Wissensnetzwerke im Grid (WisNetGrid) – **Prototypische Implementierung des Schlussfolgerungs- und Vermittlungsdienstes** –

Bericht D2.2.2 zu Arbeitspaket 2.2

Juni 2011

Martin Junghans, Steffen Stadmüller

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Ontologien haben sich als Werkzeug in Wissensumgebungen etabliert. Als formales Modell einer bestimmten Domäne fördern sie die Interoperabilität der Ressourcen, die sie beschreiben, und die Integration heterogener Quellen. Die im WisNetGrid entwickelten Ontologiedienste basieren auf einer gemeinsamen Modellierungssprache für Ontologien: OWL (Web Ontology Language).

Speicherungs- und Verzeichnisdienste für OWL-Ontologien wurden bereits im Laufe des Projekts entwickelt. Im Fokus dieses Berichtes sind der Schlussfolgerungs- und Vermittlungsdienst. Der Schlussfolgerungsdienst erlaubt es, Schlussfolgerungen aus Ontologien zu ziehen. D.h., weiteres Wissen aus dem in einer Ontologie enthaltenen Wissen abzuleiten. Der Vermittlungsdienst wird benötigt, um Heterogenitäten zwischen verschiedenen bereits vorhandenen, unabhängig voneinander erstellten Ontologien zu überbrücken.

In diesem Bericht werden die Prototypen dieser Komponenten dokumentiert. Es werden die Funktionalitäten, die Schnittstellen der entwickelten Komponenten sowie deren Verwendung in Abschnitt 2 aufgezeigt um eine Weiterentwicklung darauf aufbauender Komponenten im WisNetGrid zu ermöglichen und zu fördern. Technische Details zur Installation der Komponenten sind in Abschnitt 3 beschrieben.

Kapitel 2

Funktionalität des Prototyps

2.1 Ontologie-Verzeichnis

In diesem Abschnitt geben wir einen kurzen Überblick über die Funktionalität des Speicherungs- und Verzeichnisdienstes für Ontologien. Diese Komponente wurde im Rahmen des Meilensteins "D2.2.1 Prototypische Implementierung des Speicherungs- und Verzeichnisdienstes" entwickelt. Die in diesem Bericht vorgestellten Schlussfolgerungs- und Vermittlungsdienste bauen auf diese Basisfunktionalität auf.

Diese Funktionalität ist konzeptionell analog zum Speicherungs- und Verzeichnisdienst für Dienstbeschreibungen, welche im Bericht "D3.2.3 Dienste-verzeichnis" [AHM10] dokumentiert wurde. Die Implementierung des Ontologieverzeichnisses umfasst Methoden zum Erstellen, Lesen, Ändern sowie Löschen von Ontologien in der D-Grid Infrastruktur. Die Ontologien als auch deren Metadaten werden im D-Grid gespeichert. Dazu werden die vom AP1 des WisNetGrid entwickelten Schnittstellen zur Grid-Infrastruktur verwendet um Dateien zur Beschreibung der Ontologien und deren Metadaten im Grid zu modifizieren. Neben dieser Basisfunktionalität können die Ontologien von Nutzern exklusiv geblockt und freigegeben werden sowie über Änderungen an den Ontologien durch eine Email benachrichtigt werden.

Im einzelnen bietet der Speicherungs- und Verzeichnisdienst für Ontologien die in Listing 2.1 gezeigten Methoden an. Die Klasse `OntologyResource` kapselt die Ontologie mit dessen Metadaten in ein gemeinsames Ressourcen-

Listing 2.1: Schnittstelle des Ontologieverzeichnisses

```
1  package edu.kit.aifb.suprime.repository;
2
3  import java.net.URI;
4  import java.util.Set;
5  import edu.kit.aifb.suprime.modeling.ontology.OntologyResource;
6  import edu.kit.aifb.suprime.modeling.user.DGridUser;
7
8  public interface OntologyRepository {
9
10     URI create(OntologyResource ontologyResource);
11
12     URI create(OntologyResource ontologyResource, boolean lockOntology,
           DGridUser user);
13
14     Set<URI> read();
15
16     OntologyResource read(URI ontologyId);
17
18     OntologyResource read(URI ontologyId, boolean lockOntology, DGridUser
           user);
19
20     boolean update(OntologyResource ontologyResource, boolean lockOntology,
           DGridUser user);
21
22     boolean delete(URI ontology);
23
24     boolean subscribe(URI ontology, DGridUser user);
25 }
```

Objekt (in Analogie zu den Dienstbeschreibungsressourcen [JA11]). Beim Erstellen und Lesen der Ontologien kann für diese Ressourcen eine exklusive Änderungsberechtigung für einen gegebenen Nutzer `user` gewährt werden.

