Master Thesis

Automated Ontology Mapping of Tagged Data in a Pandisciplinary Repository for Research Projects

M. Politze

Master Thesis DKE 12-10

Thesis submitted in partial fulfillment
of the requirements for the degree of Master of Science
of Artificial Intelligence at the Department of Knowledge
Engineering of the Maastricht University

Maastricht University
Faculty of Humanities and Sciences
Department of Knowledge Engineering
Master Artificial Intelligence

July 2, 2012

This thesis was carried out at:

Center for Computing and Communication
RWTH Aachen University
Seffenter Weg 23, 52074 Aachen

This thesis was supervised by:

Dr. N. Roos	B. Decker
(Maastricht University)	(RWTH Aachen)

Preface

This master thesis was written at the Department of Knowledge Engineering of Maastricht University. It was carried out as a part of the research project *Projekt Repository* of the Center for Computing and Communication of RWTH Aachen University. It also concludes my studies at the Department of Knowledge Engineering.

At first I want to thank my supervisor Nico Roos for his guidance during the research but also for the inspiring discussions about the topic and the inspiring ideas. My appreciations also go to Frederik Schadd for being the second supervisor of this thesis.

Moreover I would like to express my appreciation for my colleagues at the Center for Computing and Communication who did not only support me mentally during my studies but also provided the technical means for the practical parts of this thesis. Especially I want to thank Bastian Küppers for seemingly endless hours, nights and days of discussions and hacking im Dienste der Wissenschaft.

Marius Politze
Aachen, July 2 2012

Abstract

The master thesis is set as a part of a research project at RWTH Aachen University. The aim of this project is to build a web based pandisciplinary repository for research projects that shall become a central component of scientific cooperation and for long term storage of research data in different projects at the university.

The described product is currently developed by the Center for Computing and Communication of RWTH Aachen University and is supported by the German Research Foundation. It will be based on Microsoft SharePoint 2010 that is extended by several features concerning the tagging and formal retrieval of data. This thesis presents the implementation of several extensions for Microsoft SharePoint 2010 that realize some of these goals based on ontology definitions in the web ontology language.

Apart from implementing and testing the product this thesis covers the theoretical aspects of ontology matching, a task that has to be performed to relate two ontologies that cover the same domain but are modelled differently or changed over time. The matching is done on structural basis, therefore a structural alignment algorithm is proposed. Finally the performance of several variants of this algorithm is evaluated against the datasets of the ontology alignment evaluation initiative.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In current research projects a lot of data in form of images, videos, simulation results or simply texts are produced. Subsequent projects may be based on this data and therefore need to retrieve the data from wherever it was stored. The European Science Foundation (ESF) and the European Heads of Research Councils (EUROHORCs) have therefore defined a common open and permanent access policy to access research data [1]:

> The collection of research data is a huge investment. Permanent access to such data, if quality controlled and in interoperable formats, allows other researchers to use them, allows re-analysis of, for example, long time series and could play a role in ensuring research integrity. EUROHORCs and ESF will address how to best promote and ensure such permanent access to data generated with their funding.

This thesis aims at giving a first step in standardizing the storage of research data so it can be easily retrieved for follow-up projects. It will provide an infrastructure to create, manage, maintain and apply metadata schemas within the context of a repository for research projects. These metadata schemas are described using ontologies.

In this first Chapter the research project as part of which this master thesis was carried out is described. Then the Problem statement and the research questions are formulated. Finally a short outline of the thesis is given.

1.1 Setting

This thesis is a part of the research project *Projekt Repository* funded by the German Research Foundation. The project itself was initiated by several institutes of RWTH Aachen University: the University Library [2], the Institute of Pathology [3], the Department of History of Urbanization [4], the Institute of Hydraulic Engineering and Water Resources Management [5] as users for *Projekt Repository* and the Center for Computing and Communication [6], as well as the Center for Innovative Learning Technologies [7] as developers and operators of the project.

The main goal of *Projekt Repository* is to built a web based platform that offers a low-threshold service to share, store and retrieve research data among different groups of researchers from a variety of fields. This service shall then be integrated into the IT infrastructure offered by RWTH Aachen University. Finally this support for researchers, called *eScience*, will have an equivalent standing as *eLearning*. A full description of the Project and its position at RWTH Aachen University is given in [8].

As a basis for the file storage and user management the initiators of the project decided to use Microsoft SharePoint 2010 which is also used for the *eLearning* services offered by RWTH Aachen University. Apart from storing the research data itself *Projekt Repository* aims at adding metadata to the stored files. This metadata has to be added manually to the files as they are uploaded. It usually includes information about the contents of the data, their source or even interpretations that were drawn by the researchers.

Each type of metadata has its own predefined range of values. As one example from medical imaging illustrates: The image source might be one of the following: x-ray, MRI, microscopy etc., while the region of the body from which the data was taken might refer to organs or other parts of the body. Last but not least the diagnosis that was drawn from the data will be stored.

Prior to the start of this thesis the participating groups compared different metadata schemas and decided upon the ones that will be used in their special cases [9]. They further agreed that a central and generic structure has to be added to Microsoft SharePoint in order to operate with schemas for different research projects.

1.2 Metadata and Ontologies

A metadata schema usually includes the meaning, range and the associations of the different fields of the metadata. Some examples for schemas that are used by the different research groups participating in the research project can be found in section 2.3. All of these proposed metadata schemas offer a hierarchical ordering of the terms, most of the schemas offer synonyms. Some rare cases even have cross references within their hierarchies. Since metadata schema are well structured, given such a schema different fields of semantic information about the data can be extracted.

The structure provided by the metadata schema has to be modelled in a way such that it can be created and maintained by the researchers of the different fields and then be processed by computers. Metadata in general assigns some properties to files. As ontologies offer a way to model properties and relations of objects they might offer a flexible framework to model metadata schemas used in *Projekt Repository*. However an ontology offers a wider spectrum of possibilities. A benefit of this model could be that in an ontology, not only the metadata, but also the contents of the repository can be described once they are stored in the repository.

Another application of such semantic information is the Semantic Web, an initiative lead by the World Wide Web Consortium (W3C). Its aim is to semantically describe the information, documents and services offered on current web sites. These descriptions of web resources should be machine readable which in turn would allow computers to automatically navigate through their contents. The initial vision of the Semantic Web was expressed by Tim Berners-Lee one of the initial founders of the World Wide Web [10]:

> I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialize.

To achieve this vision of a Semantic Web, resources need to be described such their the meaning can be deduced by a computer program solely from its description. To define this meaning and to exchange it among different programs the semantic web uses ontologies.

For *Projekt Repository* a machine readable description of its contents could then serve exactly this purpose. This representation would allow one repository to provide enough data for an arbitrary agent that is searching for information. All things considered, *Projekt Repository* could offer researchers of RWTH Aachen University a way to easily take advantage of the Semantic Web as it is already done by other researchers all over the world [11].

1.3 Problem Statement and Research Questions

This thesis aims at creating an extension to Microsoft SharePoint 2010 that allows researchers of different disciplines to tag multimedia data in different, user defined, ontologies.

This leads to the following research questions:

- *Which structures of Microsoft SharePoint can be used to represent ontologies?*
- *How can these structures be used to tag data saved in the repository?*

Furthermore after the data is stored in the repository there needs to be a way to retrieve the data at a later point in time.

- *How can different retrieval techniques be used to retrieve data from the repository?*

Projekt Repository will serve as a long time storage for different kinds of research data. Since ontologies may change over time, or due to other reasons, the system needs to be flexible to some extent. Thus finally multiple repositories that store data of the same topic but are using (partially) different ontologies will be considered.

- *Can structural measures describe the elements of an ontology?*
- *Can different ontologies be matched using only their structure?*

For *Projekt Repository* a matching approach is chosen to deal with ontologies that are changing over time. There exist several approaches to to this problem using different techniques and components of the ontology. For this thesis a structural matching approach is chosen as it allows matching of ontologies that are not described in a natural language but follow a formal naming scheme like the histopathological reporting schemes [12, 13] used by the Institute of Pathology. To perform this structural alignment it is necessary to find structural measures that can be used to match the elements of ontologies.

1.3.1 Practical Implementation

The work done in this thesis lays the theoretical basis for an extension for Microsoft SharePoint that will be used for the realization of *Projekt Repository*. The product, the *Semantic Repository*, should provide a work-flow that enables researchers of different disciplines to operate with ontologies to structure their research data.

The product will therefore offer the following features:

- An interface to store, retrieve and update the ontologies used in the different projects.
- A functionality that adds the structure provided by the ontology to an existing repository.
- Ways of retrieving data using the structural information provided by the ontology.
- An interface that allows multiple repositories to exchange data and meta-data.

The intermediate versions of the product are continuously integrated into the system starting from the time the research for this thesis began.

1.4 Outline of the Thesis

The chapters of this thesis are organized in the following way:

- Chapter 2 *Ontologies* describes the basic theories of ontologies and their representation for the semantic web that will also be used for *Projekt Repository*.

- Chapter 3 *Microsoft SharePoint* discusses the different techniques available in Microsoft SharePoint 2010 that might be useful for tagging and storing ontologies.
- Chapter 4 *Design of the* Semantic Repository describes the accommodations that were implemented for Microsoft SharePoint in order to fully deal with ontologies. Furthermore a work-flow that is used by the researchers to create and import custom ontologies is introduced.
- Chapter 5 *Ontology Matching* will first introduce structural measures for ontologies. These measures will then be used to propose an algorithm to match ontologies based on their structure.
- Chapter 6 *Experiments* evaulates the chosen techniques for representing and working with ontologies in SharePoint. Furthermore the capabilities of the ontology alignment algorithm are assessed on artificial and real world data sets.
- Chapter 7 *Conclusion* finally concludes the research done and gives an outlook to future research.

Chapter 2

Ontologies

Ontologies will be used for the representation and exchange of the metadata schemas defined in *Projekt Repository.* This chapter presents the theoretical basis of ontologies: Description Logics. After a formal definition, different representations of ontologies with more and less capabilities are presented and compared to each other. Finally the Web Ontology Language (OWL) and one representational syntax OWL XML will be introduced. OWL is the recommended by the W3C to represent ontologies.

2.1 Definition

The term *ontology* originates from a philosophical discipline that deals with the things that exist, their categories of being and their relations. In information science the term *ontology* was adopted in the early 1990s to describe a set concepts that can be shared among different agents [14].

A broader definition is given by Dieter Fensel [15]:

> Ontologies are introduced to facilitate knowledge sharing and reuse between various agents, regardless of whether they are human or artificial in nature. They are supposed to offer this service by providing a consensual and formal conceptialization of a certain area. In a nutshell ontologies are formal and consensual specifications of conceptualizations that provide a shared understanding of a domain, an understanding that can be communicated across people and application systems.

Therefore, in computer science, an ontology has two main tasks:

- define formal semantics that are processable by a computer.
- link the meaning of real-world human terminologies to computer-processable content.

2.1.1 Formal Definition

In order to formally express terminologies from varieties of domains an ontology o is defined over a description logic which in turn consists of sets of entities all of which are pairwise disjoint:

Classes A class or concept denotes a set of individuals that share some set of properties. The Set of classes will be denoted with C.

Individuals Real world objects are denoted as individuals. They may belong to one or multiple classes. The set of individuals will be denoted with I.

Relations Individuals have relations with other individuals. The set of relations will be denoted with R.

To be able to express relationships and properties of the elements of these sets, furthermore a set of operations among these sets is defined:

Specialization A hierarchical dependency between two classes or relations is called specialization. It is denoted by $\sqsubseteq$.

Instantiation Instantiation or typing is interpreted as membership. Individuals can be members of classes and properties are instances of relations Instantiation is denoted by :.

Complement The instances not part of a certain class are part of the complement of that class. The complement of a class is denoted by $\neg$.

Intersection An individual can be an instance of more classes. Classes can therefore have intersections containing instances that belong to both classes. This is denoted by $\sqcup$.

Union Instances belonging to either one or another class are an instance of the union of both classes. This is denoted by $\sqcap$.

Relationship Two instances may form a relationship, an instance of a relation. A relationship between classes is denoted a tuple $(\cdot, \cdot)$.

Restrictions Two kinds of restrictions exist: Universal restrictions, denoted by $\forall$ place a restriction on the target class of all instances of a relation. Existential restrictions, denoted by $\exists$, ensure that at least one relation exist with the given target class. Both restrictions form new classes, the classes of individuals satisfying the restriction.

From this informal definition a description logic can now be formally defined. This leads to the following definition:

Definition 2.1. A description logic $\mathcal{ALC}$ (attribute logic with complement) consists of:

$$C \text{ is the set of classes,} \tag{2.1}$$
$$I \text{ is the set of individuals,} \tag{2.2}$$
$$R \text{ is the set of relations,} \tag{2.3}$$
$$\sqsubseteq \text{ is a relation over } C \times C \to \{true, false\} \tag{2.4}$$
$$\neg \text{ is a relation over } C \to C, \tag{2.5}$$
$$: \text{ is a relation over } (I \times C) \cup (I \times I \times R) \to \{true, false\} \tag{2.6}$$
$$\sqcup \text{ is a relation over } C \times C \to C \tag{2.7}$$
$$\sqcap \text{ is a relation over } C \times C \to C \tag{2.8}$$

Together with the operators it is now possible to define the extended set of classes $\mathcal{C}$. $\mathcal{C}$ contains at least the two special classes $\top$ (everything) and $\bot$ (nothing) and every class from C. Furthermore it contains every possible intersection, union, complement and restriction such that.

Definition 2.2. Given the sets C and R the set $\mathcal{C}$ is recursively defined as:

$$C \subseteq \mathcal{C} \tag{2.9}$$
$$\top \in \mathcal{C}, \bot \in \mathcal{C} \tag{2.10}$$
$$\text{If } c \in \mathcal{C} \text{ and } d \in \mathcal{C} \text{ then } \neg c \in \mathcal{C}, c \sqcap d \in \mathcal{C} \text{ and } c \sqcup d \in \mathcal{C} \tag{2.11}$$
$$\text{If } c \in C \text{ and } r \in R \text{ then } \exists r.c \in \mathcal{C} \text{ and } \forall r.c \in \mathcal{C} \tag{2.12}$$
$$\text{Nothing else bolongs to } \mathcal{C} \tag{2.13}$$

2.1.2 Ontology Semantics

To fully define the operations of the ontology an interpretation function π is used to map the entities from the ontology to real world objects.

Definition 2.3. An interpretation $I = \langle O, \pi \rangle$ consists of an interpretation function π that maps the entities from from C, R and I to a non empty set of real world objects O such that:

$$\text{for all } c \in C : \pi(c) \subseteq O, \tag{2.14}$$
$$\text{for all } r \in R : \pi(r) \subseteq O \times O, \tag{2.15}$$
$$\text{for all } i \in I : \pi(i) \in O, \tag{2.16}$$

Using the interpretation it is now possible to define the operator semantics for the description logic. However to interpret the concepts of the extended set of classes $\mathcal{C}$ an extended interpretation π^* is defined:

Definition 2.4. Given an interpretation $I = \langle O, \pi \rangle$ the extended interpretation π^* is defined such that:

Given $c \in C$, $r \in R$ and $i \in I$

$$\pi^*(c) = \pi(c) \text{ iff } c \in \mathcal{C} \tag{2.17}$$
$$\pi^*(\top) = O \tag{2.18}$$
$$\pi^*(\bot) = \emptyset \tag{2.19}$$
$$\pi^*(\neg c) = O - \pi^*(c) \tag{2.20}$$
$$\pi^*(c \sqcap d) = \pi^*(c) \cap \pi^*(d) \tag{2.21}$$
$$\pi^*(c \sqcup d) = \pi^*(c) \cup \pi^*(d) \tag{2.22}$$
$$\pi^*(\exists r.c) = \{x \in O \mid \text{for some } y \in O : (x,y) \in \pi(r) \text{ and } y \in \pi^*(c)\} \tag{2.23}$$
$$\pi^*(\forall r.c) = \{x \in O \mid \text{for all } y \in O : (x,y) \in \pi(r) \text{ implies } y \in \pi^*(c)\} \tag{2.24}$$

The given set of expressions can be extended further by quantifiers or other logical connectors in the same manner. Nevertheless the operations in the definitions above are sufficient to define the general notion of an ontology. Some of these extensions are presented in section 2.1.4.

Using the description logic $\mathcal{ALC}$ and a set of real world objects O, it is now possible to define assumptions that hold within the context of an ontology o. These assumptions can be used to model the domain of the ontology. This model is defined in the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.

2.1.3 Terminology and Assertions

On top of description logic an ontology builds a terminology and assertions to describe properties of things described. The terminology or TBox introduces the classes and properties that exist in the ontology. Furthermore the TBox introduces the properties of classes such as their hierarchical relationship or restrictions on their properties. The assertions or ABox define the knowledge about individuals and their relations within the ontology. Together the TBox and the ABox build the knowledge base $\mathcal{K}$ of an ontology. In general the interpretation I should satisfy the knowledge base $\mathcal{K}$:

Definition 2.5. An interpretation $I = \langle O, \pi \rangle$ satisfies a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

$$I \models \mathcal{K} \text{ iff } I \models \mathcal{T} \text{ and } I \models \mathcal{A} \tag{2.25}$$
$$I \models \mathcal{T} \text{ iff for every } \tau \in \mathcal{T} : I \models \tau \tag{2.26}$$
$$I \models \mathcal{A} \text{ iff for every } \alpha \in \mathcal{A} : I \models \alpha \tag{2.27}$$
$$I \models c \sqsubseteq d \text{ iff } \pi^*(c) \subseteq \pi^*(d) \tag{2.28}$$
$$I \models c = d \text{ iff } \pi^*(c) = \pi^*(d) \tag{2.29}$$
$$I \models a : c \text{ iff } \pi(a) \in \pi^*(c) \tag{2.30}$$
$$I \models (a,b) : r \text{ iff } (\pi(a), \pi(b)) \in \pi^*(r) \tag{2.31}$$

To clarify the roles of the TBox and the ABox for the ontology consider the following example:

Given an ontology o and a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The TBox $\mathcal{T}$ may define the following assertions:

$$\begin{aligned}\mathcal{T} = \{&Person \sqsubseteq \top,\\ &Scientist \sqsubseteq Person \sqcap \forall meetsAtBar.Waiter,\\ &Waiter \sqsubseteq Person, \}\end{aligned} \tag{2.32}$$

This TBox builds a class hierarchy of persons, being either scientists or waiters but not both since they exclude each other. Furthermore the object property *meetsAtBar* is defined with a restricted range (waiters) and Domain (scientists). The TBox basically defines the schema of the data described by the ontology.

It is now possible to introduce the ABox $\mathcal{A}$ to describe the actual instances that populate the ontology.

$$\begin{aligned}\mathcal{A} =\{&Sheldon : Scientist,\\ &Leonard : Scientist,\\ &(Leonard, Penny) : meetsAtBar\}\end{aligned} \tag{2.33}$$

The given ABox describes three persons and their relation. *Sheldon* and *Leonard* are defined to be scientists. Furthermore the *meetsAtBar* relation between *Leonard* and *Penny* is defined.

Using the whole knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ even from this small example conclusions can be drawn. In equation 2.32 the TBox defines everyone being in a *meetsAtBar* relation with a scientist as a waiter. As a human it is quite easy to conclude that therefore *Penny* must be a waitress.

2.1.4 Expressiveness

As already mentioned in the section 2.1.2 the description logic on which the ontology is based can be extended by several logical connectors and quantifiers. To be able to quickly determine which capabilities a certain model of an ontology has a special naming convention is introduced.

$\mathcal{ALC}^{+}$ or $\mathcal{S}$ is the basic form of description logics it allows negotiation of atomic classes, intersection between classes, quantifiers and complements. The $^{+}$ denotes the possibility to define properties as transitive.

$\mathcal{I}$ defines the notion of inverse properties.

$\mathcal{N}$ defines numeral quantifiers such as "at least 2" or "between 1 and 3"

$\mathcal{Q}$ extension to $\mathcal{N}$, numeral quantifiers can have a type.

$\mathcal{O}$ defines class enumerations: "one of".

$\mathcal{F}$ defines functional properties.

$\mathcal{H}$ defines a simple hierarchy for relations.

$\mathcal{R}$ extension to $\mathcal{H}$, defines complex relations between relations.

$^{(\mathcal{D})}$ allows usage relations with data types such as numbers or text.

2.1.5 Forms of Ontologies

Given the formal definition of an ontology, different ways of using it to define semantics for real world data can now be considered. In [16] a set of different forms of ontologies are presented. These different manifestations offer a range of expressive features from a very limited to a full representation of the formal features defined in section 2.1.1.

Controlled Vocabularies are the most simple variety of ontologies. They only know classes and individuals belonging to a single class. Thus their expressiveness is the most limited as only the instantiation of individuals is modelled. Nevertheless controlled vocabularies are popular for tagging applications.

Taxonomies are an extension to controlled vocabularies. They do not only offer a set of classes but also include the notion of specialization. Thus a taxonomy cannot only assign a class to an individual but also has a hierarchical interclass relation thus classes can be a subclass of a single other class. Taxonomies are quite popular in biological and medical applications where they are used since the 18th century.

Thesauri can be seen as another extension to controlled vocabulary. Instead of introducing a hierarchy they introduce the possibility to define equal classes. Among the equality relation other relations between classes, such as parent-child, may be introduced. Together with these other relations the thesauri then even extend taxonomies. Nevertheless the interclass relations have to be defined beforehand and cannot be defined by the thesaurus itself. Among the structural relations thesauri also introduce the possibility to describe the classes in natural language.

Topic Maps are one of the most recently developed forms of ontologies. Instead of giving a set of classes, topic maps define relations among individuals. Although this might seem more restrictive than a thesaurus, if used correctly, topic maps offer a wider range of expressions to be made. Topic maps regard classes as meta-individuals that can be connected with a *subclass of* relation, whereas individuals might be connected to classes using an *instance of* relation. A similar relation can be introduced for equality of classes or individuals. Topic maps can define these relations and introduce new relations if needed.

Ontologies itself define the most flexible framework for semantic description. Classes and relations can be in hierarchies but can also have more than one parent to form a graph of classes and relations. By using formal logic classes can be described and therefore implicitly assigned to individuals based on their relations. Ontologies as described in section 2.1.1 can express all the other semantic schemas discussed above and, with the usage of description logic, offer an even wider range of expressions.

2.1.6 Ontologies and Metadata

The definition of ontologies can now be used to describe how metadata will be mapped to ontologies. In general metadata gives additional information about a subject, in computer science usually a file. This information is well structured and therefore divided into fields with a certain range of values.

In an ontology it is possible to define individuals which in turn can be more closely described by adding properties that either form a relation to another individual or some value.

To translate metadata to an ontology for every field of the metadata schema a relation is introduced in the ontology. Data ranges from the metadata schema are converted to data types. The files described in the metadata are modelled as an instance in the ontology.

This conversion shows clearly that an ontology is capable of expressing the semantics from the metadata schema. Nevertheless ontologies also define how the files are to be represented and additionally offer the possibility to build cross references among instances.

