- `owl:backwardCompatibleWith`
- `owl:incompatibleWith`
- `owl:DeprecatedClass`
- `owl:DeprecatedProperty`
- `owl:imports`

Beziehungen zwischen Individuen

- `owl:sameAs`
- `owl:differentFrom`
- `owl:AllDifferent` (zusammen mit `owl:distinctMembers`)

Vorgeschriebene Datentypen

- `xsd:string`
- `xsd:integer`

OWL Sprachelemente

Klassenkonstruktoren und -beziehungen

- `owl:Class`
- `owl:Thing`
- `owl:Nothing`
- `rdfs:subClassOf`
- `owl:disjointWith`
- `owl:equivalentClass`
- `owl:intersectionOf`
- `owl:unionOf`
- `owl:complementOf`

Rollenrestriktionen

- `owl:allValuesFrom`
- `owl:someValuesFrom`
- `owl:hasValue`
- `owl:cardinality`
- `owl:minCardinality`
- `owl:maxCardinality`
- `owl:oneOf`

OWL Sprachelemente

Rollenkonstruktoren, -beziehungen und -eigenschaften

- `owl:ObjectProperty`
- `owl:DatatypeProperty`
- `rdfs:subPropertyOf`
- `owl:equivalentProperty`
- `owl:inverseOf`
- `rdfs:domain`
- `rdfs:range`
- `rdf:resource="owl:TransitiveProperty"`
- `rdf:resource="owl:SymmetricProperty"`
- `rdf:resource="owl:FunctionalProperty"`
- `rdf:resource="owl:InverseFunctionalProperty"`

Weiterführende Literatur

- http://www.w3.org/2004/OWL/
  zentrale W3C Webseite für OWL.
- http://www.w3.org/TR/owl-features/
  Überblick über OWL.
- http://www.w3.org/TR/owl-ref/
  vollständige Beschreibung der OWL-Sprachkomponenten.
- http://www.w3.org/TR/owl-guide/
  zeigt, wie OWL zur Wissensmodellierung verwendet werden kann.
- http://www.w3.org/TR/owl-semantics/
  beschreibt die Semantik von OWL, die wir auf andere Weise später behandeln werden. Es beschreibt außerdem die abstrakte Syntax für OWL DL, die wir hier später noch ansprechen.
- Deutsche Übersetzungen mancher W3C Dokumente findet man unter http://www.w3.org/2005/11/Translations/Lists/ListLang-de.html