2.2 Ontologie-Schlussfolgerung

Schlussfolgern aus Ontologien bedeutet, dass weiteres Wissen aus dem in der Ontologie enthaltenen Wissen abgeleitet wird. Dazu wird der HermiT Reasoner[1] verwendet. Die Funktionalität zum Schlussfolgern aus gegebenen Ontologien wurde ebenfalls in die Ontologieverzeichnis-Komponente integriert. Die dafür bereitgestellten Methoden sind in Listing 2.2 sind daher eine Erweiterung der Klasse `OntologyRepository` von Listing 2.1. Es

[1] HermiT ist ein Reasoner, der für die Web Ontology Language (OWL) entwickelt wurde `http://hermit-reasoner.com`.

können die folgenden Schlüsse aus einer mit dem Parameter `ontologyId` spezifizierten Ontologie gefolgert werden.

- Abfrage der Sub-Klassen einer (unbenannten) komplexen Klasse (Zeile 13)
- Abfrage der äquivalenten Klassen einer komplexen Klasse (Zeile 15)
- Abfrage der Individuen von einer komplexen Klasse (Zeile 17)
- Abfrage der zu einem Individuum äquivalenten Individuen (Zeile 19)
- Abfrage der Individuen verschieden von einem gegeben Individuum (Zeile 21)
- Abfrage der Sub-Objekt-Relationen einer Objekt-Relation (Zeile 23)
- Abfrage der Sub-Datentyp-Relationen einer Datentyp-Relation (Zeile 25)
- Abfrage der Werte einer Objekt-Relationen eines Individuums (Zeile 27)
- Abfrage der Werte einer Datentyp-Relationen eines Individuums (Zeile 29)
- Überprüfung der Konsistenz einer Ontologie (Zeile 11)

Komplexe Klassen, die als Parameter `classExpr` übergeben werden, können entweder benannte Klassen der Ontologie sein oder in einem komplexen Ausdruck aus bestehenden Klassen der Ontologie komponiert werden. Sind die Klassen C und D in einer Ontologie explizit definiert, so können zum Beispiel die Sub-Klassen von C abgefragt werden. Es ist auch möglich die Sub-Klassen einer komplexe Klasse $\mathsf{C} \sqcup \mathsf{D}$ abzufragen, selbst wenn diese komplexe Klasse in der Ontologie keinen Namen besitzt.

Die Rückgabewerte der Methoden sind jeweils Mengen von IRIs (Internationalized Resource Identifiers). Jede IRI beschreibt eindeutig eine Entität der Ontologie, wie z.B. ein Individuum oder eine Klasse.

Listing 2.2: Methoden der Ontologie-Schlussfolgerung

```
1  package edu.kit.aifb.suprime.repository;
2
3  import java.net.URI;
4  import java.util.Set;
5
6
7  public interface OntologyRepository {
8
9      //  ...
10
11     boolean isConsistent(URI ontologyId);
12
13     Set<String> getSubclasses(String classExpr, URI ontologyId);
14
15     Set<String> getEquivalentClasses(String classExpr, URI ontologyId);
16
17     Set<String> getInstances(String classExpr, URI ontologyId);
18
19     Set<String> getSameIndividuals(String namedIndividualIRI, URI ontologyId)
           ;
20
21     Set<String> getDifferentIndividuals(String namedIndividualIRI, URI
           ontologyId);
22
23     Set<String> getSubObjectProperties(String objectPropertyIRI, URI
           ontologyId);
24
25     Set<String> getSubDataProperties(String dataPropertyIRI, URI ontologyId);
26
27     Set<String> getObjectPropertyValues(String namedIndividualIRI, String
           objectPropertyIRI, URI ontologyId);
28
29     Set<String> getDataPropertyValues(String namedIndividualIRI, String
           dataPropertyIRI, URI ontologyId);
30 }
```

2.3 Ontologie-Verzeichnis Web Dienst

Das Ontologieverzeichnis, wie in den Abschnitten 2.1 und 2.2 beschrieben, bietet seine Funktionalität über eine Web Dienst Schnittstelle an. Dadurch können beliebige Anwendungen aufbauend auf dem Ontologieverzeichnis nach dem Paradigma der dienst-orientierten Softwarearchitektur entwickelt werden. Der im Folgenden vorgestellte Ontologie-Editor implementiert einen Web Dienst Client und benutzt darüber die Web Dienst Schnittstelle zum entfernten Aufruf des Ontologieverzeichnisses.

Der Web Dienst kapselt die Funktionalität des Ontologieverzeichnis und bietet die Methoden des Ontologieverzeichnisses (vgl. Listing 2.1 und Listing 2.2) öffentlich an. Basierend auf den Informationen zu Typ und Pfad der Methodenaufrufe sowie den Bezeichnern der Parameter (JAX-RS Annotationen der Methoden [Bur10]) kann ein beliebiger Client des Ontologieverzeichnisdienstes im Rahmen weiterer Applikationen entwickelt werden. Die Methoden des Web Dienstes sind im Anhang A, Listing A.1, dokumentiert.