2.2 Representation

As the ontologies itself offer the widest possible spectrum of expressions it now is crucial to find a representation of ontologies that can be processed by a computer. There are several formats that offer more or less complete support of the different features of ontologies. Nevertheless the World Wide Web Consortium (W3C) gave a recommendation for semantic web applications to use the Web Ontology Language (OWL) [17].

From the list describing the different description logics it gets quite clear that some of the extensions of description logic are needed to be available to actually work with ontologies. The OWL language specification [17] defines three basic editions of OWL:

OWL Lite is the most basic variant using $\mathcal{SHIF}^{(\mathcal{D})}$

OWL DL extends OWL Lite by using $\mathcal{SHOIN}^{(\mathcal{D})}$

OWL 2 is the most recent development and therefore allows the most powerful expressions. $\mathcal{SROIQ}^{(\mathcal{D})}$

When following the development of the different versions it gets clear that the expressiveness gradually increases from OWL Lite to OWL 2. Nevertheless the languages are syntactically compatible in such a way that a document designed for OWL Lite will be interpreted in the same way when using an OWL 2 aware program.

Even though the expressiveness increases among the different versions. The more expressive the representation gets, reasoning gets computationally more intensive to decide weather a statement is valid or not. Thus the variant to be supported has to be chosen carefully.

2.2.1 Web Ontology Language

The Web Ontology Language (OWL) XML is an XML-based syntax to describe ontologies. Its main features and their correspondence in the formal definition of ontologies are presented in this section. The OWL syntax is based on the Resource Description Framework (RDF) [18] and thus uses some of the elements from RDF. The OWL XML syntax is defined by the W3C in [17].

Before having a deeper look at OWL syntax a first basic part of the OWL entities is introduced: entity names. Entity names can occur on every type of entity, they provide a unique name for classes, individuals, relations and datatypes. Usually this name is given in form of the Uniform Resource Identifier (URI) of the XML document plus a fragment denoting the actual name of the entity. [19] defines the URI as displayed in Figure 2.1.

Even though the most common structures of OWL are introduced below this list is far from complete. Please refer to [17, 20] for a full set of axiomatic rules and expressions.

Listing 2.1 shows a declaration of a class in OWL syntax. The tag *owl:Class* refers to the class as it is defined in the OWL namespace. The OWL namespace as well as *owl:Class* can be identified by their URI `http://www.w3.org/2002/07/owl#` or `http://www.w3.org/2002/07/owl#Class` respectively. Further information on the general XML format and XML namespaces can be obtained from [21].

```
<owl:Class rdf:about="http://www.owl-ontologies.com/travel.owl#Destination">
</owl:Class>
```

Listing 2.1: An example of a class declaration in OWL taken from the "Travel Ontology".

The example in Listing 2.2 shows the syntax of a class specialization. The tag *rdfs:subClassOf* is defined in the RDF schema namespace and is another indicator that OWL itself is based on RDF. Analogous to the class specialization OWL relations can be specialized.

```
<owl:Class rdf:about="http://www.owl-ontologies.com/travel.owl#Beach">
    <rdfs:subClassOf
        rdf:resource="http://www.owl-ontologies.com/travel.owl#Destination"/>
</owl:Class>
```

Listing 2.2: An example of a subclass relation in OWL taken from the "Travel Ontology".

$$\underbrace{\texttt{http:}}_{\text{scheme}}\underbrace{\texttt{//www.w3.org}}_{\text{authority}}\underbrace{\texttt{/2002/07/owl}}_{\text{path}}\texttt{\#}\underbrace{\texttt{Class}}_{\text{fragment}}$$

Figure 2.1: Different parts of a URI displayed as an example from the definition of the OWL class entity.

The instantiation of a class is done with the tag *owl:NamedIndividual*. Listing 2.3 shows the instatiation of an indidual belonging to the class *Beach* from listing 2.2. The defintion of the class instantiated by the individual is given in the *rdf:type* tag. An alternative to the instantiation with the tag *owl:NamedIndividual* is to directly use the name of the class that is instantiated as a tag. Relations can be instantiated in the same two ways.

```
<owl:NamedIndividual
    rdf:about="http://www.owl-ontologies.com/travel.owl#BondiBeach">
    <rdf:type rdf:resource="http://www.owl-ontologies.com/travel.owl#Beach"/>
</owl:NamedIndividual>
```

Listing 2.3: An example of an individual in OWL taken from the "Travel Ontology".

When it comes to relations OWL distinguishes between interindividual relations and individual-datatype relations. Inter-individual relations are denoted as object properties. They define a range and a domain in form of an OWL class and thus delimit the number of individuals the property might be applied to. A basic declaration of an object property is displayed in listing 2.4, nevertheless the property may be declared to have more detailed characteristics such as transitivity or symmetry.

```
<owl:ObjectProperty
    rdf:about="http://www.owl-ontologies.com/travel.owl#hasActivity">
    <rdfs:range
        rdf:resource="http://www.owl-ontologies.com/travel.owl#Activity"/>
    <rdfs:domain
        rdf:resource="http://www.owl-ontologies.com/travel.owl#Destination"/>
</owl:ObjectProperty>
```

Listing 2.4: An example of a object relation in OWL taken from the "Travel Ontology".

In contrast to the object property, listing 2.5 defines a datatype property. These are used to describe relations between individuals and values of a specific datatype. As in the case of object properties domain and range for the datatype property are defined but this time the range has to be a datatype. The datatypes that can be used as a range are a subset of the standard XML datatypes.

```
<owl:DatatypeProperty
    rdf:about="http://www.owl-ontologies.com/travel.owl#hasEMail">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain
        rdf:resource="http://www.owl-ontologies.com/travel.owl#Contact"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

Listing 2.5: An example of a datatype relation in OWL taken from the "Travel Ontology".

Last but not least the code in listing 2.6 gives an example of a logical expression that can be used to limit the usage of certain classes. Apart from the disjoint operator that separates the individuals of two classes set operations like intersection, union and complement as well as the full enumeration of possible individuals can be used. Furthermore restrictions on the properties can be defined in the classes.

```
<owl:Class rdf:about="http://www.owl-ontologies.com/travel.owl#RuralArea">
    <rdfs:subClassOf
        rdf:resource="http://www.owl-ontologies.com/travel.owl#Destination"/>
    <owl:disjointWith
        rdf:resource="http://www.owl-ontologies.com/travel.owl#UrbanArea"/>
</owl:Class>
```

Listing 2.6: An example of a logic expression in OWL taken from the "Travel Ontology".

The description of these complex classes requires an inference engine to find the individuals belonging to such a class. Nevertheless they can also be declared explicitly.

2.2.2 Other Syntaxes for Ontology Representation

OWL XML is not the only syntax that is used to represent ontologies. Multiple other formats exist and are commonly used in different disciplines to represent ontologies. In general these representational languages can be divided in two classes:

Explicit Semantics These syntaxes define an explicit semantic for the usual structures of ontologies as defined in 2.1.1. These syntaxes are usually explicitly designed with the theoretical construct of an ontology in mind and therefore well suited to represent ontologies. They also aim to be human readable. Examples of languages following these concepts are OWL XML, OWL functional syntax [22], Manchester OWL Syntax [23], OBO Flat File Syntax [24], KRSS Syntax [25] and Turtle [26].

Hidden Semantics This set of languages does not explicitly define semantics for the constructs of the language. Even though it is possible to define individuals of certain types or classes as well as relations there are usually only very few relations that have a specific meaning such as specialization. Deriving an ontology in this case is no longer straight forward and usually needs additional reasoning. Examples from this set of languages are XML Schema (XSL) [27] and custom CSV or XML files.

Even though languages with hidden semantics are far from optimal, many ontologies from social sciences like LIDO and ISO 19115 are expressed in XML Schema (see section 2.3).

2.2.3 Software

In the process of designing an ontology for *Projekt Repository* it is not feasible for the final users to write the ontology files manually. Therefore, together with the semantic functionality, a software tool will be introduced to create, display and edit ontologies.

The software that will be used fot this purpose is Protégé-OWL [28]. Protégé-OWL is a free and open source software that allows GUI based editing of ontologies. Other than that OWL Protégé-OWL has the ability to save and read ontologies in a variety of different formats. This is especially useful as existing ontologies that will be used for *Projekt Repository* need to be converted to follow OWL syntax.

2.3 Ontologies used in Projekt Repository

Even though *Projekt Repository* is meant to be used with any ontology this section will briefly present the ontologies used by the different participating research groups at the current state of the project. In general ontologies with two different aims can be found: General ontologies that describe digital data and its properties whereas domain specific ontologies usually describe the contents of the data.

2.3.1 General Ontology

The general ontology that is used by every project group is the Dublin Core Metadata Element Set [29]. The traditional set contains 15 properties that shall be defined for every document in a collection. Table 2.1 gives a brief overview of the core fields.

Apart from being used by a variety of organizations, universities and libraries all over the world, the Dublin Core Metadata Element Set was ratified as a standard by several standardisation organisations as IETF RFC 5013, ANSI/NISO Standard Z39.85-2007, and ISO Standard 15836:2009. Furthermore all of the standards include the option to add additional metadata information as needed.

Though very general the acceptance and flexibility of the Dublin Core Metadata Element Set makes it an ideal base format to interchange tagged elements.

2.3.2 Domain Ontologies

Apart from the Dublin Core Metadata Element Set every project group defined a domain ontology that they want to use to organize their data. These ontologies are usually much bigger and less standardised.

Title	Creator	Subject
Description	Publisher	Contributor
Date	Type	Format
Identifier	Source	Language
Relation	Coverage	Rights

Table 2.1: List of the 15 core metadata fields of the Dublin Core Metadata Element Set.

MESH The Medical Subject Headings is a popular medical thesaurus that is maintained by the U.S. National Library of Medicine. MESH offers a twelve level hierarchy leading to a total of about 26,000 terms as well as about 177,000 synonyms for these terms. The most prominent application of MESH is PubMED, a meta database that makes available medical articles [30].

LIDO Schema Lightweight Information Describing Objects is a metadata schema that defines properties to be defined by the user for the stored data instances. In contrast to MESH it does not define many terms but rather defines the structure of the metadata. LIDO was proposed in 2010 by the International Council of Museums (ICOM) for the description of historical artefacts. Figure 2.2 gives a brief overview of the different fields defined by LIDO.

ISO 19115 Is a metadata scheme for geographic information [32]. It provides structures to add spatial, qualitative and temporal information to digital geographic data. ISO 19115 is the mandatory metadata schema for geographic publications in the European Union and is therefore widely known and used. The original metadata schema is defined in XML Schema but inoffical conversions to OWL exist.

2.3.3 Ontology Conversion

To convert ontologies with hidden semantics to ontologies with explicit semantics some domain knowledge is required. For the conversion of XML Schema into an OWL ontology two similar approaches were developed independently by [33, 34]. The most important operations to convert the XML Schema to an ontology are summarized in table 2.2 based on [34].

While XML Schema is quite structured other formats, especially those for word lists, like MESH, do not have a standardized structure. However for simple controlled vocabularies two simple conversions can be performed: Every entry in the thesaurus is defined as a class or as an individual. In the first approach the instance to be tagged will be an instance of the generated class. The second

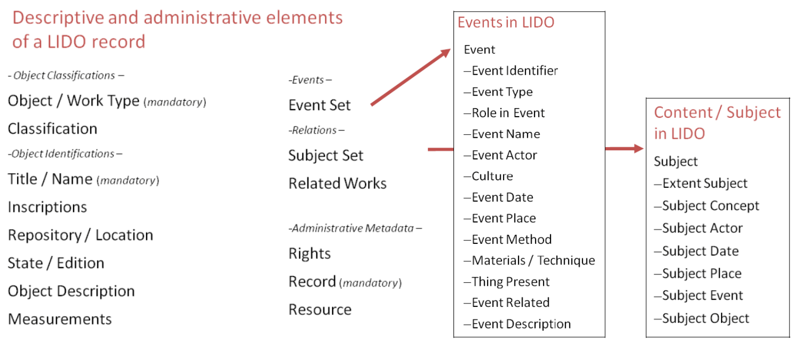


Figure 2.2: Elements of the LIDO metadata schema. Source: [31]

XML Schema	OWL	Shared semantics
element|attribute	DatatypeProperty ObjectProperty	Relation between instances
complexType|group	Class	Contextual restrcitions
complexType//element	Restriction	Restriction of a relation
restriction@base	subClassOf	Specialisation
sequence choice	intersectionOf unionOf	Combinations of relations

Table 2.2: Some rules used to convert implicit semantics in XML Schema to explicit semantics in OWL.

approach is more flexible. For every list of vocabulary a single class and an object property having this class as domain is defined. Every word in the list is then represented as an individual of the class. The instance to be tagged then has to define a relation using the object property.

Chapter 3

Microsoft SharePoint

Since it will be the basis for the *Semantic Repository*, this chapter will give an overview of the current state of the Microsoft SharePoint platform. At first the general structure of a SharePoint website will be described. Then some technical details required to extend the SharePoint document management platform will be presented. The last section in this chapter will discuss the features for semantic content currently available in SharePoint and give an impulse on where the current structures actually can be extended to work with ontologies.

3.1 Structure of a SharePoint Website

Microsoft SharePoint is designed to be an enterprise-scale system that can be used to support various processes. The default SharePoint configuration provides document and content management and collaboration features on a web based platform. All of the offered services are integrated with other active directory services offered by other Microsoft products such as Windows Server 2008.

When running SharePoint on a very large scale multiple servers form a server farm. Since all servers of a farm offer the same services when seen from the outside, they behave like a single server. A schema of the SharePoint server architecture is displayed in figure 3.1. Apart from the physical setup of multiple servers with different capabilities, such as web and database servers (2,7,9), a SharePoint server farm is logically divided into multiple independent web services (4). A web service is then divided into multiple web applications (8) which in turn consist of multiple site collections (10).

This logical structure can also be observed in the URI of the SharePoint Website. The domain name denotes the web service, whereas the web application is identified by the port. Finally the site collection and its subsites are identified by the path.

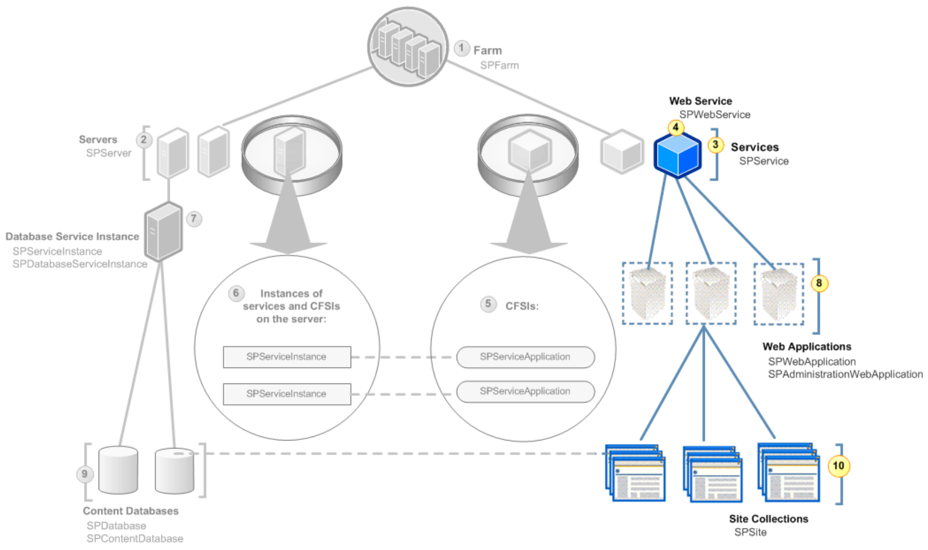


Figure 3.1: Physical (gray) and logical (colored) architecture of the SharePoint platform. Source: [35]

3.1.1 Document Management

Projekt Repository mainly aims to use SharePoint as a document management system. Every research project using *Projekt Repository* will have its own site collection containing multiple lists of stored documents (see figure 3.2: 3).

Each of these lists can store uploaded files as items (5). Furthermore meta-data of the uploaded files, such as image resolution or upload date, is saved in several named fields (4). Among the default fields for some types of data, such as image resolution for images, it is also possible to define custom fields to store arbitrary text or restrict its contents to special data types such as numbers or dates.

All of these document management capabilities come with the default installation of SharePoint and build the basis for storing the research data in the repositories.

3.1.2 Extensibility

Obviously the goal to add features to deal with semantic information from ontologies to SharePoint requires a high degree of extensibility. Among the web front-end that is offered by SharePoint it is possible to access all the functionalities via a well documented API using the C# programming language. This API allows all features of SharePoint to be customized. The next paragraphs will point out some of the most interesting features:

Lists are the basic organisational unit that is delivered by SharePoint. A list stores data and metadata of a certain type. Each item in a list can be considered an individual having multiple properties. The properties are

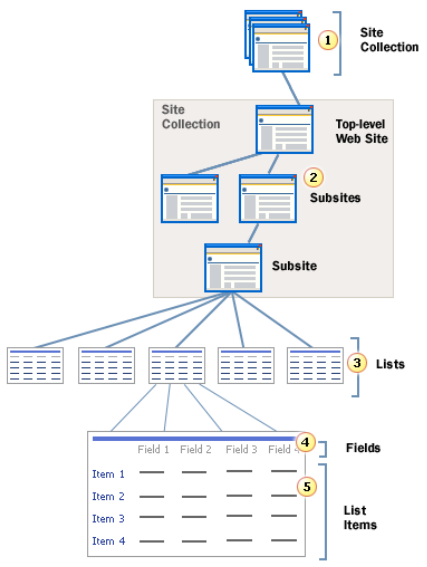


Figure 3.2: Architechture of a SharePoint site collection. Source: [35]

denoted by various fields that can be added to the list. The built in fields have various types such as numerical, text or date.

Custom Fields denote field types of a list that can be automatically checked for validity. The actual kind of validity checks has to be implemented as an SharePoint extension which is then loaded into SharePoint server as an additional program library. Using this type of extension it is virtually possible to provide arbitrary field types.

Web Parts are, in contrast to custom fields, not only an extension to the already existing lists but rather provide a custom user interface on the SharePoint website.

Services make up a different way of customizing the way SharePoint works. Services do not tackle the front end or user interface but rather build a back end interface. A service thus can be used to offer custom web pages or XML data that can be read by other programs.

Independent of the actual type of extension it is possible to define the scope of the extension. The broadest variant is to distribute the extension on the complete server farm but it can be reduced to web service, web application or even site collection scope. Also when deploying it to a complete server farm extensions can be activated for single services, applications or collections respectively granting the possibility to centrally control the number of extensions running on the different site collections.

3.2 Semantic Features

Also in the domain of semantics SharePoint offers some features and APIs to be used. In this section some of the features already mentioned in section 3.1.2 will be revisited and each of these features will be examined for their capability to work with semantic data.

The selected feature should provide the means to interact with the semantic information from the ontology such as class selection or instantiation of relationships. Therefore the feature needs to be flexibly customizable in terms of the data that is stored.

3.2.1 Custom Fields

As already discussed in section 3.1.2 custom fields offer a possibility to add custom formated data to lists. These fields then can be used to add data properties that are restricted to some set of values.

Apart from just checking the validity of the entered data, a custom field can also perform auto completion and display suggestions for partially entered data. Even though all functionalities of custom fields need to be programmed, their flexibility makes them a possible choice for controlled vocabularies as well as other semantic structures.

3.2.2 Taxonomy Store

The central taxonomy store of SharePoint can be used to maintain a custom taxonomy. The taxonomy itself can be entered or changed manually from within SharePoint or can be imported from a CSV file.

The imported taxonomy can then be made available as a taxonomy bound field in a list. When it comes to the offered features, the taxonomy fields is quite equal to custom fields as they also support auto completion and suggestions for values. Additionally they do not require any coding to be done before they can be used.

3.2.3 Evaluation

The discussed semantic features offer different possibilities to extend the semantic capabilities of SharePoint. Nevertheless both variants have some drawbacks.

The custom fields can hold arbitrary values, but have to be implemented manually for each range to be covered. Custom fields therefore are very flexible but by themselves cannot be changed by the users of *Projekt Repository*. Nevertheless they offer the possibility to implement generic fields to map generic properties that are read from an ontology.

The taxonomy store has clearly the advantage that it is fully integrated in SharePoint. This enables the user of *Projekt Repository* to interact with this feature easily. Nevertheless the taxonomy store is limited to handle taxonomies.

Even though taxonomies will suit for most schemes used a less limited approach is preferred.

Considering the discussed semantic features it seems logical to introduce a custom semantic field to fully represent ontologies. The implementation of this feature is discussed in Chapter 4.

Chapter 4

Design of the *Semantic Repository*

As already discussed in section 3.2 SharePoint offers different semantic features. Nevertheless none of them is capable of representing an ontology. This chapter describes the steps taken to implement an ontology store for Microsoft SharePoint.

The first section describes the structures that make the ontologies accessible within the program. The second section gives an overview of how the different features of SharePoint are used to enable the user to work with the ontology. Finally the last section describes a work-flow designed for the users to create a repository for their project by supplying an ontology.

After storing the research data in the repository it shall be retrieved using different techniques. The last two sections will describe how the usage of ontologies can influence and support different ways of data retrieval and search in the repository. The former shows different ways for the user to search in the system to retrieve data. The last section will describe how the information from the ontologies is used to give a machine readable output containing the data stored in the repository.

4.1 The *Semantic Repository*

Before being able to work with ontologies a representation was implemented in order to access the different parts of the ontologies. This representation was designed in such a way that the full spectrum of capabilities of ontologies can be used intuitively when programming applications that deal with ontologies.

Since the extensions of SharePoint need to be programmed in a .NET language a small research has been performed to find an already existing library for handling ontologies. Apparently all of the implementations found were either based on Java or have not been maintained for several years. Thus it was necessary to implement a .NET based API to access ontologies.

The presented API is based on the structure of OWL API, an open source Java API that is widely seen as a reference implementation of the OWL 2 format for the Java programming language [36].

4.1.1 OWL API for .Net

The OWL API for .Net offers several interfaces each of which aim to represent a specific part of an ontology. All of the interfaces are designed in a way that they can be implemented for arbitrary sources supplying the actual ontology. As a reference and for the further application for *Projekt Repository* every interface was implemented to comply with input formatted in the OWL 2 Syntax as presented in section 2.2.1. Also the conversion from XML Schema to an ontology was implemented as described in section 2.3.3.

Working with the OWL API for .Net

Besides the interfaces and classes for representing the ontologies, OWL API for .Net also offers structures for parsing input files such as an OWL 2 XML file. Again, as the interfaces for representation of the ontology, these interfaces are independent of the actual format and can be implemented for other sources than OWL 2 XML such as Turtle [26].

The interface `IOntologyParser` provides different functionalities to load an ontology from different sources. It therefore returns an instance of `IOntology` that in turn represents the specific ontology that was loaded from the file. At the current state of the development it is not possible to manipulate the ontologies structure once it was loaded from the file.

Interfaces defined by the OWL API for .Net

Among the `IOntology` interface the API defines the interfaces `IDataProperty`, `IObjectProperty` that represent the individual-datatype and the inter individual relations defined in the ontology. The interface `IAnnotationProperty` serves annotations that can be attached to any other interfaces of the library. These annotation properties usually contain meta information such as an explanation or a human readable or even localized name.

Classes are represented by the interface `IClass`. Apart from the sole definition of the class it also builds a hierarchy of the classes by giving a set of sub- and super classes as well as equivalent classes. Using the reference `Thing` to the top most class of an ontology the class hierarchy can be traversed. An instance of a class is represented by an implementation of the `IIndividual` interface.

The datatypes defined by the ontology are represented by the `IDataType` interface. A single value of such a datatype is represented by an instance of the `IValue` interface.

The ontology itself forms a structure that is composed of the parts that were already introduced in section 2.1.1 as depicted in Figure 4.1. Since the parts of `IOntology` closely relate to the formal definition it is easy to use and thus knowledge about formal ontologies can be used to operate the library.

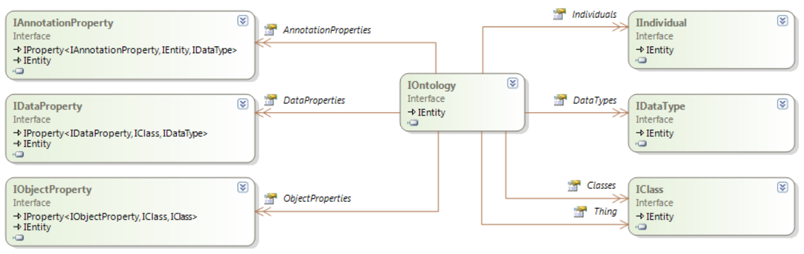


Figure 4.1: Class diagram of the OWL API of .Net

Usage Example

For *Projekt Repository* two implementations of the presented interfaces were developed. One for XML Scheme and the other for OWL 2 XML syntax as they were presented in sections 2.2.1 and 2.2.2 respectively.

In general both implementation follow the same pattern. At first an instance of a parser is created, which is then used to load the file and build up the ontology. The parsers implement the `IOntologyParser` interface defining function to load the ontologies from various sources such as files and memory streams. Listing 4.1 gives an example on how the OWL XML parser class is used in C# to load an ontology from a file.

