

Ringvorlesung Bausteine für Semantic Web Anwendungen

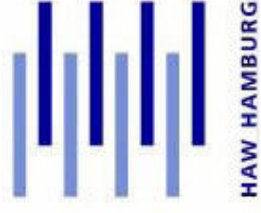
Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

Ziele des Vortrags



- Überblick der Technologien des Semantic Web
- Vorstellung Entwicklungstools
- Arbeit im Projekt
- Ausblick Master Thesis

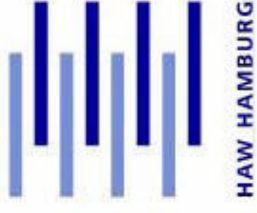
Idee des Semantic Web



- Daten mit Kontext → Information
- Information ist durch Maschinen
verarbeitbar
- Schlussfolgerungen auf Basis der
Informationen möglich → Wissen
generieren

Quelle: [1]

Semantic Web Initiative



Ziel ist die Entwicklung von Standards mit denen sich semantische Zusammenhänge im Web repräsentieren lassen und maschinell verarbeitet werden können.

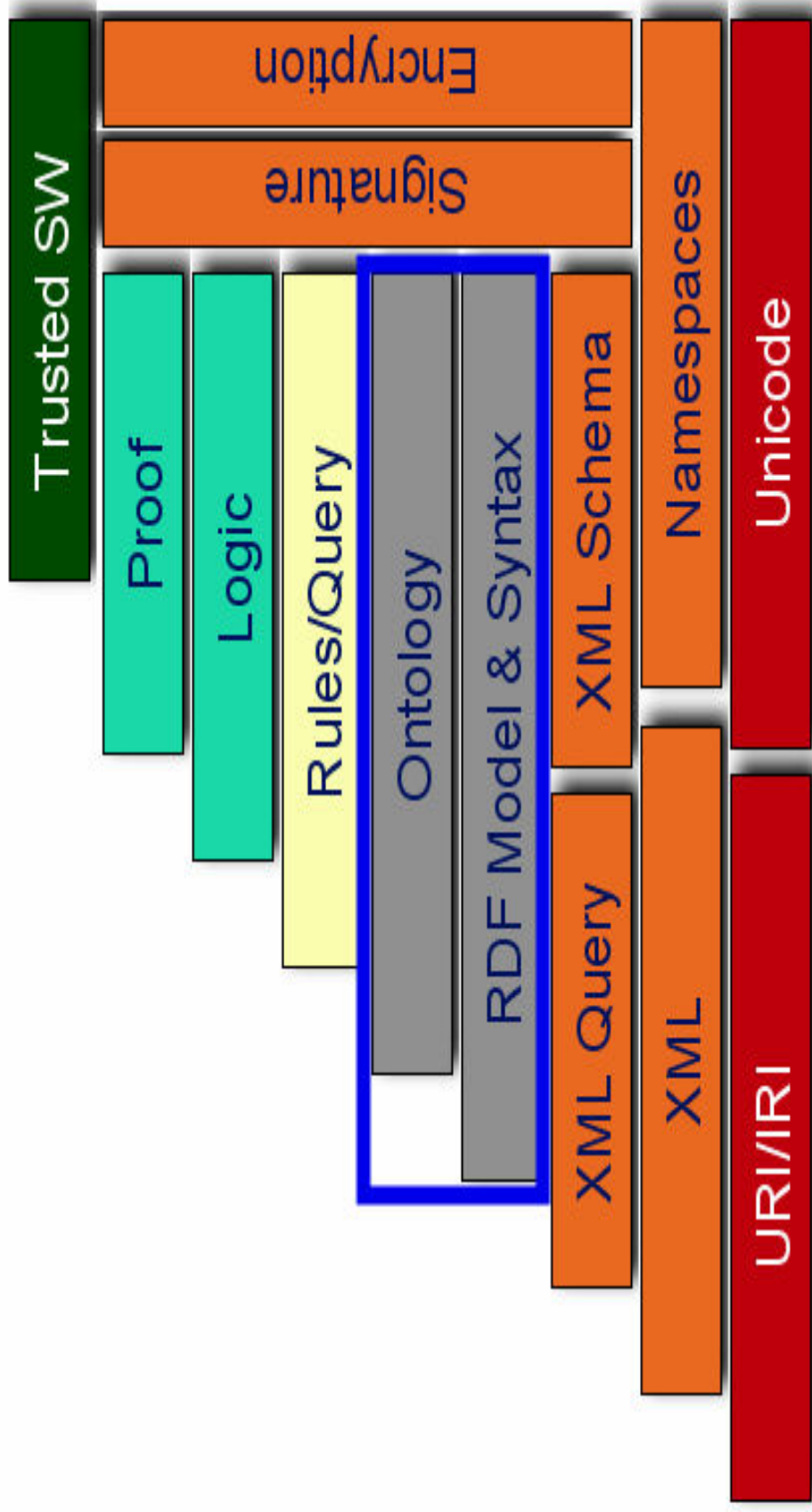
Wie weit sind die entsprechenden Technologien entwickelt ?

Quelle: [1]

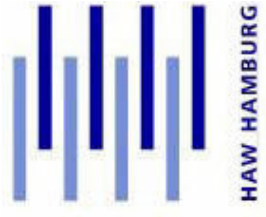
Gliederung

- Semantic Web Technologie Stack
 - RDF/Ontologiesprachen
 - RDF-Querysprachen/Regelsprachen
 - Reasoning/Inferenzmaschinen
- Entwicklungstools
 - Beispielarchitektur
 - Semantic Web Framework Jena 2
 - Ontologie Editor Protégé 3.2
 - DL Inferenzmaschine RacerPro
- Arbeit im Projekt
- Ausblick Masterarbeit

Semantic Web Stack



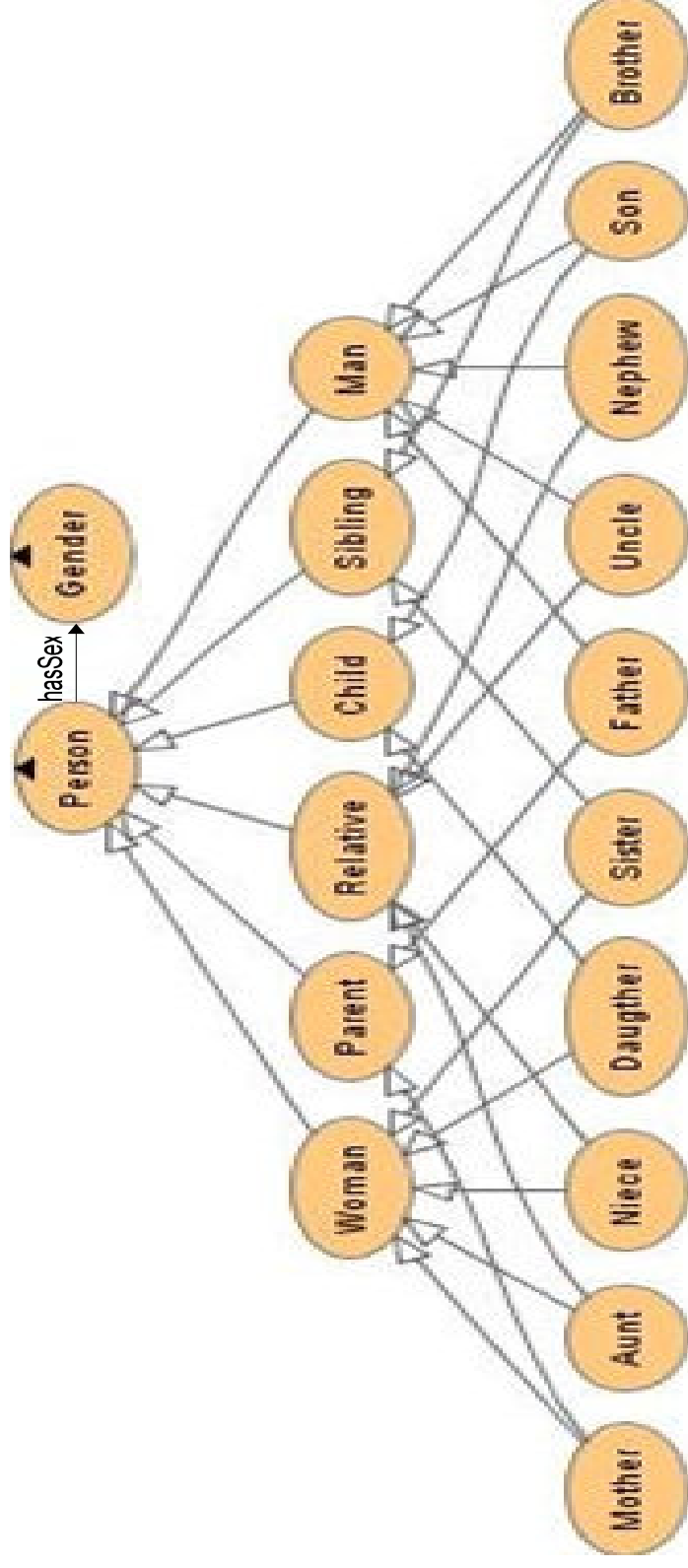
Beispielontologie Familie



- Beschreibt Personen im Kontext von Familienbeziehungen
- Personen haben ein Geschlecht (weiblich oder männlich)
- Eine instanziierte Person kann mehreren Konzepten (Klassen) angehören (sofern diese nicht disjunkt sind)