2.4 Ontologie-Vermittlung

Die Ontologie-Vermittlung überbrückt Heterogenitäten zwischen verschiedenen bereits vorhandenen, unabhängig voneinander erstellten Ontologien. Die Beziehungen zwischen den modellierten Objekten einer Quellontologie O_s und einer Zielontologie O_t lassen sich durch das Hinzufügen weiterer Ausdrücke integrieren, wodurch eine vereinigte homogene Ontologie O_m entsteht. Zum Beispiel, kann über die zusätzliche Modellierung der Teilmengenbeziehung von einen Konzept C der Ontologie O_s zu einem anderen Konzept D einer Ontologie O_t die Beziehung $\mathsf{C} \sqsubseteq \mathsf{D}$ zwischen beiden Konzepten ausgedrückt werden. Diese Vermittlung beschränkt sich nicht nur auf das Modellieren von Beziehungen zwischen Objekten aus zwei Ontologien. Es können beliebig viele Ontologien durch eine Ontologie-Vermittlung in Beziehung gebracht werden.

Die Vermittlungsinformation ist eine Menge von Ontologie-Axiomen unter Wiederverwendung der Entitäten aus n existierenden Ontologien O_s mit $(1 < s \leq n)$. Daher modellieren wir die Vermittlung abermals als eine OWL-Ontologie O_m, welche auf den Informationen aus den Ontologien O_s aufbaut. Die bereits in den obigen Abschnitten vorgestellte Funktionalität zur

Ontologie-Speicherung im Ontologieverzeichnis wird für die Speicherung der Vermittlungsontologien verwendet. Im folgenden Abschnitt wird der entwickelte Ontologie-Editor zur Modellierung von Ontologien und damit auch zur Modellierung der Vermittlungsontologien sowie zur Benutzung des Schlussfolgerungsdienstes vorgestellt.

2.5 Ontologie-Modellierung

Zur Modellierung von OWL Ontologien wurde ein leicht-gewichtiger Editor entworfen. Dieser ermöglicht die Erledigung häufig gestellter Aufgaben beim Betrachten, Ändern und Erstellen von Ontologien in einer übersichtlichen graphischen Benutzerschnittstelle. Bei komplexen Aufgaben kann auf existierende ausgewachsene Modellierungswerkzeuge wie Protégé[2], NeOn Toolkit[3] oder die OWL-API[4] zurückgegriffen werden.

Der Ontologie-Editor besitzt eine Web-basierte Benutzerschnittstelle. Dadurch können Ontologien in einem Web-Browser inspiziert und geändert werden ohne dass ein spezielles Programm installiert werden muss. Die Funktionalität des Ontologie-Editors umfasst derzeit:

- Anzeige verfügbarer Ontologien im WisNetGrid Ontologieverzeichnis
- Durchstöbern von Klassen, Relationen und Individuen einer ausgewählten Ontologie
- Hinzufügen von Klassen, Relationen und Individuen
- Speicherung von Ontologien im WisNetGrid Ontologieverzeichnis

Das zu verwendende Ontologieverzeichnis ist beim Start der Nutzung des Editors zu spezifizieren, da verschiedene Grid-Communities eigene Ontologieverzeichnisse verwalten können. Der Endpunkt des Verzeichnisses wird wie in Abbildung 2.1 gezeigt als URL eingegeben. Danach kann eine Ontologie von der obigen Liste ausgewählt und vom Ontologieverzeichnis geladen werden. Im zentralen Bereich des Ontologie-Editors stehen nun drei Ansichten

[2]Protégé Ontology Editor and Knowledge Acquisition System `http://protege.stanford.edu`

[3]NeOn Toolkit ontology engineering environment `http://neon-toolkit.org`

[4]OWL Java API `http://owlapi.sourceforge.net`

'Concepts', 'Properties' und 'Individuals' zur Anzeige der in der gewählten Ontologie spezifizierten Klassen, Relationen und Individuen. Abbildung 2.2 zeigt die erste Ansicht mit einer Klassenhierarchie im linken Bereich. Sub-, Super-, äquivalente, und disjunkte Klassen der im linken Bereich selektierten Klasse werden im rechten Bereich angezeigt.

Um eine neue disjunkte Klasse zur Ontologie hinzuzufügen, wählt man den entsprechenden Knopf und erstellt im Textfeld des Dialoges einen komplexen Klassenausdruck im Manchester Syntax [HDG+06]. Die Auswahl existierender Klassen, wie in Abb. 2.3 gezeigt, unterstützt die Eingabe existierender Klassennamen.