```
OwlXmlParser p = new OwlXmlParser();
IOntology o = p.Load(@"PATH TO OWL FILE");
foreach (IClass c in o.Classes)
{
  Console.WriteLine(c.Label);
}
```

Listing 4.1: A code snippet showing how to parse an XML file containing an ontology in OWL syntax.

Listing 4.2 gives another example of working with classes. Instead of simply giving a flat list as in the previous example, this example indents subclasses when printing them. Analogous to the subclasses set every class has a superclasses set. These sets can be used to navigate through the class graph.

```
...
IOntology o = ...
PrintClasses(o.Thing, "\t");
...
private void PrintClasses(IClass c, string space)
{
  string nextSpace = '\t' + space;
  foreach (IClass childClass in c.SubClasses)
  {
    Console.WriteLine(childClass.Label);
    PrintClasses(childClass, nextSpace);
  }
}
```

Listing 4.2: A code snippet showing how to display a full class hierarchy from of an ontology.

The given examples show only a small portion of the possibilities offered by the API. There are more ways of dealing with classes, individuals, data and object relations. A full class diagram containing all the functions and properties defined by the interfaces can be found in Appendix A.

4.1.2 Ontology Requirements

To ensure that the SharePoint extension is able to work with ontologies some technical restrictions are posed on the ontology to comply with assumptions of the SharePoint extension. First of all the ontology is supposed to be in OWL 2 XML representation as presented in section 2.2.1. If the source ontology does not comply with this syntax it can be converted. Especially for ontology syntaxes with explicit semantics most editors can convert between formats [28]. For implicit semantics conversion can be performed as described in section 2.3.3. As the main purpose of the implementation of OWL for .NET is mainly used to get information about taxonomies and, compared to general description logics, quite simple metadata schemas the current implementation only supports the expressiveness of the most limited variant OWL-Lite: $\mathcal{SHIF}^{(\mathcal{D})}$. Unknown terms and definition in the input file will be ignored.

Apart from restrictions on the expressiveness some formal restrictions are imposed on the declaration of the classes. The URI of an entity should be unique within the ontology. Additionally, if available, the labels defined for the OWL entity will be used for on screen rendering.

4.2 Ontology Representation in SharePoint

In this section two issues of the implementation of ontologies for *Projekt Repository* are addressed. The first covers which parts of the ontology are used in the repository on a theoretical basis. The second covers the way of making this information available to the user.

Primarily the classes, datatype and object properties will be read from the ontology. Also these are the structures used for the semantic tagging. In general the following assumption will be made: Every uploaded file in the repository is an instance in the ontology the repository shall then enable the user to specify data and object properties as well as the instantiation relation.

The second problem encountered when trying to built ontology support for SharePoint was to find the means offered by SharePoint to make the information actually accessible for the user. Therefore for the features already discussed in section 3.1.2 it needs to be evaluated if they can be used to make certain features of the ontology accessible. Two different ways of accessing the ontology from SharePoint were selected to be further investigated for the development of the *Semantic Repository*:

Class instantiation or *is-a-relationship* is a very easy way of adding semantic information. For each uploaded file in a SharePoint list an additional field is offered. This field then can be filled with the name of one ore more classes. This method itself is inspired by traditional tagging with a fixed dictionary as it is done in many other applications already available.

List from ontology is another way of using the information provided by the ontology in the repository. This method uses the properties defined in the ontology and adds them as fields to a list. Using this method an ontology can be used to define and create the complete structure of a repository.

4.2.1 Class Instantiation

Compared to the expressive power of ontologies this way of working with them is quite constrained. Nevertheless this method of dealing with semantic information is quite common and thus it is easy for the user to adopt it.

For example class instantiation allows the user to specify a class for an uploaded file in the repository. Suppose the TBox defines a class called *LicenceDocument*. If the user has uploaded a file named *GPLLicense* the following statement can be added by class instantiation to the ABox:

$$\mathcal{A} = \{\ldots, GPLLicence : LicenseDocument\} \tag{4.1}$$

Apart from the usability issue the class instantiation approach can make use of high level techniques offered by SharePoint such as the taxonomy store. Classes are read from the ontology and then are added to a taxonomy in the taxonomy store. This allows the use of the taxonomy field types that are already available. Still it requires that the class structure is expressed as a taxonomy or needs to be converted in some way into a taxonomy without loosing to much information.

As a first approach it was chosen to do this transformation by doing a full breadth first traversal of the class graph from the ontology. Every class that is encountered is then added to the taxonomy store. After this procedure each of the classes is at a topmost position in the hierarchy and thus this approach results in the flattest possible hierarchy. In case a conflict occurs, that is a class with two or more parents, the flattest possible hierarchy will be preferred. If that ties the first sub class relation found in the file will be used.

In the course of the development of *Projekt Repository* it turned out that almost all of the structures used by the different researches are hierarchies. Since structures that were hierarchies in the first place will be stored exactly as defined, this approach seemed suitable for the conversion.

The information that was added to the items by specifying the classes can then be used to filter the list contents or for the retrieval of certain items from the repository as it will be discussed in section 4.4.

4.2.2 List from Ontology

The second approach that makes ontologies available in the repository uses more of the expressive power as the pure class instantiation. This approach aims at describing the items in the repository in more detail. This approach uses the object and datatype properties from the ontologies.

Given a that a TBox defines the following properties: a data property *releasedAt* connecting an instance and a date and an object property *hasLicense* connecting an instance and its license. Based on these relations it is now possible to define properties of an uploaded file such that:

$$\mathcal{A} = \{\ldots, (file, "2012-05-15") : releasedAt\} \tag{4.2}$$

Furthermore when combined with the information from equation 4.1 relations among different instances can be modelled such as:

$$\mathcal{A} = \{\ldots, (file, GPLLicense) : hasLicense\} \tag{4.3}$$

For the realization within SharePoint this gives the user the following possibility: They can specify the structure and the information that shall be provided for the items in the ontology and once this is done the repository automatically maps the values of the fields in a list to the properties of the ontology.

The ontology further defines the range of the field as such. For data properties the most common basic data types were manually matched from the data types offered in OWL to the data types offered by SharePoint. Table 4.1 gives a brief overview of the different types that were matched. All other data types are represented by SharePoint as a string. A special case is formed by the object properties as form them a special lookup field is generated that enables the user to select another item already contained in the list. This selection then initiates the relation among the files.

The information added by this approach can then be used by a reasoning engine to extract certain information such as the membership to some class that is restricted on the value of certain properties. In section 4.4 several ways of working with this kind of structured and tagged data are discussed.

4.3 Working with Ontologies

After introducing the different ways of accessing the information stored in the ontologies in SharePoint this section shows some practical examples of how the

SharePoint	**OWL 2**
Integer	xsd:integer
Number	xsd:double
Text	xsd:string
URL	xsd:anyURI
DateTime	xsd:dateTime
Boolean	xsd:boolean
Choice	owl:oneOf

Table 4.1: Field types from SharePoint and their counterparts from the OWL 2 specification.

different techniques are implemented in SharePoint. The first example shows the storage of ontologies in SharePoint. The second and the third example show how to work with ontologies based on the two methods *class instantiation* and *list from ontology* as presented in section 4.2.

4.3.1 Ontology Storage

Since the list is the most natural way of storing files and other information in SharePoint, the ontologies are stored in a dedicated list. Apart from the raw XML data of the OWL file the list also stores some meta information about the ontology such as its name and its description. This information is read from the ontologies annotation properties. When accessing the ontology it is parsed from the XML data and made accessible via the OWL API for .Net.

This way of storing the ontologies in a separated part of the SharePoint repository also allows to keep track of changes or updates of ontologies. Two ontologies with the same URI can be compared and, if only minor changes are present, can be updated. This update can also include the already created lists, property fields and taxonomy store entries. This allows the users to gradually update the stored ontologies as changes occur. This update process may already need some reasoning to match the instances tagged in the old version of the ontology to the new version as it will be addressed in chapter 5.

4.3.2 Instantiation

After the ontology is uploaded the user can decide to either create a SharePoint taxonomy from the ontology or create a list to store files. The first choice will immediately create the taxonomy and make it available as a field that can be manually added to an existing list whereas the latter will prompt the user to specify the properties that shall be added to the list. An example of the user interface for instantiating an ontology from within SharePoint is given in figure 4.2. An example of a resulting list is displayed in figure 4.3.

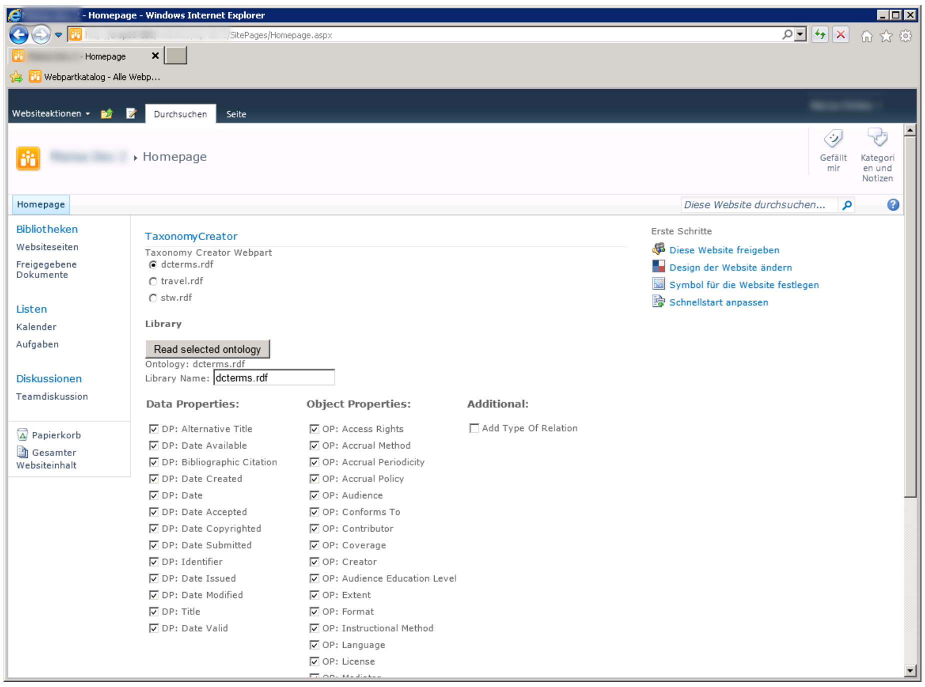
Type Property

Apart from the properties specified in the OWL file, the user may also chose to add the *rdf:label* property and the *rdf:type* relation to explicitly specify the class of an item. The *rdf:type* relation will then also create the taxonomy in the same manner as the class instantiation method would do.

Data and Object Properties

All other properties will be created as a custom field. For data properties the user may directly enter the desired values that will be range checked based on the type conversion in table 4.1. If the range check fails, the changes will not be saved in the SharePoint database.

In contrast to the data properties the object properties range is depending on the items already in the list: The *rdf:type* property may assign a class to another



TaxonomyCreator

Taxonomy Creator Webpart

(●) dcterms.rdf
( ) travel.rdf
( ) stw.rdf

Library

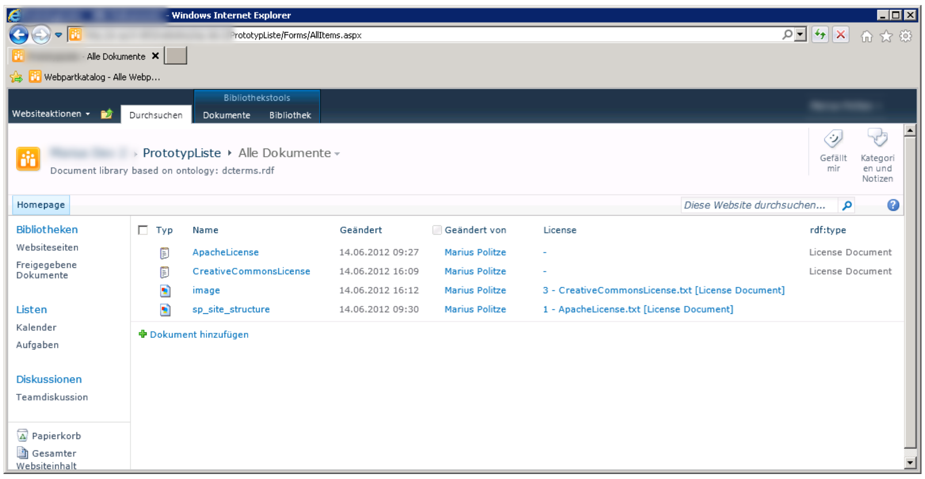
Read selected ontology

Ontology: dcterms.rdf

Library Name: dcterms.rdf

Data Properties:	Object Properties:	Additional:
☑ DP: Alternative Title	☑ OP: Access Rights	☐ Add Type Of Relation
☑ DP: Date Available	☑ OP: Accrual Method	
☑ DP: Bibliographic Citation	☑ OP: Accrual Periodicity	
☑ DP: Date Created	☑ OP: Accrual Policy	
☑ DP: Date	☑ OP: Audience	
☑ DP: Date Accepted	☑ OP: Conforms To	
☑ DP: Date Copyrighted	☑ OP: Contributor	
☑ DP: Date Submitted	☑ OP: Coverage	
☑ DP: Identifier	☑ OP: Creator	
☑ DP: Date Issued	☑ OP: Audience Education Level	
☑ DP: Date Modified	☑ OP: Extent	
☑ DP: Title	☑ OP: Format	
☑ DP: Date Valid	☑ OP: Instructional Method	
	☑ OP: Language	
	☑ OP: License	

Figure 4.2: Screenshot of the user interface created to instantiate an ontology within SharePoint. Shows the data and object properties that can be instantiated from the Dublin Core Metadata Set (left and middle column). Also shows an option to instantiate the type of relation (right column).



PrototypListe ▸ Alle Dokumente

Document library based on ontology: dcterms.rdf

Typ	Name	Geändert	Geändert von	License	rdf:type
	ApacheLicense	14.06.2012 09:27	Marius Politze	-	License Document
	CreativeCommonsLicense	14.06.2012 16:09	Marius Politze	-	License Document
	image	14.06.2012 16:12	Marius Politze	3 - CreativeCommonsLicense.txt [License Document]	
	sp_site_structure	14.06.2012 09:30	Marius Politze	1 - ApacheLicense.txt [License Document]	

Dokument hinzufügen

Figure 4.3: Screenshot of the list created by the ontology initiator. The list was populated with some example files which also define some of the properties from the ontology.

instance. This, in turn, may add the other instance as a possible candidate for an object property. Based on a custom field a lookup field was created that allows to edit this kind of property by selecting the target instance, see figure 4.4.

4.3.3 Export

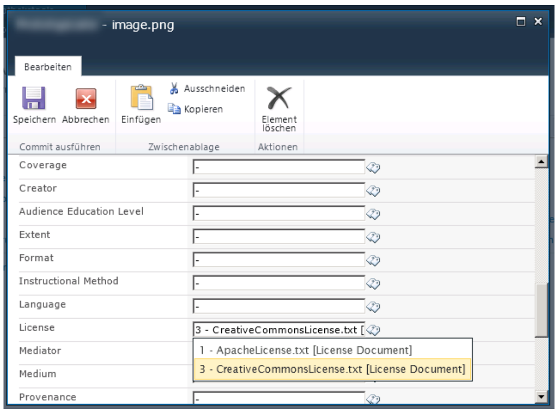
Once items are stored in the repository they are regarded as individuals belonging to some ontology. Therefore the export feature allows to get the OWL representation not only for the ontology but also for the contents of the repository. The resulting XML file can then be loaded into other software products to work with the contents of the repository.

A more detailed approach of how to use this representation is described in section 4.5.

4.4 Retrieval of Tagged Data

The retrieval of data is one of the most important functions that the different research groups developing *Projekt Repository* want to use. The most basic way of finding information within a large set of data is search.

Search usually defines a query as a string or some other human enterable representation. The search then leads to a set of items matching the query. In the following sections three different ways of defining a query in *Projekt Repository* will be presented: the full text query searches through the labels of all fields of the data stored, the field query restricts the full text query to single fields and the query by OWL class lets the user specify the query as an OWL class.



Coverage	-
Creator	-
Audience Education Level	-
Extent	-
Format	-
Instructional Method	-
Language	-
License	3 - CreativeCommonsLicense.txt [
Mediator	
Medium	
Provenance	-

1 - ApacheLicense.txt [License Document]
3 - CreativeCommonsLicense.txt [License Document]

Figure 4.4: Screenshot of the object property field in SharePoint. The field can be used to define object properties respecting their domain.

4.4.1 Full Text Query

A full text query is the most basic type of query that can be used for search. It will simply return all the elements that have the query term present in one of their fields. In case multiple terms are given only elements having both of them in one field will be retrieved thus a query term cannot be formulated to be split over the different fields of an element. However this process itself is widely supported by SharePoint but has another clear disadvantage: since all the fields have to be processed this type of search is usually quite slow or requires sophisticated indexing strategies to speed up the process.

4.4.2 Field Based Query

In contrast to the full text query the field based query is restricted to some field of interest. The amount of computing time needed to search through the items is reduced as only certain explicitly defined parts of the data need to be searched. This type of query is the first to take some advantage from the information of the ontologies.

As [37] suggests taxonomies are ideal as a basic structure that helps the user to formulate a query. Of course such information can also be obtained from ontologies as they offer a restriction on the values of certain fields. For this application, the most interesting restriction is the limitation to a small set of values for a certain field. Apart from search using a query term, this allows the system to suggest potential values of the field that can be selected by the user. This process is called filtering or faceted search. It allows the user to quickly change the restrictions on the items currently displayed by defining a very clear query.

Based on the examples from [38] and the techniques already offered by SharePoint itself an interface was created to allow the user to explore the data by filtering for certain values. It uses information about the type of the field from the ontology. Figure 4.5 shows screen shots of different interfaces that were implemented.

4.4.3 Query by Class Definition

The most powerful type of query is the query by class definition. This feature allows the user to specify a class definition in OWL or in SPARQL, a recursive acronym for SPARQL Protocol And RDF Query Language. It is a query language specifically designed to place queries to ontologies. Also SPARQL is a recommendation of the W3C [39]. However both ways of formulating the query need some reasoning capabilities to find the matching entities.

Nevertheless this type of query requires the user to know the query language. As the prospective users of *Projekt Repository* will usually have no computer science background, this query will most likely not be used directly for the project nor be accessible from the SharePoint front end.

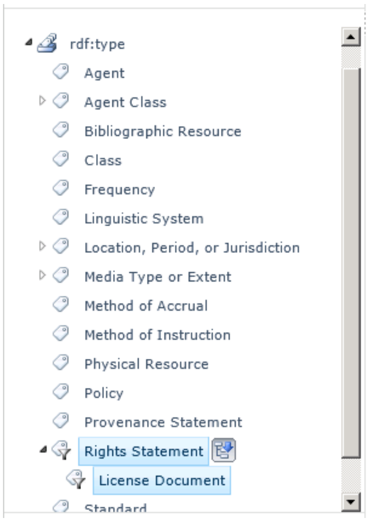


Figure 4.5: Different search facets generated by SharePoint.

4.5 OWL Representation

The OWL representation of a repository exports the contents of a repository, the metadata and its structure as an OWL file. This file can then be read directly by other tools dealing with ontologies, such as Protégé or other software agents, such as semantic search engines like Watson or Swoogle [40, 41]. As the export capability gives other agents the capability to access the contents of the repository it is a valuable tool to build multiple linked repositories which is a feature that is be considered for the future development of *Projekt Repository*.

4.5.1 Representation of Files Stored in the Repository

As described in section 4.2 there are two ways of dealing with the data from ontologies: class instantiation with files from a list and creation of a list structure based on the relations in the ontology. Nevertheless both of these ways regard one file as an instance, having properties that relate that instance to other instances or values of some data type. As already explained in section 2.2.1 these relations can be expressed by an ontology.

Since OWL is the representational language of choice for the project each of the files stored in the repository are represented as an *owl:NamedIndividual*. The URI, which is also the ID of the named individuals, is chosen from the URL of the file within the SharePoint repository. This connects the actual file and the exported metadata.

Furthermore this allows the export of the metadata to be independent of the actual access rights of the file itself. This preservation of access rights is especially important as access to the files stored in the repository may be restricted by legal or copyright issues and therefore permission to access the data itself may only be granted upon request or even payment. Still this approach makes it possible to publish and retrieve files based on their metadata. If a repository contains a file that is interesting access rights for a specific file can be granted from the copyright holder or owner of the repository.

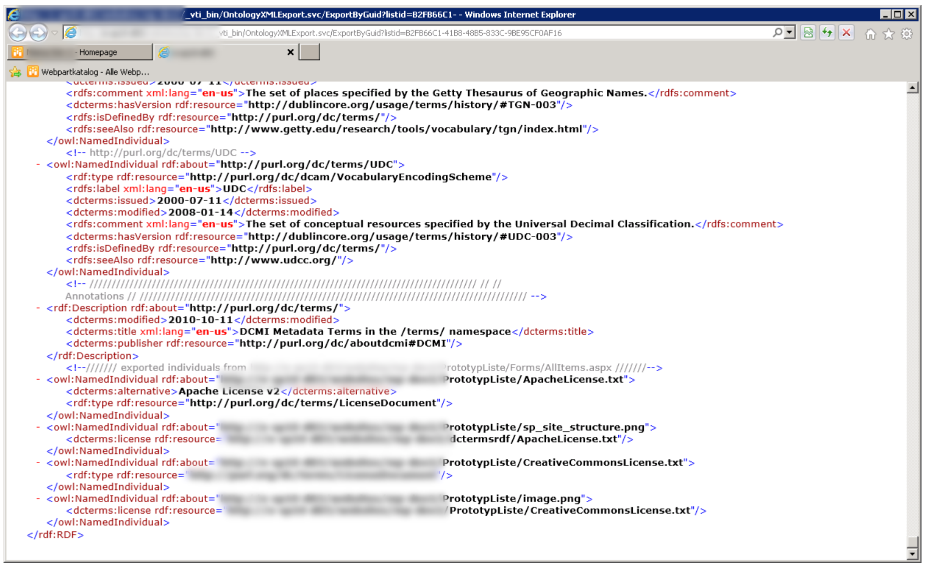
4.5.2 REST Web Service

As the data and metadata stored in the repository are subject to constant change it is important that the representation is always up to date. That is why the ontology representation of a repository is implemented as a REST service.

REST, Representational State Transfer, is a programming paradigm originally introduced by [42]. It describes a way to implement a web service that can be queried by computer programs as well as human users. REST itself does not define a concrete protocol but is usually implemented on top of the HTTP protocol which in turn is probably the most commonly used technology of the world wide web.

The concrete implementation of the REST service offers a URL for each repository that, when accessed via HTTP GET, returns the contents of the repository in OWL representation. The information is directly generated from the contents of the repository so that the user or software accessing the service will always receive the most recent state of the repository. Figure 4.6 gives an impression of the exported contend of the SharePoint list previously shown.

Apart from the information about the instances the web service also transmits the ontologies used to tag the metadata. Thus the OWL representation itself completely defines the ontology that is needed for an external observer to understand the information from the object and data properties of the individuals. This export can then be fed into the search index of the semantic web search engines.


[redacted]_vti_bin/OntologyXMLExport.svc/ExportByGuid?listid=B2FB66C1- - Windows Internet Explorer
[redacted]vti_bin/OntologyXMLExport.svc/ExportByGuid?listid=B2FB66C1-41B8-48B5-833C-9BE95CF0AF16
[redacted] - Homepage
Webpartkatalog - Alle Webp...