Karlsruhe Institute of Technology

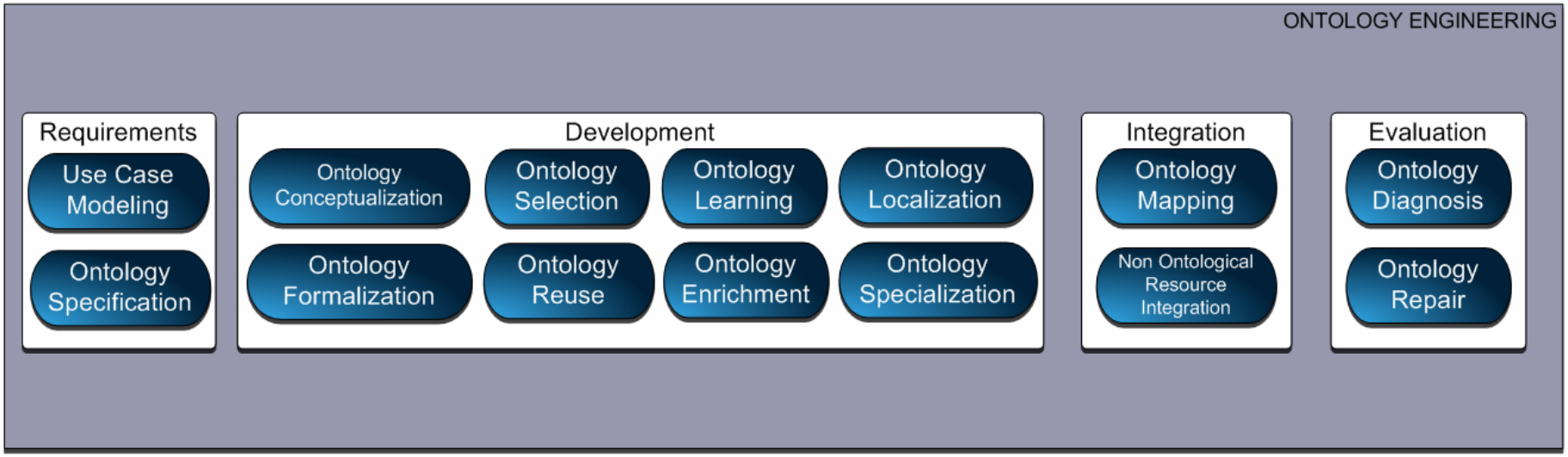
AIFB

Semantic Web Technologies II

23.04.2008

Ontology Engineering

Ontology Engineering Activities



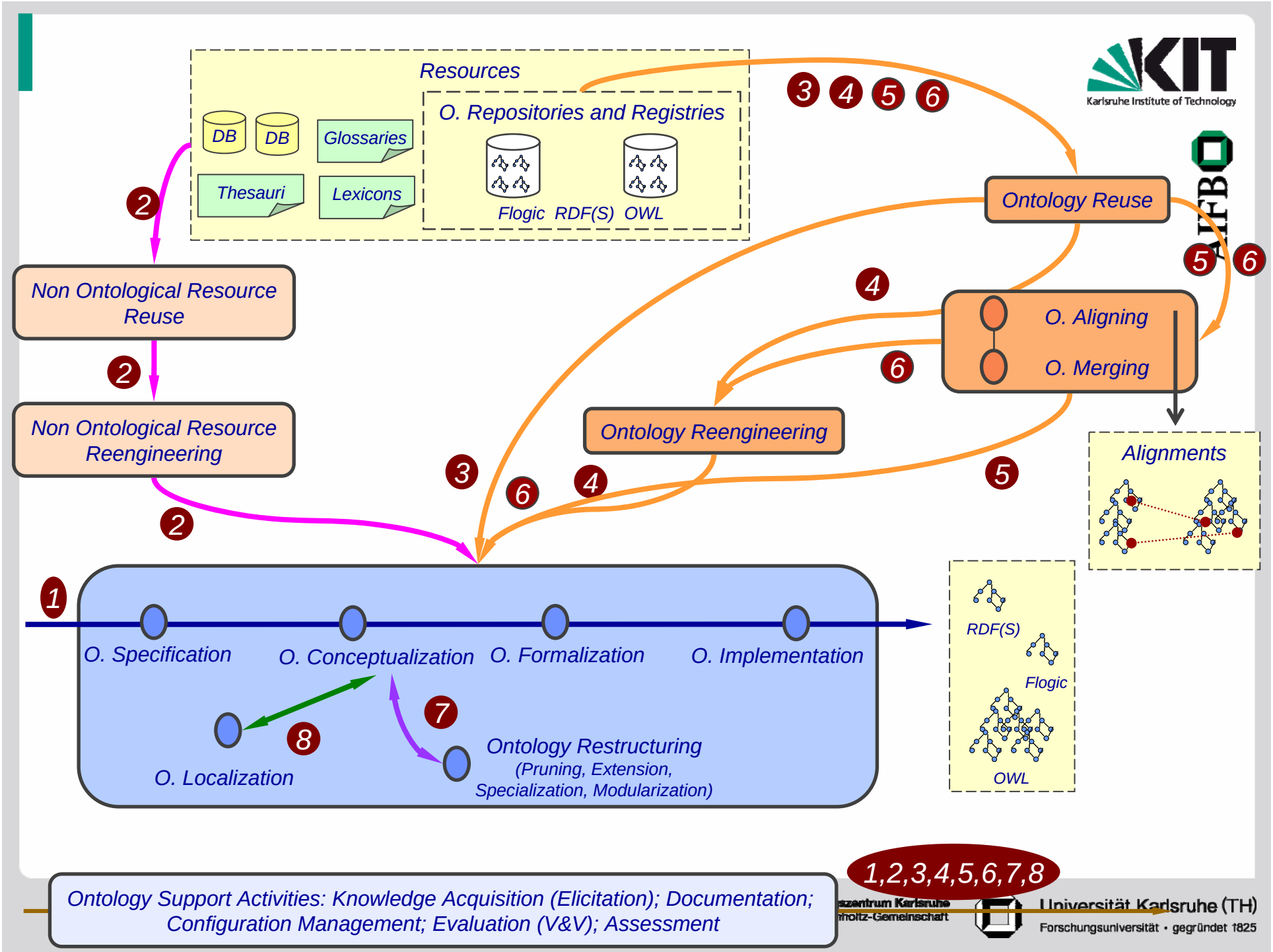
Scenarios

Karlsruhe Institute of Technology

AIFB

Building ontologies

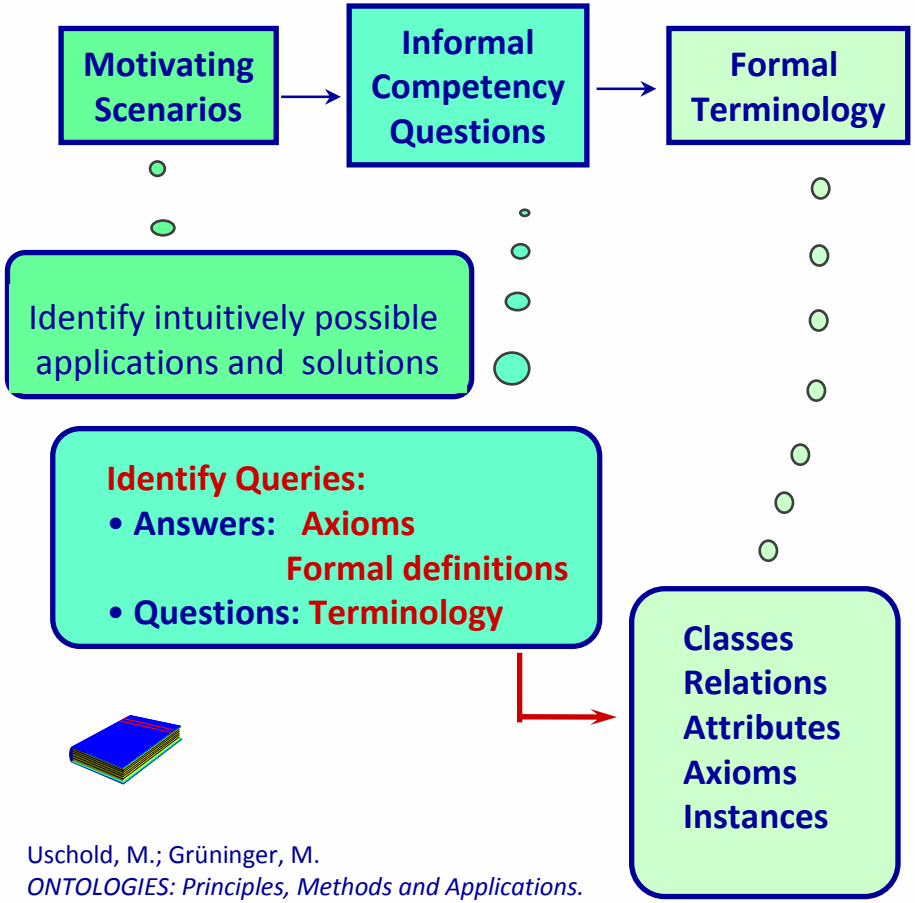
- from scratch without reusing existing resources
- by reusing non ontological resources
- by reusing ontologies
- by reusing and reengineering ontologies
- by reusing and merging ontology
- by reusing, merging and reengineering ontologies
- by restructuring ontologies or
- by localizing ontologies


Resources
DB
DB
Glossaries
Thesauri
Lexicons
O. Repositories and Registries
Flogic RDF(S) OWL
Ontology Reuse
Non Ontological Resource Reuse
Non Ontological Resource Reengineering
Ontology Reengineering
O. Aligning
O. Merging
Alignments
O. Specification
O. Conceptualization
O. Formalization
O. Implementation
O. Localization
Ontology Restructuring
(Pruning, Extension, Specialization, Modularization)
RDF(S)
Flogic
OWL
Ontology Support Activities: Knowledge Acquisition (Elicitation); Documentation; Configuration Management; Evaluation (V&V); Assessment
1,2,3,4,5,6,7,8

Ontology Specification

- Produce an Ontology Specification Document (OSD)
- Content
  - Purpose
  - Scenarios of use
  - Possible end users
  - Level of formality of the ontology
  - Scope
  - Granularity
- Technique
  - Competency Questions
- Output

Getting Terminology using Competency Questions



Uschold, M.; Grüninger, M.
ONTOLOGIES: Principles, Methods and Applications.
Knowledge Engineering Review.
Vol. 11; N. 2; June 1996.

Find documents written by Person P

Identify Queries:
- **Questions: Document, Person, writes**
- **Answers: Document D1 is written by P1**

Classes: Document, Person
Relations: Writes, written by
Attributes: ---
Axioms
Instances: P1, S1

Basic OWL Development Methodology

1. Build glossary of terms
   - Identify the entities one can encounter in the universe of discourse
   - Identify concepts that group these values.
   - Distinguish independent concepts from relationship-roles
2. Build a taxonomy of concepts.
   - Revisit this later and consider issues such as disjointness and covering for subconcepts.
3. Describe binary relations and attributes
   - Identify ‘properties’ of entities, and then identify general relationships in which objects participate.
4. „Formalize“ the ontology
   - E.g. determine local restrictions involving roles such as cardinality limits and value restrictions.
5. Describe instances
   - Identify any individuals that are of interest in all states of the world in this universe of discourse