Analog lassen sich Informationen zu Relationen und Individuen anzeigen und modifizieren (vgl. Abbildungen 2.4 und 2.5). In Abbildung 2.5 wird die Ontologie so erweitert, dass ein gewähltes Individuum mit einem Wert in Relation gesetzt wird. Der Schlussfolgerungsdienst, wie in Abschnitt 2.2 beschrieben, kann im unteren Teil des Editors aufgerufen werden. Es stehen verschiedene Anfragetypen zur Auswahl (vgl. die Liste der Anfragetypen im linken Teil der Abb. 2.6). Bei der Abfrage von Sub-Klassen einer Klasse ist es wieder möglich, eine komplexe Klasse aus existierenden Klassen der Ontologie zu definieren. Die Ergebnisse einer solchen Anfrage werden dann nach der Auswertung in der Liste am rechten Rand des Ausschnitts angezeigt.

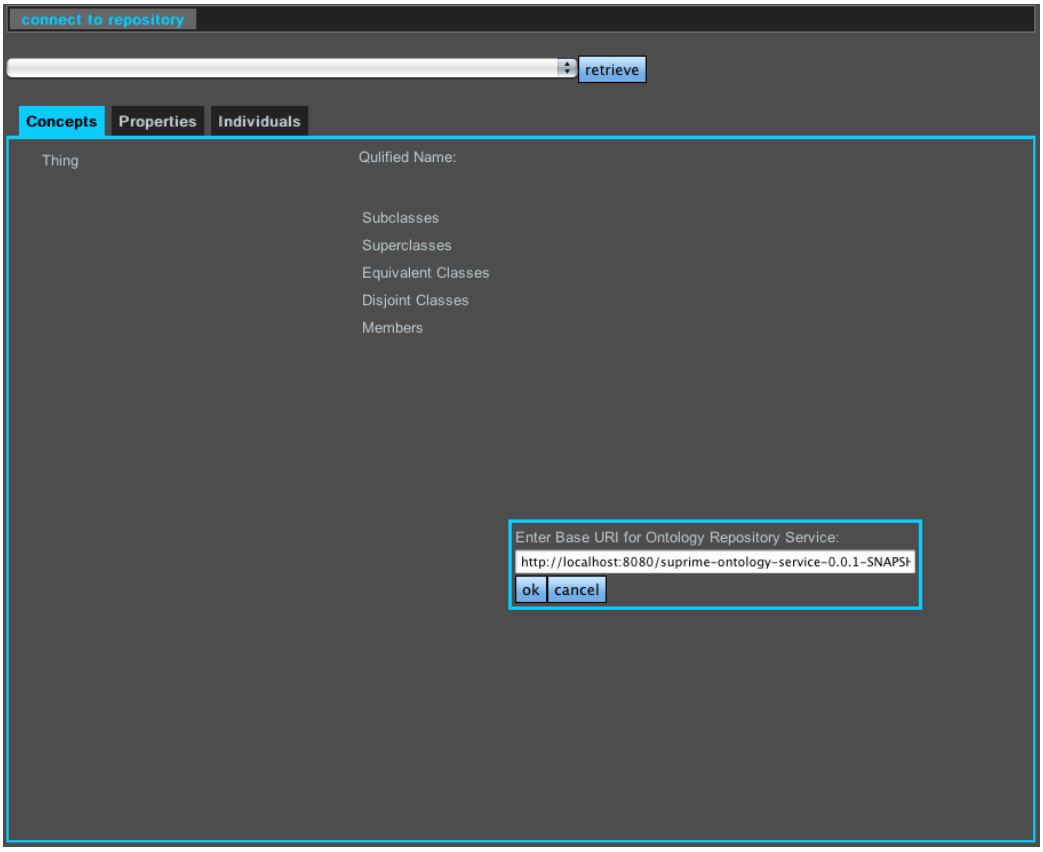

Abbildung 2.1: Ansicht des Ontologie-Editors beim Start der Anwendung: Aufforderung zur Eingabe des Ontologieverzeichnis-Endpunkts.