```
        <rdfs:comment xml:lang="en-us">The set of places specified by the Getty Thesaurus of Geographic Names.</rdfs:comment>
        <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#TGN-003"/>
        <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/terms/"/>
        <rdfs:seeAlso rdf:resource="http://www.getty.edu/research/tools/vocabulary/tgn/index.html"/>
    </owl:NamedIndividual>
        <!-- http://purl.org/dc/terms/UDC -->
  - <owl:NamedIndividual rdf:about="http://purl.org/dc/terms/UDC">
        <rdf:type rdf:resource="http://purl.org/dc/dcam/VocabularyEncodingScheme"/>
        <rdfs:label xml:lang="en-us">UDC</rdfs:label>
        <dcterms:issued>2000-07-11</dcterms:issued>
        <dcterms:modified>2008-01-14</dcterms:modified>
        <rdfs:comment xml:lang="en-us">The set of conceptual resources specified by the Universal Decimal Classification.</rdfs:comment>
        <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#UDC-003"/>
        <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/terms/"/>
        <rdfs:seeAlso rdf:resource="http://www.udcc.org/"/>
    </owl:NamedIndividual>
        <!-- /////////////////////////////////////////////////////////////////////////////////////// // //
        Annotations // /////////////////////////////////////////////////////////////////////////////////////// -->
  - <rdf:Description rdf:about="http://purl.org/dc/terms/">
        <dcterms:modified>2010-10-11</dcterms:modified>
        <dcterms:title xml:lang="en-us">DCMI Metadata Terms in the /terms/ namespace</dcterms:title>
        <dcterms:publisher rdf:resource="http://purl.org/dc/aboutdcmi#DCMI"/>
    </rdf:Description>
        <!--/////// exported individuals from [redacted]rototypListe/Forms/AllItems.aspx ///////-->
  - <owl:NamedIndividual rdf:about="[redacted]PrototypListe/ApacheLicense.txt">
        <dcterms:alternative>Apache License v2</dcterms:alternative>
        <rdf:type rdf:resource="http://purl.org/dc/terms/LicenseDocument"/>
    </owl:NamedIndividual>
  - <owl:NamedIndividual rdf:about="[redacted]PrototypListe/sp_site_structure.png">
        <dcterms:license rdf:resource=[redacted]dctermsrdf/ApacheLicense.txt"/>
    </owl:NamedIndividual>
  - <owl:NamedIndividual rdf:about=[redacted]PrototypListe/CreativeCommonsLicense.txt">
        <rdf:type rdf:resource=[redacted]"/>
    </owl:NamedIndividual>
  - <owl:NamedIndividual rdf:about="[redacted]PrototypListe/image.png">
        <dcterms:license rdf:resource=[redacted]/PrototypListe/CreativeCommonsLicense.txt"/>
    </owl:NamedIndividual>
</rdf:RDF>
```

Figure 4.6: Screenshot of the OWL XML export of the repository contents as seen in figure 4.3

Chapter 5

Ontology Matching

The central problem that is dealt with in this chapter is the matching of two different ontologies. This problem arises due to the fact that it is possible to define multiple ontologies for a single domain and it especially occurs when two research groups independently built an ontology or if an ontology changes over time. Especially the latter can occur if the repository is used for long time storage and needs to be accessed years after the actual project was finished. To automatically retrieve data using a query defined in one ontology it is necessary to find an alignment that connects the two ontologies.

This chapter will first introduce the ontology alignment problem in general. Then several statistical measures that are be used to measure structural properties of classes of ontologies are presented in the second section. In the last section an approach will be proposed that performs an ontology alignment based on the presented measures.

5.1 The Ontology Alignment Problem

Before proposing an algorithm to perform an ontology alignment the general process of ontology alignment is defined:

The goal of the alignment process is to produce a mapping **A** from an ontology o_1 to an ontology o_2. In general every entity in the ontologies might be matched. For a pair of entities e_1 from o_1 and e_2 from o_2 the mapping **A** determines if they match.

In general **A** can determine all kinds of relations among the entities such as equality, hierarchies or partial overlap or every other relation that can be expressed by an ontology. However for practical reasons in the following description will focus on defining the alignment problem for exactly matching classes. [43] defines the alignment problem for more general cases.

Definition 5.1. For an alignment **A** of two ontologies o_1 and o_2 an alignment function f is defined such that for every class c_1 from o_1 and some class c_2 from o_2:

$$f(c_1) = \begin{cases} c_2 & \text{if } \mathbf{A} \text{ exactly maps } c_1 \text{ to } c_2 \\ \perp & \text{otherwise} \end{cases} \tag{5.1}$$

The process of creating the alignment **A** can be based on several resources and parameters and thus is not unique in any way. However a reference alignment can be created which gives the correct mapping of the classes. For evaluation purposes this is done manually using domain knowledge.

5.1.1 Ontology Alignment Evaluation Initiative

To evaluate different matching techniques the Ontology Alignment Evaluation Initiative (OAEI) [44] offers an annual evaluation event where multiple approaches for matching ontologies are compared using different ontologies. The programs submitted during the last years mostly focussed on alignment by using the labels of the classes and properties and used structural measures mainly for partitioning [45] the ontology. With these approaches good results have already been achieved [45, 46].

Furthermore the OAEI offers a hub where papers describing the different matching approaches can be found. These papers work as a good start to gather information about current state of the art ontology matching techniques.

5.1.2 Matching Techniques

The similarity measures performed for the matching are divided into four classes by [45]: terminological, extensional, semantic and structural. Table 5.1 gives a short overview over the different approaches.

The terminological algorithms aim at matching the ontologies based on the labels or comments given in the ontology. For this purpose they use different proximity measures on strings such as n-gram or the edit-distance. A more advanced way of string matching can be performed when using a thesaurus

Type	Operates on	Sample algorithms
Terminological	strings, labels, comments	n-gram, edit/Levenshtein distance, WordNet
Extensional	(common) instances	Naïve Bayes, object (dis)similarity
Semantic	logical structure	rule-based or fuzzy inference
Structural	relational structure, hierarchies	graph similarity, structural proximities, descendant and sibling similarities

Table 5.1: Comparison of different approaches to match ontologies

to match synonyms. Instead of labels the extensional approach depends on instances that can be found in both ontologies. Based on this information data mining techniques can be used to find a similarity measure.

In contrast to the two content based approaches the semantic and the structural approach only use the information present in the pure ontology definition. A semantic reasoner may construct a model of the ontology and then match the model using a rule based or fuzzy logic approach. This step of semantic verification is often carried out as a separate process that requires user interaction to verify some of the matchings.

Finally the structural approach will only use the structures presented in the ontology. These structures can come from two sources: the class hierarchy and the relational structure presented by the different object properties. This class of algorithms makes use of measures for graph similarity which are also widely used in other disciplines in computer science such as computer vision and social network analysis [47, 48, 49, 50].

5.2 Structure Based Measures

In this section structure based measures will be closer examined. This section will first introduce the measures used to describe structural properties of the ontologies. The next section will focus on how these measures are used to match the entities of an ontology.

From the ontology a graph representation is constructed on which the structural measures then operate on. Originally some of the presented measures, centrality and modularity, are used for structural description of graphs and were neither designed for matching graph nodes nor ontologies.

5.2.1 Adjacency Matrix

To retrieve a graph from the definition of the ontology an adjacency matrix A_x is proposed where x denotes the object property that is used as the source for the graph. In this graph the classes of the ontology are used as a node. Classes connected by the property are transferred to nodes connected by an edge in the graph.

Definition 5.2. To define an adjacency matrix A_x based on an ontology o and an object property $x \in R$ the following steps are performed:
Without the loss of generality it is assumed that for every $c \in C$ there exists a bijective mapping I to an index such that $I : c \rightarrow \mathbb{N}$. Furthermore this mapping has an inverse mapping $I^{-1} : \mathbb{N} \rightarrow c$.
The adjacency matrix A_x is then defined as follows:

$$A_x^{ij} = \begin{cases} 1 & \text{if } i \neq j \text{ and } o \models (\alpha, \beta) : x \text{ for all } \alpha : I^{-1}(i),\ \beta : I^{-1}(j) \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

The definition above assembles A_x so the corresponding graph is directed and unweighted. Using this graph several statistical measures such as the number of neighbours can be used to determine its properties. The ontology matching algorithm will make use of these properties to match the graphs and therefore the classes of the ontology. In the same manner an adjacency matrix $A_{subClass}$ is proposed based on the class hierarchy which leads to definition 5.3.

Definition 5.3. To define an adjacency matrix $A_{subClass}$ based on an ontology o the following steps are performed:
Again the mapping from definition 5.2 is used to map classes to indexes of the matrix. The adjacency matrix A_x is then defined as follows:

$$A^{ij}_{subClass} = \begin{cases} 1 & \text{if } i \neq j \text{ and } o \models I^{-1}(i) \sqsubseteq I^{-1}(j) \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

Since the graph of $A_{subClass}$ is also directed and unweighted, $A_{subClass}$ is regarded as a special case of A_x. Nevertheless since it has the same properties the following techniques will only be discussed for an arbitrary A_x which may also be $A_{subClass}$.

5.2.2 Direct Neighbours

Using the definition of the graph of the ontology a direct neighbourhood of one class can be defined as the direct neighbourhood of the corresponding node in the graph. In [51] the definition of direct neighbours was successfully used for graph matching.

Definition 5.4. Given an adjacency matrix A_x and a class c from an ontology, the direct ancestors D of the class c are defined as the classes whose nodes in the graph have a common edge with c. Again the mapping I from definition 5.2 is used:

$$D_c = \{c' \in C | A_x^{I(c),I(c')} = 1\} \tag{5.4}$$

As a measure based on the direct neighbourhood the number of successors and the relative number of successors is defined.

Definition 5.5. Given an adjacency matrix A_x and a class c from an ontology and the mapping I from definition 5.2 the number of successors of a class c is defined as follows:

$$k_c = |D_c| = \sum_j A_x^{I(c),j} \tag{5.5}$$

Furthermore the relative number of successors is determined as follows:

$$k_c^{rel} = \frac{k_c}{\max_{c'} k_{c'}} \tag{5.6}$$

5.2.3 Extended Neighbourhood

Apart from the direct neighbours as defined in [51] a notion of the extended neighbourhood is proposed. The extended neighbourhood consists of all the classes that can be reached by any path from a given class c. Within this extended neighbourhood two kinds of classes are especially important: the classes that have no successors (leafs, sinks) and the classes that are not a successor of any other class (roots, sources).

Definition 5.6. Given an adjacency matrix A_x and a class c from an ontology and the mapping I from definition 5.2 a sink is defined as follows:

$$c \text{ is a sink iff } k_c = 0 \tag{5.7}$$

Definition 5.7. Given an adjacency matrix A_x and a class c from an ontology and the mapping I from definition 5.2 a source is defined as follows:

$$c \text{ is a source iff } \forall c' A_x^{I(c),I(c')} = 0 \tag{5.8}$$

which is equivalent to:

$$c \text{ is a source iff } \sum_i A_x^{i,I(c)=0} \tag{5.9}$$

For a measure of the structure of the extended neighbourhood the number of sinks s_c is counted. Furthermore the relative number of sinks s_c^{rel} is defined equivalently to definition 5.5.

5.2.4 Centrality

While the neighbourhood measures determine properties of single nodes in contrast to their direct successors the centrality measure assigns values relative to all the other nodes in the network. The notion of centrality is based on the centrality measure presented in [52] which in turn is based on the PageRank algorithm from [53].

The centrality measure uses a random walk model to determine classes that are referred most often which leads to definition 5.8. The underlying model assumes that one node is first randomly chosen. Then based on the of outgoing edges of the chosen node the probability is calculated that one of the connected nodes is chosen randomly. If there are no outgoing edges again another random node is chosen. This process is repeated until the probabilities of being chosen converge. Apart from the described iterative process [53] shows that the given probabilities can also be determined by the biggest eigenvector of the adjacency matrix.

After all the of centrality of a class provides a measurement of how likely it is for all other classes to be in a relation with the given class.

Definition 5.8. Given an adjacency matrix A_x from an ontology and the mapping I from definition 5.2 the centrality vector v is defined as follows:
Let V be the eigenvectors of A_x. Furthermore assume the eigenvectors v_i in V

are ordered according to the absolute value of the corresponding eigenvalue λ_i such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Then the centrality vector of A_x is defined as $v = v_1$.
The centrality z_c of a class c of an ontology is then defined as: $z_c = v^{I(c)} = v_1^{I(c)}$.

Definition 5.9. Given the centrality z_c of a class c of an ontology a relative centrality is defined as follows:

$$z_c^{rel} = \frac{z_c}{\max_{c'} z_{c'}} \tag{5.10}$$

5.2.5 Modularity

The modularity measure is based on the modularity algorithm in [52]. It determines the partitions the graph can be divided into by separating as few successor nodes as possible. Apparently [52] defines modularity only for undirected graphs. Nevertheless modularity respects connected nodes in general which suggests that the directed graph of the ontology is converted into a undirected graph.

Definition 5.10. The adjacency matrix A'_x of the undirected graph G' for the directed graph G is defined as follows:

$$A'_x = A_x + A_x^T \tag{5.11}$$

To fully define the modularity measure an intermediate matrix has to be defined.

Definition 5.11. Given an adjacency matrix A'_x and two classes c, c' from an ontology and the mapping I from definition 5.2 the expected number of relations with another class is calculated:

$$P_x^{I(c),I(c')} = \frac{k_c k_{c'}}{\sum_{ij} A_x'^{ij}} \tag{5.12}$$

The expected number of relations among two classes forms a model of the graph. The expectancy matrix P_x determines how likely it is that an edge is encountered among two nodes of the graph. This model is the basis for calculating the modularity of the graph.

Definition 5.12. Given an adjacency matrix A'_x and the matrix of expected edges among the nodes of the graph P_x we can define the modularity matrix B_x as:

$$B_x = A'_x - P_x \tag{5.13}$$

Using this modularity matrix we can now define a modularity value for each node determining two (or more) clusters of nodes. Nodes within the clusters are highly connected whereas there are as few connections as possible to nodes outside of the cluster. Apparently the exact solution is NP-hard, nevertheless [52] proposes an approximation method:

Definition 5.13. Given a modularity matrix B_x from an ontology and the mapping I from definition 5.2 the centrality vector v is defined as follows:
Let U be the eigenvectors of B_x. Furthermore assume the eigenvectors u_i in U are ordered according to the absolute value of the corresponding eigenvalue β_i such that $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$. Then the modularity vector of B_x is defined as $u = u_1$.
The modularity m_c of a class c of an ontology is then defined as: $m_c = |u_{I(c)}|$.

As done with the measures before also a relative modularity is defined:

Definition 5.14. Given the modularity m_c of a class c of an ontology the relative modularity is defined as follows:

$$m_c^{rel} = \frac{m_c}{\max_{c'} m_{c'}} \tag{5.14}$$

5.3 Structure Based Matching Algorithm

In this section a framework that uses the different presented measures to align two ontologies is proposed. Several variations of the proposed framework were implemented and then evaluated against the test sets of the OAEI in section 6. The presented structural measures are not designed to do graph or ontology matching with them. However the structural matching algorithm assumes that matching classes in the ontologies will have similar structural properties. This is used by the alignment algorithm proposed in this thesis.

5.3.1 Prerequisites

The proposed matching algorithm makes heavy use of the measures presented in definitions 5.5, 5.6, 5.9 and 5.14. To compensate for different sizes of the ontologies the relative measures are used. Apparently the relative measures cannot compensate big differences in size of the ontologies to be matched. Therefore the matching algorithm makes several assumptions:

The ontologies have about the same size. As already mentioned this requirement is based on the dependency of the presented measures on the graph size.

The ontologies deal with the same topic. Since the algorithm compares only structural measures ontologies with only a small overlap will not be aligned correctly — especially if the non-overlapping classes provide roughly the same structure it will lead to an incorrect alignment.

The ontologies have approximately the same granularity. If one of the ontologies models many details in a complex hierarchy whereas the other models the classes in a list the structure of the ontologies obviously cannot be matched.

The classes of the ontologies have a 1:1 matching or no match. The algorithm assumes that either two classes match or do not match. While a sub-graph to class mapping is quite realistic this option is not considered by the algorithm.