Quelle: [2]

Beispielontologie Familie (2)

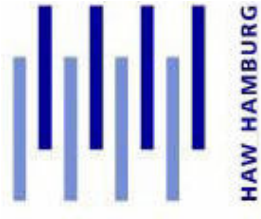


Quelle: [2]

RDF

- Metadatenmodell über Ressourcen
- Basis des Modells sind Aussagen über Ressourcen (Subjekte)
- Aussagen sind aufgebaut als Subjekt-Prädikat-Objekt Triple
- Aussagen bilden (gerichtete) Graphen
- Man kann Aussagen über Aussagen formulieren (Reification)

RDF: Beispiel



Aussage:

„Mary ist eine Frau“

RDF Statement in N3 Notation:

<<http://a.com/ontology#Mary>>

<<http://a.com/ontology#hasSex>>

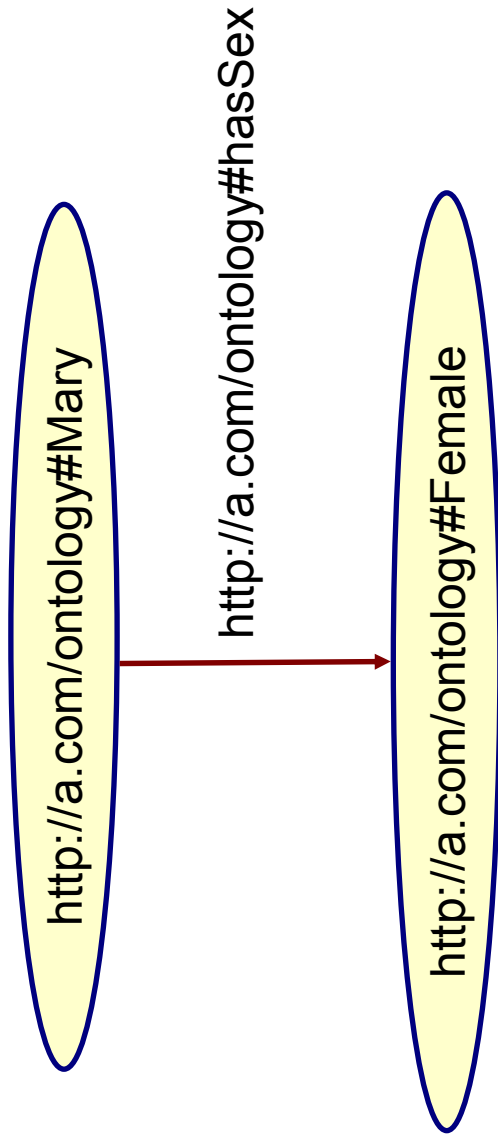
<<http://a.com/ontology#Female>>

← subject
← predicate
← object

1.1 RDF

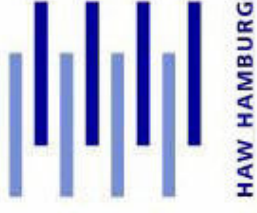
RDF Notation: Gerichteter Graph

RDF modelliert Statements mit Knoten und Pfeilen:



1.1 RDF

RDF Notationen: RDF/XML



RDF/XML Notation der gleichen Aussage:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:family="http://a.com/ontology#">
  <rdf:Description rdf:about="http://a.com/ontology#Mary">
    <family:hasSex rdf:resource="http://a.com/ontology#Female " />
  </rdf:Description>
</rdf:RDF>
```

Ontologiesprachen

- Semantische Modellierung der durch RDF beschriebenen Aussagen
- Mapping von Ontologien
- Bestehen aus Klassen, deren Eigenschaften und Relationen
- Instanz wird über `<rdf:type>` erzeugt
- Quasi Standards RDFS und OWL
- Dabei gilt:

RDF(S) < OWL Lite < OWL DL < OWL Full
„<“ = syntaktisch und semantisch enthalten

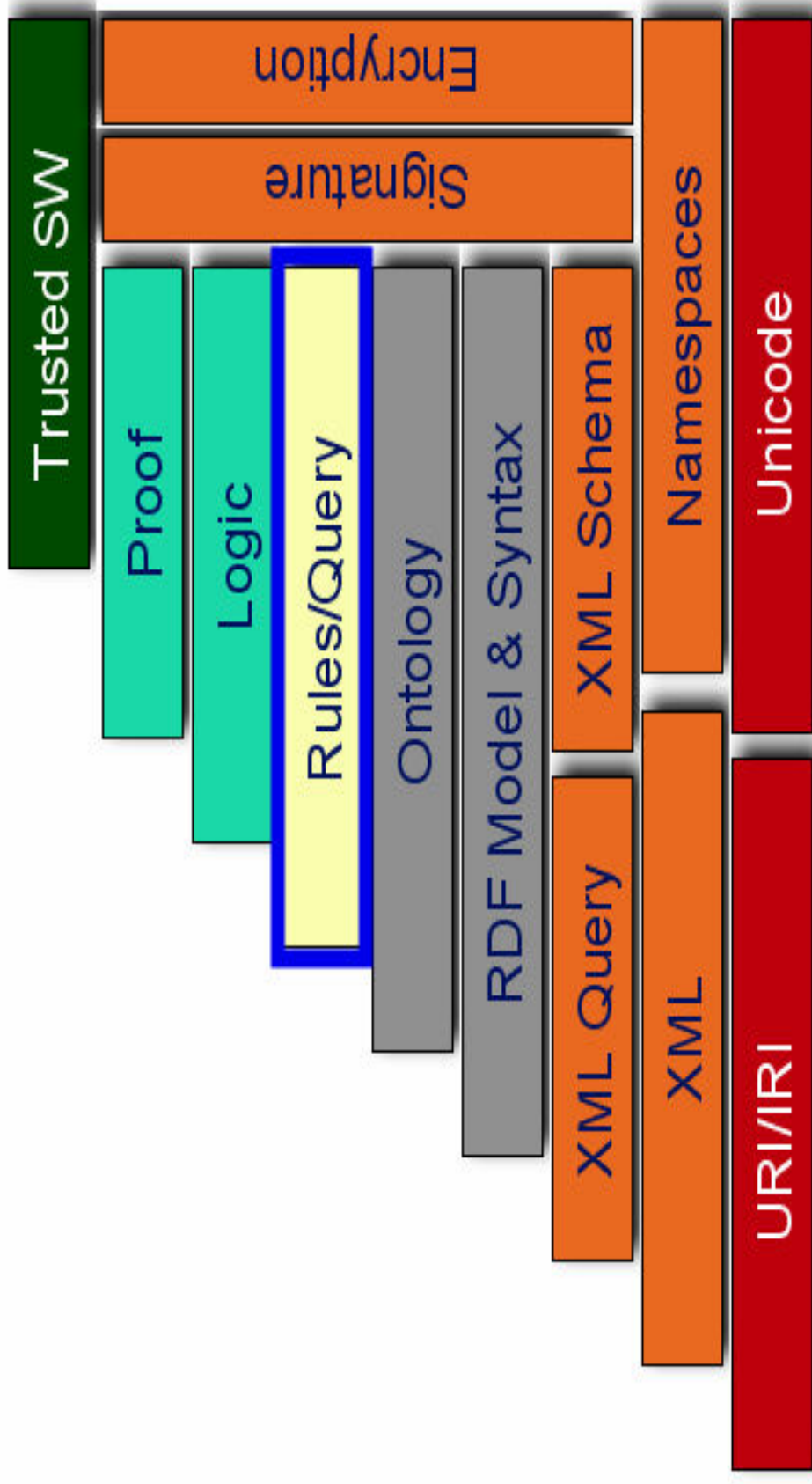
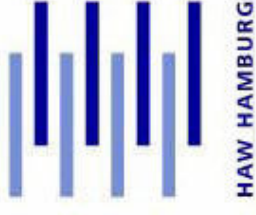
Fazit

- RDF Triples
 - Instanzen eines Wissensmodells
 - RDFS/OWL
 - Modellierung des Wissensmodells
-
- Technische Grundlage für Semantik
 - Formale Grundlage für logische Inferenz

Gliederung

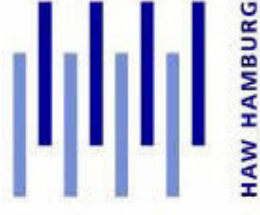
- Semantic Web Technologie Stack
 - RDF/Ontologiesprachen
 - RDF-Querysprachen/Regelsprachen
 - Reasoning/Inferenzmaschinen
- Entwicklungstools
 - Beispielarchitektur
 - Semantic Web Framework Jena 2
 - Ontologie Editor Protégé 3.2
 - DL Inferenzmaschine RacerPro
- Arbeit im Projekt
- Ausblick Masterarbeit

Semantic Web Stack



Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

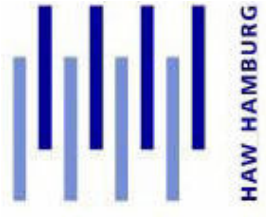
RDF Query Sprachen



Funktion:

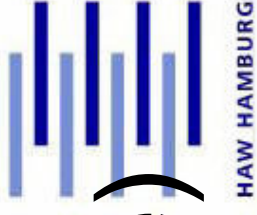
- Abfrage von RDF-Graphen
(Wissensmodell + Instanzen)
- Mehrere konkurrierende Vorschläge für
eine einheitliche Anfragesprache (RQL,
SPARQL, SeRQL, RDQL,...)

SPARQL



- SPARQL (Simple Protocol and RDF Query Language)
- W3C Working Draft
- Spezifiziert folgendes:
 - Eine Abfragesprache für RDF
 - Ein Protokoll für RDF
 - SPARQL Query Results in XML

SPARQL (Query Language)



Syntax:

Prologue (optional)

Query Result forms (required, pick 1)

BASE *<uri>*
PREFIX *prefix*: *<uri>* (repeatable)
SELECT (DISTINCT)sequence of *?variable*
SELECT (DISTINCT)*
DESCRIBE sequence of *?variable* or *<uri>*
DESCRIBE *
CONSTRUCT { *graph pattern* }
ASK

Query Dataset Sources (optional)

Add triples to the background graph (repeatable):

FROM *<uri>*
Add a named graph (repeatable):
FROM NAMED *<uri>*

Graph Pattern (optional, required for ASK)

WHERE { *graph pattern* [FILTER *expression*] }

Query Results Ordering (optional)

ORDER BY ...

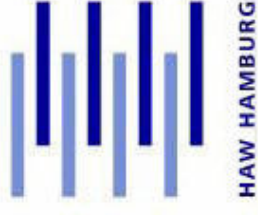
Query Results Selection (optional)

LIMIT *n*, OFFSET *m*

Quelle: [3]

Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

Arten von Queries

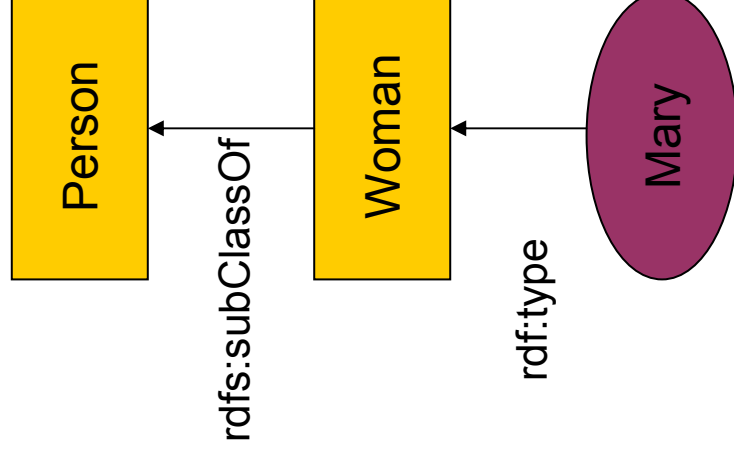


1. select, liefert ein Result Set für gültige Ersetzungen der Query Variablen (vergleichbar mit SQL)
Beispiel: *select ?a ?b where {?a rdf:type ?b}*
2. construct, konstruiert einen RDF Graphen basierend auf dem angegebenen Pattern
Beispiel: *CONSTRUCT { ?a foaf:knows ?b }
WHERE { ?a ex:KnowsQuiteWell ?b }*
3. describe, liefert einen RDF Graphen, der die angegebene resp. gesuchte Ressource beschreibt
Beispiel: *DESCRIBE ?person
WHERE { ?person foaf:name "Dave" }*
4. ask, true falls mindestens eine gültige Ersetzung für das Graph Pattern vorhanden ist
Beispiel: *ASK WHERE { ?a rdf:type foaf:Person }*

Einfacher select Query

Einfaches Beispiel eines RDF Graphen:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix : <xmlns:family="http://a.com/ontology/family.owl#">
:Person rdf:type owl:Class
:Woman rdf:type owl:Class
:Woman rdfs:subClassOf :Person
:Mary rdf:type :Woman
```