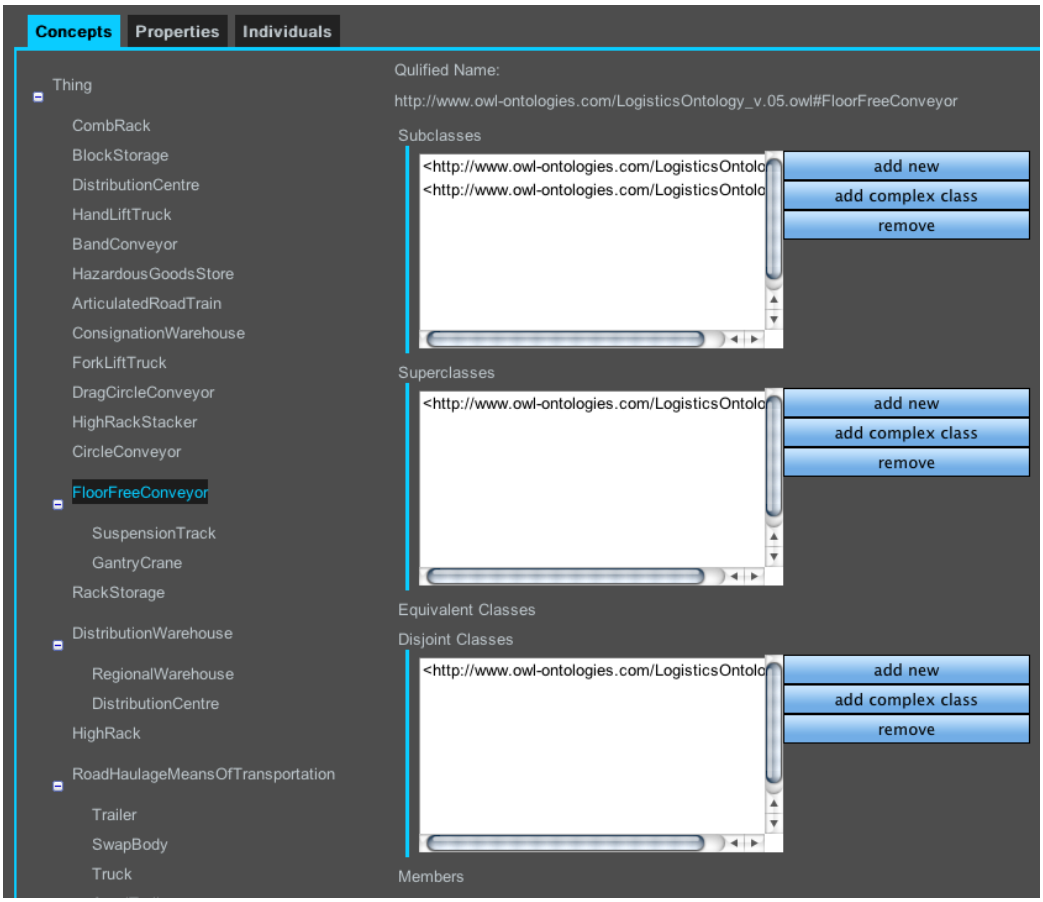

Abbildung 2.2: Informationen über Klassen einer Ontologie.

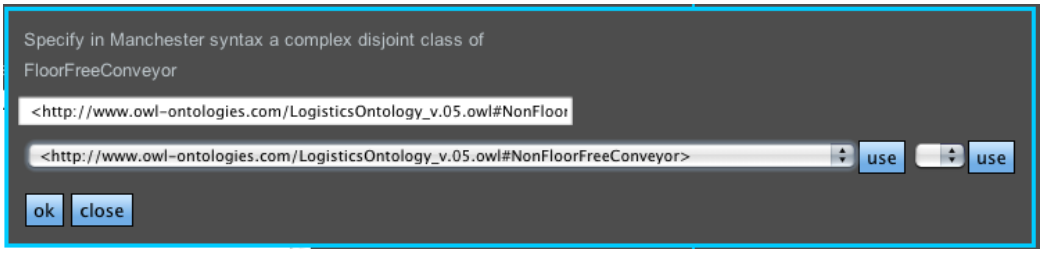

Abbildung 2.3: Hinzufügen einer komplexen disjunkten Klasse.

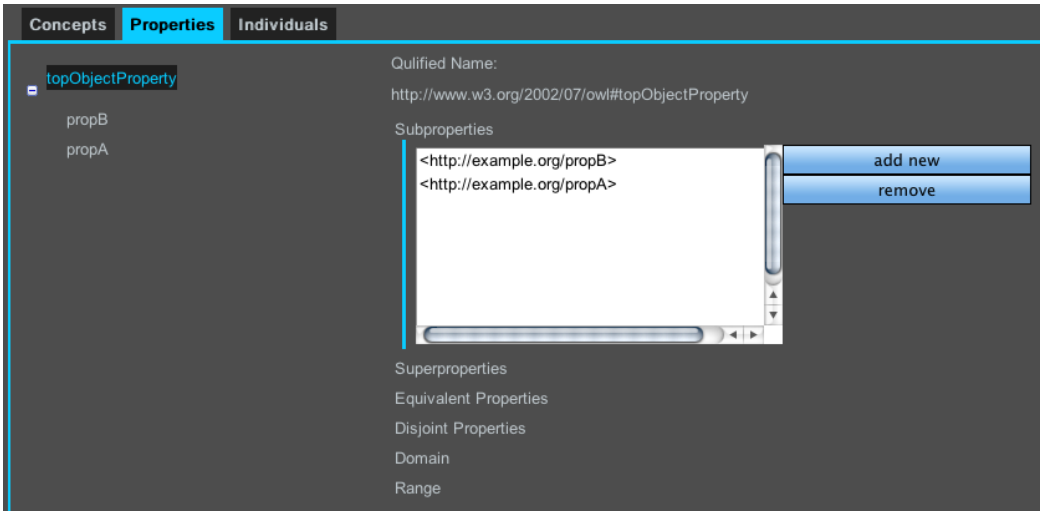

Abbildung 2.4: Informationen über Relationen einer Ontologie.