5.3.2 Alignment Algorithm

To perform the alignment of the ontologies the algorithm performs several steps on the classes of both ontologies:

1. For every class from o_1 and o_2 a value vector $n_c = (k_c^{rel}, s_c^{rel}, z_c^{rel}, m_c^{rel})^T$ is calculated. This vector is the basis for the matching process.
2. The algorithm generates a possible alignment.
3. (Optional) The alignment is checked for consistency. This step only allows alignments that meet a consistency rule:
   If two classes c_1 from o_1, c_2 from o_2 are aligned any successor class $c_1' \sqsubseteq c_1$ may only be aligned with c_2' if $c_2' \sqsubseteq c_2$.
4. Then for every pair of classes c_1 from o_1, c_2 from o_2 in the alignment a distance $d(c_1, c_2)$ is calculated. This distance is based on the value vectors n_{c_1} and n_{c_1}. The actual distance measure may be varied.
5. The quality of the alignment is based on the accumulated distance. As with the distance function itself the accumulation function may be varied.
6. (Optional) Perform local search and select an alignment from the neighbourhood of the current alignment. Continue at step 2 until the accumulated distance reaches a certain threshold or a maximum number of iterations is reached.
7. The algorithm returns the best alignment that was found according to the distance and accumulation function.

From the description of the algorithm it gets quite clear that there are many varieties that will inflict its performance. The main fields that should be considered are:

The distance measure. The distance measure is crucial for the solution to be found by the search algorithms. The distance measure for this special case can be any function $d : \mathbb{R}^4 \times \mathbb{R}^4 \rightarrow \mathbb{R}$. By convention distance functions should also meet the additional requirements such as $d(a, b) = 0$ iff $a = b$ and $d(a, b) > 0$ iff $a \neq b$.

The accumulation function. Since the distance function only judges the quality of a matched pair of nodes, the accumulation function needs to accumulate the distances of the node pairs to generate a notion of quality for the complete alignment. The accumulation function can be any function $s : \mathbb{R}^{\max(|C_1|,|C_2|)} \rightarrow \mathbb{R}$ where C_1 and C_2 are the sets of classes from ontology o_1 or o_2 respectively. Again the accumulation function should follow the convention that for two alignments a, b $s(a, b) = 0$ iff $a = b$ and $s(a, b) > 0$ iff $a \neq b$

The generation of a neighbourhood of alignments. For local optimization it is crucial that based on one alignment it is possible to generate neighbouring alignments which in turn can be inspected for their quality.

The technique used to (repetitively) generate alignments. This task can be performed by multiple search techniques. These approaches usually include the necessity to search in the alignments neighbourhood for alignments with a better accumulated distance.

5.3.3 Implementation

For the realization of the alignment algorithm the three modules mentioned in the previous section have to be implemented. For this thesis the selection is mainly motivated by some core characteristics of the ontologies:

- Ontologies are likely to differ in size. The measures used should compensate this.
- Ontologies are likely to have several thousand classes leading to combinatorial explosion.

Distance Measure

In general information retrieval tasks the cosine similarity is commonly used to measure similarity between vectors. Cosine similarity c of two vectors v and u is defined ad follows:

$$c = \frac{v \cdot u}{\|u\|\|v\|} \tag{5.15}$$

Cosine similarity has two desirable properties: It is robust against vectors of different length as the distance is based on the angle between the vectors and it has a fixed range from -1 to 1. Since all similarity measures are positive $c \in [0, 1]$. To convert the cosine similarity to a distance d the formula $d = 1 - c$ can be used.

Accumulation

The accumulation is done simply by adding the distances. Alignments with a smaller sum of distances will be preferred by the alignment algorithms.

Alignment generation

For alignment generation best-first-search is used. Even though in general anything but the first choice this approach is fast and therefore generates alignments of large ontologies in reasonable time. Nevertheless it is very likely to find only local optima, the general tendency is good enough to evaluate the quality of the presented similarity measures.

As a first extension to the best-first-search algorithm tabu search is used. Tabu search is a local search strategy that can be used to improve the results from best-first-search. To improve the result x from the result set X tabu search needs a function $S(u) \subset X$ giving the neighbours of a solution u. Tabu search furthermore maintains a *tabu list* T of solutions already visited. The function C gives the costs of a certain solution, whereas *optimum* gives the best solution from a set of solutions. The basic tabu search algorithm can is described by [54, 55] in the following way:

1. Select an initial $x \in X$ and let $x* := x$. Set the iteration counter $k = 0$ and begin with $T = empty$.
2. If $S(x) - T$ is empty, go to Step 4. Otherwise, set $k := k + 1$ and select $s_k \in S(x) - T$ such that $s_k(x) = optimum(s(x) : s \in S(x) - T)$.
3. Let $x := s_k(x)$. If $C(x) < c(x*)$, where $x*$ denotes the best solution currently found, let $x* := x$.
4. If a chosen number of iterations has elapsed either in total or since $x*$ was last improved, or if $S(x) - T = 0$ upon reaching this step directly from Step 2, stop. Otherwise, update T (as subsequently identified) and return to step 2.

To use tabu search for ontology alignment the elements X, $S(u)$ and C need to be properly defined. T, *optimum* and x then follow directly from these definitions. X is the set of all possible alignments whereas $x \in X$ is the alignment found by best-first-search. C is the distance based on the distance measure used for the alignment. The *optimum*-function will then select the alignment with the lowest accumulated distance. Finally $S(u)$ is defined by every alignment that can be reached by pairwise swapping of two aligned classes.

The second extension to the best-first-search is simulated annealing [56] which is also used for optimization of solutions generated by the best-first-search. Simulated annealing also is a local search algorithm that tries to enhance the given result by examination of single elements of the neighbourhood of the current solution $x \in X$. Based on a solution u simulated annealing needs to generate a sequence of neighbouring elements $s_k(u) \in X, k \in \mathbb{N}$. Furthermore the solutions are again evaluated using a quality measure C which again is based on the distance measure used. A function P gives an acceptance probability for the neighbouring solution. This probability depends on the quality of the current solution $C(u)$, the quality of the current neighbour $C(s_n(u))$ and a temperature t that is lowered slowly during the process. Simulated annealing then performs the following steps:

1. Select an initial $x \in X$ and let $x* := x$. Set the iteration counter $k = 0$ and begin with $t = t_max$.
2. Calculate $e = C(x)$ and $e' = C(s_n(x))$ for some $n \in \mathbb{N}$
3. With probability $P(e, e', t)$ set $x = s_n(x)$. If $C(x) < C(x*)$ set $x* = x$
4. If $t = 0$ stop. Otherwise, decrease t and return to step 2.

For the implementation of simulated annealing the function P was imeplemented such that:

$$P(e, e', t) = \begin{cases} 1 & \text{if } e' < e \\ \exp(\frac{e-e'}{t}) & \text{otherwise} \end{cases} \tag{5.16}$$

Neighbourhood

To search for local optimizations the alignments within a neighbourhood of a given alignment need to be evaluated. To generate the neighbouring alignments the aligned classes are swapped pairwise. Given the alignment below

$$f(c_1^1) = c_2^1, f(c_1^2) = c_2^2, f(c_1^3) = c_2^3 \tag{5.17}$$

its neighbourhood consists of three other alignments and can be generated as

$$f(c_1^1) = c_2^2, f(c_1^2) = c_2^1, f(c_1^3) = c_2^3 \tag{5.18}$$
$$f(c_1^1) = c_2^3, f(c_1^2) = c_2^2, f(c_1^3) = c_2^1 \tag{5.19}$$
$$f(c_1^1) = c_2^1, f(c_1^2) = c_2^3, f(c_1^3) = c_2^2 \tag{5.20}$$

Additionally to the plain swapping a consistency requirement can be placed on the validity of the neighbouring alignments: If two classes c_1 from o_1, c_2 from o_2 are aligned any successor class $c_1' \sqsubseteq c_1$ may only be aligned with c_2' if $c_2' \sqsubseteq c_2$.

However this consistency requirement will most often be violated by swapping classes. It is therefore necessary to extend the notion of the neighbourhood such that every successor class of the swapped classes will be realigned in a consistent way. In the implementation tested this realignment is done by performing a constant best-first-search on the successor classes.

Chapter 6

Experiments

In this section the previously described structures will be evaluated based on several experiments. The experiments were performed in three settings: At first the product was tested in a workshop together with the researchers of the other projects. Then the results of an ontology conversion from XML Schema to OWL are presented. The third test considers the matching algorithm described in chapter 5.

6.1 Workshop

To gain insight into the application of the *Semantic Repository* when used by the researchers of the different disciplines participating in *Projekt Repository* workshops were organized to test the software. In these workshops the researchers got to work with the product and had to comment on their impressions.

This first evaluation is based on the qualitative feedback obtained from the workshops. They were thematically focussed on creation and import of the ontologies as well as creating a repository structure from the ontology and finally the steps needed to store and tag data. The researchers were explicitly asked to give their feedback on how they think the *Semantic Repository* can be used for daily research activities. These workshops itself gave some valuable input on how to further improve the product. Figure 6.1 gives an overview of the testing process.

6.1.1 Workshop Activities

At first an ontology from the ones listed in section 2.3 was selected based on the discipline of the researcher. If for some reason the research group has no ontology previously defined a custom ontology is defined (1). The ontology was then reviewed using Protégé before being uploaded into *Projekt Repository* (2,3). Using the ontology, the two methods, class instantiation and list creation from ontology (see section 4.2) were used to add the semantic information to the repository (4,5). Last but not least several files were uploaded and tagged

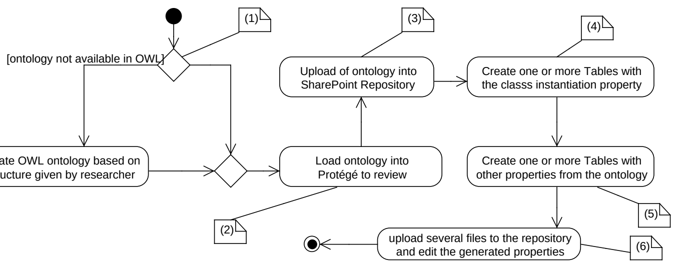

(1)
[ontology not available in OWL]
Create OWL ontology based on Structure given by researcher
(2)
Load ontology into Protégé to review
(3)
Upload of ontology into SharePoint Repository
(4)
Create one or more Tables with the classs instantiation property
Create one or more Tables with other properties from the ontology
(5)
upload several files to the repository and edit the generated properties
(6)

Figure 6.1: Activity diagram for the different action performed with the other researchers during the workshop. During all of the steps the comments were recorded.

within the repository to gain experience with the work-flow that will be mainly performed when working with *Projekt Repository* during research projects.

6.1.2 Workshop Evaluation

To gather the results and opinions of the researchers during the workshop two approaches were used: the Think-Aloud method during the activities and targeted questioning after the activities were finished. The Think-Aloud method is a securely established method for software testing and user interface design [57] that demands the users of the system to speak out loud what they think when using the system. Common phrases heard during this method are for instance:

- *I want to save the file so I click on the button that has the disk icon on it*
- *I am doing a right click on this entry to open the context menu but it does not open*

The targeted questions asked after the activities were focussed on the daily application of the *Semantic Repository* to store research data:

- *Compared to your current storage technique how well does the program integrate in your work-flow?*
- *What was the most interesting feature encountered during the workshop?*
- *What would you change to make the program fit your needs better?*

The evaluation of the workshop can obviously only be done on qualitative basis. There were several key points mentioned by the researchers which raised many ideas for enhancements of the product:

Protégé does not integrate well with the process Most of the researchers found it annoying to switch from the web based repository to a desktop application, or even have to install this application. This is not only a media disruption but also conflicts with the goal of the project to offer low-threshold access to the repository and its features. However the specification of the ontology and therefore the structure of the repository are usually only specified at in the beginning of a project.

OWL Export is interesting for round trip engineering The export feature was found quite useful not only for making the data accessible from outside of the SharePoint repository but also as a round trip engineering feature. An exported ontology is then loaded into a second repository. This allows tagging and referring of data across different repositories. The goal of a cross-linked repository was also already mentioned in the initial proposal for *Projekt Repository*. This feature of the *Semantic Repository* was regarded as an initial step for its realization.

The tagging process is intuitive and time efficient This is probably the most important feature that was noticed during the workshop. The impression of the whole tagging process was that is does not interfere with the research tasks of the users. This is quite important as the project can only be successful if it is used by the researchers to tag, store and work with their data.

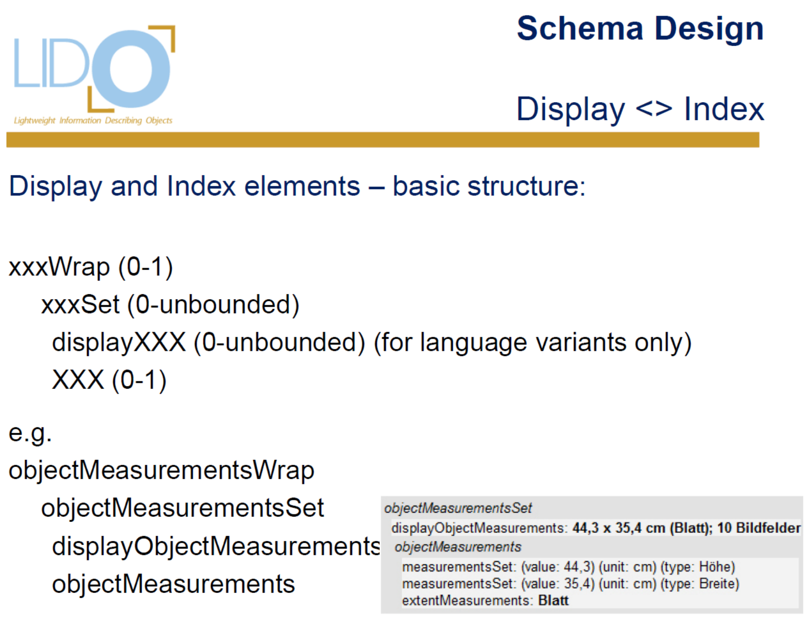
6.2 Ontology Conversion

In this section some considerations about the conversion from XML Schema to OWL are presented. To explain some of the phenomena an example from the LIDO metadata schema is presented. The considerations are mostly based on the way LIDO is structured, these structures are converted to OWL which in turn is interpreted by the SharePoint extension. They are therefore not of general nature but rather specific for the *Semantic Repository*.

In figure 6.2 the general structure of LIDO is presented to be formed by *Wraps* which in turn are formed of *Sets*. Thus multiple measurements, e.g. of the physical dimensions of an object, are arranged in a *MeasurementSet*. These measurements are tied to the object which contains the *MeasurementWrap*, that contains the actual *MeasurementSet*s.

In general this is a valid approach for structuring metadata. Using the conversion presented in section 2.3.3 each of the Wraps and Sets will be converted into a Class in the ontology. Additionally object properties will be generated to link the object and its metadata. For the example given in figure 6.2 the following TBox will be created:

$$\begin{aligned}\mathcal{T} = \{&LidoObject \sqsubseteq \exists hasMeasurementWrap.MeasurementWrap,\\ &MeasurementWrap \sqsubseteq \exists hasMeasurementSet.MeasurmentSet,\\ &MeasurementSet \sqsubseteq \exists hasMeasurement.Measurement,\\ &Measurement \sqsubseteq (\exists hasValue.\top) \sqcap (\exists hasUnit.\top) \sqcap (\exists hasType.\top)\}\end{aligned} \tag{6.1}$$


Schema Design

Display <> Index

Display and Index elements – basic structure:

xxxWrap (0-1)
  xxxSet (0-unbounded)
    displayXXX (0-unbounded) (for language variants only)
    XXX (0-1)

e.g.
objectMeasurementsWrap
  objectMeasurementsSet
    displayObjectMeasurements
    objectMeasurements

objectMeasurementsSet
displayObjectMeasurements: **44,3 x 35,4 cm (Blatt); 10 Bildfelder**
objectMeasurements
measurementsSet: (value: 44,3) (unit: cm) (type: Höhe)
measurementsSet: (value: 35,4) (unit: cm) (type: Breite)
extentMeasurements: **Blatt**

Figure 6.2: General structure of the LIDO metadata schema. Source: [58]

To fully model the example given in figure 6.2 various objects need to be defined in the ABox:

$$
\begin{aligned}
\mathcal{A} = \{&w : MeasurementWrap,\\
&s : MeasurementSet,\\
&m_H : Measurement,\\
&m_W : Measurement,\\
&l : LidoObject,\\
&(l, w) : hasMeasurementWrap,\\
&(w, s) : hasMeasurementSet,\\
&(s, m_H) : hasMeasurement,\\
&(s, m_B) : hasMeasurement,\\
&(m_H, 44.3) : hasValue\\
&(m_H, "cm") : hasUnit,\\
&(m_H, Height) : hasType,\\
&(m_B, 35.5) : hasValue,\\
&(m_B, "cm") : hasUnit,\\
&(m_B, Width) : hasType\}
\end{aligned}
\tag{6.2}
$$

While this again is completely valid and has the advantage that an OWL document generated using the converted schema is likely to be syntactically compatible with the original schema, since OWL can express anonymous objects, a problem arises when using the converted schema as an ontology in the *Semantic Repository*. All the intermediate objects have to be created as a list item by the user of the product. This is a protracted process especially due to the fact that the *Measurement* objects will most likely be used only once and therefore have to be created for every tagged object. This behaviour is quite inconvenient but could be improved by the detection of these intermediate objects. However this issue was out of the scope of the *Semantic Repository.*

6.3 Ontology Matching

To evaluate the matching process several experiments based on two of the datasets of the Ontology Alignment Evaluation Initiative were carried out. The experiments were then evaluated against the reference alignment given for the datasets.

6.3.1 Evaluation Measures

Even though the reference alignment can be used to automatically evaluate the performance of an alignment algorithm based on the number of direct matches, for the structural matching process this type of direct evaluation was not expressive as it did not allow giving a proximity how far the alignment of a class missed the actual class. Therefore the direct matching evaluation was extended to a measure that respects the distance between the proposed alignment and the reference alignment.

In accordance to the definitions in section 5.2 a proximity matrix P_x is defined. This proximity matrix is defined as follows:

Definition 6.1. The Proximity matrix P_x of a graph of an ontology is defined such that for every two of classes c, c' from the ontology:

$$P_x^{I(c),I(c')} = \min_{c'' \in N_c} P_x^{I(c''),I(c')} + 1 \tag{6.3}$$

$$\text{with } N_c = \left\{ c'' \in C | A_x'^{I(c),I(c'')} \neq 0 \right\} \tag{6.4}$$