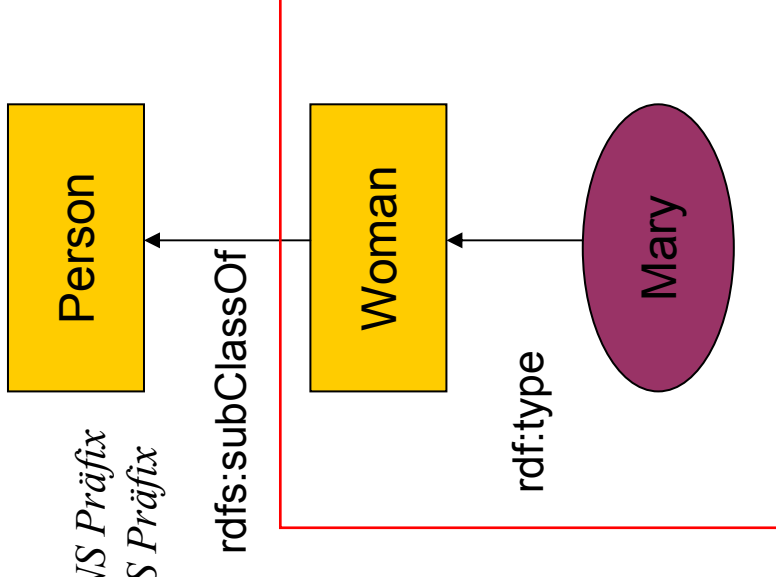
Einfacher select Query

Select Query über den Graphen:

```
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> → NS Präfix  
prefix : <xmlns:family="http://a.com/ontology/family.owl#"> → NS Präfix  
select ?c → Query Variable  
where { :Mary rdf:type ?c } → Graph pattern
```

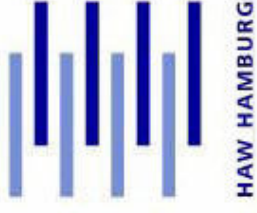
Resultat:

$c \rightarrow \text{:Woman}$



Subgraph match:
c wird an „Woman“ gebunden

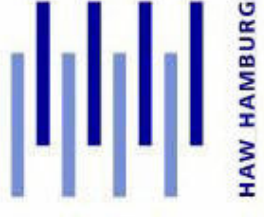
Regelsprachen



Aufgaben:

- Erstellung expliziter Regeln
- Verbesserung OWL Ausdrucksmächtigkeit
- Regeln drücken Zusammenhänge aus, die in OWL nicht modellierbar sind

SWRL



- Semantic Web Rule Language
- Kombiniert OWL DL und OWL Lite mit Datalog RuleML
- *SWRL = OWL DL/OWL Lite + Datalog RuleML*
- Bisher kein W3C Standard (Member Submission)
- Eventuelles Input in W3C Prozess

Quelle: [4]

Beispielregel $\text{hasUncle}(?x, ?z)$

Ein Onkel ist der Bruder eines Elternteils.

Regel:

Kopf	Rumpf
$\text{hasUncle}(?x, ?z) \leftarrow$	$\text{hasParent}(?x, ?y) \wedge \text{hasBrother}(?y, ?z)$

1.2 Regelsprachen

SWRL mit Protégé

family Protégé 3.2 beta (file: C:\Programme\Protégé_3.2_beta\family.pprj, OWL / RDF Files)

File Edit Project OWL Code Tools Window Help

OWL Classes Properties Forms Individuals Metadata SWRL Rules OWL Viz

SWRL Rules

Name	Expression
Def-hasAunt	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{hasSister}(?y, ?z) \rightarrow \text{hasAunt}(?x, ?z)$
Def-hasBrother	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasBrother}(?x, ?y)$
Def-hasDaughter	$\rightarrow \text{hasChild}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasDaughter}(?x, ?y)$
Def-hasFather	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasFather}(?x, ?y)$
Def-hasMother	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasMother}(?x, ?y)$
Def-hasNephew	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{hasSon}(?y, ?z) \rightarrow \text{hasNephew}(?x, ?z)$
Def-hasNiece	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{hasDaughter}(?y, ?z) \rightarrow \text{hasNiece}(?x, ?z)$
Def-hasParent	$\rightarrow \text{hasConsort}(?y, ?z) \wedge \text{hasParent}(?x, ?y) \rightarrow \text{hasParent}(?x, ?z)$
Def-hasSibling	$\rightarrow \text{hasChild}(?y, ?x) \wedge \text{hasChild}(?y, ?z) \wedge \text{differentFrom}(?x, ?z) \rightarrow \text{hasSibling}(?x, ?z)$
Def-hasSister	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasSister}(?x, ?y)$
Def-hasSon	$\rightarrow \text{hasChild}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasSon}(?x, ?y)$
Def-hasUncle	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{hasBrother}(?y, ?z) \rightarrow \text{hasUncle}(?x, ?z)$

Query

```
SELECT ?Name ?Gender
WHERE ( ?subject hasSex ?Gender . ?subject :name ?Name
```

Execute Query

SPARQL

Results

Name	Gender
Surrey	Female
Tom	Male
Marilyn	Female
Jimmy	Male
Elizabeth	Female
Ronald	Male

Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

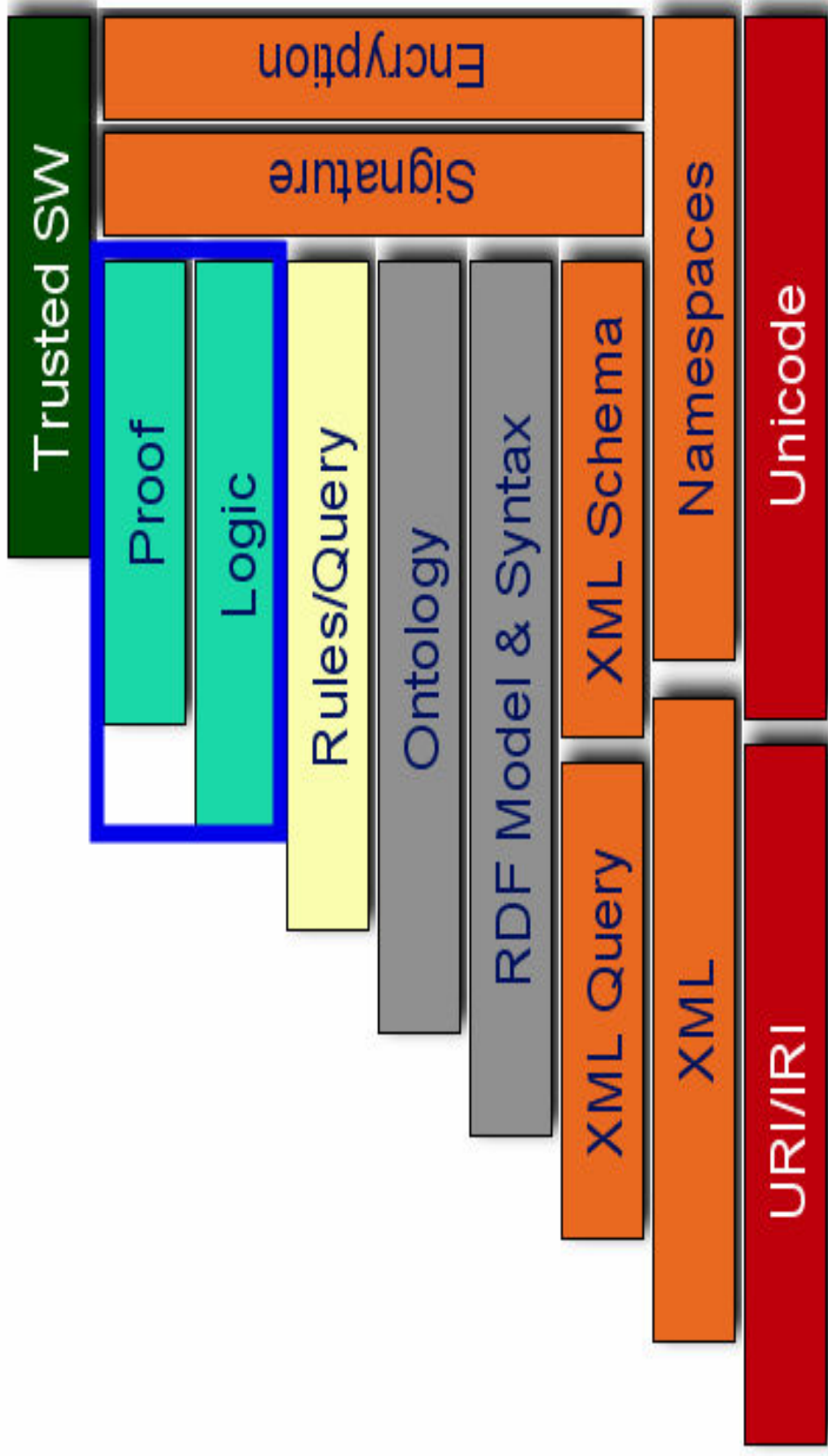
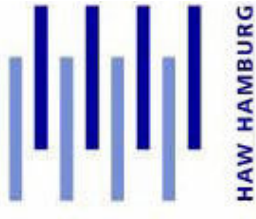
Fazit

- SWRL erhöht die Ausdrucksmächtigkeit von OWL
- Damit aber auch die Komplexität
- Derzeit unterstützen Inferenzmaschinen keine Regelsprachen

Gliederung

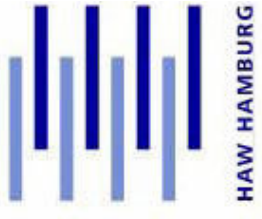
- Semantic Web Technologie Stack
 - RDF/Ontologiesprachen
 - RDF-Querysprachen/Regelsprachen
 - Reasoning/Inferenzmaschinen
- Entwicklungstools
 - Beispielarchitektur
 - Semantic Web Framework Jena 2
 - Ontologie Editor Protégé 3.2
 - DL Inferenzmaschine RacerPro
- Arbeit im Projekt
- Ausblick Masterarbeit

Semantic Web Stack



Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

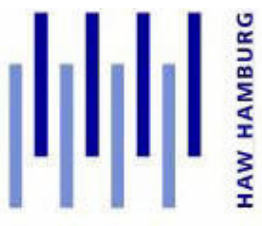
Reasoning



Aufgaben:

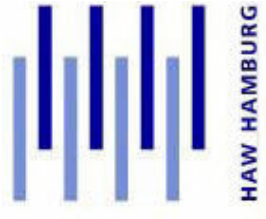
- Abgeleitete Bedingungen ermitteln → neues Wissen
- Konsistenz gewährleisten
- Klassifikation
- Äquivalenzen ermitteln

Inferenzmaschinen



- Komplexität von Inferenzmaschinen
 - Higher Order Logic
 - Full First Order Logic (Prädikatenlogik)
 - Description Logic
 - Logic Programming (Horn Logic)
 - Generiertes Wissen als „virtuelle“ Triples
 - Abfrage über RDF Queries
 - Inferenzmaschinen arbeiten meist mit Description Logic (RacerPro, Pellet, FaCT++)
- } Nicht entscheidbar

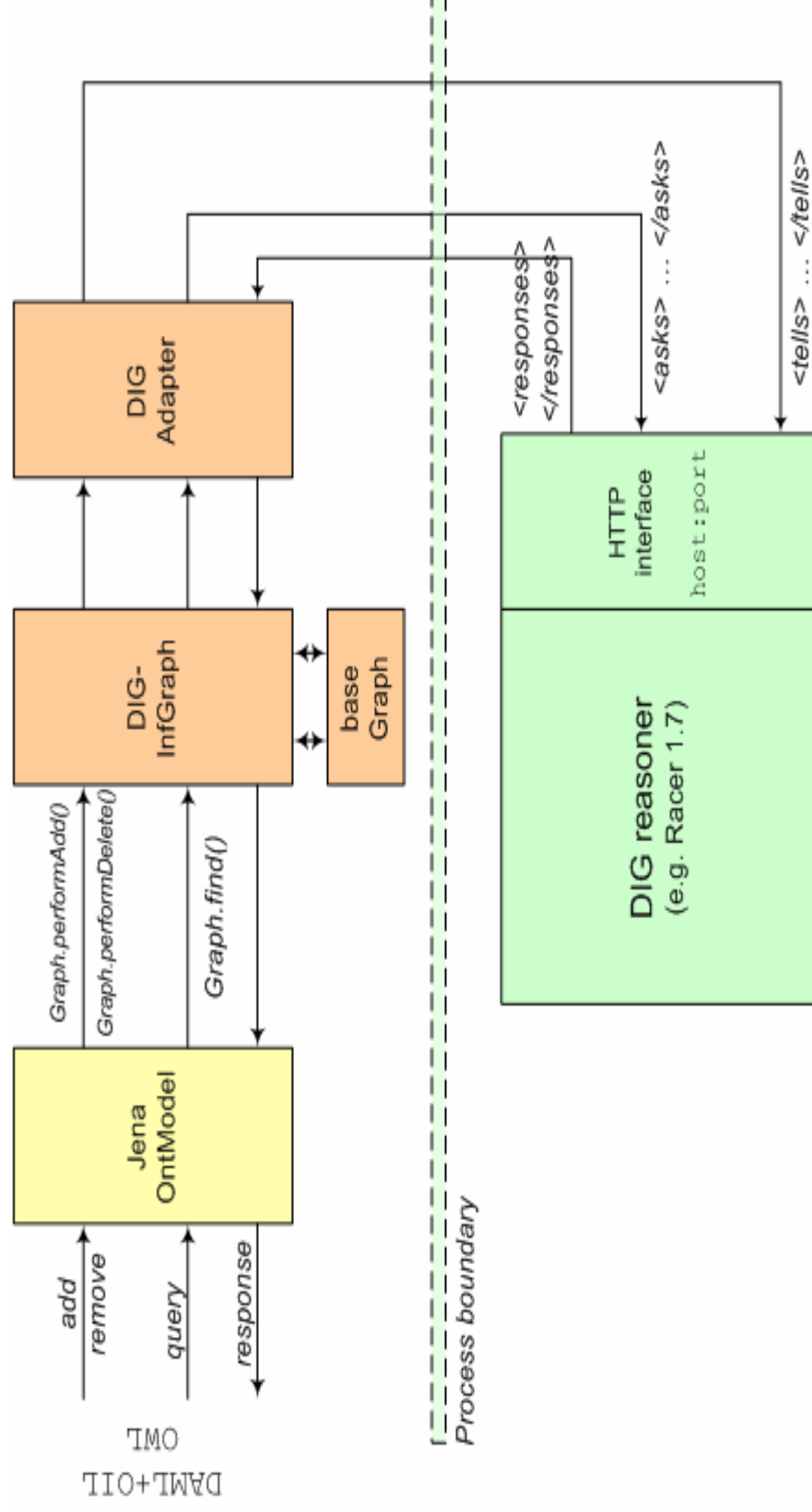
DIG Description Logic Interface



- Interface für die Anbindung von DL Inferenzmaschinen
- DL Inferenzmaschinen unterstützen DIG
- DIG basiert auf HTTP und XML
- DIG = DL Implementation Group