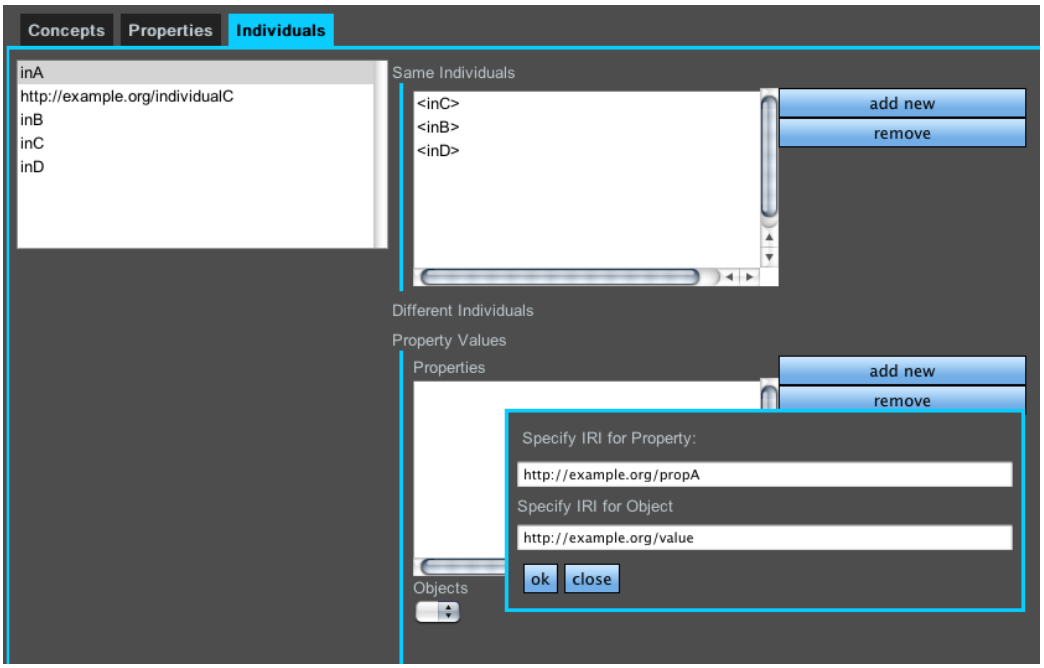

Abbildung 2.5: Informationen über Individuen einer Ontologie.

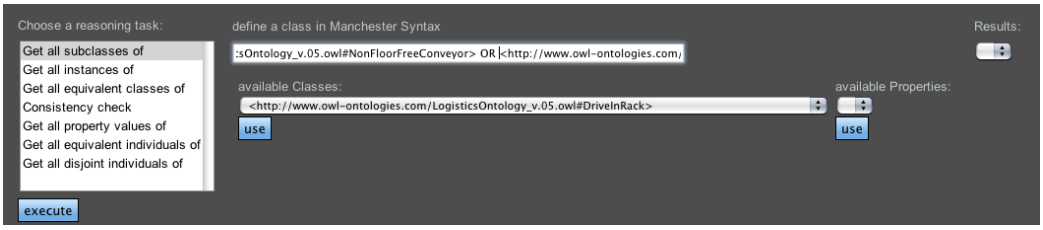

Abbildung 2.6: Abfrage der Sub-Klassen einer komplexen Klasse.

Kapitel 3

Installation der Komponenten

Die im Bericht beschriebenen Komponenten wurden in Java entwickelt und lassen sich mit dem Apache Maven Software Projekt Management Werkzeug kompilieren. Die Abhängigkeiten werden durch die Verwendung von Maven automatisch mit der Hilfe von mehreren Maven Repositories aufgelöst. Dazu muss die lokale Maven Instanz, wie in Anhang B beschrieben, konfiguriert werden.

Der Ontologie-Editor benutzt einen Web Dienst Client (basierend auf Jersey Client API[1]) um mit dem Ontologieverzeichnis-Dienst Ontologien auszutauschen und die Schlussfolgerungensmethoden aufzurufen. Die Web-basierte Benutzerschnittstelle wurde mit Hilfe der Google Web Toolkit[2] Bibliothek entwickelt. Zur Installation des Ontologie-Editors wird `mvn install` im Hauptverzeichnis des Projekts ausgeführt. Die notwendigen Komponente des Ontologieverzeichnisses müssen dafür weder kompiliert werden noch lokal verfügbar sein. Danach wird im Verzeichnis `./target/` eine Datei mit der Endung ".war" (Web Application Archive) erzeugt. Diese Datei wird in einem Servlet Container, wie z.B. Apache Tomcat, deployed. Danach ist der Ontologie-Editor mit einem Web Browser verwendbar. Zur Benutzung des Editors ist der Zugriff auf eine laufende Instanz eines Ontologieverzeichnis Web Dienstes notwendig.