This recursive definition can be informally expressed as $P_x^{I(c),I(c')}$ and it denotes the length of the shortest path between the class c and c' in the graph of the ontology.

In contrast to a simple comparison to the reference alignment, the distance based evaluation measure allows evaluating the quality of the alignment of a single class. Consider the following example: A class c_a is part of ontology A whereas classes c_b^1, c_b^2 and c_b^3 are part of ontology B. The proximity matrix of B is given by:

$$P_B = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix} \tag{6.5}$$

Together with a reference alignment any other alignment differing from the reference alignment can now be evaluated quantitatively. If c_b^1 is the reference alignment for c_a the following three cases can be observed:

- If c_a is matched to c_b^1 then the distance to the reference alignment of c_a is $d_{c_a} = P_B^{1,1} = 0$
- If c_a is matched to c_b^2 then $d_{c_a} = P_B^{1,2} = 1$
- If c_a is matched to c_b^1 then $d_{c_a} = P_B^{1,3} = 2$ respectively

Since many structural measures are almost equal for instance for sibling nodes, this distance based evaluation is especially useful for comparing structural ontology matches. Nevertheless it also preserves the notion of an exact match with the distance of 0. In general a higher measure can be regarded as a worse match.

Precision and Recall

Based on the introduced evaluation measure it is now possible to define a more flexible notion of precision and recall. This is especially useful since the structural matching algorithm cannot distinguish between nodes on the same level having the same properties, especially sink nodes. To compensate for this it is now possible to not only consider a correct match as aligned but also every sibling node of the matching node:

Definition 6.2. Given the sub sets of classes $C'_{o_1} \subseteq C_{o_1}$ from an ontology o_1 and $C'_{o_2} \subseteq C_{o_2}$ and $C''_{o_2} \subseteq C_{o_2}$ from an ontology o_2, a bijective alignment function $f : C'_{o_1} \to C'_{o_2}$ and a bijective reference alignment function $g : C'_{o_1} \to C''_{o_2}$
The precision measure p is defined as the fraction of relevant matches in the alignment such that

$$p = \frac{|\left\{c \in C'_{o_1} | P_{o_2}^{I(f(c)),I(g(c))} \leq k\right\}|}{|\left\{c \in C'_{o_1} | \exists c' \in C'_{o_2}, f(c) = c'\right\}|} \tag{6.6}$$

The recall measure r is defined in the same manner:

$$r = \frac{|\left\{c \in C'_{o_1} | P_{o_2}^{I(f(c)),I(g(c))} \leq k\right\}|}{|\left\{c \in C'_{o_1} | \exists c' \in C''_{o_2}, g(c) = c'\right\}|} \tag{6.7}$$

Where k can be chosen freely from $\mathbb{N}$. The higher k is chosen the fuzzier the quality measure will be. Nevertheless for $k = 0$ these notions of precision and recall are compatible with the classical expressions.

Definition 6.2 can also be less formally expressed as the ratio of the number of classes that are within k-steps to their reference class to the number of classes in the alignment from the algorithm (precision) or to the number of classes in the reference alignment (recall).

6.3.2 The Benchmark Dataset

The Benchmark dataset is an artificial test set that can be used to systematically determine strengths and weaknesses of an alignment algorithm. This set of

33 ontologies and reference alignments is one of the datasets provided by the OAEI. As the set is artificially generated each of the ontologies has their own speciality against which the algorithm can then be tested. For the evaluation of the proposed matching algorithm a subset of ontologies from the Benchmark dataset was chosen, see table 6.1.

6.3.3 Results on the Benchmark Dataset

The results generated for the Benchmark dataset were evaluated using the evaluation measure and precision and recall as described in section 6.3.1. All graphs shown in this section show only a selection of the experiments that were actually carried out. The full set of experiments can be found in appendix C.

Evaluation of Distance Measures

The first experiment using dataset 101 is used to check the general functionality of the alignment algorithm. Therefore the reference ontology is aligned with itself. The graph in figure 6.3 shows that the matching algorithm is independent of the similarity measure used capable of matching the ontology against itself. Since this basic requirement is met combinations of similarity measures were tested. The results of this experiment are displayed in figure 6.4. From this experiment it gets clear that the combination of the different measures is possible. However the true benefit of the diffent combinations of measures cannot be seen from this example but gets clear for the other datasets.

In a second experiment the ontologies 201 and 202 were aligned with the reference ontology. Figure 6.5 gives the results for ontology 201, whereas figure 6.6 gives the results for ontology 202. These ontologies differ from 101 in a way that all the names (201 and 202) and all the comments (202 only) were scrambled. Furthermore the order of the classes was altered. Since the matching of the classes is on structural basis only neither the order nor the labels should change the result significantly. Still when compared to figure 6.4 the differences are quite clear.

#	**Description**
101	Reference ontology. All other ontologies will be aligned against this one.
201	Same structure as reference alignment but all names are replaced by random strings. The order of class definition is shuffled
202	Same as 201 but comments were removed. Again shuffled differently.
221	Hierarchies are completely removed.
222	Hierarchy is flattened, some intermediate levels were removed.
223	Hierarchy is extended , additional levels are introduced.

Table 6.1: Properties of some ontologies from the Benchmark Dataset.

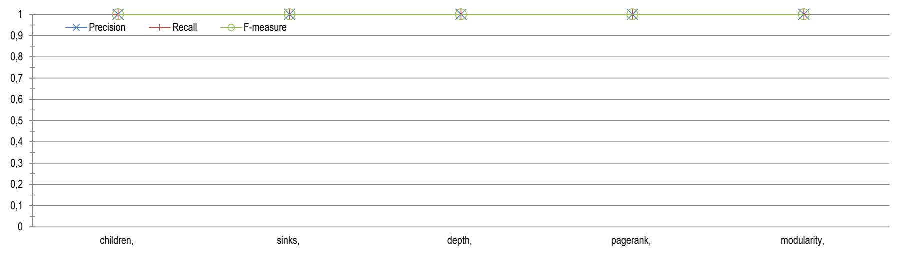


Figure 6.3: Results of the matching algorithm when matching the reference ontology 101 against itself using the different similarity measures.

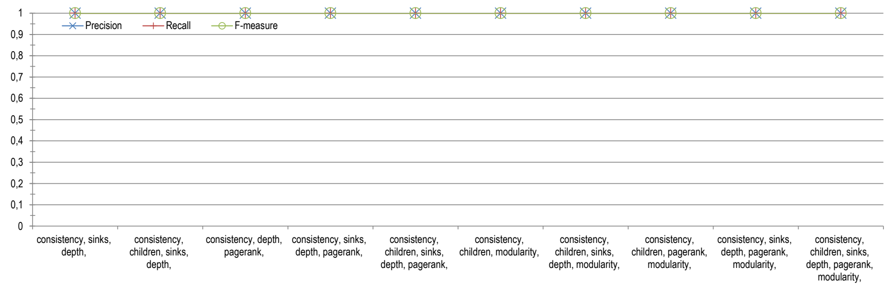


Figure 6.4: Results of the matching algorithm when matching the reference ontology 101 against itself using different combined similarity measures.

There are also noticeable differences between the results of 201 and 202. These differences can be explained by the order in which the classes are processed. This effect of different orderings will be examined closer in the following section.

Even though some results of the single measures are quite bad, the cumulation of distance measures performs quite reasonable in both cases.

Finally the three datasets are considered that have been changed in their structure. Dataset 221 has no hierarchy at all. This lead to a total failure of the alignment as the algorithm is dedicated to structural mapping. 222 was flattened by removing a class layer from the ontology. This leads to failure all measures that depend on the descendants of the node. The measure incorporating consistency, number of children, pagerank and modularity gives the best but not outstanding results, see figure 6.7. The best alignment of set 223 is even worse in quality: consistency, number of children, and modularity again give the best but not an outstanding result, see figure 6.8. When comparing to figure C.4, dataset 223 is the only one that performs better on average when turning consistency off.

Evaluation of Ordering

As already mentioned above the order the classes are processed in during alignment plays a significant role for the outcome of the alignment. Based on the

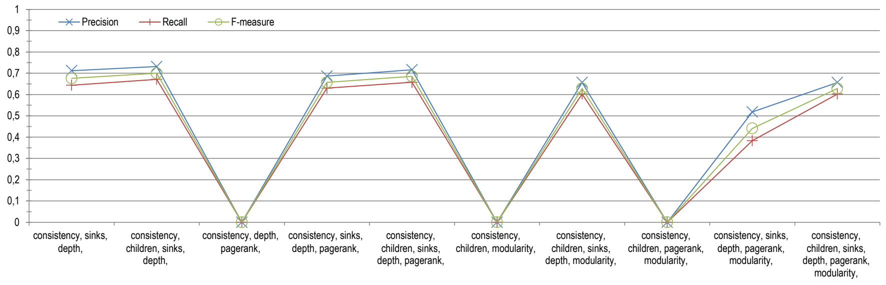


Figure 6.5: Results of the matching algorithm when matching 101 against 201 using different combined similarity measures.

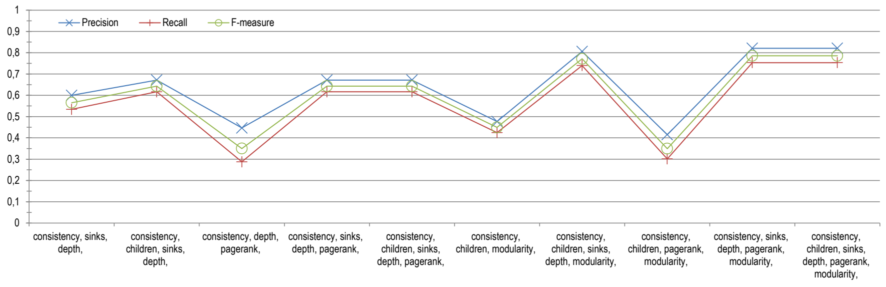


Figure 6.6: Results of the matching algorithm when matching 101 against 202 using different combined similarity measures.

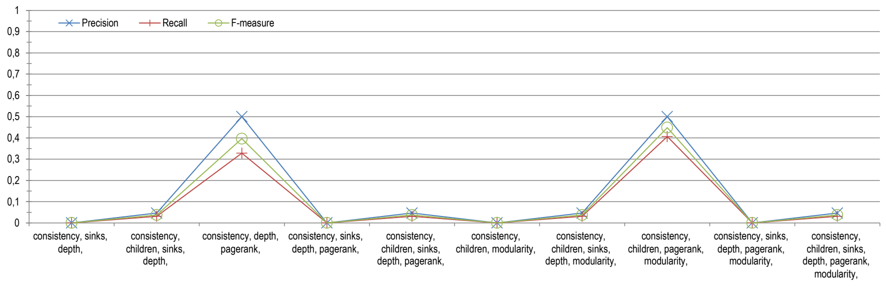


Figure 6.7: Results of the matching algorithm when matching 101 against 222 using different combined similarity measures.

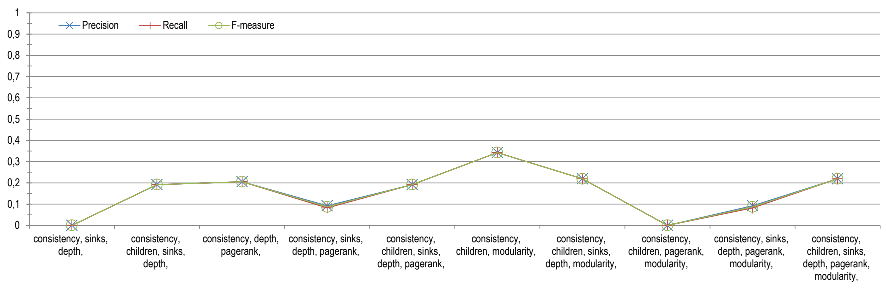


Figure 6.8: Results of the matching algorithm when matching 101 against 223 using different combined similarity measures.

dataset 201 different orderings are tested for their influence on the results. Finally the insights gained are used on sets 202, 222 and 223 for verification. Figure 6.9 shows a plot of the results using the fully combined distance measure (consistency, number of children, depth, number of sinks, pagerank and modularity).

Looking at the results the influence of the ordering shows a tremendous effect. However the ordering by pagerank seems to be the most beneficial for the alignment. Figure 6.10 shows that the ordering by pagerank increases also the result for 202 and, more importantly for dataset 222. Apparently no improvement can be obtained for dataset 223, however the quality stays approximately at the same level.

Evaluation of Tabu Search

Tabu search was proposed as an extension to the best-first-search used so far in section 5.3.3. To evaluate the performance of tabu search two experiments with a different number of iterations were carried out on the datasets. The first experiment performs only one iteration and thus only searches the direct neighbourhood of the first solution whereas the second approach does ten iterations.

Figure 6.11 shows the results when using a combination of the distance measures consistency, number of sinks, depth, pagerank. Especially for the datasets 222 and 223 it is very beneficial to use tabu search to enhance the result. However the other, already quite well aligned datasets do not profit much from the tabu search.

As a reference the results of the fully combined distance measure in combination with tabu search are given in figure 6.12. However these results show that tabu search is not always beneficial — in the case of dataset 223 the overall quality even decreases. In this example the fully combined distance measure obviously does not reflect the actual alignment correctly. However this can also be guessed by the first results for this measure in figure 6.8.

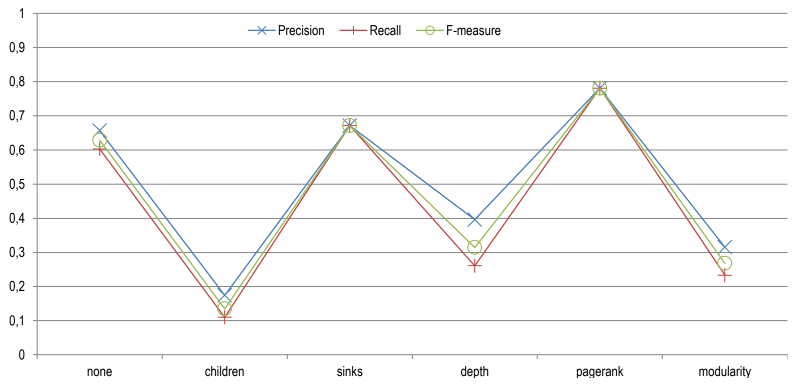


Figure 6.9: Results of the matching algorithm using when matching 101 against 201 using the different ordering of the classes. Before the alignment the classes were ordered according to the given criteria.

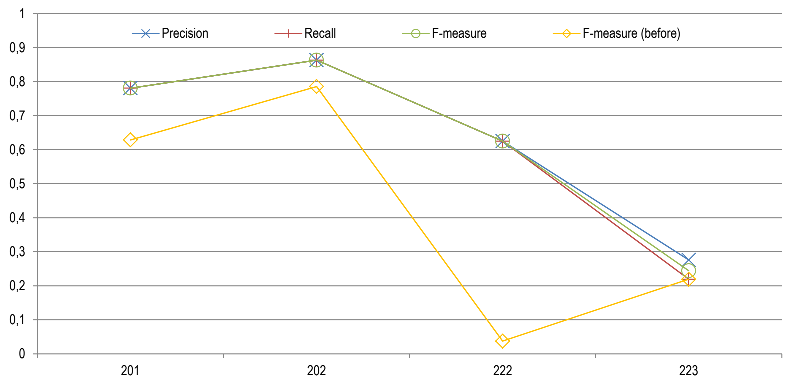


Figure 6.10: Results of the matching algorithm when matching 101 against the given ontology using the class ordering by pagerank.

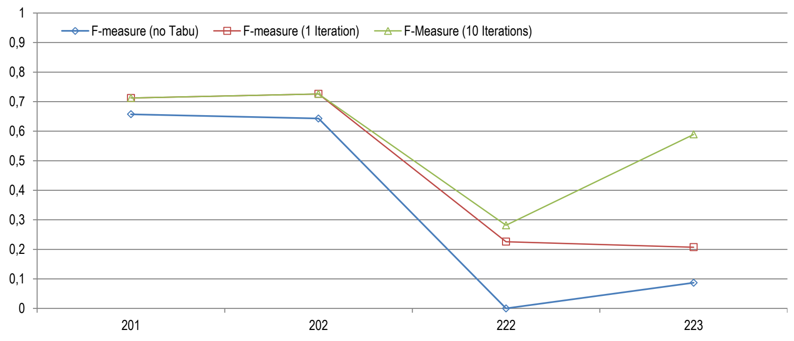


Figure 6.11: Results of the matching dataset 101 against the given ontology using consistency, number of sinks, depth, pagerank and tabu search with the given number of iterations.

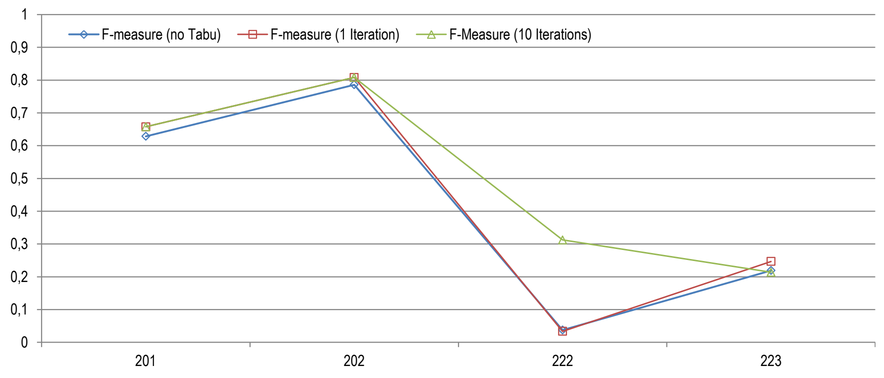


Figure 6.12: Results of the matching dataset 101 against the given ontology using the fully combined distance measure and tabu search with the given number of iterations.

Combination of Ordering and Tabu Search

Finally the combination of both techniques is examined. Compared with the results using only ordering only slight improvements could by obtained by combining both enhancements, see figure 6.13. However one slight improvement could be noticed at dataset 223. This effect is even present using a single iteration of tabu search.

Putting all the things together this combination of ordering by pagerank and one iteration of tabu search is the most stable and most promising approach for the different kinds of alignment problems.

Simulated Annealing

The second search technique tested is simulated annealing. Figure 6.14 depicts the performance of the algorithm compared to the tabu search with ordering. It is quite clear that simulated annealing is, without further optimizations, quite close to the results that were obtained when using tabu search.

Considerations

When looking at the results one may notice that precision and recall measures are very close together. While this is usually rather abnormal in the special case of the considered algorithm it is the expected behaviour. The best-first-search attempts to match as many classes as possible and is not limited by a threshold. Therefore all classes in the ontologies will be aligned to some class. Only if consistency checks fail for every remaining class these classes are not aligned and therefore precision and recall deviate.

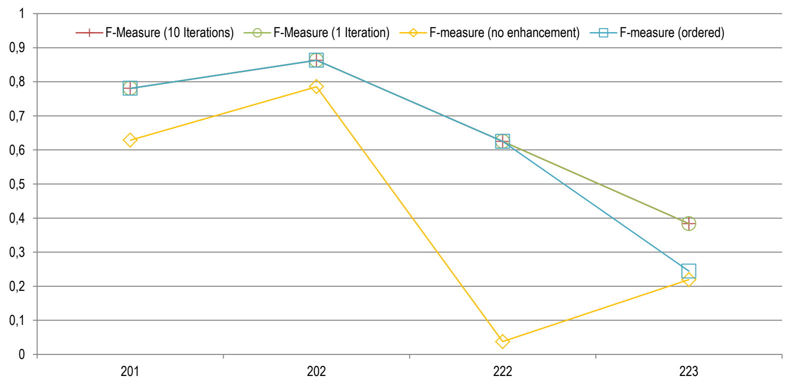


Figure 6.13: Results of the matching dataset 101 against the given ontology using the fully combined distance measure, ordering by pagerank and tabu search with the given number of iterations.

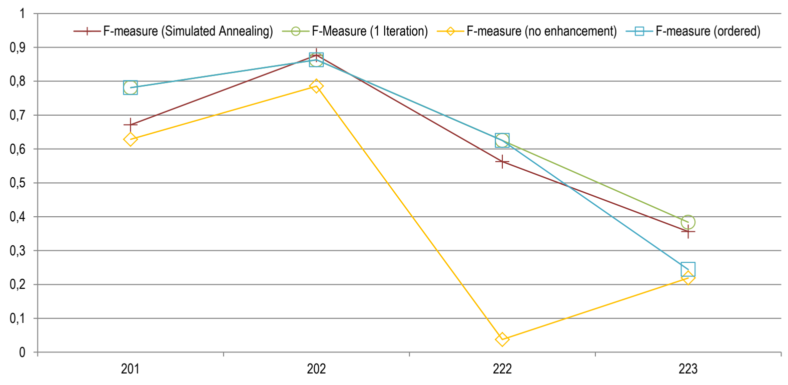


Figure 6.14: Results of the matching dataset 101 against the given ontology using the fully combined distance measure and simulated annealing.

6.3.4 The Anatomy Dataset

The Anatomy dataset consists of two ontologies. One is describing the anatomy of an adult mouse whereas the other one describes the human anatomy. Both ontologies describe relations among the anatomical parts, especially a *part of* relation is used to describe anatomical sub parts. Both ontologies contain about 3000 classes and are therefore quite large.

Apart from the ontologies the OAEI committee provides a reference alignment that is based on expert knowledge. This alignment is used to check the performance of the implemented mapping algorithm.

6.3.5 Results on the Anatomy Dataset

In this section some results of the matching algorithm presented in section 5.3 are presented. All the results shown were evaluated with the previously presented distance based measure.

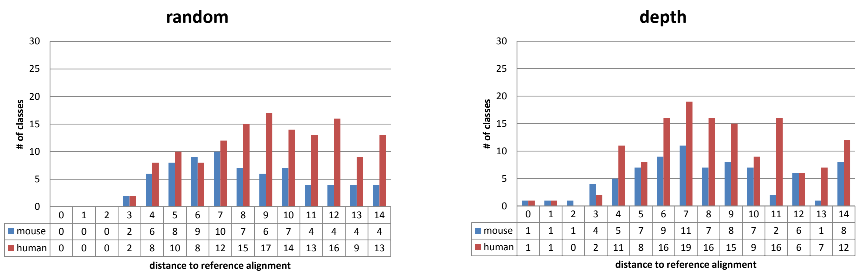
Similarity measures

In a first approach the presented similarity measures are evaluated based on the quality of the alignment produced when only using the discussed similarity measure for a greedy alignment. Figures 6.15, 6.16 and 6.17 show the results using the presented evaluation method. To provide a better overview the maximum distance in the graphs is limited to 14. Table 6.2 gives more details on the statistics of the results presented. These statistics are based on the complete alignment. As a further reference among the results generated using the different similarity measures a random alignment generated from the mean of 50 trials is presented in figure 6.15.

As a general tendency it can be observed that either similarity measure contracts the distribution and shifts the mean to a smaller distance. Both properties are desired nevertheless the effect of the single measures is quite small. To gain a maximum effect of the different measures they need to be combined in order

Measure	**Min**	**Max**	μ	σ	**Min**	**Max**	μ	σ	Σ
Random	3	18	8.0	6.7	3	24	12.3	6.4	102
Depth	1	18	8.0	3.8	1	24	9.3	5.5	216
Children	0	16	7.4	4.0	0	24	10.4	5.4	161
Sinks	0	14	6.5	2.9	0	22	8.5	5.8	89
Pagerank	0	18	8.0	3.8	0	24	9.2	5.5	203
Modularity	2	18	8.1	3.7	2	23	8.5	5.3	307

Table 6.2: Statistical measures for the distribution of the distances to the reference alignment for the single measures. The left half of the table shows the minimum and maximum distance as well as the mean (μ) and the standard deviation σ for the distances in the mouse ontology whereas the right half shows the respecting values for the human ontology. The column Σ denotes the total number of class alignments found.



random

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
🟦 mouse	0	0	0	2	6	8	9	10	7	6	7	4	4	4	4
🟥 human	0	0	0	2	8	10	8	12	15	17	14	13	16	9	13

depth

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
🟦 mouse	1	1	1	4	5	7	9	11	7	8	7	2	6	1	8
🟥 human	1	1	0	2	11	8	16	19	16	15	9	16	6	7	12

Figure 6.15: Distance to the reference alignment in the given ontology when using random assignments (mean over 50 trials, left) and the depth measure (right) for the assignment.

to compensate for their individual weaknesses. However before investigating the effects of combination of the measures some of the properties of the single measures will be pointed out at first.

The most obvious difference of the different outcomes of the experiments is the total number of correctly or closely aligned instances. It turns out that even when approximately equally distributed the total number of classes which were actually connected to the reference alignment varies greatly over the different measures used. Figure 6.16, for example, shows the difference in the number of classes that could be at least closely aligned quite clearly.

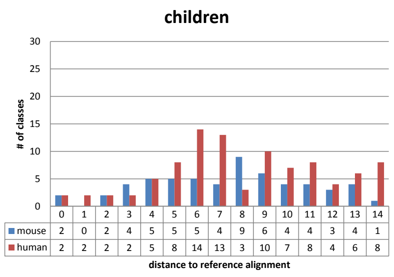
Figure 6.18 (left) shows the distance distribution when using all of the proposed similarity measures. As already proposed it is quite obvious that the different measures, when used together compensate each others weaknesses and produce a quite reasonnable alignment. Table 6.3 provides some more detailed statistical information for the distribution shown. What can be seen clearly is that the number of close matches increased and also for the first time a noticeable amount of correct matches was found.

Consistency Check

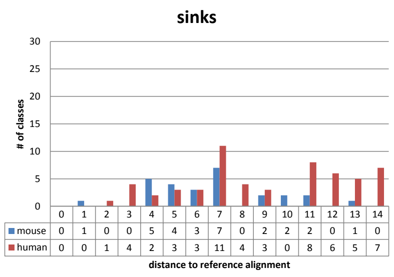
Among the different similarity measures for the classes in section 5.3 a special consistency check was introduced. This would allow the alignment of two subclasses only if at least one of their superclasses were already aligned with each other. Figure 6.18 (right) shows the distance diagram for the alignment using all similarity measures and only allowing consistent alignments. The statistics in table 6.4 shows the different measures for the consistent alignment according to table 6.2.

Measures	**Min**	**Max**	μ	σ	**Min**	**Max**	μ	σ	Σ
Combined	0	19	7.5	3.7	0	25	9.3	5.4	310

Table 6.3: Statistical measures for the distribution of the distances to the reference alignment when accumulation all similarity measures.

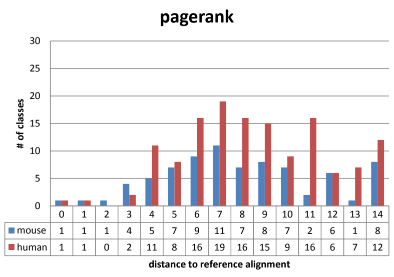


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
🟦 mouse	2	0	2	4	5	5	5	4	9	6	4	4	3	4	1
🟥 human	2	2	2	2	5	8	14	13	3	10	7	8	4	6	8

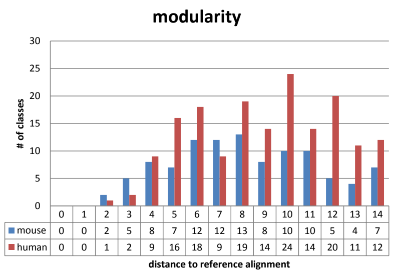


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
🟦 mouse	0	1	0	0	5	4	3	7	0	2	2	2	0	1	0
🟥 human	0	0	1	4	2	3	3	11	4	3	0	8	6	5	7

Figure 6.16: Distance to the reference alignment in the given ontology when using number of children (left) or number of sinks (right) for the assignment.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
🟦 mouse	1	1	1	4	5	7	9	11	7	8	7	2	6	1	8
🟥 human	1	1	0	2	11	8	16	19	16	15	9	16	6	7	12



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
🟦 mouse	0	0	2	5	8	7	12	12	13	8	10	10	5	4	7
🟥 human	0	0	1	2	9	16	18	9	19	14	24	14	20	11	12

Figure 6.17: Distance to the reference alignment in the given ontology when using pagerank (left) or modularity (right) for the assignment.

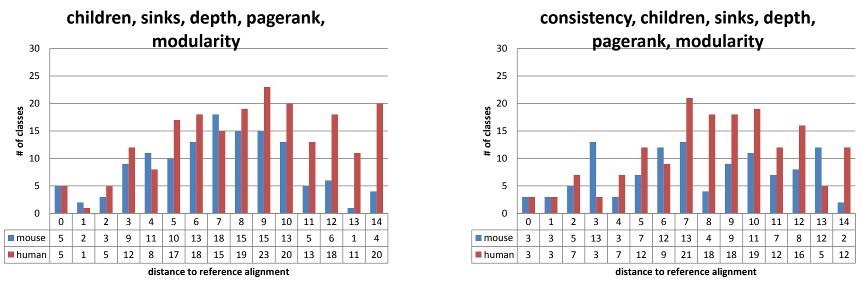