1.3 Inferenzmaschinen

Beispiel Jena and RACER via DIG



Quelle: [5]

Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

Tell Funktion

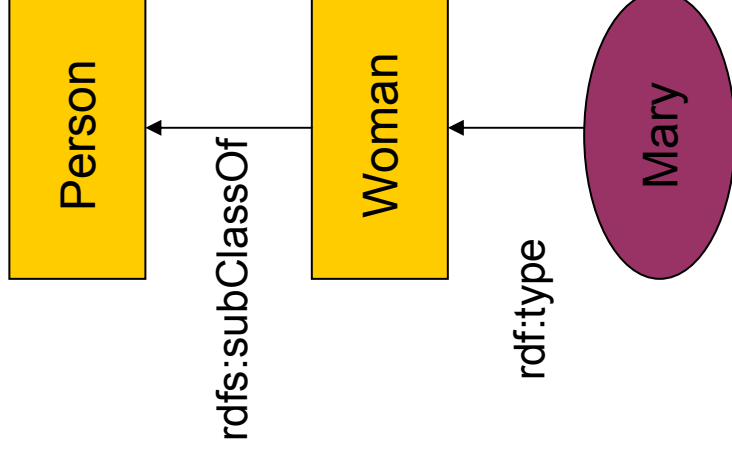
```
<tells>
  <defconcept name="Woman"/>
<equalc>
  <catom name="Woman"/>
<and>
  <some>
    <ratom name="hasSex"/>
    <iset>
      <individual name="Female"/>
    </iset>
  </some>
  <catom name="Person"/>
</and>
</equalc>
<defconcept name="Person"/>
<equalc>
  <catom name="Person"/>
<and>
  <or>
    <catom name="Man"/>
    <catom name="Woman"/>
  </or>
</and>
</equalc>
</tells>
```

Quelle: [6]

Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

Ask Funktion

```
<asks>
<satisfiable id="q1">
  <catom name="Person"/>
</satisfiable>
<descendants id="q2">
  <and>
    <some>
      <ratom name="hasSex"/>
    <iset>
      <individual name="Female"/>
    </iset>
    </some>
  <catom name="Person"/>
</and>
</descendants>
<types id="q3">
  <individual name="Mary"></individual>
</types>
</asks>
```



Fazit

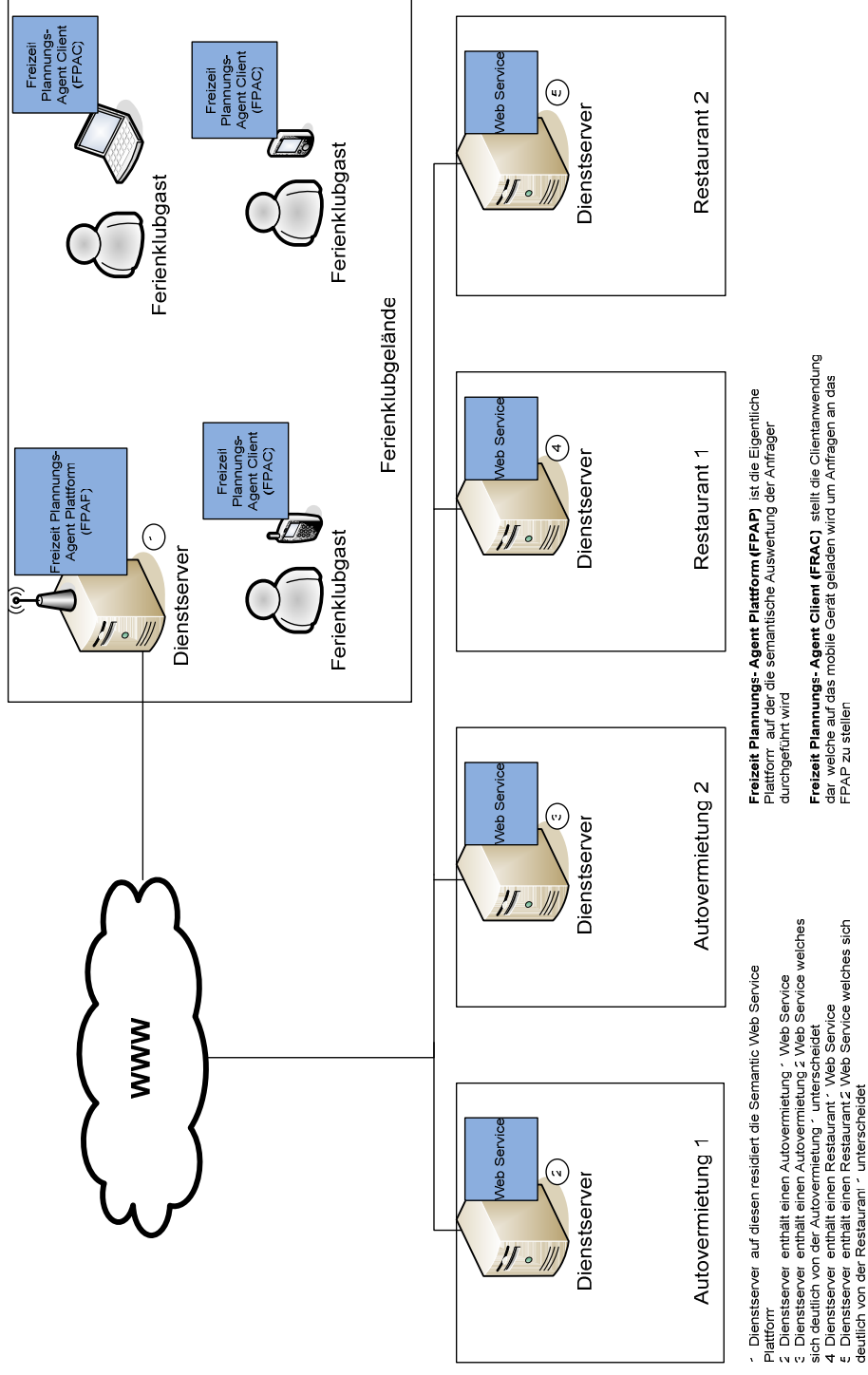
- Es gibt mehrere erprobte Inferenzmaschinen
- Meist arbeiten sie auf Basis von DL
- Über DIG Interface können Inferenzmaschinen an Semantic Web Applikationen angebunden werden

Gliederung

- Semantic Web Technologie Stack
 - RDF/Ontologiesprachen
 - RDF-Querysprachen/Regelsprachen
 - Reasoning/Inferenzmaschinen
- **Entwicklungstools**
 - **Beispielarchitektur**
 - Semantic Web Framework Jena 2
 - Ontologie Editor Protégé 3.2
 - DL Inferenzmaschine RacerPro
- Arbeit im Projekt
- Ausblick Masterarbeit