Der Ontologieverzeichnis Web Dienst wird analog mit `mvn install` kompiliert und die erzeugte war-Datei in einem Servlet Container deployed.

[1] Jersey ist eine Referenz-Implementierung von JAX-RS `http://jersey.java.net`

[2] Google Web Toolkit ist ein Werkzeug zur Entwicklung komplexer Internet-Anwendungen `http://code.google.com/webtoolkit`

Das Ontologieverzeichnis baut auf die OWL-API sowie den HermiT Ontologie-Reasoner auf. Diese Komponenten werden ebenfalls durch Maven automatisch während der Kompilierung von einem Maven Repository nachgeladen.

Anhang A

Methoden des Ontologieverzeichnis Web Dienstes

Listing A.1: Methoden des Ontologie-Verzeichnis Web Dienstes

```
1  package edu.kit.aifb.suprime.repository.ontologies.service.rest;
2
3  @Path("/")
4  public class OntologyRepositoryService {
5
6      @POST
7      @Path("createOntology")
8      public Response createOntology(
9              @FormParam("ontology") String ontology,
10             @FormParam("ontologyId") String ontologyId);
11
12     @POST
13     @Path("createAndLockOntology")
14     @Produces("application/xml")
15     public Response createAndLockOntology(
16             @FormParam("ontology") String ontology,
17             @FormParam("ontologyId") String ontologyId,
18             @FormParam("lockOntology") String lockOntology,
19             @FormParam("userId") String userId);
20
21     @GET
22     @Path("getIds")
23     @Produces("application/xml")
24     public ResultURIs getIds();
25
26     @GET
27     @Path("readOntology")
28     @Produces("application/xml")
29     public String readOntology(
30             @QueryParam("ontologyId") String ontologyId);
31
```

```
32      @GET
33      @Path("readAndLockOntology")
34      @Produces("application/xml")
35      public String readAndLockOntology(
36              @QueryParam("ontologyId") String ontologyId,
37              @QueryParam("lockOntology") String lockOntology,
38              @QueryParam("userId") String userId);
39
40      @POST
41      @Path("updateOntology")
42      @Produces("application/xml")
43      public Response updateOntology(
44              @FormParam("ontology") String ontology,
45              @FormParam("ontologyId") String ontologyId,
46              @FormParam("lockOntology") String lockOntology,
47              @FormParam("userId") String userId);
48
49      @DELETE
50      @Path("deleteOntology")
51      @Produces("application/xml")
52      public Response deleteOntology(
53              @FormParam("ontologyId") String ontologyId);
54
55      @POST
56      @Path("subscribeToOntology")
57      @Produces("application/xml")
58      public Response subscribeToOntology(
59              @FormParam("ontologyId") String ontologyId,
60              @FormParam("userId") String userId);
61
62      /*
63       * Methoden des Schlussfolgerungsdienstes
64       */
65      @GET
66      @Path("isConsistent")
67      @Produces("application/xml")
68      public Response isConsistent(
69              @QueryParam("ontologyId") String ontologyId);
70
71      @GET
72      @Path("getSubclasses")
73      @Produces("application/xml")
74      public ResultURIs getSubclasses(
75              @QueryParam("classExpr") String classExpr,
76              @QueryParam("ontologyId") String ontologyId);
77
78      @GET
79      @Path("getEquivalentClasses")
80      @Produces("application/xml")
81      public ResultURIs getEquivalentClasses(
82              @QueryParam("classExpr") String classExpr,
83              @QueryParam("ontologyId") String ontologyId);
84
85      @GET
86      @Path("getInstances")
87      @Produces("application/xml")
88      public ResultURIs getInstances(
89              @QueryParam("classExpr") String classExpr,
90              @QueryParam("ontologyId") String ontologyId);
91
92      @GET
93      @Path("getSameIndividuals")
```

```
94      @Produces("application/xml")
95      public ResultURIs getSameIndividuals(
96              @QueryParam("namedIndividualIRI") String namedIndividualIRI,
97              @QueryParam("ontologyId") String ontologyId);
98
99      @GET
100     @Path("getDifferentIndividuals")
101     @Produces("application/xml")
102     public ResultURIs getDifferentIndividuals(
103             @QueryParam("namedIndividualIRI") String namedIndividualIRI,
104             @QueryParam("ontologyId") String ontologyId);
105
106     @GET
107     @Path("getSubObjectProperties")
108     @Produces("application/xml")
109     public ResultURIs getSubObjectProperties(
110             @QueryParam("objectPropertyIRI") String objectPropertyIRI,
111             @QueryParam("ontologyId") String ontologyId);
112
113     @GET
114     @Path("getSubDataProperties")
115     @Produces("application/xml")
116     public ResultURIs getSubDataProperties(
117             @QueryParam("dataPropertyIRI") String dataPropertyIRI,
118             @QueryParam("ontologyId") String ontologyId);
119
120     @GET
121     @Path("getObjectPropertyValues")
122     @Produces("application/xml")
123     public ResultURIs getObjectPropertyValues(
124             @QueryParam("namedIndividualIRI") String namedIndividualIRI,
125             @QueryParam("objectPropertyIRI") String objectPropertyIRI,
126             @QueryParam("ontologyId") String ontologyId);
127
128     @GET
129     @Path("getDataPropertyValues")
130     @Produces("application/xml")
131     public ResultURIs getDataPropertyValues(
132             @QueryParam("namedIndividualIRI") String namedIndividualIRI,
133             @QueryParam("dataPropertyIRI") String dataPropertyIRI,
134             @QueryParam("ontologyId") String ontologyId);
135 }
```