For all of the single distance measures as well as the combined distance measure the results show that the average distance gets a bit smaller when compared to the results without the consistency check. Even more interesting is the number of classes that actually were aligned by the algorithm. Generally the consistency check leads to a smaller number of classes aligned. Since the average distance is also decreased the overall quality of the consistent alignment is higher.

6.3.6 Evaluation of the Presented Results

The results presented in this section focus on three topics: evaluation of the structural distance measures, evaluation of search techniques to produce an ontology alignment and the performance of the alignment algorithm presented on a real world dataset.

Especially in the evaluation of the Anatomy dataset it becomes clear that the single distance measures do have some relationship with the alignment of the ontologies. However it is shown in the Benchmark dataset, that the performance can be greatly increased by combining the single measures.

The evaluation of the algorithm on the Benchmark dataset has also shown that it is very sensitive to the order in which the classes are processed. Ordering the classes by pagerank achieved the best results. Doing local optimization of



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
🟦 mouse	5	2	3	9	11	10	13	18	15	15	13	5	6	1	4
🟥 human	5	1	5	12	8	17	18	15	19	23	20	13	18	11	20

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
🟦 mouse	3	3	5	13	3	7	12	13	4	9	11	7	8	12	2
🟥 human	3	3	7	3	7	12	9	21	18	18	19	12	16	5	12

Figure 6.18: Distance to the reference alignment in the given ontology when using all measures with equal weight (left) and the same setting with the additional consistency check (right).

Measure	Min	Max	μ	σ	Min	Max	μ	σ	Σ
Depth	0	15	7.8	4.0	0	21	11.1	5.6	105
Children	1	15	7.3	3.8	1	21	10.6	5.4	56
Sinks	3	13	6.3	3.1	3	19	9.9	4.4	40
Pagerank	0	20	7.8	4.0	0	23	11.0	5.6	105
Modularity	3	17	8.9	3.9	3	24	12.3	5.0	98
Combined	0	18	7.1	3.2	0	25	11.2	5.7	290

Table 6.4: Statistical measures for the distribution of the distances to the reference alignment for the different measures when also checking consistency.

the alignment using tabu search further enhanced the results. Also simulated annealing gave reasonable results even though not as good as the combination of ordering, best-first search and tabu search.

6.3.7 Final Considerations

When comparing the presented results to the results of other algorithms it becomes quite clear that the overall performance of the pure structural matching is not satisfying on both test sets. Other algorithms using word similarity measures such as [46] on average perform much better on the artificial examples from the Benchmark dataset as well as the more real world anatomy dataset.

Nevertheless the experiments have shown that structural matching usually will find a neighbourhood class of the real alignment. This could be used to help to reduce the search space or partition the classed of the ontology for other approaches and thus increase the overall quality of the alignment.

Chapter 7

Conclusion

This chapter concludes the research done for this thesis. It will first give a short overview on the topics that were dealt with by answering the research questions and the problem statement. At the end some ideas are given that show promising directions for future research.

7.1 Answering the Research Questions

In section 1.3 five research questions are given. These questions will now be answered according to the results of the proceeding chapters and especially the results of chapter 6.

- *Which structures of Microsoft SharePoint can be used to represent ontologies?*

In chapter 3 a closer look was taken at the state of the art technologies available in Microsoft SharePoint. SharePoint itself only offers the Taxonomy store, a component capable of using simple taxonomies for tagging. However the notion of taxonomies usually is not enough. SharePoint also offers several basic interfaces like custom fields, services and web parts that can be extended to represent ontologies as it was done in chapter 4. All of these components find their data basis in OWL for .NET, a program library especially implemented for the *Semantic Repository* to work with ontologies from within .NET programming languages.

- *How can these structures be used to tag data saved in the repository?*

As already mentioned a basic tagging functionality was already available in SharePoint. Nevertheless most of the functionality had to be adapted in order to work with ontologies. In chapter 4 several ways to instantiate a tagging system based on an ontology are presented: class instantiation, a method closely related to traditional tagging and the list from ontology method that sets up the complete structure of the SharePoint repository based on an ontology. The two approaches were tested for usability together witch researchers from different fields and were found convenient to use for daily routines.

A lot of different formats for storing metadata were encountered during the implementation of the *Semantic Repository*. For most formats giving explicit semantics there exists a standard way to convert between formats. However this task is more complicated when no explicit semantics are defined. Among other ways of representing metadata schema ontologies offer the widest expressiveness and therefore can represent metadata schemas defined in any other format. While it is formally possible to transfer some of these metadata schemas into an ontology the results in section 6.2 show that this conversion is not always satisfying for the application in the project.

- *How can different retrieval techniques be used to retrieve data from the repository?*

The retrieval of the data is one of the final considerations for the *Semantic Repository*. After all three retrieval techniques are proposed in section 4.4: Full text search as the most basic variant, faceted search that can greatly take advantage of the formal definition of the ontology as well as an Export into the OWL format to make the metadata available in semantic web search engines an other programs.

- *Can structural measures describe the elements of an ontology?*

In chapter 5 different measures to express the structural properties of classes are proposed. Using these measures the classes in an ontology can be described and classes having similar structural properties can be found in the same but also in other ontologies. How these measures and their combinations relate to ontology alignments was checked against reference problems of the OAEI in section 6.3. Some of the measures describe the class better than others, yet they can be combined to lead to the best structural description.

- *Can different ontologies be matched using only their structure?*

From the experimental results in chapter 6 it gets quite clear that only based on the structural measures presented, ontology alignments can be performed. However it could also be observed that the structural alignment algorithm is very sensitive to the order in which the classes are processed. This issue could partially be avoided by using search algorithms, tabu search and simulated annealing, to optimize the result. Compared to other alignment algorithms the structural alignment algorithm presented does not lead to satisfying results. Still it is possible to enhance the algorithms and to combine them with other similarity measures to boost the overall performance.

7.2 Problem Statement and the *Semantic Repository*

The overall goal of the thesis in the context of *Projekt Repository* was to develop an extension to Microsoft SharePoint 2010 that allows researchers of different disciplines to tag multimedia data in different user defined ontologies. Among the theoretical basis for this extension the *Semantic Repository* was implemented, rolled out and tested with the researchers.

A first prototype of the product was released during the thesis and has already been evaluated as stated in section 6.1. Nevertheless there are still some open practical considerations in many details of the product that will be addressed in the further proceeding of *Projekt Repository*.

7.3 Future Research

Finally some directions of research are given that point to future challenges that can be addressed based on the issues discussed in this thesis. At first some of the open practical considerations are presented for the *Semantic Repository*. The second section points at improvements of the ontology alignment algorithm. The final section presents some prospects of the Semantic Web.

7.3.1 Practical considerations

Based on the presented prototype it is now possible to collect user experiences when working with the system. The prototype will be improved to provide a low-threshold access method to create and maintain semantic data for a variety of researchers. The future challenges before finally introducing the product are:

- Building a more seamless integration of ontology creation and handling in the SharePoint UI to provide a low-threshold to access the semantic tools.
- Adoption of features already supplied by SharePoint like advanced search techniques, indexing and custom data providers.
- Further tests to raise usability and acceptance of the system to be able to implement it for more research groups at RWTH Aachen University.

7.3.2 Ontology Alignment

The different measures for structural ontology matching were based on simple statistical measures and rather trivial approaches for the matching function. Also the algorithm was focussed solely on aligning the ontologies based on their structure.

It is very well possible that the presented similarity measures can be extended by other, more sophisticated, measures from graph theory. Furthermore the alignment algorithm would profit from the use of a more advanced search technique to find the best possible alignment as the best first search used for the experiments turned out to be quite prone to the order in which the classes are processed. However the improvements achieved by using tabu search and simulated annealing show that it is very likely to find more robust alignments using other search techniques.

The combination of structural and terminological or semantic alignment approaches will most probably lead to better results as the weaknesses and strengths of the different approaches are likely to compensate. The combination of multiple alignment techniques could for instance be achieved by majority voting or learning algorithms.

Also during the thesis research the area of social network analysis drew some attention as it also covers different graph based algorithms. The task to find correspondences in different social networks is very close to alignment of ontologies. Looking at this field might as well be rewarding as this area of research is very active.

7.3.3 Semantic Web

Using the *Semantic Repository* a user interface to the Semantic Web was developed — allowing users to interactively make semantic definitions. Maybe this application can reduce the fear of contact with the Semantic Web, OWL and RDF as it was stated by Tim Bray in one of his blog entries [59]:

> [. . .] RDF has ignored what I consider to be the central lesson of the World Wide Web, the "View Source" lesson. The way the Web grew was, somebody pointed their browser at a URI, were impressed by what they saw, wondered "How'd they do that?", hit View Source, and figured it out by trial and error.
>
> This hasn't happened and can't happen with RDF [. . .]

However the need to retrieve information from a tremendous and still growing mass of data as well as the increasing number of internet services offered by companies and governments raise the need for semantic tagging as the number of alternatives and choices overwhelms the intellectual grasp of the user. This development will make the Semantic Web inevitable and therefore boost the meaning of ontologies as a structural interchange format among agents.

Appendix A

OWL API for .Net Class Diagrams

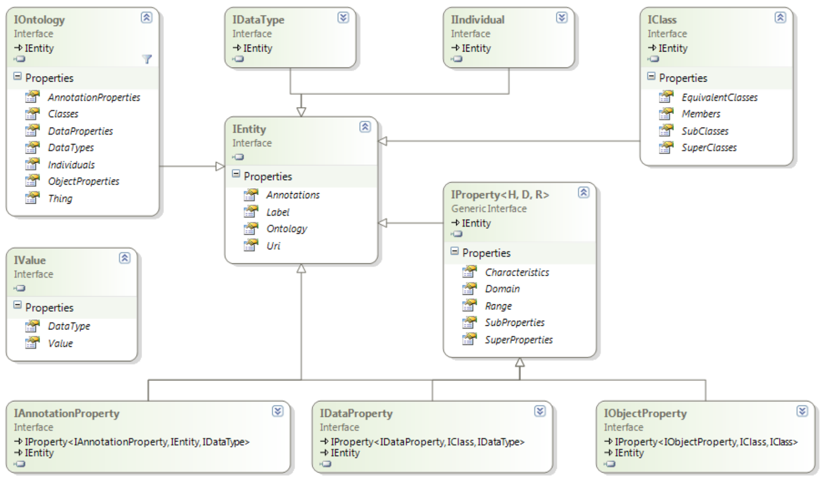


Figure A.1: Class diagram of OWL API for .Net showing the inheritance relations of the implemented interfaces.

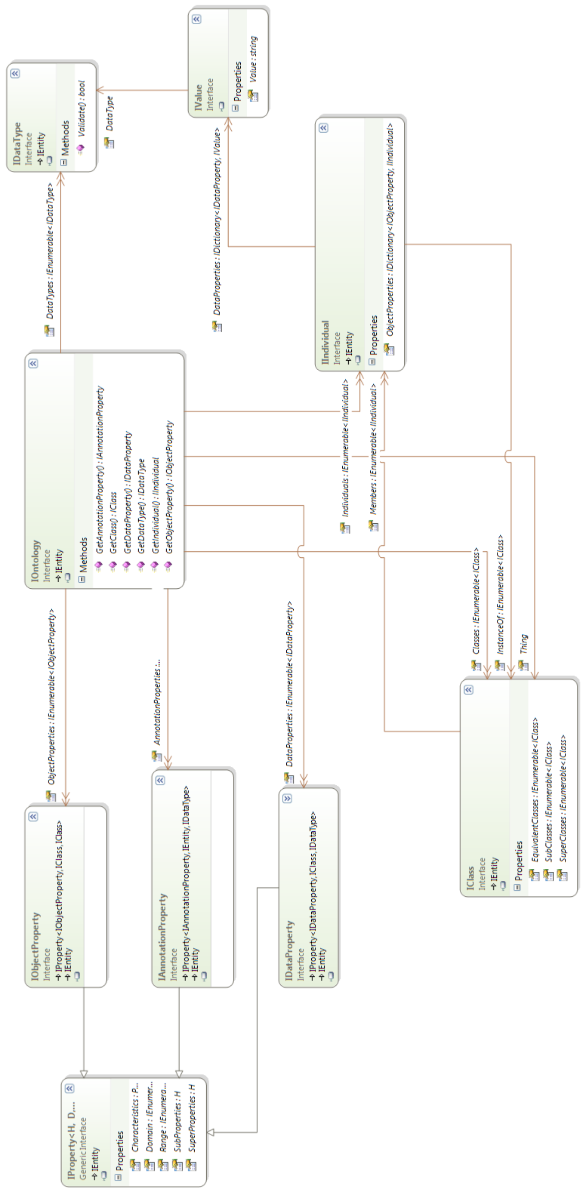

IDataType
Interface
IEntity
Methods
Validate() : bool
DataType
IValue
Interface
Properties
Value : string
IIndividual
Interface
IEntity
Properties
ObjectProperties : IDictionary<IObjectProperty, IIndividual>
DataProperties : IDictionary<IDataProperty, IValue>
DataTypes : IEnumerable<IDataType>
IOntology
Interface
IEntity
Methods
GetAnnotationProperty() : IAnnotationProperty
GetClass() : IClass
GetDataProperty() : IDataProperty
GetDataType() : IDataType
GetIndividual() : IIndividual
GetObjectProperty() : IObjectProperty
Individuals : IEnumerable<IIndividual>
Members : IEnumerable<IIndividual>
Classes : IEnumerable<IClass>
InstanceOf : IEnumerable<IClass>
Thing
ObjectProperties : IEnumerable<IObjectProperty>
AnnotationProperties :...
DataProperties : IEnumerable<IDataProperty>
IClass
Interface
IEntity
Properties
EquivalentClasses : IEnumerable<IClass>
SubClasses : IEnumerable<IClass>
SuperClasses : IEnumerable<IClass>
IObjectProperty
Interface
IProperty<IObjectProperty,IClass,IClass>
IEntity
IAnnotationProperty
Interface
IProperty<IAnnotationProperty,IEntity,IDataType>
IEntity
IDataProperty
Interface
IProperty<IDataProperty,IClass,IDataType>
IEntity
IProperty<H, D,...
Generic Interface
IEntity
Properties
Characteristics : P...
Domain : IEnumer...
Range : IEnumera...
SubProperties : H
SuperProperties : H

Figure A.2: Class diagram of OWL API for .Net showing the cardinality of the relations among the defined interfaces.

Appendix B

CodePlex Repository

The source codes of the practical implementations done for the thesis are available at CodePlex:

- OWL for .NET: `http://owl4net.codeplex.com/`
  Contains the library OWL for .NET as well as the matching features presented in section 5.2.
- Semantic Repository: `http://semrepo.codeplex.com/`
  Contains the code for the SharePoint extensions implemented for the *Semantic Repository* as presented in chapter 4.

Both projects are published under an open source license. Especially the *Semantic Repository* will be developed further as part of *Projekt Repository*

Appendix C

Experiment Results

In this chapter the full result graphs of the experiments presented in section 6.3.2 are depicted for further reference.

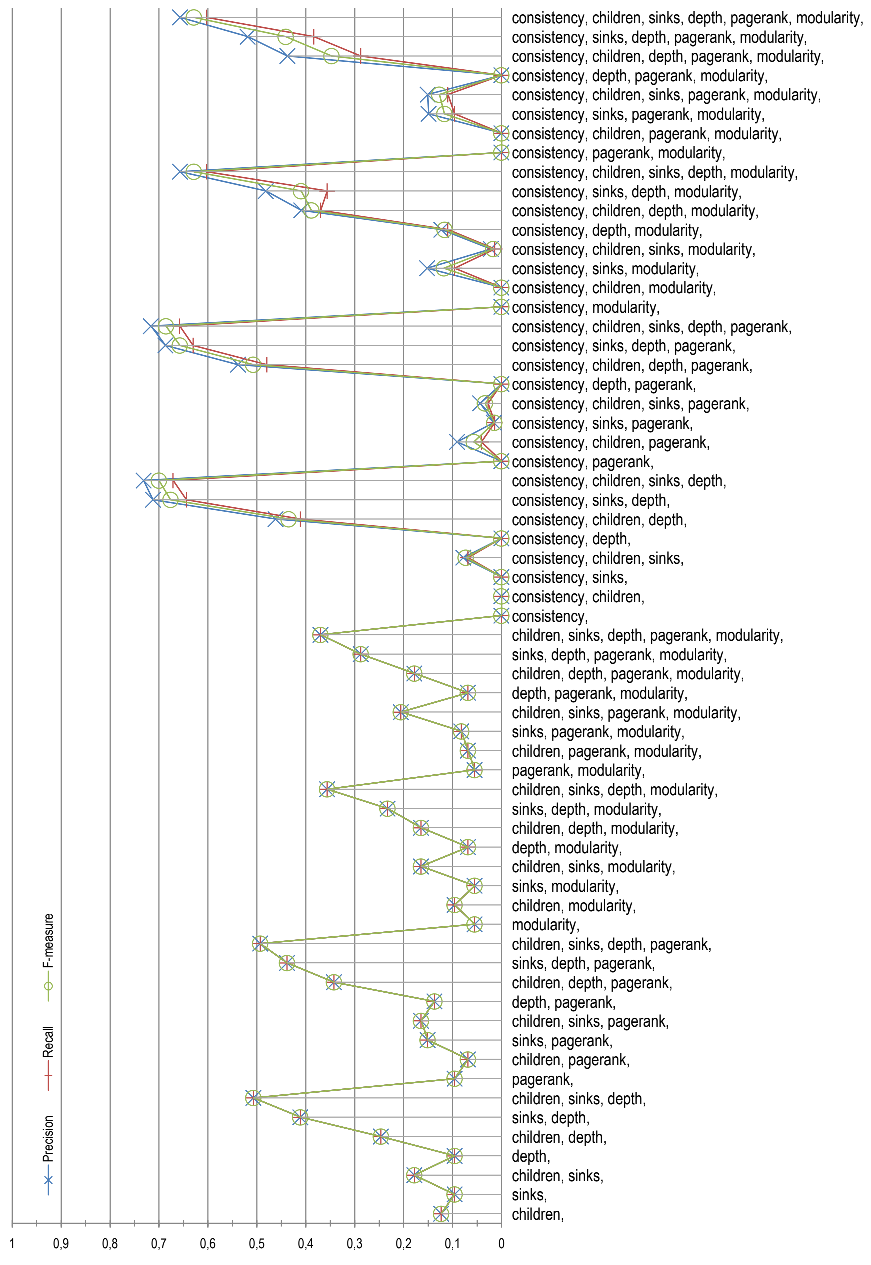


Figure C.1: Full diagram of the results of the matching algorithm matching 101 against 201 using the different similarity measures.

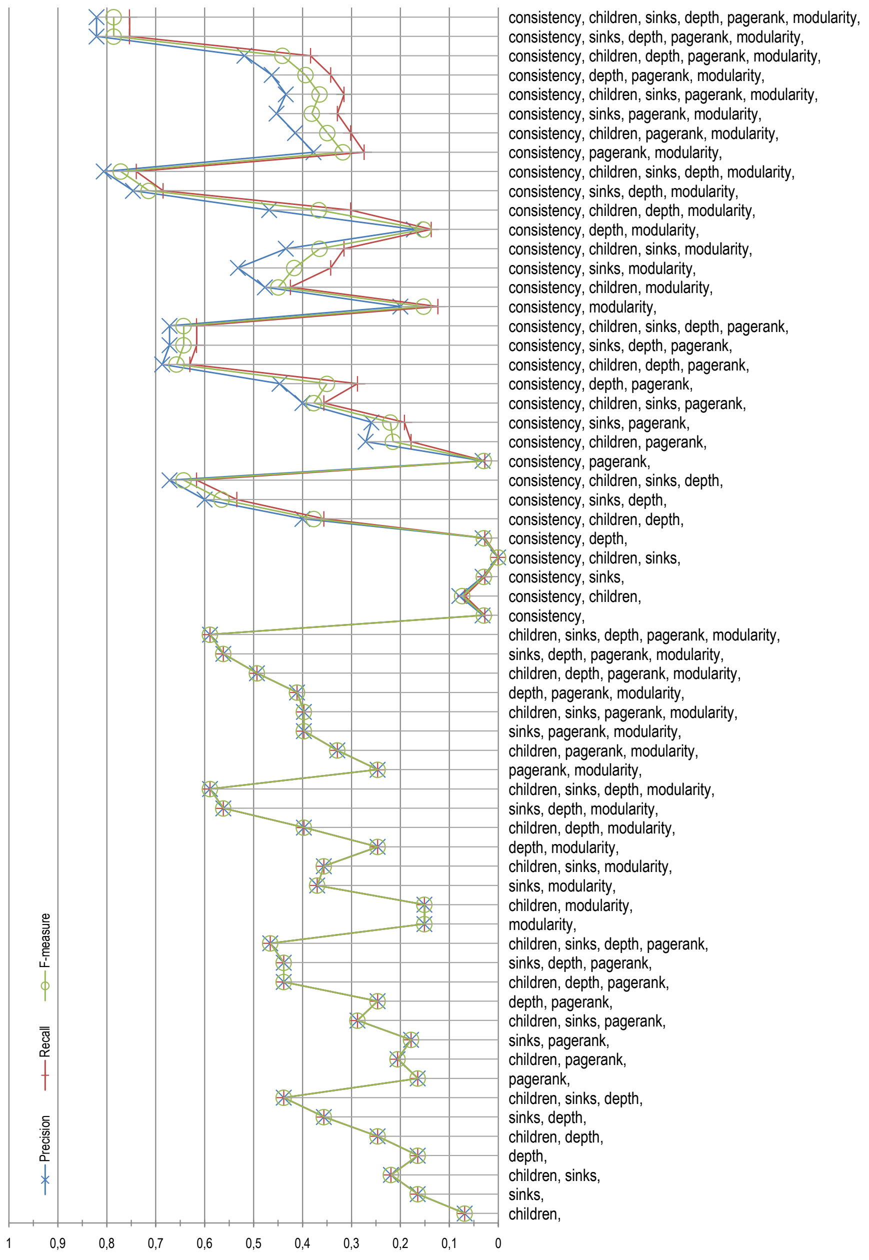


Figure C.2: Full diagram of the results of the matching algorithm matching 101 against 202 using the different similarity measures.

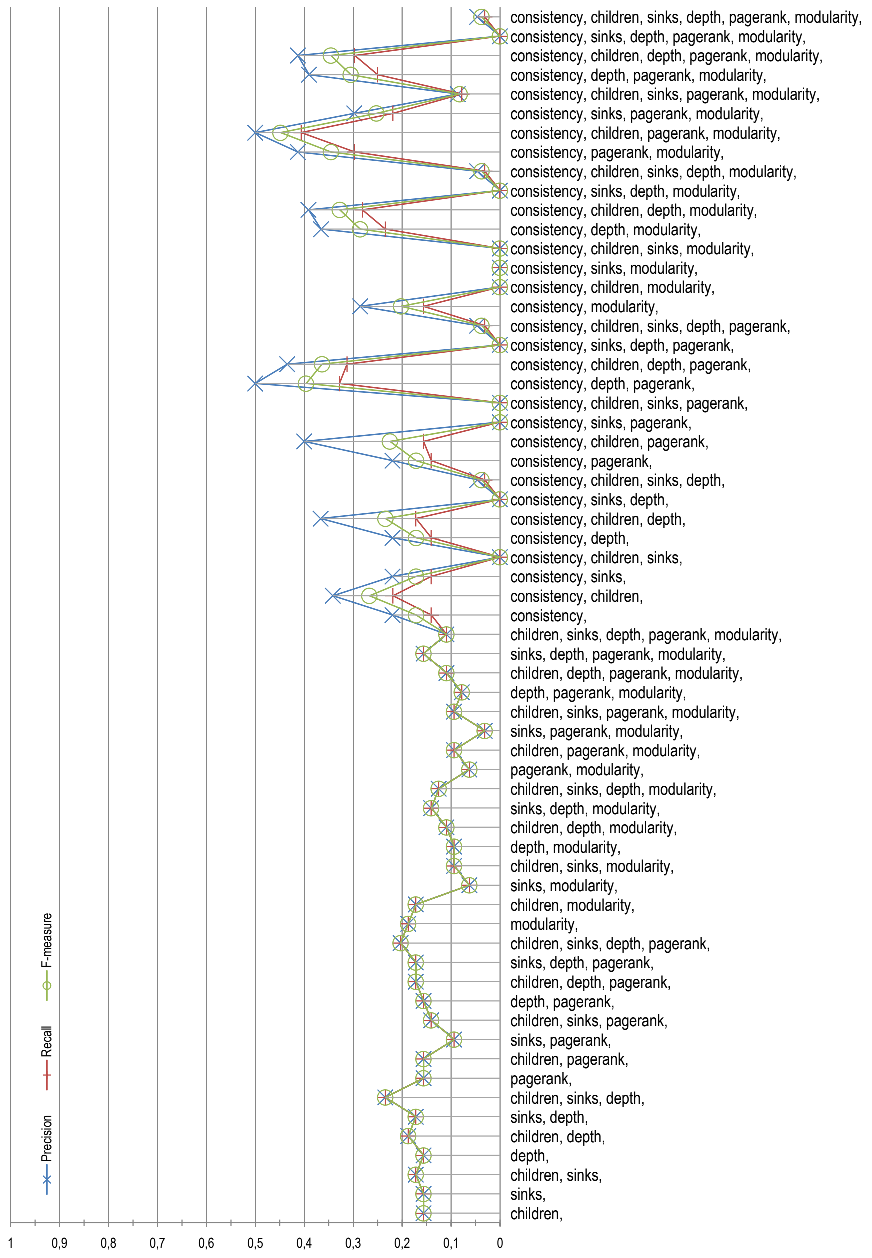


Figure C.3: Full diagram of the results of the matching algorithm matching 101 against 222 using the different similarity measures.

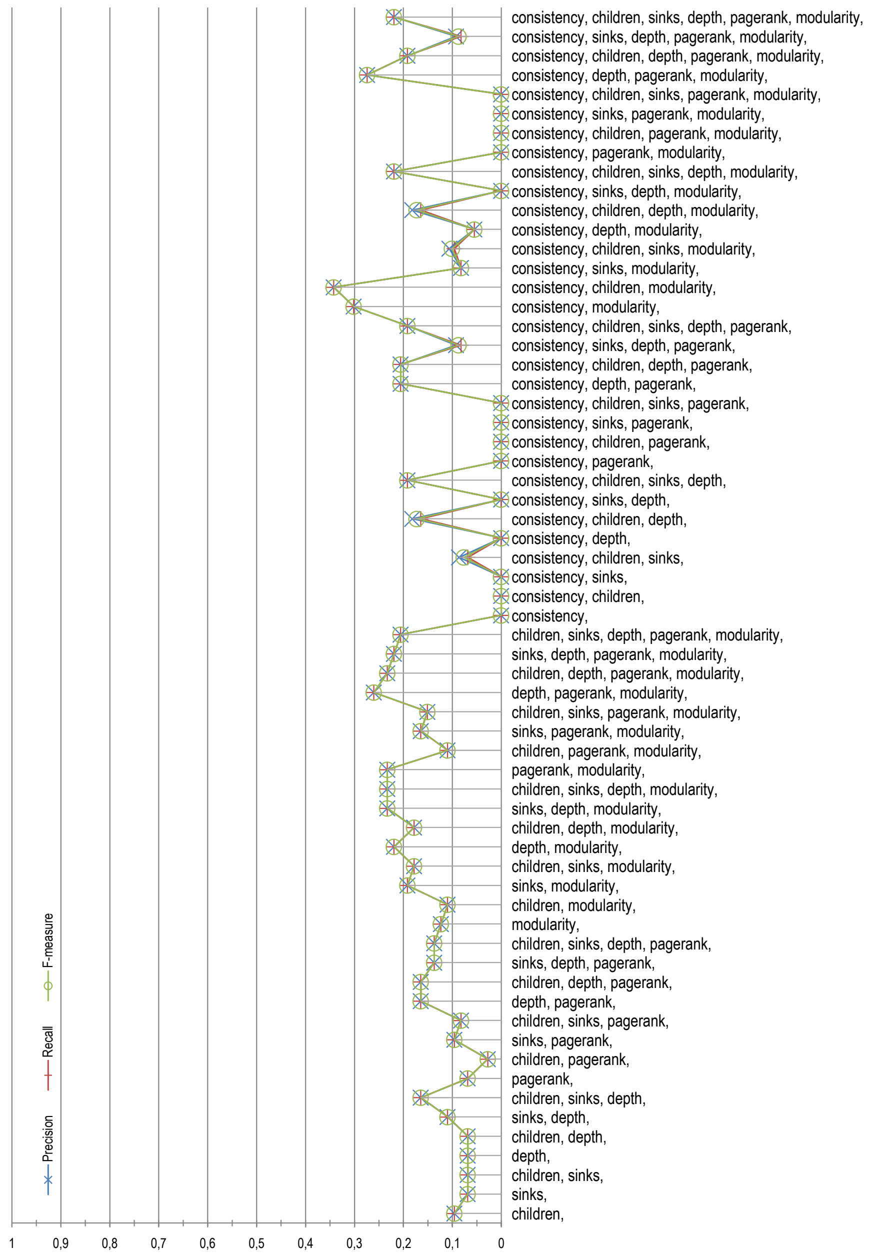
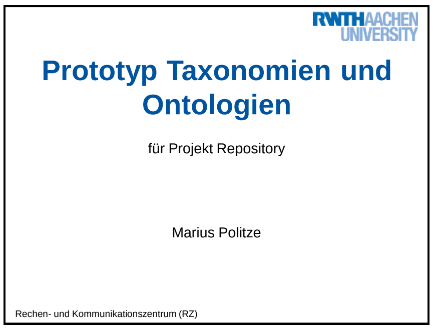


Figure C.4: Full diagram of the results of the matching algorithm matching 101 against 223 using the different similarity measures.

Appendix D

Prototype Presentation

This chapter shows the slides used to introduce the product to the researchers participating in *Projekt Repository*. They were used during the workshops described in section 6.1.



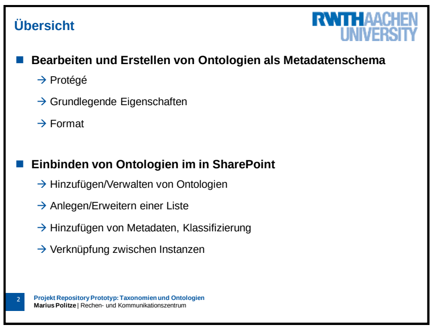
RWTH AACHEN UNIVERSITY

Prototyp Taxonomien und Ontologien

für Projekt Repository

Marius Politze

Rechen- und Kommunikationszentrum (RZ)

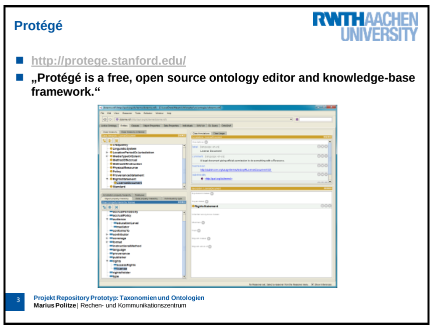


Übersicht

RWTH AACHEN UNIVERSITY

- **Bearbeiten und Erstellen von Ontologien als Metadatenschema**
  - → Protégé
  - → Grundlegende Eigenschaften
  - → Format
- **Einbinden von Ontologien im in SharePoint**
  - → Hinzufügen/Verwalten von Ontologien
  - → Anlegen/Erweitern einer Liste
  - → Hinzufügen von Metadaten, Klassifizierung
  - → Verknüpfung zwischen Instanzen

2 **Projekt Repository Prototyp: Taxonomien und Ontologien**
Marius Politze | Rechen- und Kommunikationszentrum

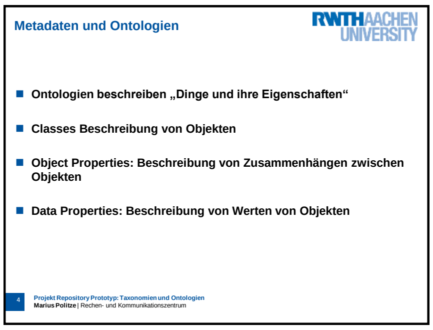


Protégé

RWTH AACHEN UNIVERSITY

- http://protege.stanford.edu/
- **„Protégé is a free, open source ontology editor and knowledge-base framework.“**

3 **Projekt Repository Prototyp: Taxonomien und Ontologien**
Marius Politze | Rechen- und Kommunikationszentrum

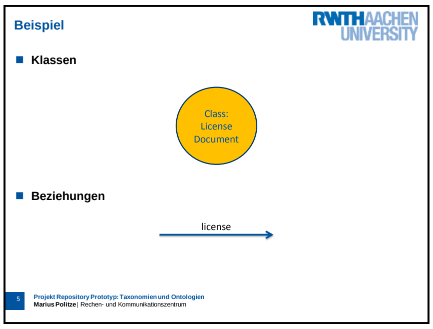


Metadaten und Ontologien

RWTH AACHEN UNIVERSITY

- **Ontologien beschreiben „Dinge und ihre Eigenschaften“**
- **Classes Beschreibung von Objekten**
- **Object Properties: Beschreibung von Zusammenhängen zwischen Objekten**
- **Data Properties: Beschreibung von Werten von Objekten**

4 **Projekt Repository Prototyp: Taxonomien und Ontologien**
Marius Politze | Rechen- und Kommunikationszentrum



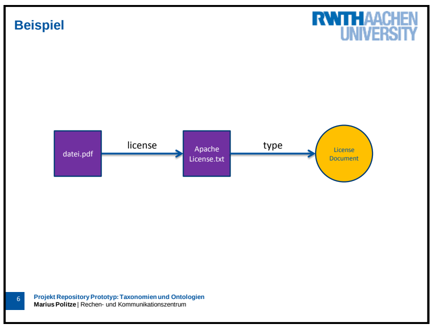
Beispiel

RWTH AACHEN UNIVERSITY

- **Klassen**

- **Beziehungen**

5 **Projekt Repository Prototyp: Taxonomien und Ontologien**
Marius Politze | Rechen- und Kommunikationszentrum

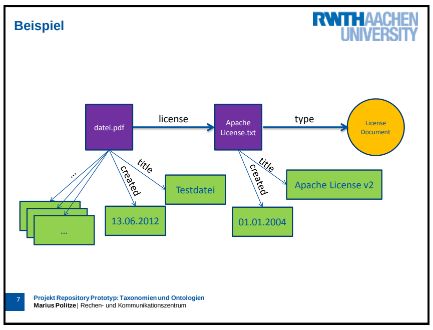


Beispiel

RWTH AACHEN UNIVERSITY

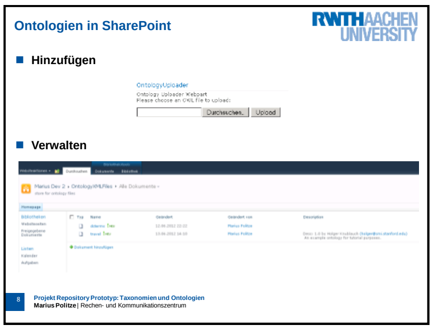
6 **Projekt Repository Prototyp: Taxonomien und Ontologien**
Marius Politze | Rechen- und Kommunikationszentrum

Figure D.1: Slides 1-6 of the presentation of the *Semantic Repository* for the collaborating researchers.



Beispiel

7 Projekt Repository Prototyp: Taxonomien und Ontologien
Marius Politze | Rechen- und Kommunikationszentrum



Ontologien in SharePoint

- **Hinzufügen**

OntologyUploader
Ontology Uploader Webpart
Please choose an OWL file to upload:

- **Verwalten**

Typ	Name	Geändert	Geändert von	Description
	dcterms	12.06.2012 22:22	Marius Politze	
	travel	13.06.2012 14:10	Marius Politze	Desc: 1.0 by Holger Knublauch (holger@smi.stanford.edu) An example ontology for tutorial purposes.

8 Projekt Repository Prototyp: Taxonomien und Ontologien
Marius Politze | Rechen- und Kommunikationszentrum

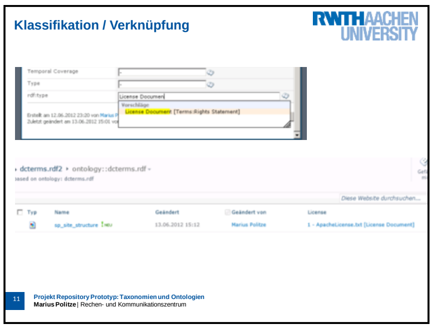
Listen anlegen / erweitern

- **Properties auswählen:**
- **Liste erzeugen**

9 Projekt Repository Prototyp: Taxonomien und Ontologien
Marius Politze | Rechen- und Kommunikationszentrum

Beschreiben der Dateien

10 Projekt Repository Prototyp: Taxonomien und Ontologien
Marius Politze | Rechen- und Kommunikationszentrum



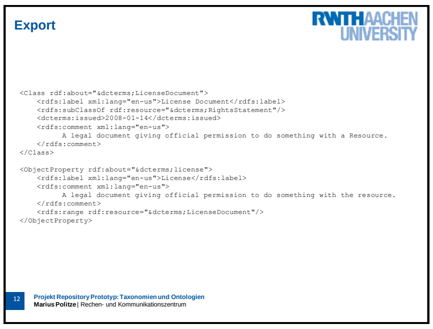
Klassifikation / Verknüpfung

Temporal Coverage
Type
rdf:type
License Document
Vorschläge
License Document [Terms:Rights Statement]
dcterms.rdf2 › ontology::dcterms.rdf
based on ontology: dcterms.rdf
Diese Website durchsuchen...

Erstellt am 12.06.2012 23:20 von Marius P
Zuletzt geändert am 13.06.2012 15:01 vo

Typ	Name	Geändert	Geändert von	License
	sp_site_structure	13.06.2012 15:12	Marius Politze	1 - ApacheLicense.txt [License Document]

11 Projekt Repository Prototyp: Taxonomien und Ontologien
Marius Politze | Rechen- und Kommunikationszentrum



Export