2.1 Beispielarchitektur

Problemstellung Projekt

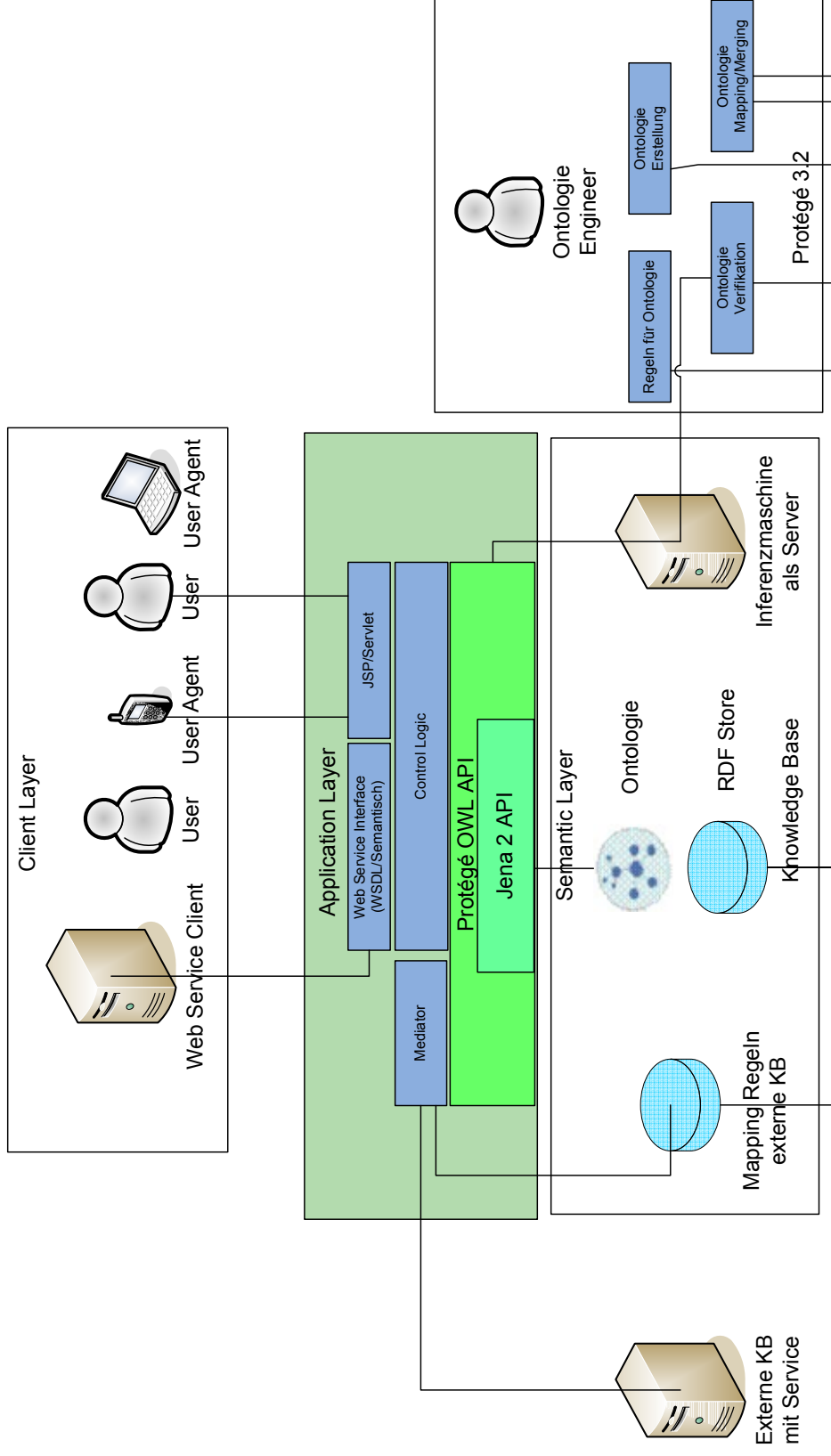


Quelle: [7]

Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

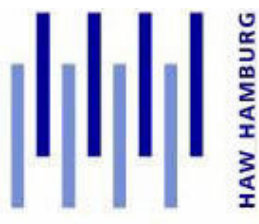
2.1 Beispielarchitektur

Beispielarchitektur für Projektproblematik



Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

Hauptentwicklungstasks



Im Projekt:

- Ontologie Engineering
 - Wissensextraktion und -repräsentation
- Ontologie Mapping

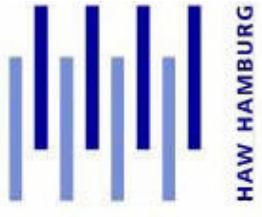
In der Realität bei großen Datenmengen:

- Annotation von Ressourcen
 - Manuell
 - (Halb)automatisch

Gliederung

- Semantic Web Technologie Stack
 - RDF/Ontologiesprachen
 - RDF-Querysprachen/Regelsprachen
 - Reasoning/Inferenzmaschinen
- **Entwicklungstools**
 - Beispielarchitektur
 - **Semantic Web Framework Jena 2**
 - Ontologie Editor Protégé 3.2
 - DL Inferenzmaschine RacerPro
- Arbeit im Projekt
- Ausblick Masterarbeit

Jena 2.3

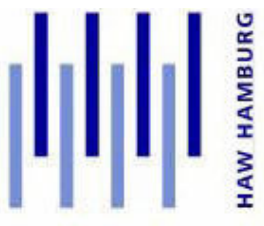


- Programmiertool für Semantic Web Anwendungen
- Ursprünglich an den HP Labs Semantic Web Research entwickelt
- Open source gehostet bei Sourceforge
- Basiert auf Java

Quelle: [8]

Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

Jena 2.3: Features



- RDF API
- OWL API
- SPARQL Implementation (ARQ)
- RDQL Implementation
- Eigene Inferenzmaschine (OWL Lite)
- DIG Interface implementiert
- Model Storage (In Memory und Persistent)

Gliederung

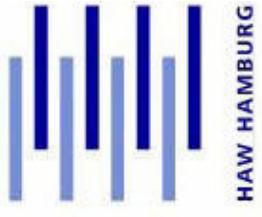
- Semantic Web Technologie Stack
 - RDF/Ontologiesprachen
 - RDF-Querysprachen/Regelsprachen
 - Reasoning/Inferenzmaschinen
- **Entwicklungstools**
 - Beispielarchitektur
 - Semantic Web Framework Jena 2
 - **Ontologie Editor Protégé 3.2**
 - DL Inferenzmaschine RacerPro
- Arbeit im Projekt
- Ausblick Masterarbeit

Protégé 3.2

- Ontologieerstellungstool
 - Erstellung von Ontologien/Instanzen
 - Mapping von Ontologien
 - Erstellen von Queries
 - Plugin-Erweiterbarkeit
- Open Source
- Java Anwendung
- Leicht benutzbare GUI
- Schnittstellen zu Inferenzmaschinen (RACER)
via DIG

Quelle: [9]

Protégé 3.2 [2]

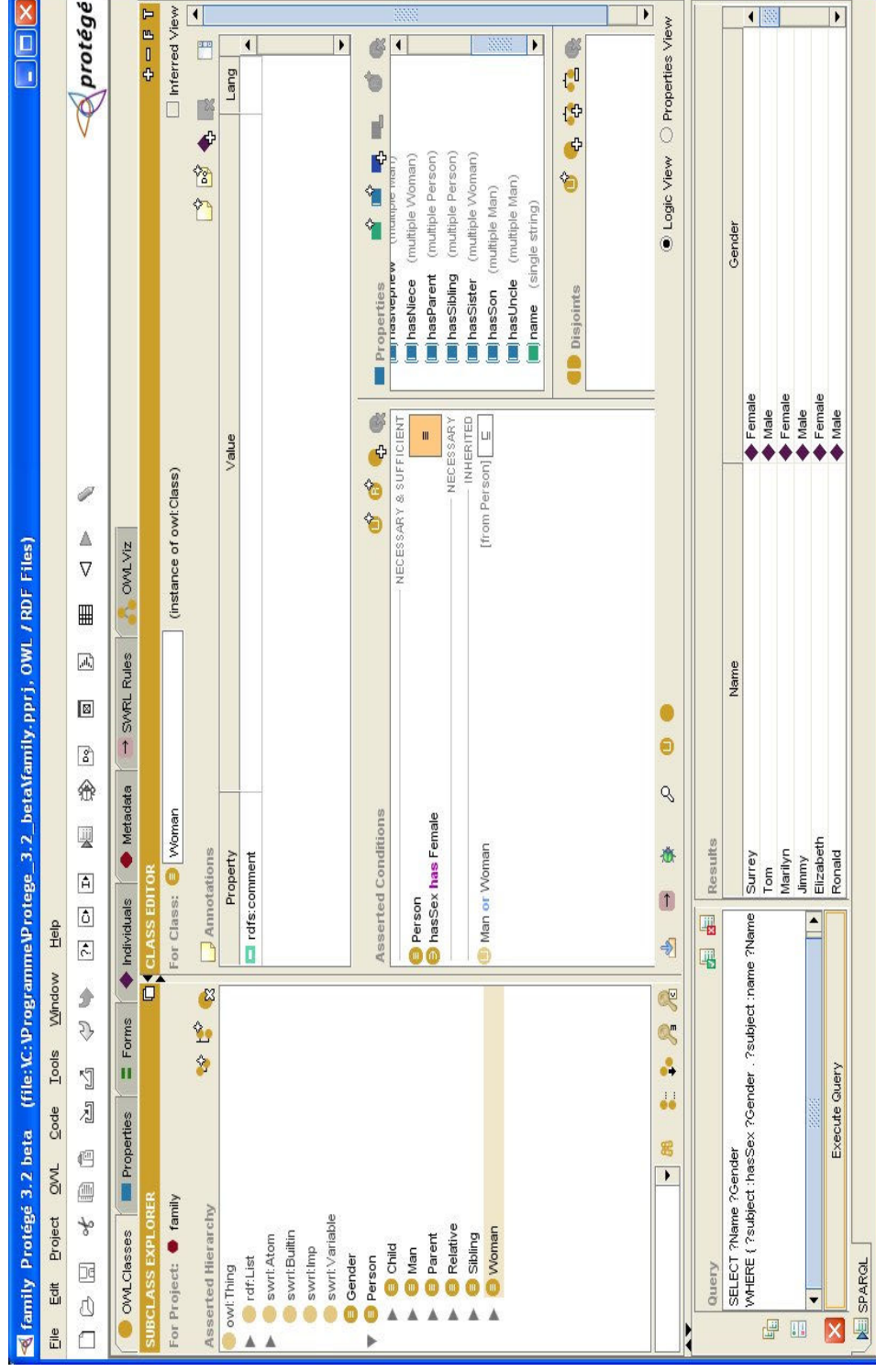


Bedeutende Plugins:

- Protégé-OWL Plugin
- Prompt (Ontologie merging)
- nRQL Plugin (Plugin für **new RACER Query Language**)
- OntoViz (grafische Darstellung von Ontologien)
- OWL-S Editor
- Jess

2.3 Protégé 3.2

Demo Protege 3.2 beta



The screenshot displays the Protégé 3.2 beta interface with the following components:

- Top Panel:** File, Edit, Project, OWL, Code, Tools, Window, Help.
- Left Panel:** OWL Classes Explorer showing a hierarchy for 'family' (owl:Thing, rdf:List, swrt:Atom, swrt:BuiltIn, swrt:Imp, swrt:Variable, Gender, Person, Child, Man, Parent, Relative, Sibling, Woman).
- CLASS EDITOR:** For Project: family. For Class: Woman. Annotations: rdfs:comment. Value: (Instance of owl:Class).
- ASSERTED CONDITIONS:** Person hasSex has Female. Man or Woman. Necessary & Sufficient conditions: NECESSARY, INHERITED, [from Person].
- Properties:** hasNiece (multiple Woman), hasParent (multiple Person), hasSibling (multiple Person), hasSister (multiple Woman), hasSon (multiple Man), hasUncle (multiple Man), name (single string).
- RESULTS:** A table showing results for a SPARQL query.

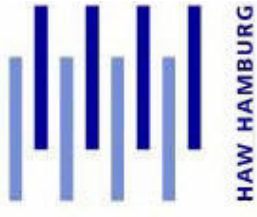
Name	Gender
Surrey	Female
Tom	Male
Marilyn	Female
Jimmy	Male
Elizabeth	Female
Ronald	Male

Query: SELECT ?Name ?Gender WHERE { ?subject hasSex ?Gender . ?subject :name ?Name }

Execute Query

Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

Fazit Protégé 3.2



- Unterstützt die standardisierten Technologien für Semantic Web
- Unterstützt die populärsten der nicht standardisierten Technologien
- Implementiert DIG Schnittstelle
- Stellt eine komplette Entwicklungsumgebung für Ontologien dar
- Liefert API's für die Programmierung von Semantic Web Anwendungen

Gliederung

- Semantic Web Technologie Stack
 - RDF/Ontologiesprachen
 - RDF-Querysprachen/Regelsprachen
 - Reasoning/Inferenzmaschinen
- **Entwicklungstools**
 - Beispielarchitektur
 - Semantic Web Framework Jena 2
 - Ontologie Editor Protégé 3.2
 - **DL Inferenzmaschine RacerPro**
- Arbeit im Projekt
- Ausblick Masterarbeit

RacerPro

- Renamed ABox and Concept Expression Reasoner Pro
- Basiert auf der Beschreibungslogik *SHIQ*
- Unterstützt RDF, DAML-OIL, OWL (DL)
- Implementiert DIG-Interface
- Ursprünglich an der TUHH entwickelt
- Inzwischen kommerziell

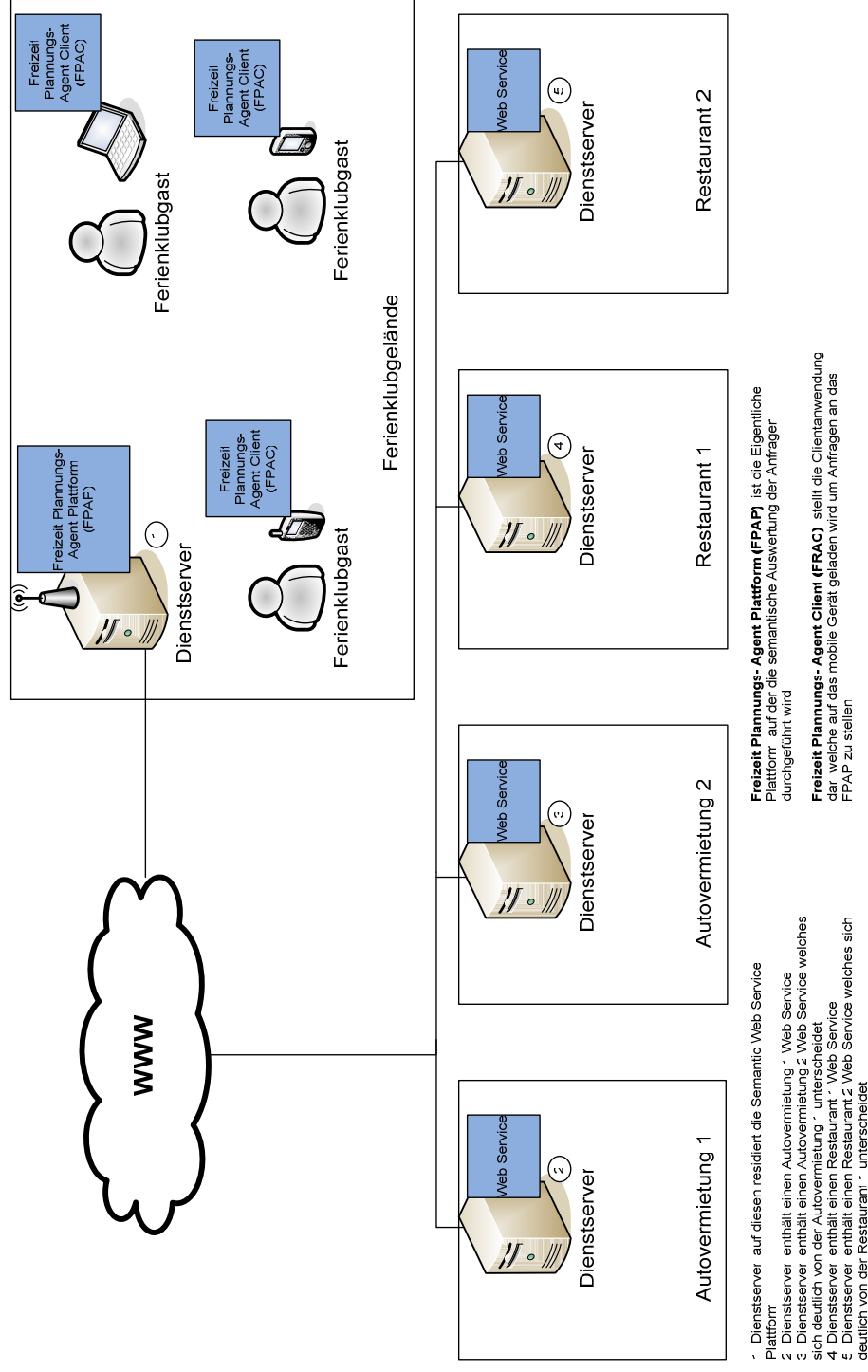
Quelle: [10]

Gliederung

- Semantic Web Technologie Stack
 - RDF/Ontologiesprachen
 - RDF-Querysprachen/Regelsprachen
 - Reasoning/Inferenzmaschinen
- Entwicklungstools
 - Beispielarchitektur
 - Semantic Web Framework Jena 2
 - Ontologie Editor Protégé 3.2
 - DL Inferenzmaschine RacerPro
- **Arbeit im Projekt**
- Ausblick Masterarbeit