Anhang B

Maven Repository Einstellungen

Maven2 benötigt die Informationen über den Zugriff auf die zu verwendenden Maven Repositories, in denen sich die notwendigen Software-Artefakte befinden, in der Datei `settings.xml` im Nutzerverzeichnis des Betriebssystems (`~/.m2/settings.xml`). Listing B.1 zeigt die notwendige Einstellungen. Die ausgelassen Passwörter können bei `junghans@kit.edu` erfragt werden.

Listing B.1: Konfiguration von Maven2 (settings.xml)

```
1  <settings xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.
       org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/
       POM/4.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">
2
3      <activeProfiles>
4          <activeProfile>suprime</activeProfile>
5      </activeProfiles>
6      <localRepository>/opt/development/mavenrepo</localRepository>
7
8      <profiles>
9          <profile>
10             <id>suprime</id>
11             <repositories>
12                 <repository>
13                     <id>Maven2 repository</id>
14                     <url>http://repo1.maven.org/maven2</url>
15                 </repository>
16                 <repository>
17                     <id>suprime-public</id>
18                     <url>http://suprimeworkbench.aifb.uni-karlsruhe.de/nexus-
                           webapp-1.8.0/content/groups/public</url>
19                 </repository>
20                 <repository>
21                     <id>suprime-releases</id>
22                     <url>http://suprimeworkbench.aifb.uni-karlsruhe.de/nexus-
                           webapp-1.8.0/content/repositories/releases</url>
23                 </repository>
```

```
24                 <repository>
25                     <id>suprime-snapshots</id>
26                     <url>http://suprimeworkbench.aifb.uni-karlsruhe.de/nexus-
                           webapp-1.8.0/content/repositories/snapshots</url>
27                 </repository>
28
29                 <repository>
30                     <id>unicore.eu</id>
31                     <name>UNICORE repository</name>
32                     <url>http://unicore-dev.zam.kfa-juelich.de/maven</url>
33                 </repository>
34             </repositories>
35         </profile>
36     </profiles>
37
38     <servers>
39         <server>
40             <id>suprime-public</id>
41             <username>suprimo</username>
42             <password>***</password>
43         </server>
44         <server>
45             <id>suprime-snapshots</id>
46             <username>suprimo</username>
47             <password>***</password>
48         </server>
49         <server>
50             <id>suprime-releases</id>
51             <username>suprimo</username>
52             <password>***</password>
53         </server>
54     </servers>
55
56 </settings>
```

Literaturverzeichnis

[AHM10] Agarwal, Sudhir ; Harms, Patrick ; Michels, Carolin: *D3.2.3 Diensteverzeichnis*. Juni 2010. – verfügbar unter `http://www.wisnetgrid.org/`

[Bur10] Burke, Bill: *RESTful Java with JAX-RS - Designing and Developing Distributed Web Services*. O'Reilly, 2010. – I–XX, 1–288 S. – ISBN 978–0–596–15804–0

[HDG+06] Horridge, Matthew ; Drummond, Nick ; Goodwin, John ; Rector, Alan L. ; Stevens, Robert ; Wang, Hai: The Manchester OWL Syntax. In: Grau, Bernardo C. (Hrsg.) ; Hitzler, Pascal (Hrsg.) ; Shankey, Conor (Hrsg.) ; Wallace, Evan (Hrsg.): *OWLED* Bd. 216, CEUR-WS.org, 2006 (CEUR Workshop Proceedings)

[JA11] Junghans, Martin ; Agarwal, Sudhir: *D3.2.6 Evaluierung der Suchfunktion*. Juni 2011. – verfügbar unter `http://www.wisnetgrid.org/`