```
<Class rdf:about="&dcterms;LicenseDocument">
    <rdfs:label xml:lang="en-us">License Document</rdfs:label>
    <rdfs:subClassOf rdf:resource="&dcterms;RightsStatement"/>
    <dcterms:issued>2008-01-14</dcterms:issued>
    <rdfs:comment xml:lang="en-us">
          A legal document giving official permission to do something with a Resource.
    </rdfs:comment>
</Class>

<ObjectProperty rdf:about="&dcterms;license">
    <rdfs:label xml:lang="en-us">License</rdfs:label>
    <rdfs:comment xml:lang="en-us">
          A legal document giving official permission to do something with the resource.
    </rdfs:comment>
    <rdfs:range rdf:resource="&dcterms;LicenseDocument"/>
</ObjectProperty>
```

12 Projekt Repository Prototyp: Taxonomien und Ontologien
Marius Politze | Rechen- und Kommunikationszentrum

Figure D.2: Slides 7-12 of the presentation of the *Semantic Repository* for the collaborating researchers.

Appendix E

Bibliography

[1] M. Kleiner, P. Omling, I. Halliday, P. Gruss, G. Makara, M. Makarow, A. Migus, E. Nexø, and J. Marks, “The EUROHORCs and ESF Vision on a Globally Competitive ERA and their Road Map for Actions to Help Build It,” *Science Policy Briefing*, no. 33, Jun. 2008.

[2] U. Eich, “RWTH Aachen, Hochschulbibliothek,” University Library at RWTH Aachen University, 2012. [Online]. Available: http://www.bth.rwth-aachen.de/

[3] R. Knüchel-Clarke, “Head and Secretary [UK Aachen],” Institute of Pathology, University Hospital Aachen, 2012. [Online]. Available: http://www.pathologie.ukaachen.de

[4] M. R. N. Jansen, “RWTHsbg,” Department of History of Urbanization, RWTH Aachen University, 2012. [Online]. Available: http://sbg.arch.rwth-aachen.de/

[5] H. Schüttrumpf, “IWW-Homepage,” Institute of Hydraulic Engineering and Water Resources Management, RWTH Aachen University, 2012. [Online]. Available: http://www.iww.rwth-aachen.de/

[6] K. Brühl, “Rechen- und Kommunikationszentrum der RWTH Aachen,” Center for Computing and Communication of RWTH Aachen University, 2012. [Online]. Available: http://www.rz.rwth-aachen.de/

[7] U. Schröder, “CIL,” Center for Innovative Learning Technologies, RWTH Aachen University, 2012. [Online]. Available: http://www.cil.rwth-aachen.de/

[8] C. Bischof, U. Eich, R. Knüchel-Clarke, M. Jansen, and H. Schüttrumpf, “ProjektRepository, Ein pandisziplinäres Repository für Forschungsprojekte als Komponente einer niederschwelligen webbasierten Kooperationsinfrastruktur,” 2009, DFG-Antrag.

[9] U. Schröder, U. Eich, R. Knüchel-Clarke, M. Jansen, and H. Schüttrumpf, “ProjektRepository, Ein pandisziplinäres Repository für Forschungsprojekte als Komponente einer niederschwelligen webbasierten Kooperationsinfrastruktur,” 2011, Zwischenbericht.

[10] T. Berners-Lee and M. Fischetti, *Weaving the web: The original design and ultimate destiny of the world wide web by its inventor.* San Francisco: Harper, 1999.

[11] L. Feigenbaum, I. Herman, T. Hongsermeier, E. Neumann, and S. Stephens, "The Semantic Web in Action," *Scientific American*, vol. 297, pp. 90–97, Dec. 2007.

[12] *Pathology Reporting in Breast Cancer Screening*, 2nd ed., ser. UK: Breast Screening Publications. National Coordinating Group for Breast Screening Pathology, 1995.

[13] *European Guidelines for Quality Assurance in Mammography Screening.* European Commission, 1996.

[14] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Aquisition*, vol. 5, pp. 199–220, 1993.

[15] D. Fensel, *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce.* Springer-Verlag, 2004.

[16] L. M. Garshol, "Metadata? Thesauri? Taxonomies? Topic Maps! Making Sense of it all," *Journal of Information Science*, vol. 30, no. 4, pp. 378–391, Aug. 2004.

[17] W3C OWL Working Group, "OWL 2 Web Ontology Language Document Overview," W3C, Tech. Rep., Oct. 2009. [Online]. Available: http://www.w3.org/TR/2009/REC-owl2-overview-20091027/

[18] G. Klyne and J. J. Carroll, "Resource Description Framework (RDF): Concepts and Abstract Syntax," W3C, Tech. Rep., Feb. 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[19] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," Internet Society (ISOC), Internet Engineering Task Force RFC 3986, Jan. 2005. [Online]. Available: http://tools.ietf.org/html/rfc3986

[20] W3C OWL Working Group, "OWL Web Ontology Language Semantics and Abstract Syntax," W3C, Tech. Rep., Feb. 2004. [Online]. Available: http://www.w3.org/TR/owl-semantics/

[21] T. Bray, J. P. Paoli, C. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (XML) 1.0," W3C, Tech. Rep., 11 2008. [Online]. Available: http://www.w3.org/TR/2008/REC-xml-20081126/

[22] B. Motik, P. F. Patel-Schneider, and B. Parsia, "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax," W3C, Tech. Rep., Oct. 2009. [Online]. Available: http://www.w3.org/TR/owl2-syntax/

[23] M. Horridge and P. F. Patel-Schneider, "OWL 2 Web Ontology Language Manchester Syntax," W3C, Tech. Rep., Oct. 2009. [Online]. Available: http://www.w3.org/TR/owl2-manchester-syntax/

[24] J. Day-Richter, "The OBO Flat File Format Specification, version 1.2," The Gene Ontology Consortium, Tech. Rep., Nov. 2004. [Online]. Available: http://www.geneontology.org/GO.format.obo-1_2.shtml

[25] P. F. Patel-Schneider and B. Swartout, "Description-Logic Knowledge Representation System Specification from the KRSS Group of the ARPA Knowledge Sharing Effort," •, Tech. Rep., Nov. 1993.

[26] W3C RDF Working Group, "Turtle," W3C, Tech. Rep., Aug. 2011. [Online]. Available: http://www.w3.org/TR/2011/WD-turtle-20110809/

[27] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition," W3C, Tech. Rep., Oct. 2004. [Online]. Available: http://www.w3.org/TR/xmlschema-1/

[28] Stanford Center for Biomedical Informatics Research, "The Protégé Ontology Editor and Knowledge Acquisition System," 2012. [Online]. Available: http://protege.stanford.edu

[29] "DCMI Home: Dublin Core® Metadata Initiative (DCMI)," Dublin Core Metadata Initiative. [Online]. Available: http://dublincore.org/

[30] J. McEntyre and D. Lipman, "PubMed: bridging the information gap," *Canadian Medical Association Journal*, vol. 164, pp. 1317–1319.

[31] *LIDO (Lightweight Information Describing Objects): Making it easier to deliver information to portals*, International Council of Museums. [Online]. Available: http://www.lido-schema.org/documents/LIDO-Handout.pdf

[32] ISO-19115, "ISO 19115:2003 Geographic information – Metadata," Tech. Rep., 2003.

[33] M. Ferdinand, C. Zirpins, and D. Trastour, "Lifting XML Schema to OWL," in *Web Engineering - 4th International Conference, ICWE 2004, Munich, Germany, July 26-30, 2004, Proceedings*, N. Koch, P. Fraternali, and M. Wirsing, Eds. Springer Heidelberg, 2004, pp. 354–358.

[34] R. G. González, "A semantic web approach to digital rights management," Ph.D. dissertation, Universitat Pompeu Fabra, Barcelona, Nov. 2005.

[35] Microsoft Cooperation, *Server and Site Architecture: Object Model Overview*, May 2010. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms473633.aspx

[36] M. Horridge and S. Bechhofer, "The OWL API: A Java API for Working with OWL 2 Ontologies," in *OWLED*, ser. CEUR Workshop Proceedings, R. Hoekstra and P. F. Patel-Schneider, Eds., vol. 529, Oct. 2008.

[37] O. Ben-Yitzhak, N. Golbandi, N. Har'El, R. Lempel, A. Neumann, S. Ofek-Koifman, D. Sheinwald, E. Shekita, B. Sznajder, and S. Yogev, "Beyond basic faceted search," in *Proceedings of the international conference on Web search and web data mining*, ser. WSDM '08. New York, NY, USA: ACM, 2008, pp. 33–44. [Online]. Available: http://doi.acm.org/10.1145/1341531.1341539

[38] M. A. Hearst, "Design recommendations for hierarchical faceted search interfaces," in *SIGIR, Workshop on Faceted Search*, Aug. 2006, pp. 26–30.

[39] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," W3C, Tech. Rep., Jan. 2008. [Online]. Available: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

[40] M. d'Aquin, M. Sabou, M. Dzbor, C. Baldassarre, L. Gridinoc, S. Angeletou, and E. Motta, "Watson: a gateway for the semantic web," in *The 4th Annual European Semantic Web Conference*, ser. ESWC 2007, 2007.

[41] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: a search and metadata engine for the semantic web," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, ser. CIKM '04. New York, NY, USA: ACM, 2004, pp. 652–659.

[42] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000.

[43] J. Euzenat and P. Shvaiko, *Ontology matching.* Heidelberg (DE): Springer-Verlag, 2007.

[44] P. Shvaiko and J. Euzenat, "Ontology Alignment Evaluation Initiative::Home," 2012. [Online]. Available: http://oaei.ontologymatching.org

[45] P. Shvaiko and J. Euzenat, "Ontology matching: state of the art and future challenges," *IEEE Transactions on knowledge and data engineering*, 2012, in press. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.253

[46] *MaasMatch results for OAEI 2011*, ser. The Sixth International Workshop on Ontology Matching. Bonn, Germany: ISWC, Oct. 2011.

[47] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 10, no. 5, pp. 695–703, 1988.

[48] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, no. 4, pp. 377–388, 1996.

[49] M. Mukherjee and L. Holder, "Graph-based data mining on social networks," Ph.D. dissertation, University of Texas at Arlington, 2004.

[50] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe, "A framework for community identification in dynamic social networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2007, pp. 717–726.

[51] S. Kpodjedo, P. Galinier, and G. Antoniol, "Enhancing a tabu algorithm for approximate graph matching by using similarity measures," *Evolutionary Computation in Combinatorial Optimization*, pp. 119–130, 2010.

[52] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, 2006. [Online]. Available: doi:10.1103/PhysRevE.74.036104

[53] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep. 1999-66, November 1999. [Online]. Available: http://ilpubs.stanford.edu:8090/422/

[54] F. Glover, "Tabu search-part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[55] F. Glover, "Tabu search-part ii," *ORSA Journal on computing*, vol. 2, no. 1, pp. 4–32, 1990.

[56] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[57] M. Jaspers, T. Steen, C. Bos, and M. Geenen, "The think aloud method: a guide to user interface design," *International journal of medical informatics*, vol. 73, no. 11, pp. 781–795, 2004.

[58] *Contributing Contributing Content to Content to Cultural Heritage Repositories*, International Council of Museums. [Online]. Available: http://www.lido-schema.org/documents/LIDO-Introduction.pdf

[59] T. Bray, "The RDF.net Challenge," 2003. [Online]. Available: http://www.tbray.org/ongoing/When/200x/2003/05/21/RDFNet