3. Arbeit im Projekt

Arbeit im Projekt

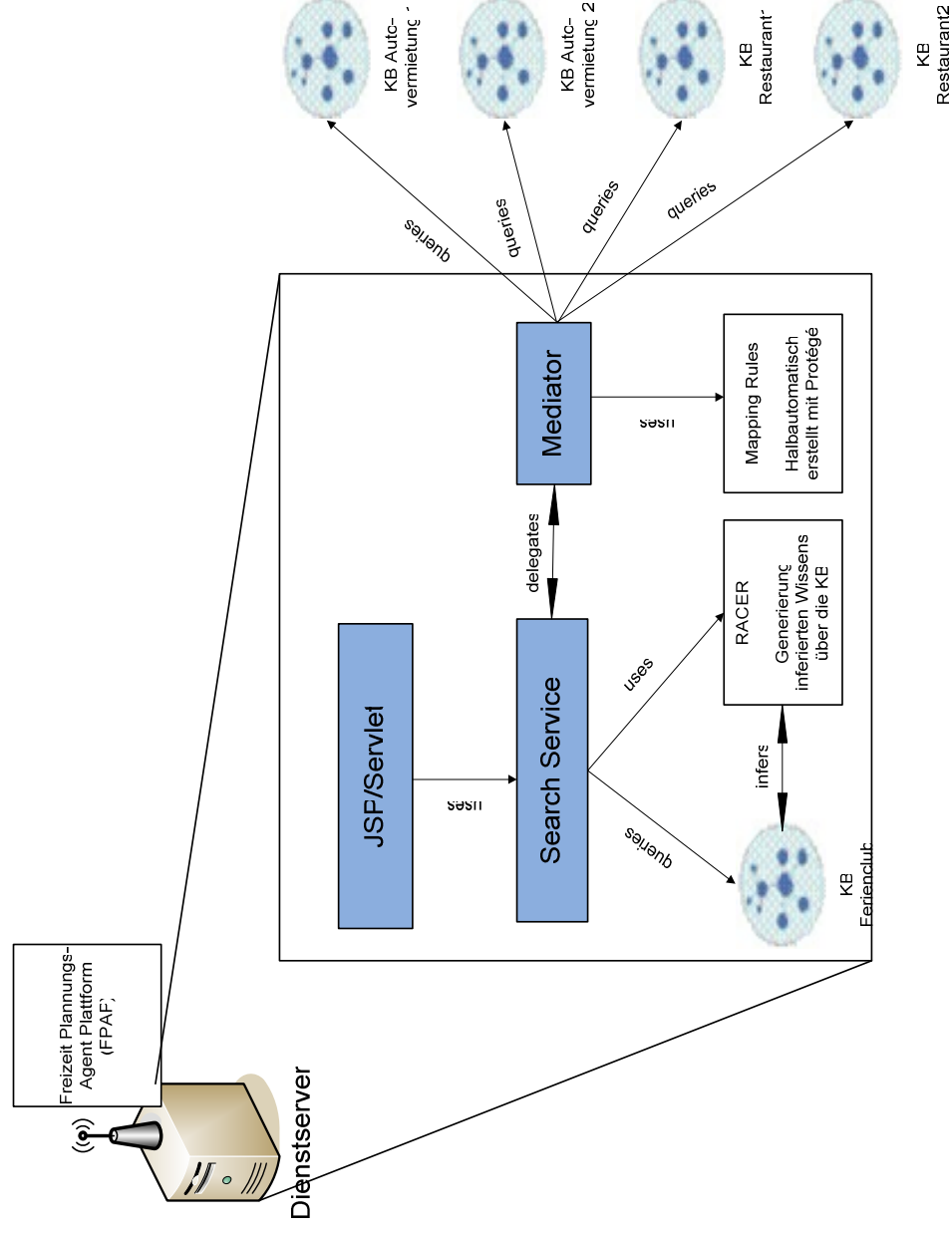


Quelle: [7]

Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

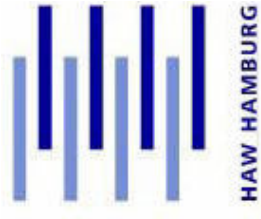
3. Arbeit im Projekt

Arbeit im Projekt (2)



Ringvorlesung Vortrag Gerrit Diederichs
Bausteine für Semantic Web Anwendungen

Arbeit im Projekt (3)



- Entwicklung von Ontologien für die verschiedenen Dienste
- Mapping der Ontologien
- Anbindung einer Inferenzmaschine (RacerPro)
- Informationsportal für Clubbesucher, das auf der Semantic Web Infrastruktur aufsetzt

Gliederung

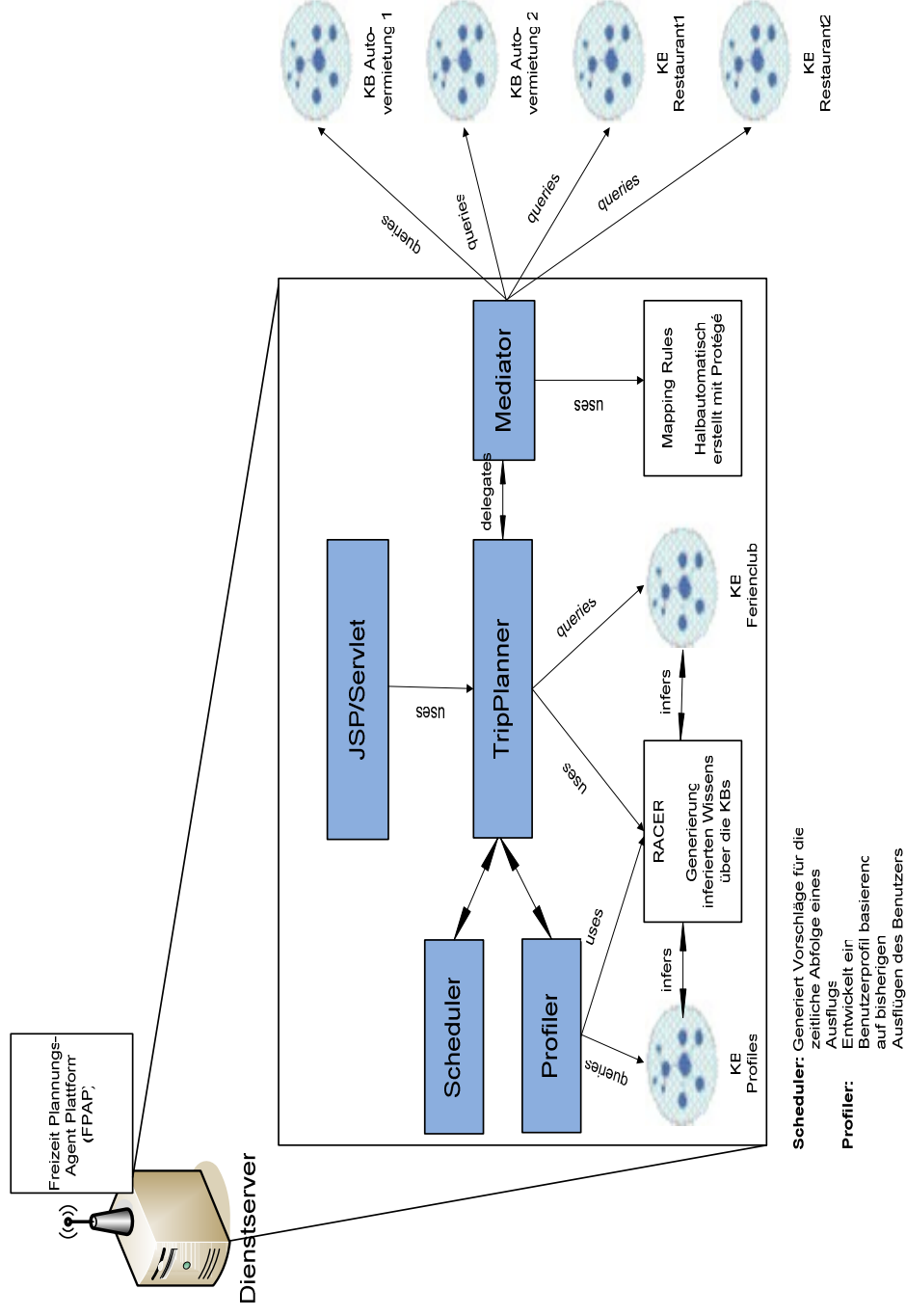
- Semantic Web Technologie Stack
 - RDF/Ontologiesprachen
 - RDF-Querysprachen/Regelsprachen
 - Reasoning/Inferenzmaschinen
- Entwicklungstools
 - Beispielarchitektur
 - Semantic Web Framework Jena 2
 - Ontologie Editor Protégé 3.2
 - DL Inferenzmaschine RacerPro
- Arbeit im Projekt
- **Ausblick Masterarbeit**

Ausblick Masterarbeit

- Fortsetzung der Arbeit aus dem Projekt
- Erweiterung der semantischen Infrastruktur um Benutzerprofile, Scheduling
- Entwicklung eines Ausflugsplaners, der auf personalisierten Angeboten beruht und auf der semantischen Infrastruktur aufsetzt
- Angelehnt an „Granite Nights“

4. Ausblick Masterarbeit

Ausblick Masterarbeit (2)

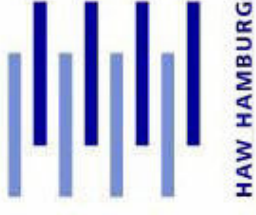


Quellen

Links:

- [1] Projekt Semantic Web (WS 2005/6) http://www.inf.fu-berlin.de/inst/ag-nbi/lehre/0506/P_SW/fohlen/Einleitung.pdf
- [2] Protégé OWL Plugin – SWRL Editor; <http://protege.stanford.edu/plugins/owl/swrl/>
- [3] SPARQL Reference 1.7; <http://www.dajobe.org/2005/04-sparql/SPARQLreference-1.7.pdf>
- [4] Regelsprachen – Seminarvortrag FU Berlin
http://www.inf.fu-berlin.de/inst/ag-nbi/lehre/05/S_MOD/Regelsprachen_180505.pdf
- [5] HOWTO use Jena with an external DIG reasoner; <http://jena.sourceforge.net/how-to/dig-reasoner.html>
- [6] The DIG Description Logic Interface: DIG/1.1; <http://dl-web.man.ac.uk/dig/2003/02/interface.pdf>
- [7] Piotr Wendt: Semantic Web Services Discovery – Seminarvortrag HAW Hamburg;
<http://users.informatik.haw-hamburg.de/~ubicompp/projekte/master05-06/wendt/slides.pdf>
- [8] Jena – A Semantic Web Framework for Java; <http://jena.sourceforge.net/>
- [9] The Protégé Ontology Editor and Knowledge Acquisition System; <http://protege.stanford.edu/>
- [10] Keno Selzer: Entwicklung und Evaluierung einer Test-Infrastruktur für den Racer-Server
Studienarbeit an der TU Hamburg-Harburg

Vielen Dank für die Aufmerksamkeit



Fragen ?